Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 1

Н. В. Батин, Н. В. Хаджинова

ПРИМЕНЕНИЕ ПАКЕТА MS OFFICE ДЛЯ ОБРАБОТКИ ИНФОРМАЦИИ

Рекомендовано УМО по образованию
в области информатики и радиоэлектроники для специальности 1-53 01 02
«Автоматизированные системы обработки информации»
в качестве учебно-методического пособия

Минск БГУИР 2013

УДК 004(076)
ББК 32.973.202–018.2я73
К63

Р е ц е н з е н т ы:

кафедра информатики и вычислительной техники государственного учреждения образования «Институт подготовки научных кадров Национальной академии наук Республики Беларуси (протокол №9 от 28.05.2012);

заведующий кафедрой экономической информатики учреждения образования «Белорусский государственный экономический университет», кандидат технических наук, доцент Б. А. Железко

Компьютерные информационные технологии. Лабораторный К63 практикум : учеб.-метод. пособие. В 2 ч. Ч. 1 : Применение пакета MS Office для обработки информации / Н. В. Батин, Н. В. Хаджинова. – Минск : БГУИР, 2013. – 139 с. : ил.
ISBN 978-985-488-638-1 (ч.1).

Часть 1 учебно-методического пособия состоит из описания восьми лабораторных работ, теоретических сведений и методических указаний к каждой работе. Цель лабораторных работ – приобретение практических навыков применения развитых возможностей пакета MS Office для решения задач обработки информации.

Издание предназначено для студентов, изучающих современные компьютерные технологии обработки информации и их применение для решения задач автоматизации управления.

УДК 004(076)
ББК 32.973.202–018.2я73

ISBN 978-985-488-638-1 (ч.1)
ISBN 978-985-488-639-8

ЛАБОРАТОРНАЯ РАБОТА №1

РАЗВИТЫЕ ВОЗМОЖНОСТИ ТАБЛИЧНОГО ПРОЦЕССОРА MS EXCEL: БАЗЫ ДАННЫХ

Цель работы – Изучение возможностей табличного процессора MS Excel для работы с базами данных: сортировка, группирование, поиск, выборка данных по заданным критериям.

1.1 Понятие базы данных в MS Excel

Под базой данных в Excel понимают данные, введенные в несколько столбцов, с заголовками в первой строке (рисунок 1.1). Столбцы называются полями базы данных (в данном примере это «Фамилия», «Категория», «Отдел» и «Заработная плата»), а строки – записями.

Примечание – Если на рабочем листе вместе с базой данных размещаются еще какие-либо данные, то они должны быть отделены от базы данных хотя бы одной строкой и столбцом.

В данной работе на примерах рассматриваются основные возможности MS Excel по работе с базами данных. В примерах, если не указано иное, используется база данных, приведенная на рисунке 1.1.

Основные команды для работы с базами данных находятся в меню **Данные** (хотя для операций с базами данных используются и элементы других меню). Кроме того, имеется набор функций для работы с базами данных, рассматриваемый в подразделе 1.7.

	A	B	C	D	E
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**	
2	Андреев	штатный	1	240	
3	Воробьев	стажер	4	100	
4	Галкин	внештатный		270	
5	Гурин	совместитель	1	270	
6	Иванов	штатный	2	200	
7	Ковалев	штатный	1	500	
8	Котов	штатный	3	430	
9	Петров	штатный	2	450	
10	Семенов	совместитель	3	320	
11	Сидоров	штатный	2	380	
12	Синицын	стажер	2	120	
13	Сорокин	внештатный		320	
14	Степанов	штатный	4	420	
15	Яковлев	стажер	3	280	
16					

Рисунок 1.1 – Пример базы данных в MS Excel

1.2 Выбор отображаемых столбцов

Пример 1.1 – Требуется отображать только фамилии и заработную плату сотрудников.

1 Поместить курсор в любую ячейку столбца, который требуется скрыть (в данном случае – в столбец В). Выбрать **Формат – Столбец – Скрыть**. Аналогично можно скрыть несколько столбцов, если это требуется.

2 Чтобы скрытый столбец снова отображался, выбрать любые две ячейки в соседних столбцах (в данном примере – А и С), *между которыми* располагался скрытый столбец, и выбрать команду **Формат – Столбец – Отобразить**.

1.3 Сортировка данных

Пример 1.2 – Отсортировать список работников по номеру отдела, а в пределах каждого отдела – по фамилии.

1 Поместить курсор в любую ячейку базы данных.

2 Выбрать команду **Данные – Сортировка**. Убедиться, что база данных автоматически выделена.

Примечание – Если появляется сообщение о том, что обнаружены данные вне указанного выделения, то следует выбрать предлагаемое действие **Автоматически расширить выделенный диапазон** (чтобы была автоматически выделена вся база данных) и нажать кнопку **Сортировка**. Если появляется сообщение «Список не найден», то необходимо убедиться, что база данных отделена свободными ячейками от других данных, а также что курсор установлен в пределах базы данных, и снова выбрать команду **Данные – Сортировка**.

3 В появившемся окне **Сортировка диапазона** указать критерии сортировки. Для этого в поле **Сортировать по** выбрать *Отдел*, установить переключатель **По возрастанию**. В поле **Затем по** выбрать *Фамилия*; также установить переключатель **По возрастанию** (чтобы данные были отсортированы по алфавиту). Нажать **ОК**.

1.4 Фильтрация данных

Фильтрация данных – выделение части строк (записей) по некоторым условиям. В Excel для этого имеются два основных способа: автофильтр (стандартный набор возможностей фильтрации) и расширенный фильтр (задание условий фильтрации по выбору пользователя).

1.4.1 Автофильтр

Для перехода в режим автофильтра необходимо поместить курсор в любую ячейку базы данных и выбрать команду **Данные – Фильтр – Автофильтр**. Возле каждого заголовка появляется кнопка вызова окна условий фильтрации для соответствующего поля (столбца).

Необходимо учитывать, что если условия фильтрации заданы для нескольких полей, то отбираются только строки, соответствующие *всем* этим условиям. Другими словами, условия, заданные для нескольких полей, связаны союзом И (а не ИЛИ).

Чтобы восстановить отображение всех данных, требуется выбрать **Данные – Фильтр – Отобразить все**. Чтобы выйти из режима автофильтра (убрать кнопки у заголовков полей), требуется выбрать **Данные – Фильтр** и снять флажок **Автофильтр.**

Пример 1.3 – Получить список работников, работающих в отделах 1 и 3 и имеющих заработную плату не ниже 300 ден. ед. Для этого выполнить следующее.

1 Перейти в режим автофильтра.

2 Нажать кнопку заголовка *Отдел*. Из появившегося меню выбрать **Условие**. В появившемся окне выбрать условие **равно** и ввести **1**. Установить переключатель **ИЛИ**. Установить второе условие – **равно 3**. Нажать **ОК**. Убедиться, что отображаются данные только о работниках отделов 1 и 3.

3 Нажать кнопку заголовка *Заработная плата*. Выбрать **Условие.** Установить условие **больше или равно 300**. Нажать **ОК**. Убедиться, что отображаются данные в соответствии с заданными условиями.

4 Восстановить отображение всех данных. Выйти из режима автофильтра.

Пример 1.4 – Получить список работников отдела 2, имеющих зарплату от 250 до 400 ден. ед. Для этого выполнить следующее.

1 Перейти в режим автофильтра.

2 Для поля *Отдел* установить **Условие – равно 2.**

3 Для поля *Заработная плата* установить **Условие – больше или равно 250 И меньше или равно 400**.

4 Убедиться, что отображаются данные в соответствии с заданными условиями. Восстановить отображение всех данных.

Пример 1.5 – Получить список из трех работников, имеющих наименьшую зарплату. Для этого нажать кнопку заголовка *Заработная плата*. Выбрать **Первые 10**. В появившемся окне установить: **Показать 3 наименьших элементов списка**. Нажать **ОК**. Убедиться, что отображаются требуемые данные. Восстановить отображение всех данных.

Пример 1.6 – Получить список работников, составляющих 25 % всех работающих на предприятии и имеющих наибольшую зарплату. Для этого нажать кнопку заголовка *Заработная плата*. Выбрать **Первые 10**. В появившемся окне установить: **Показать 25 наибольших % от количества элементов**. Нажать **ОК**. Убедиться, что отображаются требуемые данные. Восстановить отображение всех данных.

Пример 1.7 – Получить список из трех работников отдела 2, имеющих максимальную зарплату (в этом отделе).

Эту операцию невозможно выполнить с помощью автофильтра непосредственно. Порядок выполнения операции может быть, например, следующим.

1 Для поля *Отдел* установить **Условие – равно 2**.

2 Скопировать полученные данные (включая заголовки) на новый рабочий лист.

3 На новом рабочем листе для поля *Заработная плата* выбрать условие отбора **Первые 10** и установить **Показать 3 наибольших элементов списка**.

4 Убедиться, что отображаются данные в соответствии с заданными условиями. Восстановить отображение всех данных. Вернуться на рабочий лист с базой данных и также восстановить отображение всех данных.

Примечания

1 Вместо копирования данных на другой рабочий лист можно, отобрав работников отдела 2, просто отсортировать полученный список по полю *Заработная плата* и просмотреть данные о необходимом числе работников с максимальной зарплатой.

2 Как упомянуто выше, если условия фильтрации заданы для нескольких полей, то отбираются только строки, соответствующие *всем* этим условиям. Поэтому, если в данном примере просто установить для поля *Отдел* условие отбора **равно 2**, а затем (на этом же рабочем листе) применить для поля *Заработная плата* условие отбора **Первые 10** – **Показать 3 наибольших элементов списка**, то результат будет неправильным: будут показаны работники отдела 2, входящие в число трех работников предприятия, имеющих максимальную зарплату.

Задания для самостоятельного выполнения:

– получить список всех работников – стажеров и совместителей, работающих в отделах 1 и 3. Отсортировать его по заработной плате. Поле *Категория* не должно отображаться;

– из всех штатных работников предприятия отобрать 25 %, имеющих наименьшую зарплату, и отсортировать их список по фамилиям (по алфавиту).

1.4.2 Расширенный фильтр

Расширенный фильтр применяется в случаях, когда требуется фильтрация по сложным условиям, которые нельзя задать с помощью автофильтра, например:

– более двух условий для одного поля;

– несколько диапазонов условий для одного поля;

– отбор данных, соответствующих заданным условиям хотя бы для одного поля;

– задание условия с помощью формулы и т. д.

Пример 1.8 – Получить список работников отделов 1, 3 и 4.

В данном случае для одного поля (*Отдел*) имеются три условия отбора, а автофильтр позволяет задать не более двух. Поэтому необходимо использовать расширенный фильтр.

1 В рабочем листе с базой данных указать условия отбора (расширенный фильтр). Они должны включать имена полей, по которым выполняется отбор, и собственно условия. Условия отбора должны быть отделены от базы

данных хотя бы одной свободной строкой или столбцом. Для данной задачи условия отбора могут иметь, например, такой вид, как показано на рисунке 1.2 (в данном примере условия отбора указаны в ячейках D17:D20).

2 Установить курсор в любую ячейку базы данных. Выбрать **Данные – Фильтр – Расширенный фильтр**.

3 На экран выводится окно **Расширенный фильтр**. Убедиться, что в поле **Исходный диапазон** указаны ячейки с базой данных (в данном примере – диапазон A1:D15). В поле **Диапазон условий** указать диапазон ячеек, содержащий условия отбора (в данном примере – D17:D20). Установить переключатель **Обработка: Фильтровать список на месте** (будет отфильтрована база данных) или **Скопировать результат в другое место** (база данных будет отображаться полностью, а отфильтрованные данные будут выведены в другое место). Если выбрано **Скопировать результат в другое место**, то в поле **Поместить результат в диапазон** необходимо указать начальную ячейку области, куда требуется вывести результаты фильтрации. Нажать **ОК**.

4 Убедиться, что отображаются данные в соответствии с заданными условиями. Восстановить отображение всех данных, выбрав **Данные – Фильтр – Отобразить все**.

Пример 1.9 – Получить список следующих работников: работники отдела 1 с зарплатой от 250 ден. ед. и выше, отдела 2 – с зарплатой от 150 до 400 ден. ед., отдела 3 – от 300 ден. ед. и выше.

Подготовить расширенный фильтр, как показано на рисунке 1.3. В окне **Расширенный фильтр** в поле **Диапазон условий** указать диапазон C18:E21.

	A	B	C	D	E
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**	
2	Андреев	штатный	1	240	
3	Воробьев	стажер	4	100	
4	Галкин	внештатный		270	
5	Гурин	совместитель	1	270	
6	Иванов	штатный	2	200	
7	Ковалев	штатный	1	500	
8	Котов	штатный	3	430	
9	Петров	штатный	2	450	
10	Семенов	совместитель	3	320	
11	Сидоров	штатный	2	380	
12	Синицын	стажер	2	120	
13	Сорокин	внештатный		320	
14	Степанов	штатный	4	420	
15	Яковлев	стажер	3	280	
16					
17				Отдел	
18				1	
19				3	
20				4	
21					

Рисунок 1.2 – Расширенный фильтр для примера 1.8

	A	B	C	D	E
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**	
2	Андреев	штатный	1	240	
3	Воробьев	стажер	4	100	
4	Галкин	внештатный		270	
5	Гурин	совместитель	1	270	
6	Иванов	штатный	2	200	
7	Ковалев	штатный	1	500	
8	Котов	штатный	3	430	
9	Петров	штатный	2	450	
10	Семенов	совместитель	3	320	
11	Сидоров	штатный	2	380	
12	Синицын	стажер	2	120	
13	Сорокин	внештатный		320	
14	Степанов	штатный	4	420	
15	Яковлев	стажер	3	280	
16					
17					
18			**Отдел**	**Заработная плата**	**Заработная плата**
19			1	>=250	
20			2	>=150	<=400
21			3	>=300	
22					

Рисунок 1.3 – Расширенный фильтр для примера 1.9

Пример 1.10 – Получить список следующих работников: все работники отделов 1 и 4; стажеры из отдела 2; совместители и стажеры из отдела 3.

Расширенный фильтр для этой задачи показан на рисунке 1.4.

Пример 1.11 – Получить список следующих работников: все работники отдела 3, а также все совместители и стажеры (из всех отделов).

Расширенный фильтр для этой задачи показан на рисунке 1.5.

Отдел	Категория
1	
2	стажер
3	совместитель
3	стажер
4	

Рисунок 1.4 – Расширенный фильтр для примера 1.10

Отдел	Категория
3	
	стажер
	совместитель

Рисунок 1.5 – Расширенный фильтр для примера 1.11

Пример 1.12 – Получить список работников, имеющих зарплату выше средней по предприятию.

Расширенный фильтр для этой задачи показан на рисунке 1.6. В данном случае расширенный фильтр будет задан с использованием формулы.

1 В ячейке A18 ввести произвольную подпись для расширенного фильтра, например, «Условие». Эта подпись должна *отличаться* от подписей полей базы данных (т. е. нельзя использовать, например, подпись «Заработная плата»).

2 В ячейке A19 ввести формулу условия отбора (расширенного фильтра): **=D2>СРЗНАЧ($D$2:$D$15).** Здесь D2 – ссылка на *первую* ячейку поля, к которому относится заданное условие (в данном случае – поля *Заработная плата*). Эта ссылка должна быть *относительной* (т. е. необходимо указать именно D2, а не $D$2). Остальные ссылки в формуле должны быть *абсолютными* (т. е. необходимо указать именно **$D$2:$D$15**, а не **D2:D15**). В ячейке E17 в данном случае выводится значение ЛОЖЬ, так как величина в ячейке D2 (зарплата работника по фамилии Андреев) ниже средней зарплаты по предприятию.

Примечание – Размещение расширенного фильтра и других данных, конечно, может отличаться от приведенного в данном примере. Обязательно только, чтобы база данных была отделена от всех других данных (в том числе от расширенного фильтра) хотя бы одной свободной строкой и столбцом.

3 Установить курсор в любую ячейку базы данных. Выбрать **Данные – Фильтр – Расширенный фильтр**. В появившемся окне **Расширенный фильтр** убедиться, что в поле **Исходный диапазон** указаны ячейки с базой данных (A1:D15). В поле **Диапазон условий** указать диапазон ячеек, содержащий условия отбора (A18:A19). Используя переключатель **Обработка,** выбрать место для вывода результатов фильтрации. Нажать **OK.**

4 Убедиться, что отображаются данные в соответствии с заданными условиями. Восстановить отображение всех данных.

Пример 1.13 – Получить список работников отдела 2, имеющих зарплату выше средней по предприятию.

Расширенный фильтр для этой задачи показан на рисунке 1.7. Формула расширенного фильтра – такая же, как в примере 1.12.

	A	B	C	D
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**
2	Андреев	штатный	1	240
3	Воробьев	стажер	4	100
4	Галкин	внештатный		270
5	Гурин	совместитель	1	270
6	Иванов	штатный	2	200
7	Ковалев	штатный	1	500
8	Котов	штатный	3	430
9	Петров	штатный	2	450
10	Семенов	совместитель	3	320
11	Сидоров	штатный	2	380
12	Синицын	стажер	2	120
13	Сорокин	внештатный		320
14	Степанов	штатный	4	420
15	Яковлев	стажер	3	280
16				
17				
18		Условие		
19		=D2>СРЗНАЧ($D$2:$D$15)		
20				

Рисунок 1.6 – Расширенный фильтр для примера 1.12

	A	B	C	D
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**
2	Андреев	штатный	1	240
3	Воробьев	стажер	4	100
4	Галкин	внештатный		270
5	Гурин	совместитель	1	270
6	Иванов	штатный	2	200
7	Ковалев	штатный	1	500
8	Котов	штатный	3	430
9	Петров	штатный	2	450
10	Семенов	совместитель	3	320
11	Сидоров	штатный	2	380
12	Синицын	стажер	2	120
13	Сорокин	внештатный		320
14	Степанов	штатный	4	420
15	Яковлев	стажер	3	280
16				
17				
18	**Отдел**	**Условие**		
19	2	=D2>СРЗНАЧ($D$2:$D$15)		

Рисунок 1.7 – Расширенный фильтр для примера 1.13

Таким образом, можно сформулировать следующие **правила построения расширенных фильтров**:

– условия, связанные союзом ИЛИ, размещаются в разных строках. Так, в примере 1.8: отбор работников отдела 1, ИЛИ отдела 3, ИЛИ отдела 4. В примере 1.11: отбор работников отдела 1, ИЛИ стажеров, ИЛИ совместителей;

– условия, связанные союзом И, размещаются в разных столбцах. Так, в примере 1.9 – отбор работников, соответствующих условиям: отдел 2, И зарплата не менее 150 ден. ед., И зарплата не более 400 ден. ед. При этом, если несколько условий относятся к одному полю, то они также располагаются в разных столбцах, но с одинаковыми заголовками (в примере 1.9 – два столбца для поля *Заработная плата*);

– заголовки столбцов расширенного фильтра должны совпадать с заголовками полей, кроме случаев, когда условие в расширенном фильтре задается в виде формулы (в этом случае, наоборот, заголовок расширенного фильтра должен отличаться от заголовков полей, см. примеры 1.12, 1.13).

Задания для самостоятельного выполнения:

– получить список всех штатных работников, а также всех работников с заработной платой от 200 до 400 ден. ед.;

– получить список всех штатных работников и совместителей с заработной платой от 200 до 400 ден. ед.;

– получить список всех стажеров, имеющих зарплату выше минимальной по предприятию (для определения минимальной зарплаты использовать функцию МИН).

1.5 Промежуточные итоги

Операция получения промежуточных итогов позволяет вычислять итоговые значения (сумму, количество и т. д.) по некоторым полям для данных, сгруппированных по определенному признаку. Перед операцией вычисления промежуточных итогов обычно необходима операция сортировки для группирования данных по желаемому признаку.

Примечания

1 Если требуется в одной базе данных выполнить несколько операций подсчета промежуточных итогов, то обычно эти операции выполняются последовательно по одной.

2 Если требуется использовать сложные критерии отбора данных для суммирования (например, с учетом нескольких полей, диапазонов значений и т. д.), то следует воспользоваться функциями для работы с базой данных, рассматриваемыми в подразделе 1.7.

Пример 1.14 – Вычислить сумму заработной платы для работников каждого отдела.

1 Так как при наличии пустых ячеек некоторые операции подсчета итогов выполняются неверно, для работников, у которых номер отдела не указан (т. е. внештатных), в поле *Отдел* указать прочерк (знак «тире»).

2 Выполнить сортировку базы данных по номеру отдела.

3 Установить курсор в любую ячейку базы данных. Выбрать **Данные – Итоги.**

4 В появившемся окне в поле **При каждом изменении в** выбрать *Отдел* (так как итог должен вычисляться для каждого отдела). В поле **Операция** выбрать **Сумма**. В поле **Добавить итоги по** установить флажок *Заработная плата* (чтобы суммировалась именно заработная плата). Установить флажок **Итоги под данными**. Нажать **OK.**

Пример 1.15 – В списке с суммами зарплат по отделам, полученном в примере 1.14, подсчитать также количество работников каждого отдела.

1 Установить курсор в любую ячейку базы данных. Выбрать **Данные – Итоги.**

2 В появившемся окне в поле **При каждом изменении в** выбрать *Отдел*. В поле **Операция** выбрать **Количество**. В поле **Добавить итоги по** установить *любой* флажок (так как подсчитывается количество строк, и безразлично, по какому столбцу выполняется подсчет). Сбросить флажок **Заме-**

нить существующие итоги (если не сделать этого, то имеющиеся итоги, т. е. сумма зарплат по отделам, будут удалены). Установить флажок **Итоги под данными**. Нажать **OK.**

3 Убедиться, что требуемые величины подсчитаны. Удалить все итоги, выбрав **Данные – Итоги – Убрать все.**

Пример 1.16 – Подсчитать количество работников каждого отдела, а в каждом отделе – количество работников каждой категории.

1 Выполнить сортировку базы данных по номеру отдела, а в пределах каждого отдела – по категории.

2 Установить курсор в любую ячейку базы данных. Выбрать **Данные – Итоги.**

3 В появившемся окне в поле **При каждом изменении в** выбрать *Отдел*. В поле **Операция** выбрать **Количество**. В поле **Добавить итоги по** установить любой флажок. Установить флажок **Итоги под данными**. Нажать **OK**. Убедиться, что подсчитано количество работников каждого отдела.

4 Снова установить курсор в любую ячейку базы данных и выбрать **Данные – Итоги**. В появившемся окне в поле **При каждом изменении в** выбрать *Категория*. В поле **Операция** выбрать **Количество**. В поле **Добавить итоги по** установить любой флажок. Сбросить флажок **Заменить существующие итоги**. Установить флажок **Итоги под данными**. Нажать **OK.**

5 Убедиться, что подсчитано количество работников каждой категории в каждом отделе. Удалить все итоги, выбрав **Данные – Итоги – Убрать все.**

***Задание для самостоятельного выполнения*:** подсчитать количество и сумму заработной платы работников каждой категории.

Примечание – Прежде чем приступать к рассмотрению операций, описываемых далее в данной работе, можно удалить прочерки в поле *Отдел* для внештатных работников, так как все последующие операции выполняются правильно и при наличии пустых ячеек.

1.6 Консолидация

Операция консолидации предназначена для обработки (обычно – для суммирования) однородных данных, расположенных в различных частях рабочей книги, например, на разных рабочих листах.

Пример 1.17 – Пусть на трех рабочих листах расположены данные о заработной плате работников предприятия (например, на каждом листе – данные о зарплате за некоторый месяц). Требуется вычислить суммарную заработную плату каждого работника.

1 Скопировать данные о работниках (см. рисунок 1.1) еще на два рабочих листа. На втором и третьем рабочих листах изменить заработную плату (изменения могут быть любыми). Кроме того, на одном из рабочих листов удалить данные о каком-либо работнике, а на другом рабочем листе – добавить произвольные данные еще об одном работнике. Присвоить рабочим листам с исход-

ными данными имена *Январь*, *Февраль* и *Март*. Чтобы присвоить имя рабочему листу, дважды щелкнуть по его ярлыку и ввести желаемое имя.

2 Перейти на свободный рабочий лист (если свободных рабочих листов в книге уже нет, создать его командой **Вставка – Лист**). Присвоить листу имя *Итог*. Установить курсор в ячейку A1 этого листа.

3 Выбрать команду **Данные – Консолидация.**

4 В появившемся окне **Консолидация** выполнить следующее:

– в поле **Функция** выбрать **Сумма**;

– указать диапазоны данных для консолидации. Для этого установить курсор в поле **Ссылка** и перейти на рабочий лист *Январь*, щелкнув по его ярлыку. Выделить базу данных (если данные введены согласно рисунку 1.1, то должны быть выделены ячейки A1:D13). Нажать кнопку **Добавить**. Выбранный диапазон указывается в области **Список диапазонов**. Снова перейти в поле **Ссылка** и аналогично выбрать диапазоны на рабочих листах *Февраль* и *Март*;

– в области **Использовать в качестве имен** установить флажки **Подписи верхней строки** и **Значения левого столбца**;

– нажать **OK.**

5 Убедиться, что суммы заработных плат вычислены. Внести в полученные результаты необходимые исправления: удалить суммы в столбце *Отдел* (при консолидации автоматически суммируются все числовые данные, в том числе и номера отделов, что, конечно, не требуется). Скопировать с других рабочих листов категории работников и номера отделов.

1.7 Функции для работы с базами данных

В Excel имеется набор функций для операций с базами данных: **БСЧЁТ** (подсчет строк базы данных, содержащих заполненное числовое поле), **БСЧЁТА** (подсчет строк базы данных, содержащих заполненное поле любого типа), **ДМАКС** (выбор максимального значения поля), **ДМИН** (выбор минимального значения поля), **БДСУММ** (суммирование значений поля), **ДСРЗНАЧ** (подсчет среднего значения по полю) и другие. Эти функции объединены в категорию **Работа с базой данных**. Все эти функции имеют следующий формат:

ФУНКЦИЯ(база данных, поле, критерий).

Здесь *база данных* – диапазон ячеек, содержащих базу данных (включая заголовки). *Поле* – имя поля (или ссылка на ячейку с именем поля), для которого выполняется желаемая операция: подсчет, суммирование, выбор максимального значения и т. д. *Критерий* – диапазон ячеек, содержащий условия отбора записей для выполняемой операции. Эти условия задаются так же, как для расширенного фильтра (см. п. 1.4.2). Если операция выполняется для всей базы данных, то в качестве критерия указывается диапазон ячеек, содержащих всю базу данных.

Рассмотрим примеры, иллюстрирующие использование основных функций для работы с базами данных.

Пример 1.18 – Подсчитать количество работников отделов 1, 2 и 4.

1 В рабочем листе с базой данных указать критерий, как показано на рисунке 1.2.

2 Установить курсор в любую свободную ячейку рабочего листа.

3 Выбрать команду **Вставка – Функция**. Выбрать функцию **БСЧЁТА**.

4 В появившемся окне указать следующие параметры функции **БСЧЁТА**: **База_данных:** A1:D15; **Поле:** A1 (т. е. ячейку с именем поля, для которого выполняется подсчет значений, в данном случае подсчитываются фамилии); **Критерий:** D17:D20. Нажать **OK**.

Примечание – Можно не выбирать функцию **БСЧЁТА** через меню, а просто ввести в свободной ячейке следующую формулу: **= БСЧЁТА(A1:D15;A1;D17:D20)**. Это же относится и к другим рассматриваемым функциям (и вообще ко всем функциям в Excel).

Пример 1.19 – Подсчитать количество следующих работников: работники отдела 1 с зарплатой от 250 ден. ед. и выше, отдела 2 – с зарплатой от 150 до 400 ден. ед., отдела 3 – от 300 ден. ед. и выше.

Указать критерий, как показано на рисунке 1.3. В любой свободной ячейке ввести функцию **БСЧЁТА** со следующими параметрами: **База_данных:** A1:D15; **Поле:** A1; **Критерий:** C18:E21.

Пример 1.20 – Подсчитать количество следующих работников: все работники отделов 1 и 4; стажеры из отдела 2; совместители и стажеры из отдела 3.

Критерий для этой задачи показан на рисунке 1.4.

Пример 1.21 – Подсчитать количество следующих работников: все работники отдела 3, а также все совместители и стажеры (из всех отделов).

Критерий для этой задачи показан на рисунке 1.5.

Пример 1.22 – Подсчитать количество всех работников, кроме внештатных.

Эту задачу можно решить следующим образом: подсчитать количество работников, для которых указан какой-либо номер отдела. Для этого можно с помощью функции **БСЧЁТ** подсчитать количество записей базы данных, содержащих заполненное поле *Отдел*. В данном случае требуется использовать именно функцию **БСЧЁТ** (а не **БСЧЁТА**), так как поле *Отдел* – числовое. Критерий в этом случае указывать не требуется, так как подсчет будет выполняться по всей базе данных. Для решения задачи следует в любой свободной ячейке ввести функцию **БСЧЁТ** со следующими параметрами: **База_данных:** A1:D15; **Поле:** C1; **Критерий:** A1:D15 (т. е. вся база данных).

Примечание – Конечно, эту задачу можно решить и многими другими способами, в том числе с использованием критерия, аналогично предыдущим примерам.

Пример 1.23 – Подсчитать количество работников, имеющих зарплату выше средней по предприятию.

Критерий для этой задачи показан на рисунке 1.6. В данной задаче критерий указывается с использованием формулы. В ячейке B19 введена следующая формула критерия: **=D2>СРЗНАЧ($D$2:$D$15)**. Для решения задачи следует в любой свободной ячейке ввести функцию **БСЧЁТА** со следующими параметрами: **База_данных:** A1:D15; **Поле:** A1; **Критерий:** B18:B19.

Примечание – Следует обратить внимание, что вместо заголовка *Условие* можно было использовать любой заголовок, не совпадающий с заголовками полей базы данных. Подробнее об условиях отбора с использованием формул, в том числе об используемых заголовках, относительных и абсолютных адресах и т. д., см. п. 1.4.2 (расширенный фильтр).

Пример 1.24 – Подсчитать количество работников отдела 2, имеющих зарплату выше средней по предприятию.

Критерий для этой задачи показан на рисунке 1.7. В данной задаче критерий задается с использованием формулы. Для решения задачи следует в любой свободной ячейке ввести функцию **БСЧЁТА** со следующими параметрами: **База_данных:** A1:D15; **Поле:** A1; **Критерий:** A18:B19.

Пример 1.25 – Найти максимальную зарплату по предприятию.

Для решения задачи требуется в любой свободной ячейке ввести функцию **ДМАКС** со следующими параметрами: **База_данных:** A1:D15; **Поле:** D1 (из которого требуется выбрать максимальное значение); **Критерий:** A1:D15 (вся база данных).

Пример 1.26 – Найти сумму зарплаты, выплачиваемой всем работникам, кроме штатных.

Критерий для этой задачи показан на рисунке 1.8. Для решения задачи требуется в любой свободной ячейке ввести функцию **БДСУММ** со следующими параметрами: **База_данных:** A1:D15; **Поле:** D1 (для которого выполняется суммирование); **Критерий:** ячейки, где указан критерий, приведенный на рисунке 1.8.

Пример 1.27 – Найти среднюю зарплату штатных работников отдела 1.

Критерий для этой задачи показан на рисунке 1.9. Для решения задачи требуется в любой свободной ячейке ввести функцию **ДСРЗНАЧ** со следующими параметрами: **База_данных:** A1:D15; **Поле:** D1 (для которого вычисляется среднее); **Критерий:** ячейки, где указан критерий, приведенный на рисунке 1.9.

Категория
<>штатный

Рисунок 1.8 – Критерий для примера 1.26

Категория	Отдел
штатный	1

Рисунок 1.9 – Критерий для примера 1.27

Пример 1.28 – Подсчитать количество работников отдела 2, имеющих зарплату выше средней по своему отделу.

Решение этой задачи показано на рисунке 1.10. Задача решается в следующем порядке.

1 Используя функцию **ДСРЗНАЧ**, найти среднюю зарплату по отделу 2. На рисунке 1.10 она вычисляется в ячейке G4, а критерий для функции **ДСРЗНАЧ** задан в ячейках F1:F2.

2 Задать критерий для отбора работников отдела 2, имеющих зарплату выше средней по своему отделу (т. е. зарплату, превышающую значение ячейки G4). На рисунке 1.10 этот критерий задан в ячейках F6:G7.

3 Для подсчета количества работников отдела 2, имеющих зарплату выше средней по своему отделу, воспользоваться функцией **БСЧЁТА**. На рисунке 1.10 она задана в ячейке G9.

Примечание – Надписи, приведенные в ячейках F4 и F9, необязательны.

	A	B	C	D	E	F	G	H	I
1	**Фамилия**	**Категория**	**Отдел**	**Заработная плата**		**Отдел**			
2	Андреев	штатный	1	240		2			
3	Воробьев	стажер	4	100					
4	Галкин	внештатный		270		**Средняя з/п по отделу 2**	=ДСРЗНАЧ(A1:D15;D1;F1:F2)		
5	Гурин	совместитель	1	270					
6	Иванов	штатный	2	200		**Отдел**	**Условие**		
7	Ковалев	штатный	1	500		2	=D2>$G$4		
8	Котов	штатный	3	430					
9	Петров	штатный	2	450		**Работников в отделе 2 с з/п выше средней по отделу**	=БСЧЁТА(A1:D15;A1;F6:G7)		
10	Семенов	совместитель	3	320					
11	Сидоров	штатный	2	380					
12	Синицын	стажер	2	120					
13	Сорокин	внештатный		320					
14	Степанов	штатный	4	420					
15	Яковлев	стажер	3	280					

Рисунок 1.10 – Решение примера 1.27

Пример 1.29 – Получить список работников отдела 2, имеющих зарплату выше средней по своему отделу.

Для решения этой задачи воспользуемся расширенным фильтром. В качестве условия отбора используем критерий, заданный в примере 1.28. Таким образом, для расширенного фильтра требуется в поле **Исходный диапазон** указать ячейки с базой данных (A1:D15), а в поле **Диапазон условий** – диапазон ячеек, содержащий условия отбора (F6:G7).

Задания для самостоятельного выполнения:

– найти количество работников, получающих зарплату от 200 до 300 ден. ед.;

– найти минимальную зарплату штатного работника;

– получить список штатных работников, получающих минимальную зарплату среди работников своей категории (т. е. зарплату, найденную в предыдущем задании).

1.8 Функции с условиями

Под функциями с условиями будем понимать функции, в которых могут задаваться какие-либо условия, и возвращаемое функцией значение зависит от того, истинны или ложны эти условия. В данном подразделе рассматриваются основные из этих функций. Рассматриваемые функции применимы для обработки не только баз данных (как функции, рассмотренные в подразделе 1.7), но и любых других данных.

1.8.1 Функция СЧЁТЕСЛИ

Функция **СЧЁТЕСЛИ** предназначена для подсчета количества ячеек, соответствующих некоторому критерию. Функция входит в категорию **Статистические**.

Формат функции **СЧЁТЕСЛИ** следующий:

СЧЁТЕСЛИ (диапазон; критерий).

Здесь *диапазон* – диапазон, в котором подсчитывается количество ячеек, соответствующих *критерию*.

Пример 1.30 – Пусть по базе данных, приведенной на рисунке 1.1, требуется подсчитать количество работников, имеющих зарплату не менее 400 ден. ед.

Для этого необходимо в любой свободной ячейке вызвать функцию **СЧЁТЕСЛИ**. В окне параметров функции указать: **Диапазон**: D2:D15; **Критерий**: “>=400”. Нажать **ОК**.

Пример 1.31 – Подсчитать количество стажеров.

Для этого в любой свободной ячейке ввести: **=СЧЁТЕСЛИ(B2:B15;"стажер")**.

Пример 1.32 – Подсчитать количество работников отдела 1.

Для этого в любой свободной ячейке ввести: **=СЧЁТЕСЛИ(C2:C15;1)**.

Примечание – Примеры 1.30 – 1.32 можно решить и с помощью функции **БСЧЁТА**, как показано в подразделе 1.7.

Пример 1.33 – Подсчитать количество работников каждого из отделов.

1 В ячейках A20, A21, A22, A23 (или в других свободных ячейках в одном столбце) ввести номера отделов: 1, 2, 3, 4.

2 В ячейке B20 ввести: **=СЧЁТЕСЛИ($C$2:$C$15;A20)**. Это означает, что требуется подсчитать количество ячеек в диапазоне C2:C15, значение которых равно ячейке A20 (т. е. 1).

3 Распространить содержимое ячейки B20 на ячейки B21:B23.

Примечание – Для функции **СЧЁТЕСЛИ** невозможно задать сложные условия (например, условия для нескольких ячеек или несколько условий для одной ячейки).

1.8.2 Функция СУММЕСЛИ

Функция **СУММЕСЛИ** предназначена для суммирования ячеек, соответствующих некоторому критерию. Функция входит в категорию **Математические**.

Формат функции **СУММЕСЛИ** следующий:

СУММЕСЛИ (диапазон отбора; критерий; диапазон суммирования).

Суммируются ячейки в *диапазоне суммирования*, если соответствующие ячейки в *диапазоне отбора* удовлетворяют *критерию*.

Пример 1.34 – Подсчитать сумму зарплат, выплачиваемых стажерам.

В свободной ячейке ввести: **=СУММЕСЛИ(B2:B15;"стажер";D2:D15)**. Это означает, что требуется вычислить сумму тех ячеек из диапазона D2:D15, для которых в соответствующих ячейках диапазона B2:B15 указано слово «стажер».

Пример 1.35 – Подсчитать сумму зарплат, выплачиваемых работникам, имеющим зарплату свыше 300 ден. ед.

В любой свободной ячейке ввести: **=СУММЕСЛИ(D2:D15;">300";D2:D15)**.

Примечание – Так как в данном случае диапазон отбора и диапазон суммирования совпадают, можно было указать только диапазон отбора: **=СУММЕСЛИ(D2:D15;">300")**. Это означает, что должны суммироваться ячейки из диапазона D2:D15, содержащие значения свыше 300.

Пример 1.36 – Подсчитать суммарную зарплату работников каждого из отделов.

Пусть номера отделов (1, 2, 3, 4) введены в ячейках A20, A21, A22, A23 (см. пример 1.33). Для подсчета суммарной зарплаты каждого отдела требуется выполнить следующее.

1 В ячейке C20 ввести: **=СУММЕСЛИ($C$2:$C$15;A20;$D$2:$D$15).** Это означает, что требуется вычислить сумму ячеек из диапазона D2:D15, для которых соответствующая ячейка диапазона C2:C15 равна ячейке A20 (т. е. содержит номер отдела 1).

2 Распространить содержимое ячейки C20 на ячейки C21:C23.

Примечание – Для функции **СУММЕСЛИ,** как и для **СЧЁТЕСЛИ,** невозможно задать сложные условия (например, условия для нескольких ячеек или несколько условий для одной ячейки)**.**

1.8.3 Функции И и ИЛИ

Эти функции возвращают значение ИСТИНА или ЛОЖЬ в зависимости от заданных условий. Функции входят в категорию **Логические**. Формат функций следующий:

И (условие_1; условие_2; …;условие_n)
ИЛИ (условие_1; условие_2; …;условие_n)

Функция **И** возвращает значение ИСТИНА, если истинны все заданные в ней условия; если хотя бы одно из них ложно, то возвращается значение ЛОЖЬ. Функция **ИЛИ** возвращает значение ИСТИНА, если истинно хотя бы одно из заданных в ней условий; если все эти условия ложны, то возвращается значение ЛОЖЬ.

Пример 1.37 – Пусть, например, в ячейку A1 введено значение 7, в ячейку B1 – значение 5, в ячейку C1 – значение 2. Если ввести в любую свободную ячейку функцию **=И(A1>3;B1<10;C1=2)**, то будет получено значение ИСТИНА, так как все заданные условия верны. Если ввести **=И(A1>3;B1<4;C1=2)**, то будет получено значение ЛОЖЬ, так как одно из условий (B1<4) неверно.

Функция **=ИЛИ(A1>3;B1<4;C1=2)** возвращает значение ИСТИНА, так как среди указанных условий есть верные. Функция **=ИЛИ(A1>10;B1<4;C1>7)** возвращает значение ЛОЖЬ, так как все условия, указанные в ней, ложны.

Функции **И** и **ИЛИ** обычно применяются совместно с другими функциями для указания сложных условий.

1.8.4 Функция ЕСЛИ

Формат функции **ЕСЛИ** следующий:

ЕСЛИ (условие; значение_1; значение_2)

Если *условие* истинно, то функция возвращает *значение_1*, если ложно – *значение_2*. *Условие* может включать функции **И** и **ИЛИ**. *Значение_1* и *значение_2* могут представлять собой как конкретные значения (числа или текст), так и формулы, в том числе с использованием вложенных функций **ЕСЛИ**, а также функций **И** и **ИЛИ**. Допустимая глубина вложенности функций **ЕСЛИ** – до 7.

Функция **ЕСЛИ** входит в категорию **Логические.**

Пример 1.38 – Пусть по базе данных о работниках предприятия (см. рисунок 1.1) требуется вычислить налог с заработной платы каждого работника. Налог составляет 9 % при заработной плате менее 200 ден.ед, 12 % – при заработной плате 200 ден. ед. и более.

Для этого на рабочем листе с базой данных в ячейке E2 следует вызвать функцию **ЕСЛИ**. В появившемся окне указать следующие параметры функции: **Лог_выражение**: D2<200; **Значение_если_истина**: 0,09*D2; **Значение_если_ложь**: 0,12*D2. Нажать **OK**. Распространить введенную формулу на ячейки E3:E15.

Примечание – Можно не вызывать функцию **ЕСЛИ** через меню, а просто ввести в ячейке E2 следующую формулу: **=ЕСЛИ(D2<200;0,09*D2;0,12*D2)**.

Пример 1.39 – Пусть налог с заработной платы вычисляется более сложным образом: 9 % при заработной плате менее 200 ден.ед, 12 % – при заработной плате от 200 до 400 ден. ед., 15 % – при заработной плате 400 ден. ед. и выше.

На рабочем листе с базой данных в ячейке E2 ввести формулу: **=ЕСЛИ(D2<200; 0,09*D2; ЕСЛИ(И(D2>=200; D2<400); 0,12*D2; 0,15*D2))**. Распространить ее на ячейки E3:E15.

Задание для самостоятельного выполнения: используя функции **СЧЁТЕСЛИ** и **СУММЕСЛИ**, подсчитать количество работников каждой категории, а также сумму налогов с заработной платы, выплачиваемых работниками каждой категории. Для вычисления сумм налогов использовать результаты примера 1.39.

1.9 Порядок выполнения работы

Ввести в рабочий лист MS Excel базу данных, приведенную на рисунке 1.1. Выполнить для нее примеры, приведенные в лабораторной работе, включая задания для самостоятельного выполнения.

1.10 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать титульный лист, цель работы и примеры, иллюстрирующие следующие возможности работы с базами данных в MS Excel (по одному примеру на каждую рассматриваемую возможность):

а) сортировка данных;
б) автофильтр (не менее двух условий отбора);
в) расширенный фильтр (не менее двух условий отбора, связка И);
г) расширенный фильтр (не менее двух условий отбора, связка ИЛИ);
д) промежуточные итоги;
е) консолидация данных;
ж) функции для работы с базами данных;
з) функции СУММЕСЛИ и СЧЕТЕСЛИ.

По пунктам б)–д), ж)–з) примеры должны быть приведены для заданий, указанных для самостоятельного выполнения.

Для каждого примера должно быть приведено следующее:

– постановка задачи;

– описание используемых элементов меню, функций и их параметров;
– копии экранов, иллюстрирующие ход решения задачи и ее результаты.

1.11 Контрольные вопросы

1 Понятие базы данных в MS Excel. Обзор операций с базой данных.
2 Сортировка данных.
3 Автофильтры: виды задач, способы задания условий отбора.
4 Расширенные фильтры: виды задач, способы задания условий отбора.
5 Расширенные фильтры: условия отбора со связкой И.
6 Расширенные фильтры: условия отбора со связкой ИЛИ.
7 Расширенные фильтры: условия отбора с использованием функций.
8 Промежуточные итоги: решение задач одним и несколькими уровнями группирования.
9 Консолидация данных.
10 Функции для работы с базами данных: обзор возможностей.
11 Функции для работы с базами данных: суммирование, вычисление средних.
12 Функции СУММЕСЛИ и СЧЕТЕСЛИ: примеры применения для обработки баз данных.
13 Функции И, ИЛИ, ЕСЛИ.

ЛАБОРАТОРНАЯ РАБОТА №2

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MS ACCESS: ТАБЛИЦЫ, ФОРМЫ

Цель работы:

– ознакомление с основными понятиями СУБД данных MS Access;

– освоение основных операций по созданию базы данных в среде СУБД MS Access;

– изучение основных возможностей СУБД MS Access по обеспечению целостности данных;

– изучение возможностей использования форм для заполнения и просмотра баз данных в среде СУБД MS Access.

2.1 Понятие о СУБД Access. Пример постановки задачи

Система управления базами данных (СУБД) Access предназначена для организации хранения данных, их обработки (сортировка, выборка по заданным критериям, группирование, вычисления и т. д.), представления в заданной форме и ряда других задач.

Базы данных, с которыми работает СУБД Access, состоят из одной или нескольких таблиц. Базы данных и работающие с ними СУБД, где данные представлены в виде таблиц, называются реляционными. Таким образом, Access – реляционная СУБД.

Рассмотрим работу с СУБД Access на ряде примеров, связанных с построением и использованием базы данных строительного предприятия. Пусть база данных должна содержать:

– информацию о рабочих предприятия: табельный номер, фамилия, дата рождения, профессия, разряд, дата приема на работу, допуск к работам на высоте, шифр объекта (на котором рабочий работает в данный момент);

– информацию об объектах, над которыми работает предприятие: шифр объекта, заказчик, вид объекта, стоимость контракта, дата заключения контракта, дата окончания строительства.

Здесь потребуется построить две таблицы. Одна из них будет содержать данные о рабочих, другая – об объектах. Столбцы этих таблиц (табельный номер, фамилия, шифр объекта и т. д.) называются полями. Строки таблиц называются записями. Таким образом, в данном примере каждая запись будет содержать информацию об одном рабочем или об одном объекте.

2.2 Начало работы с Access

Загрузить программу Access. Выбрать команду **Файл – Создать – Новая база данных**. В появившемся окне **Файл новой базы данных** в поле **Папка** выбрать папку, в которой будет сохраняться база данных. В поле **Имя файла** указать имя файла базы данных, например, *Стройтрест* (оставлять имя, предлагаемое по умолчанию, не рекомендуется). В поле **Тип файла**

оставить стандартное значение **База данных Access**. Нажать кнопку **Создать**.

2.3 Создание и заполнение базы данных. Таблицы

2.3.1 Пример создания таблицы: создание таблицы данных о рабочих предприятия

Рассмотрим создание таблицы данных о рабочих строительного предприятия. Таблица должна содержать следующую информацию: табельный номер, фамилия, профессия, разряд, дата приема на работу, допуск к работам на высоте, шифр объекта, на котором рабочий работает в данный момент.

Примечание – Даты рождения будут добавлены позже.

1 В окне **База данных** перейти на вкладку **Таблицы**.

2 Нажать кнопку **Создать**. В окне **Новая таблица** выбрать режим создания таблицы – **Конструктор**. Нажать **ОК**. Другой вариант – просто выбрать **Создание таблицы в режиме конструктора**.

3 В появившемся окне **Таблица** ввести описание полей создаваемой базы данных (см. таблицу 2.1). Для свойств полей, не указанных в таблице 2.1, оставить значения по умолчанию.

Таблица 2.1 – Поля базы данных

Имя поля	Тип поля	Свойства поля
Табельный номер	Числовой	Размер поля: Целое Число десятичных знаков: 0 Обязательное поле: Да Индексированное поле: Да (Совпадения не допускаются)
Фамилия	Текстовый	Размер поля: 15 Обязательное поле: Да Индексированное поле: Да (Допускаются совпадения)
Профессия	Текстовый	Размер поля: 15
Разряд	Числовой	Размер поля: Байт Число десятичных знаков: 0 Значение по умолчанию: 1 Условие на значение: >=1 and <=6 Сообщение об ошибке: Разряд может быть от 1 до 6
Дата приема на работу	Дата/время	Формат поля: Краткий формат даты
Допуск на высоту	Логический	Формат поля: Да/Нет Значение по умолчанию: Нет
Шифр объекта	Текстовый	Размер поля: 5

4 Установить поле *Табельный номер* в качестве первичного ключа. Для этого выделить это поле и щелкнуть по пиктограмме с ключом или выбрать команду **Правка – Ключевое поле**.

5 Сохранить созданную структуру таблицы (**Файл – Сохранить**). При этом запрашивается имя таблицы. Ввести для нее имя *Рабочие*. Имя, предлагаемое по умолчанию (*Таблица1*), оставлять не рекомендуется.

6 Закрыть окно с описанием таблицы *Рабочие*.

2.3.2 Изменение структуры таблицы

Под изменением структуры таблицы понимается добавление или удаление полей, а также изменение их свойств.

1 Открыть таблицу *Рабочие* в режиме конструктора. Для этого на вкладке **Таблицы** выбрать таблицу *Рабочие* и нажать кнопку **Конструктор** или выбрать команду **Вид – Режим конструктора**.

2 Вставить между полями *Фамилия* и *Профессия* поле *Дата рождения*. Для этого выделить поле, перед которым вставляется новое поле (в данном случае – выделить поле *Профессия*) и выбрать команду **Вставка – Строки**, или щелкнуть по пиктограмме **Добавить строки**. Для созданного поля выбрать **Тип данных – Дата/время**. В области **Свойства поля** установить **Формат поля – Краткий формат даты**.

3 Для созданного поля задать маску ввода. Для этого:

– в строке **Маска ввода** нажать кнопку … (вызов **Мастера масок ввода**);

– выбрать маску ввода **Краткий формат даты**. Нажать **Далее**;

– убедиться, что указана маска ввода 00/00/0000. Удалить из маски ввода два последних нуля (т. е. установить маску ввода 00/00/00), чтобы иметь возможность вводить только две последние цифры года. Убедиться, что в качестве заполнителя указан знак подчеркивания. Нажать **Далее**;

– нажать **Готово**.

4 Для поля *Дата приема на работу* также установить свойство поля **Маска ввода** (как и для поля *Дата рождения*).

5 Для поля *Профессия* предусмотреть возможность выбора из списка значений. Для этого:

– выбрать для поля *Профессия* тип данных – **Мастер подстановок**;

– в появившемся окне **Создание подстановки** установить переключатель **Будет введен фиксированный набор значений**. Нажать **Далее**;

– в очередном окне **Создание подстановки** установить **Число столбцов – 1** (оно предлагается по умолчанию). Ввести названия профессий: каменщик, маляр, сантехник, штукатур (каждое название вводится в отдельной строке). Нажать **Далее**;

– в очередном окне **Создание подстановки** установить подпись столбца подстановки – *Профессия* (эта подпись предлагается по умолчанию). Нажать **Готово**;

– в области **Свойства поля** перейти на вкладку **Подстановка**. Убедиться, что для свойства **Ограничиться списком** установлено значение **Нет** (чтобы иметь возможность указывать профессии, не содержащиеся в списке).

6 Сохранить внесенные изменения. Закрыть окно описания структуры таблицы *Рабочие*.

2.3.3 Заполнение таблицы

Чтобы ввести данные в таблицу *Рабочие*, необходимо на вкладке **Таблицы** выбрать эту таблицу, затем нажать кнопку **Открыть**, или дважды щелкнуть мышью.

Ввести в таблицу данные о рабочих (таблица 2.2). При вводе профессии использовать выбор из списка (если необходимая профессия есть в списке). При вводе разряда убедиться в невозможности ввода какого-либо числа, выходящего за диапазон от 1 до 6.

Таблица 2.2 – Данные для ввода в таблицу *Рабочие*

Табельный номер	Фамилия	Дата рождения	Профессия	Разряд	Дата приема на работу	Допуск на высоту	Шифр объекта
101	Андреев	11.05.1970	каменщик	3	12.02.1998	да	Д50
105	Семенов	05.12.1971	штукатур	5	15.04.1996	да	П100
106	Борисов	17.07.1967	штукатур	6	10.07.1995	да	Д50
110	Васильев	03.09.1975	маляр	2	17.05.2001	нет	А70
112	Гурин	10.12.1980	разнорабочий		01.06.2002	нет	П80
115	Петров	08.11.1975	каменщик	3	12.05.1995	да	П100
120	Иванов	20.08.1982	маляр	1	05.08.2002	нет	П80
121	Демин	12.07.1980	штукатур	3	10.08.2002	да	П100
122	Мишин	10.02.1980	каменщик	5	14.08.2001	нет	П100

2.3.4 Режимы работы с таблицей

Как видно из рассмотренных примеров, в СУБД Access имеются два основных режима работы с таблицами:

– режим конструктора. В этом режиме создается или изменяется структура таблицы: поля, их имена, типы, свойства;

– режим таблицы. В этом режиме выполняется ввод или изменение данных в таблице.

Если, например, в таблицу *Рабочие* потребуется добавить поле *Домашний адрес* или удалить поле *Допуск на высоту*, или изменить формат поля *Дата рождения*, то необходимо воспользоваться режимом конструктора. Если же требуется добавить в таблицу информацию о новом рабочем или удалить информацию об уволенном рабочем, или изменить разряд какого-либо рабочего, то используется режим таблицы.

Примечание – Как будет показано ниже, в СУБД Access режимы конструктора и таблицы применяются и к другим объектам базы данных (формы, запросы, отчеты)**.** Для всех этих объектов режим конструктора предназначен для изменения структуры соответствующего объекта, а режим таблицы – для работы с данными.

2.3.5 Создание таблицы данных об объектах

Эта таблица должна содержать следующую информацию: шифр объекта, заказчик, вид объекта, стоимость контракта, дата заключения контракта,

дата окончания строительства. При вводе структуры этой таблицы необходимо учесть следующее:

– рекомендуется использовать для полей имена *Шифр объекта*, *Заказчик*, *Вид объекта*, *Стоимость контракта*, *Дата заключения* и *Дата окончания*. Таблице присвоить имя *Объекты*;

– для поля *Шифр объекта* тип данных и размер поля должны быть такими же, как и для поля *Шифр объекта* в таблице *Рабочие* (так как в этих полях будут содержаться одни и те же данные – шифры объектов);

– поле *Шифр объекта* должно быть задано как ключевое;

– для полей *Заказчик* и *Вид объекта* использовать **текстовый** тип данных;

– для поля *Стоимость контракта* использовать **Тип данных – Денежный, Формат поля – Денежный, Число десятичных знаков – 0**;

– для полей *Дата заключения* и *Дата окончания* использовать **Тип данных – Дата/время, Формат поля – Краткий формат даты**. Предусмотреть маску ввода, как показано в пункте 2.3.2.

Ввести в таблицу *Объекты* данные из таблицы 2.3.

Примечание – Шифры объектов в таблице *Объекты* должны вводиться точно так же, как они введены в таблице *Рабочие*.

Таблица 2.3 – Данные для ввода в таблицу *Объекты*

Шифр объекта	Заказчик	Вид объекта	Стоимость контракта	Дата заключения	Дата окончания
А70	АО Олимп	офис	80 000 000	12.10.2009	12.08.2010
Д50	Автозавод	жилой дом	120 000 000	10.01.2009	30.12.2010
П80	Хлебозавод №2	склад	40 000 000	20.05.2009	01.03.2010
П100	Автозавод	цех	180 000 000	10.04.2009	30.11.2010

2.4 Связывание таблиц

В рассматриваемом примере в таблицах *Рабочие* и *Объекты* имеется поле *Шифр объекта*. Очевидно, что эти поля в обеих таблицах должны содержать одинаковые значения – шифры объектов. Говорят, что таблицы *Рабочие* и *Объекты* «**связаны** по полю *Шифр объекта*».

По смыслу задачи на одном объекте работает много рабочих, но каждый рабочий – только на одном объекте. Другими словами, шифр объекта, указанный в таблице *Объекты* один раз, может быть указан в таблице *Рабочие* для многих рабочих. Такая связь называется связью «один ко многим» и обозначается как 1:М.

Очевидно, что в таблице *Рабочие* можно указывать только те шифры объектов, которые имеются в таблице *Объекты* (несоблюдение этого требования означало бы, что рабочий может быть назначен на несуществующий объект). Желательно, чтобы в случае изменения шифра объекта (в таблице *Объекты*) он автоматически изменялся у всех рабочих, назначенных на этот

объект (т. е. чтобы автоматически изменялась таблица *Рабочие*). Выполнение этих требований называется обеспечением **целостности данных**.

Желательно также иметь возможность по данным из одной таблицы получать соответствующие им данные из другой таблицы. Например, желательно, чтобы для определенного рабочего можно было найти дату окончания строительства объекта, на котором он работает, или для определенного заказчика – просмотреть перечень рабочих, занятых на его объектах.

Для обеспечения целостности данных, а также для обеспечения возможности получения информации из двух (и более) таблиц применяется связывание таблиц.

Примечания

1 В данном примере поля, по которым устанавливается связь (*Шифр объекта*), имеют одинаковые имена в обеих таблицах. Это необязательно: связываемые поля могут называться и по-разному. Однако тип данных у связываемых полей должен быть одинаковым.

2 Более подробно (и более строго) понятия связи между таблицами, целостности данных и т. д. рассматриваются в курсе «Базы и банки данных».

Рассмотрим связывание таблиц *Рабочие* и *Объекты* по полю *Шифр объекта*. Связь устанавливается следующим образом.

1 Выбрать команду **Сервис – Схема данных**.

2 На экран выводится окно **Добавление таблицы**. Если его нет, следует выбрать команду **Связи – Добавить таблицу** или нажать правую кнопку мыши и выбрать команду **Добавить таблицу**. В этом окне выбрать таблицы *Рабочие* и *Объекты* (для этого нажать клавишу **Ctrl** и, не отпуская ее, с помощью мыши отметить необходимые таблицы). Нажать кнопку **Добавить**. Закрыть окно **Добавление таблицы**.

3 Чтобы установить связь между таблицами, щелкнуть мышью по полю *Шифр объекта* в таблице *Объекты* и, не отпуская кнопку мыши, поместить указатель на поле *Шифр объекта* в таблице *Рабочие*. После этого отпустить кнопку мыши. На экран выводится окно параметров связи (окно **Связи**). Выполнить следующее:

– установить флажок **Обеспечение целостности данных**, чтобы в таблицу *Рабочие* можно было вводить только шифры объектов, уже имеющиеся в таблице *Объекты*;

– установить флажок **Каскадное обновление связанных полей**, чтобы при изменении шифра объекта в таблице *Объекты* он автоматически изменялся и в таблице *Рабочие*;

– флажок **Каскадное удаление связанных полей** не устанавливать: при удалении данных об объекте не должны удаляться данные о рабочих, занятых на этом объекте;

– убедиться, что в поле **Тип отношения** указано **Один ко многим**;

– нажать кнопку **Создать**.

Примечание – Если при выполнении данной операции выводится сообщение об ошибке, это обычно означает, что в таблицах введены недопустимые данные (например,

для какого-либо рабочего указан шифр несуществующего объекта). В этом случае необходимо закрыть окно **Схема данных** и внести исправления в таблицы.

Связь между таблицами должна иметь примерно такой вид, как показано на рисунке 2.1.

2.5 Простейшие операции с данными в таблицах

Для всех рассматриваемых ниже операций требуется открыть таблицу в режиме таблицы, т. е. выделить ее и нажать кнопку **Открыть** (или просто дважды щелкнуть мышью по обозначению таблицы).

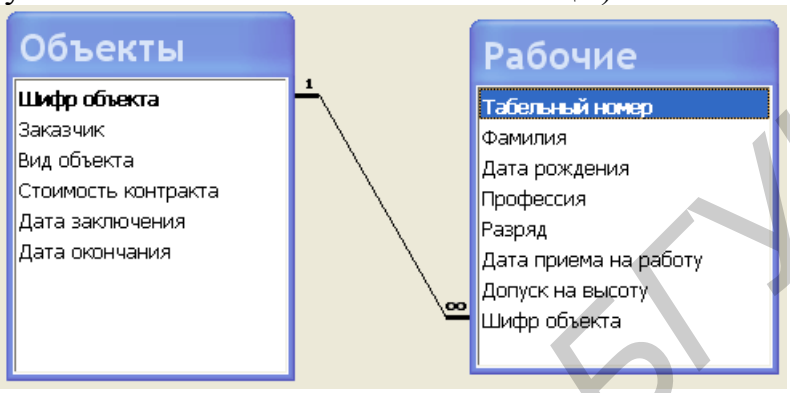

Рисунок 2.1 – Связывание таблиц

Просмотр, изменение, добавление и удаление данных

В таблице можно просматривать и редактировать имеющиеся данные. В конце таблицы имеется свободная строка для добавления новых данных.

Если таблица связана с другой таблицей, то рядом с каждой записью имеется отметка (в виде креста). Щелкнув по такой отметке, можно просмотреть записи в другой таблице, связанные с данной записью. Например, если щелкнуть по такой отметке в таблице *Объекты*, то на экран выводятся данные о рабочих, занятых на этом объекте.

Для удаления записей (одной или нескольких) следует выделить их с помощью мыши и нажать клавишу **Del** или выбрать команду **Правка – Удалить.**

Поиск данных

Пусть требуется найти в таблице *Объекты* строку с информацией об объекте П100 (это может понадобиться, если таблица большая и не помещается на экране). Для этого следует выбрать команду **Правка – Найти**. В поле **Образец** указать образец для поиска (П100). В поле **Поиск в** можно выбрать имя поля, в котором в данный момент находится курсор, или имя таблицы (в первом случае поиск будет выполняться только в текущем поле, во втором случае – во всей таблице). Имеются и другие возможности настройки параметров поиска. Для выполнения поиска нажать кнопку **Найти далее.**

Сортировка данных

Для сортировки таблицы по некоторому полю следует установить курсор в это поле и воспользоваться командой **Записи – Сортировка** или кноп-

ками с отметками **А/Я** (сортировка по возрастанию или по алфавиту), или **Я/А** (сортировка по убыванию, или в порядке, обратном алфавитному).

Выбор столбцов для просмотра

Пусть требуется просмотреть таблицу *Объекты*, но при этом не требуется, чтобы на экран выводилось поле *Шифр объекта*. Для этого следует выбрать команду **Формат – Отобразить столбцы**, снять флажок столбца *Шифр объекта* и нажать кнопку **Закрыть.**

Чтобы восстановить отображение столбца, потребуется снова выбрать команду **Формат – Отобразить столбцы** и восстановить флажок столбца.

Другой способ отмены отображения столбца – поместить курсор в этот столбец (или выделить несколько столбцов, если требуется) и выбрать команду **Формат – Скрыть столбцы**. Для восстановления отображение столбцов потребуется использовать команду **Формат – Отобразить столбцы.**

Примечание – В режиме таблицы имеется возможность изменять структуру таблицы: добавлять новые поля (команда **Вставка – Столбец**) или удалять их (команда **Правка – Удалить столбец**). Однако делать это **не рекомендуется**. Для изменения структуры таблицы следует использовать режим конструктора, как показано в п. 2.3.2.

2.6 Формы

2.6.1 Назначение, способы создания и виды форм

Формы предназначены для отображения и/или ввода данных в удобном виде.

Имеются **три способа создания форм**:

– автоматический (автоформа) – форма строится полностью автоматически и включает все поля выбранной таблицы;

– в режиме мастера – в процессе построения формы пользователю предлагаются подсказки и возможности выбора;

– в режиме конструктора – форма строится пользователем самостоятельно.

Имеются следующие **основные виды форм**:

– в столбец – все поля базы данных располагаются в столбец друг под другом (каждое поле – в отдельной строке);

– табличная – имеет такой же вид, как и таблица базы данных;

– ленточная – по виду близка к табличной;

– диаграмма – содержит график.

Кроме того, форма может быть **простой** (состоит из одного окна и обычно содержит данные из одной таблицы) или **составной** (обычно состоит из двух окон и используется для просмотра данных в двух связанных таблицах).

Простые формы обычно создаются автоматически, а составные – в режиме мастера. Затем в них при необходимости вносятся изменения в режиме конструктора.

2.6.2 Создание простой формы

Пусть требуется создать форму для просмотра и ввода данных в таблицу объектов. Форма строится следующим образом.

1 Перейти на вкладку **Формы**.

2 Нажать кнопку **Создать**.

3 Выбрать команду **Автоформа: в столбец**. В поле выбора источника данных выбрать таблицу *Объекты*. Нажать **OK.**

На экран выводится окно формы, аналогичное показанному на рисунке 2.2. С помощью кнопок, расположенных в нижней части окна, можно переходить от одной записи к другой. Можно перейти к пустой (свободной) записи для добавления новой записи, а также изменять имеющиеся данные. Если требуется удалить текущую запись из базы данных, то необходимо выделить ее (щелчком мыши по отметке в левом верхнем углу окна) и нажать клавишу **Del**.

При закрытии окна созданной формы предлагается сохранить ее. На этот вопрос следует выбрать ответ **Да**. После этого запрашивается имя формы. По умолчанию предлагается имя таблицы, с которой связана форма (в рассматриваемом примере – *Объекты*). В данном случае можно использовать предлагаемое имя. Для сохранения формы нажать **OK.**

Примечание – Не рекомендуется сохранять формы (как и другие объекты базы данных) под именами, не имеющими содержательного смысла (например *Форма 1*). Если при сохранении предлагается такое имя, то его следует изменить.

Для работы с формой следует перейти на вкладку **Формы**, выбрать желаемую форму и нажать кнопку **Открыть** (или просто дважды щелкнуть мышью).

2.6.3 Изменение вида формы

Для изменения вида формы необходимо открыть ее в режиме конструктора. Для этого следует выбрать форму и нажать кнопку **Конструктор** или выбрать команду **Вид – Конструктор**. После этого можно добавлять в форму новые элементы (поля, надписи, кнопки и т. д.), изменять или удалять имеющиеся.

Чтобы изменить или удалить элемент формы, необходимо сначала выделить его щелчком мыши. Если требуется выделить сразу несколько элементов формы, то следует нажать клавишу **Shift** и, не отпуская ее, выделять желаемые элементы с помощью мыши. Для перемещения выбранного элемента курсор должен иметь форму ладони, для изменения размеров – форму двунаправленной стрелки. Для удаления выбранного элемента требуется нажать клавишу **Del**.

Кроме того, каждый элемент формы имеет набор свойств. Для их изменения требуется щелкнуть по элементу формы правой кнопкой мыши и выбрать команду **Свойства.**

Пусть требуется внести в форму *Объекты*, построенную в пункте 2.6.2, следующие изменения: указать в верхней части формы надпись *Заказы на <текущая дата>*; указать количество дней, оставшихся до окончания строительства (разность даты окончания и текущей даты).

Оформление заголовка формы

1 Открыть форму *Объекты* в режиме конструктора (выбрать форму и нажать кнопку **Конструктор**).

2 С помощью мыши расширить область заголовка (между надписями **Заголовок формы и Область данных**).

Примечание – Если в окне формы отсутствует область заголовка (нет надписи **Заголовок формы**), то следует выбрать команду **Вид – Заголовок/примечание формы**.

3 Из панели элементов выбрать элемент **Надпись** (с отметкой **Аа**). Переместить курсор (принявший вид креста с буквой **А**) в область заголовка и щелкнуть мышью в нужном месте.

Примечание – Если на экране нет панели элементов, то следует выполнить команду **Вид – Панель элементов**.

4 В прямоугольнике, появившемся в области заголовка, ввести текст: *Заказы на*.

5 Чтобы вставить в заголовок текущую дату, выбрать команду: **Вставка – Дата и время**. Снять флажок **Формат времени**. Выбрать желаемый формат даты. Нажать **OK**.

6 Разместить отметку даты в заголовке формы. Для этого выделить отметку даты (щелчком мыши), разместить курсор так, чтобы он принял форму ладони, и переместить отметку даты в желаемое место.

7 Установить для заголовка формы (слова *Заказы на* и отметка текущей даты) жирный шрифт размером 10. Для этого выделить оба элемента (щелчками мыши при нажатой клавише **Shift**) и нажать соответствующие кнопки на панели инструментов.

8 Сохранить внесенные изменения. Закрыть форму. Открыть ее в режиме формы (кнопкой **Открыть**). Убедиться, что изменения внесены. При необходимости перейти в режим конструктора и внести необходимые изменения.

Вставка вычисляемого поля

Для отображения количества дней, оставшихся до конца строительства, следует добавить в форму вычисляемое поле. Пусть оно будет размещено в нижней части области данных (после других полей).

1 Открыть форму *Объекты* в режиме конструктора.

2 Если в нижней части области данных нет свободного места, расширить область данных, сдвинув вниз ее нижнюю границу (над надписью **Примечание формы**) с помощью мыши.

3 Из панели элементов выбрать элемент **Поле** (с отметкой ***аб***). Переместить курсор в нижнюю часть области данных и щелкнуть мышью в нуж-

ном месте. На экране появляется новое поле (с отметкой **Свободный**) и надпись к нему (слово **Поле** и некоторый номер).

4 Вызвать окно свойств надписи нового поля. Для этого щелкнуть правой кнопкой мыши на надписи (со словом **Поле**) и выбрать команду **Свойства**. В появившемся окне свойств надписи перейти на вкладку **Макет** (или вкладку **Все**). Для свойства **Подпись** ввести текст *Осталось дней:*. Закрыть окно свойств.

5 Вызвать окно свойств нового поля (самого поля, а не надписи к нему). В строке свойства **Данные** (на вкладке **Данные**) ввести формулу: *=[Дата окончания] – date()*. Закрыть окно свойств.

Примечание – Здесь *Дата окончания* – имя поля таблицы. Оно должно быть заключено в квадратные скобки и абсолютно точно совпадать с именем поля, указанным в описании таблицы. *Date()* – стандартная функция, возвращающая текущую дату. Формула должна начинаться со знака «равно». Никаких знаков в конце формулы не ставится.

6 Переместить новое поле и надпись к нему, а также изменить их размеры таким образом, чтобы они были удобно размещены в окне (в один столбец с другими полями).

7 Закрыть форму. Просмотреть ее. При необходимости вернуться в режим конструктора для внесения изменений.

В результате форма должна иметь примерно такой вид, как показано на рисунке 2.3.

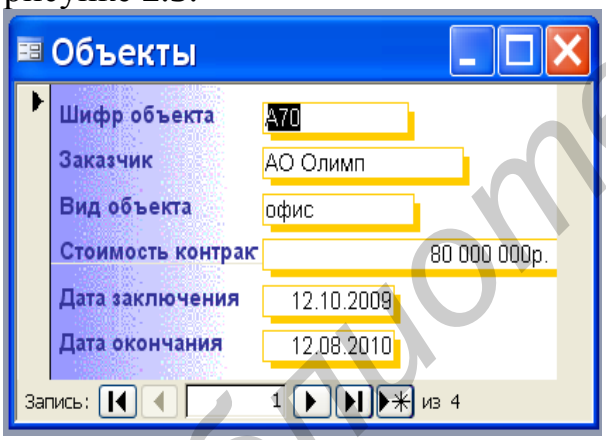

Рисунок 2.2 – Форма, созданная автоматически

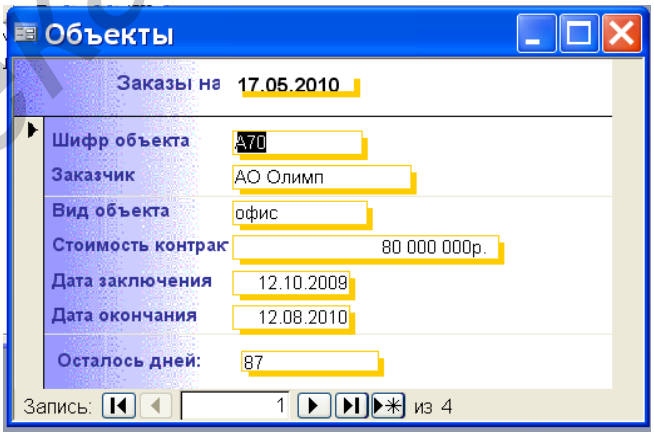

Рисунок 2.3 – Форма с изменениями

2.6.4 Создание составной формы

Пусть требуется создать форму для просмотра данных об объектах, включая данные о рабочих, занятых на этих объектах.

1 Перейти на вкладку **Формы**. Нажать кнопку **Создать.**

2 Выбрать команду **Мастер форм**. В поле выбора источника данных ничего не выбирать. Нажать **OK**.

3 В поле **Таблицы и запросы** выбрать таблицу *Объекты*. Из списка **Доступные поля** перенести все поля в список **Выбранные поля** (это можно сделать одним нажатием кнопки >>). Кнопку **Далее** не нажимать!

4 В поле **Таблицы и запросы** выбрать таблицу *Рабочие*. Из списка **Доступные поля** перенести в список **Выбранные поля** имена полей *Табельный номер*, *Фамилия* и *Профессия*. Для этого используется кнопка >. В результате в списке **Выбранные поля** должны находиться поля из обеих таблиц. Нажать **Далее**.

Примечание – Если в списке **Выбранные поля** оказывается несколько полей с одинаковым именем (из разных таблиц), то перед именами таких полей автоматически указываются имена таблиц.

5 В очередном окне **Создание форм** в списке **Выберите вид представления данных** выбрать таблицу, на основе которой будет строиться главная форма. В данном случае это таблица *Объекты*. Выбрать переключатель **Подчиненные формы**. Нажать **Далее**.

6 Выбрать вид подчиненной формы – **Ленточный** или **Табличный** (по своему выбору). Нажать **Далее**.

7 Выбрать стиль, т. е. внешний вид формы (по своему выбору). Нажать **Далее**.

8 Задать имена форм: **Форма** – *Строящиеся объекты*, **Подчиненная форма** – *Рабочие на объекте*. Установить переключатель **Открыть форму для просмотра и ввода данных**. Нажать **Готово**.

На экран выводится построенная составная форма (см. рисунок 2.4). Она состоит из двух форм: основной (*Строящиеся объекты*), где отображается информация о выбранном объекте, и подчиненной (*Рабочие на объекте*), содержащей информацию о рабочих, занятых на выбранном объекте.

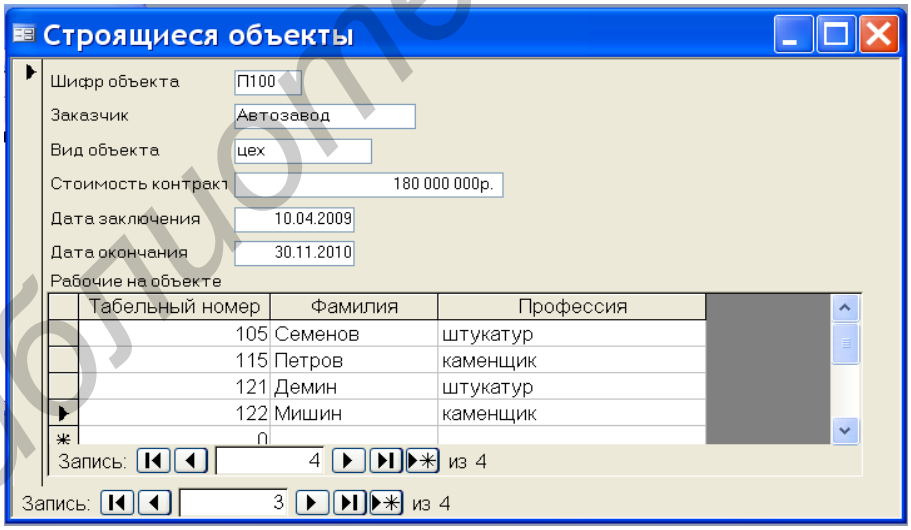

Рисунок 2.4 – Составная форма

Примечание – Существуют два типа составных форм – подчиненные и связанные. В данном примере построена подчиненная форма (тип формы выбран на шаге 5). Если выбирается связанная форма, то в ее окне отображается только основная форма, а также кнопка вызова подчиненной формы. Рекомендуется самостоятельно построить еще одну составную форму для данного примера, выбрав для нее тип «связанная».

2.6.5 Изменение свойств формы

Пусть для составной формы, построенной в пункте 2.6.4, требуется запретить внесение каких-либо изменений в данные о рабочих.

1 Перейти на вкладку **Формы**. Выбрать форму *Рабочие на объекте*. Открыть ее в режиме конструктора.

2 Щелкнуть правой кнопкой мыши по кнопке, расположенной в левом верхнем углу на пересечении верхней и левой линеек, и выбрать команду **Свойства**. Другой способ – просто щелкнуть дважды по этой кнопке. На экран выводится окно **Форма** со свойствами формы.

3 На вкладке **Данные** установить следующие свойства формы: **Разрешить изменение – Нет, Разрешить удаление – Нет, Разрешить добавление - Нет**. Закрыть окно свойств формы.

4 Сохранить внесенные изменения. Закрыть форму.

5 Открыть главную форму (*Строящиеся объекты*) в режиме формы. Убедиться в невозможности изменения данных о рабочих.

2.7 Порядок выполнения работы

1 Построить базу данных строительной организации согласно подразделу 2.1–2.3.

2 Установить связь между таблицами согласно подразделу 2.4.

3 Выполнить операции изменения, поиска, отбора и сортировки данных согласно подразделу 2.5.

4 Выполнить операции с формами согласно подразделу 2.6.

2.8 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать титульный лист, цель работы и примеры, иллюстрирующие следующие возможности работы с СУБД MS Access (по одному на каждую рассматриваемую возможность):

а) создание таблицы;
б) изменение структуры таблицы;
в) связывание таблиц;
г) построение простой формы;
д) построение составной формы;
е) создание вычисляемого поля на форме.

Для каждого примера должно быть приведено следующее:

- постановка задачи;
- описание выполняемых операций и используемых элементов меню;
- копии экранов, иллюстрирующие ход решения задачи и ее результаты.

2.9 Контрольные вопросы

1 Понятие базы данных в СУБД MS Access. Понятие поля и записи.

2 Свойства полей.

3 Работа с таблицами в режиме конструктора и режиме таблицы.

4 Изменение структуры таблицы.

5 Связывание таблиц: порядок выполнения, назначение.

6 Примеры ошибок при вводе данных, защита от которых обеспечивается связыванием таблиц.

7 Примеры каскадного обновления и каскадного удаления данных.

8 Возможности одновременного просмотра данных из нескольких таблиц.

9 Назначение и виды форм.

10 Составные формы: виды, назначение, возможности.

11 Изменение вида форм. Вычисляемые поля.

12 Изменение свойств форм.

ЛАБОРАТОРНАЯ РАБОТА №3

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MS ACCESS: ЗАПРОСЫ, ОТЧЕТЫ

Цель работы:

– изучение основных возможностей выборки и обработки данных с использованием запросов в среде MS Access;

– изучение основных возможностей представления данных в заданной форме с использованием отчетов в среде MS Access.

3.1 Запросы

3.1.1 Назначение, способы создания и виды запросов

Запросы предназначены для выборки информации из базы данных или для внесения изменений в базу данных.

В Access имеются следующие основные виды запросов:

а) по способу описания:

1) QBE-запросы (Query By Example – выборка по образцу) – вид запроса устанавливается в специальном окне (окне конструктора запросов);

2) SQL-запросы (Structured Query Language – структурированный язык запросов) – запрос описывается с помощью команд языка SQL.

Примечание – При построении любого QBE-запроса для него автоматически строится описание на языке SQL, и наоборот (за исключением некоторых SQL-запросов, которые не могут быть построены как QBE-запросы). Переход от описания в виде SQL-запроса к QBE-запросу (и наоборот) выполняется с помощью команд меню **Вид**. В данной работе SQL-запросы не рассматриваются;

б) по назначению:

1) запросы на выборку – для извлечения информации из базы данных;

2) запросы на изменение – для внесения изменений в базу данных (включая добавление, удаление, изменение записей, создание новых таблиц);

в) по содержанию:

1) обычные (подробные) – содержащие информацию из отдельных записей, извлеченных из одной или нескольких таблиц;

2) с групповыми операциями (итоговые) – запросы, в которых выполняется разбиение данных на группы по значению какого-либо поля и подсчет итогов (количества, суммы, среднего и т. д.) по этим группам;

3) перекрестные – аналогичны запросам с групповыми операциями, но разбиение на группы выполняется по значениям двух полей;

г) по виду описания условий обработки данных:

1) фиксированные – запросы, в которых условия обработки данных (т. е. их выборки или изменения) полностью заданы;

2) параметрические – запросы, в которых условия обработки данных указываются пользователем при выполнении запроса.

Имеются два способа создания запросов:

– в режиме мастера – в процессе построения запроса пользователю предлагаются подсказки и возможности выбора;

– в режиме конструктора – запрос строится пользователем самостоятельно.

Источником данных для запроса может быть как таблица, так и другой запрос, созданный ранее.

3.1.2 Создание запросов в режиме мастера

Пример 3.1 (запрос для получения данных из одной таблицы) – Пусть требуется создать запрос для вывода списка всех рабочих. В списке должна содержаться вся информация о рабочих, кроме шифра объекта, на котором он работает.

1 Перейти на вкладку **Запросы**. Нажать кнопку **Создать.**

2 Выбрать команду **Простой запрос**. Нажать **OK.**

3 В поле **Таблицы и запросы** выбрать таблицу *Рабочие*. Из списка **Доступные поля** перенести в список **Выбранные поля** обозначения всех полей, кроме поля *Шифр объекта*. Нажать **Далее.**

4 Выбрать вид отчета – **Подробный**. Нажать **Далее.**

5 Задать имя запроса *Список рабочих*. Установить переключатель **Открыть запрос для просмотра данных**. Нажать **Готово.**

6 На экран выводится запрос с данными из таблицы *Рабочие*. Просмотреть и закрыть его.

7 Чтобы снова выполнить запрос, требуется выделить его и нажать **Открыть** или просто дважды щелкнуть по отметке запроса.

Пример 3.2 (запрос для получения данных из нескольких таблиц) – Пусть требуется создать запрос для вывода списка рабочих. Для каждого рабочего указываются его табельный номер, фамилия, разряд, шифр объекта, а также вид объекта, на котором он работает.

1 Нажать кнопку **Создать**. Выбрать команду **Простой запрос**. Нажать **OK**.

2 В поле **Таблицы и запросы** выбрать таблицу *Рабочие*. Из списка **Доступные поля** перенести в список **Выбранные поля** обозначения необходимых полей. Кнопку **Далее** не нажимать!

3 В поле **Таблицы и запросы** выбрать таблицу *Объекты*. Из списка **Доступные поля** перенести в список **Выбранные поля** обозначение поля *Вид объекта*. Нажать **Далее.**

4 Дальнейшие действия выполняются точно так же, как показано выше (пример 3.1). Присвоить созданному запросу имя *Список рабочих_2*.

3.1.3 Изменение запросов в режиме конструктора

Пример 3.3 – Пусть требуется создать запрос для вывода списка объектов. При этом предусмотреть следующее: а) запрос должен быть упорядочен по названию заказчика, а для каждого заказчика – по дате окончания строительства (первыми должны указываться объекты, строительство которых необходимо закончить раньше); б) вид объекта должен указываться после его шифра.

1 Создать запрос на основе таблицы *Объекты* аналогично показанному выше. Присвоить ему имя *Список объектов*.

2 Открыть запрос *Список объектов* в режиме конструктора. Для этого выбрать запрос и нажать кнопку **Конструктор**.

3 В появившемся окне шаблона запроса с помощью мыши переместить поле *Вид объекта*, чтобы оно располагалось после поля *Шифр объекта*. Для этого сначала выделить поле *Вид объекта*: поместить указатель мыши *над* именем поля (указатель должен принять вид утолщенной черной стрелки) и нажать левую кнопку мыши. Затем снова поместить указатель мыши над именем поля (указатель должен принять вид белой стрелки) и переместить поле в желаемое место.

4 В строке **Сортировка** для полей *Заказчик* и *Дата окончания* установить значение **По возрастанию**.

5 Сохранить измененный запрос.

6 Для просмотра запроса нажать кнопку с восклицательным знаком, или выбрать команду **Вид – Режим таблицы**, или закрыть запрос и нажать кнопку **Открыть**.

Пример 3.4 (создание вычисляемого поля) – Внести изменения в созданный запрос *Список объектов*: добавить в него поле с именем *Налог*, в котором должна указываться величина, равная 10 % от стоимости.

1 Открыть запрос в режиме конструктора.

2 Чтобы создать вычисляемое поле *Налог*, поместить курсор в свободное поле в конце запроса. Ввести в этом поле следующее выражение: *[Стоимость контракта]*0,1*. После ввода этого выражения перед ним автоматически указывается имя созданного поля – *Выражение1* (это имя назначается по умолчанию). Вместо него ввести имя *Налог*.

Примечания

1 Здесь *Стоимость контракта* – имя поля, используемого в выражении. Оно должно быть заключено в квадратные скобки. Никаких знаков в конце выражения не ставится.

2 Для записи выражения можно использовать построитель (мастер) выражений. Для этого следует нажать правую кнопку мыши и выбрать команду **Построить.**

3 В столбце с введенным выражением нажать правую кнопку мыши и выбрать команду **Свойства** или выбрать из меню команду **Вид – Свойства.** Открывается окно свойств поля. В этом окне для свойства **Формат поля** вы-

брать значение **Денежный** или **С разделителями разрядов**. Закрыть окно свойств поля.

4 Сохранить внесенные изменения. Просмотреть полученный запрос.

Пример 3.5 (вычисляемое поле со сложной формулой) – Пусть в условиях примера 3.4 налог вычисляется по следующим правилам: 9 % от стоимости контракта, если эта стоимость составляет не более 100 млн ден. ед., и 11 %, если стоимость контракта превышает 100 млн ден. ед.

1 Открыть запрос *Список объектов* в режиме конструктора.

2 В поле *Налог* нажать правую кнопку мыши. Выбрать из появившегося меню команду **Построить**. Открывается окно построителя выражений.

3 В верхней части окна построителя выражений (окно формулы) удалить прежнюю формулу *[Стоимость контракта]*0,1*, оставив только имя поля *Налог*.

4 В левой нижней части окна раскрыть список **Функции**, затем – список **Встроенные функции**. В средней нижней части окна выбрать группу функций **Управление**. В правой нижней части окна выбрать функцию **IIf.** Строится следующая формула: *«Выражение» IIf («expr»; «truepart»; «falsepart»)*. Здесь *Выражение* – имя выражения (его можно будет удалить, так как в данном случае удобнее использовать имя *Налог*). Вместо отметки *expr* необходимо ввести условие, которое может быть истинным или ложным. Вместо отметок *truepart* и *falsepart* необходимо ввести формулы, которые должны вычисляться, если условие *expr* окажется соответственно истинным или ложным.

5 Ввести необходимые элементы формулы. Ее окончательный вид следующий: *Налог: IIf([Стоимость контракта]<=100000000;[Стоимость контракта]*0,08;[Стоимость контракта]*0,11)*. Никаких знаков в конце формулы не требуется.

6 В окне построителя выражений нажать кнопку **OK.**

7 Сохранить внесенные изменения. Просмотреть полученный запрос.

3.1.4 Запросы с условиями выборки

Пример 3.6 (простое условие выборки) – Создать запрос для получения списка рабочих-штукатуров, имеющих разряд не ниже пятого. В запросе должны указываться следующие данные: табельный номер, фамилия, разряд, допуск к работам на высоте, а также шифр объекта и название заказчика, у которого работает данный рабочий. Запрос должен быть упорядочен по фамилиям (по алфавиту).

1 Сделать копию запроса *Список рабочих*. Для этого отметить этот запрос и выбрать команду **Правка – Копировать**. Затем выбрать команду **Правка – Вставить**. Указать имя запроса – *Отобранные*.

2 Открыть запрос *Отобранные* в режиме конструктора.

3 Удалить лишние поля *Дата рождения* и *Дата приема на работу*. Чтобы удалить поле, необходимо поместить курсор в это поле и выбрать команду **Правка – Удалить столбцы**.

4 Из таблицы *Рабочие* (над описанием запроса) выбрать поле *Шифр объекта* и поместить его в свободный столбец в конце запроса (после столбца *Допуск на высоту*).

5 Чтобы добавить в запрос название заказчика, нажать правую кнопку мыши в области над описанием запроса и выбрать команду **Добавить таблицу**. Выбрать таблицу *Объекты*. Из этой таблицы выбрать поле *Заказчик* и поместить в свободный столбец в конце запроса.

6 В строке **Условие отбора** для поля *Профессия* указать *штукатур*. Снять флажок **Вывод на экран** для поля *Профессия* (так как в запросе будут только данные о штукатурах, выводить название профессии не требуется).

7 В строке **Условие отбора** для поля *Разряд* указать *>=5*.

8 В строке **Сортировка** для поля *Фамилия* установить значение **По возрастанию**.

9 Сохранить внесенные изменения. Просмотреть запрос.

Пример 3.7 (сложное условие выборки) – Изменить запрос *Отобранные*, созданный в примере 3.6, таким образом, чтобы получить список **штукатуров и каменщиков**, имеющих разряд не ниже пятого.

1 Открыть запрос *Отобранные* в режиме конструктора.

2 С помощью мыши переместить поле *Профессия*, чтобы оно размещалось перед полем *Разряд* (если оно там не находится).

3 В строке **Или** для поля *Профессия* указать *каменщик*. В этой же строке для поля *Разряд* указать *>=5*.

Примечания

1 Хотя условие *>=5* уже было указано для поля *Разряд* в строке **Условие отбора**, его необходимо также указать в строке **Или**. Если не сделать этого, то в запросе будет получен список штукатуров, имеющих разряд не ниже пятого, и **всех** каменщиков.

2 Указать условие отбора в данном примере можно было и по-другому: в строке **Условие отбора** для поля *Профессия* указать условие *"каменщик" Or "штукатур"*, а для поля *Разряд* – условие *>=5*. Строку **Или** в этом случае использовать не требуется.

4 Для поля *Профессия* установить флажок **Вывод на экран** (так как в запросе будут указаны рабочие двух профессий – каменщики и штукатуры, профессию также следует выводить на экран).

5 Сохранить внесенные изменения. Просмотреть запрос.

3.1.5 Запросы с групповыми операциями

Групповые операции – это операции суммирования по отдельным полям, вычисление средних, подсчет количества записей и т. д.

Пример 3.8 (создание запроса в режиме мастера) – Создать запрос для подсчета суммарной стоимости контрактов и количества контрактов каждого заказчика.

1 Нажать кнопку **Создать**. Выбрать команду **Простой запрос**. Нажать **OK**.

2 В поле **Таблицы и запросы** выбрать таблицу *Объекты*. Из списка **Доступные поля** перенести в список **Выбранные поля** обозначения полей *Заказчик* и *Стоимость контракта*. Нажать **Далее.**

3 Выбрать вид отчета – **Итоговый**. Нажать кнопку **Итоги**. В появившемся окне **Итоги** для поля *Стоимость контракта* установить флажок **Sum**. Установить также флажок **Подсчет записей**. Нажать OK. В окне **Создание простых запросов** нажать **Далее.**

4 Задать имя запроса *Стоимость контрактов по заказчикам*. Установить переключатель **Открыть запрос для просмотра данных**. Нажать **Готово.**

5 Внести изменения в созданный запрос, чтобы сделать его заголовки более понятными. Для этого открыть запрос в режиме конструктора. Для поля, в котором указывается сумма стоимостей контрактов, в строке **Поле** вместо отметки *Sum – Стоимость контракта* (т. е. заголовка, заданного по умолчанию) ввести заголовок *Стоимость контрактов*. Аналогично для поля с количеством объектов вместо стандартного заголовка *Count – Объекты* ввести заголовок *Количество*. Сохранить внесенные изменения и просмотреть запрос.

Пример 3.9 (создание запроса в режиме конструктора) – Создать запрос для подсчета количества рабочих каждой профессии. Запрос должен быть упорядочен по количеству рабочих (первыми должны выводиться профессии с максимальным количеством рабочих).

Примечание – Если в запросе требуется только операция подсчета (без каких-либо других операций), то такие запросы необходимо создавать в режиме конструктора.

1 Нажать кнопку **Создать**. Выбрать команду **Конструктор**. Нажать **OK.** Появляется шаблон запроса, а также окно **Добавление таблицы** (если его нет, то следует выбрать команду **Запрос – Отобразить таблицы**).

2 В окне **Добавление таблицы** выбрать таблицу *Рабочие*. Нажать кнопку **Добавить**. Закрыть окно **Добавление таблицы.**

3 В строке **Поле** выбрать отметки полей *Профессия* и *Табельный номер* (именно в таком порядке).

4 Выбрать команду **Вид – Групповые операции**. В шаблоне запроса появляется строка **Групповая операция.**

5 В строке **Групповая операция** для поля *Профессия* выбрать отметку **Группировка**, для поля *Табельный номер* – отметку **Count**. Это означает, что должно быть подсчитано количество табельных номеров по каждой профессии.

6 В строке **Поле** перед отметкой *Табельный номер* указать желаемое имя поля, например, *Количество*, затем – двоеточие. Таким образом, отметка в строке **Поле** будет следующей: *Количество: Табельный номер*. Если не сделать этого, то по умолчанию будет назначено имя *Count – Табельный номер.*

7 В строке **Сортировка** для поля *Количество: Табельный номер* выбрать значение **По убыванию.**

8 Сохранить запрос под названием *Количество рабочих по профессиям*. Просмотреть запрос. При необходимости внести в него изменения.

Пример 3.10 (создание запроса с источником-запросом) – Создать запрос для подсчета количества объектов у каждого заказчика, а также суммы налогов, выплачиваемых каждым заказчиком.

Так как данных о налогах нет ни в одной таблице, в качестве источника будет использоваться не таблица, а запрос, в котором такие данные есть (запрос *Список объектов* из примера 3.5).

1 Нажать кнопку **Создать**. Выбрать команду **Конструктор**. Нажать **OK**.

2 В окне **Добавление таблицы** перейти на вкладку **Запросы** (или **Таблицы и запросы**). Выбрать запрос *Список объектов*. Нажать кнопку **Добавить**. Закрыть окно **Добавление таблицы**.

3 В строке **Поле** выбрать отметки полей *Заказчик*, *Шифр объекта* и *Налог* (именно в таком порядке).

4 Выбрать команду **Вид – Групповые операции**.

5 В строке **Групповая операция** для поля *Заказчик* выбрать отметку **Группировка**, для поля *Шифр объекта* – отметку **Count**, для поля *Налог* – отметку **Sum**.

6 Для полей *Шифр объекта* и *Налог* установить имена *Количество объектов* и *Сумма налогов*, как показано в предыдущем примере. Для поля *Сумма налогов* становить формат **Денежный** (см. пример 3.4). Установить сортировку по названию заказчика (в алфавитном порядке).

7 Сохранить запрос под названием *Налоги по заказчикам*. Просмотреть запрос.

Примечание – Эту задачу можно было решить и в режиме мастера аналогично примеру 3.8.

Пример 3.11 (создание запроса для подсчета итогов по всей таблице) – Создать запрос для подсчета общей стоимости контрактов.

В режиме мастера или конструктора создать запрос, содержащий только одно поле – *Стоимость контракта* (такой запрос можно создать на основе таблицы *Объекты* или запроса *Список объектов*). Выбрать команду **Вид – Групповые операции**, чтобы в шаблоне запроса появилась строка **Групповая операция**. В этой строке для поля *Стоимость контрактов* выбрать операцию **Сумма**. Установить для поля имя *Общая стоимость*, задать денежный формат. Сохранить и просмотреть запрос.

3.1.6 Перекрестные запросы

Пример 3.12 – Создать запрос для подсчета количества рабочих каждой профессии, работающих на каждом из объектов. Результат запроса будет представлять собой таблицу, где в строках будут указаны объекты, а в столбцах – профессии.

1 Нажать кнопку **Создать**. Выбрать команду **Перекрестный запрос**. Нажать **OK**.

2 В появившемся окне **Создание перекрестных таблиц** для переключателя **Показать** выбрать значение **Таблицы** (или **Все**). Выбрать таблицу *Рабочие*. Нажать **Далее.**

3 Из списка **Доступные поля** перенести в список **Выбранные поля** отметку поля *Шифр объекта* (т. е. выбрать поле, которому в создаваемой таблице будут соответствовать строки). Нажать **Далее.**

4 Выбрать поле *Профессия* (т. е. поле, которому в создаваемой таблице будут соответствовать столбцы). Нажать **Далее.**

5 В очередном окне **Создание перекрестных таблиц** в списке **Поле** выбрать *Табельный номер*, в списке **Функции – Число** (так как требуется подсчитать количество рабочих). Установить также флажок **Вычислить итоговое значение для каждой строки** (чтобы подсчитать количество рабочих на каждом объекте). Нажать **Далее.**

6 Задать имя запроса *Распределение профессий по объектам*. Установить переключатель **Просмотреть результаты запроса**. Нажать **Готово**. Результаты запроса должны иметь примерно такой вид, как показано на рисунке 3.1.

Распределение профессий по объектам : перекрестный запрос

Объект	Итоговое значение	каменщик	маляр	разнорабочий	штукатур
А70	1		1		
Д50	4	2			2
П100	2	1			1
П80	2			1	1

Рисунок 3.1 – Результаты перекрестного запроса

Пример 3.13 – Создать запрос для подсчета количества рабочих каждой профессии, работающих на объектах каждого вида (жилые дома, офисы, цехи и т. д.).

Все данные, необходимые для построения перекрестного запроса, должны содержаться **в одной таблице или запросе** (другими словами, в перекрестном запросе невозможно использовать данные из нескольких таблиц или запросов). Поэтому сначала необходимо **создать обычный запрос**, содержащий поля *Табельный номер* и *Профессия* (из таблицы *Рабочие*) и *Вид объекта* (из таблицы *Объекты*). Присвоить ему имя *Распределение рабочих по видам объектов.*

Создать перекрестный запрос, как показано в примере 3.12. На шаге 2 потребуется установить для переключателя **Показать** значение **Запросы** (или **Все**) и выбрать запрос *Распределение рабочих по видам объектов*. В качестве строк создаваемого запроса следует выбрать виды объектов, в качестве столбцов – названия профессий (или наоборот). Присвоить созданному отчету имя *Распределение профессий по видам объектов*.

Пример 3.14 – Самостоятельно создать запрос для подсчета общей стоимости объектов каждого вида для каждого заказчика.

3.1.7 Запросы на изменение базы данных

Пример 3.15 (запрос на обновление) – Создать запрос для выполнения следующей операции: для всех каменщиков и штукатуров, имеющих разряд не ниже пятого, установить шифр объекта П80.

Условия отбора (профессии – каменщики и штукатуры, разряд – не ниже пятого) уже были заданы в запросе *Отобранные* (см. пункт 3.1.4).

1 Сделать копию запроса *Отобранные*, присвоив новому запросу имя *Направление на объект*.

2 Открыть запрос *Направление на объект* в режиме конструктора.

3 Удалить из запроса все лишние поля: оставить только поля *Профессия, Разряд* и *Шифр объекта*.

4 Выбрать команду **Запрос – Обновление**. В описании запроса появляется строка **Обновление**.

5 В строке **Обновление** для поля *Шифр объекта* указать значение *П80*.

6 Сохранить запрос *Направление на объект*. Закрыть его. Выполнить запрос (кнопкой **Открыть** или двойным щелчком мыши). При этом на экран будут выводиться предупреждения о том, что выполнение запроса приведет к изменению данных в таблице. На эти предупреждения отвечать **Да**.

7 Открыть запрос *Отобранные* (или таблицу *Рабочие*) и убедиться, что для всех каменщиков и штукатуров с разрядом не ниже пятого установлен шифр объекта *П80*.

Пример 3.16 (запрос на удаление) – Создать запрос для удаления данных обо всех рабочих, имеющих разряд не выше второго и не имеющих допуска к работам на высоте.

1 Создать обычный запрос, содержащий поля *Фамилия, Разряд* и *Допуск на высоту* (из таблицы *Рабочие*). В строке **Условие отбора** для поля *Разряд* указать условие *<=2*, для поля *Допуск на высоту* – условие *нет*. Сохранить запрос под именем *Удаление*.

2 Выполнить запрос *Удаление* (кнопкой **Открыть** или двойным щелчком мыши). Эта операция требуется **только в качестве меры предосторожности** (запрос пока выполняет не удаление, а только просмотр данных). Убедиться, что в результате выполнения запроса на экран выводится информация о рабочих, соответствующих заданным условиям (разряд не выше второго, нет допуска на высоту).

3 Открыть запрос *Удаление* в режиме конструктора. Выбрать команду **Запрос – Удаление**. Можно (но не обязательно) также удалить из запроса поле *Фамилия*.

Примечание – Если требуется снова преобразовать запрос для удаления в обычный запрос (для выборки данных), то необходимо выполнить команду **Запрос – Выборка**.

4 Сохранить запрос *Удаление*. Выполнить его. При этом на экран выводятся предупреждения. Чтобы выполнить удаление, ответить на эти предупреждения **Да**.

5 Открыть таблицу *Рабочие* и убедиться, что удаление выполнено.

3.1.8 Параметрические запросы

Пример 3.17 – Создать запрос для получения списка рабочих заданной профессии, имеющих разряд не ниже заданного. В списке, получаемом в результате выполнения запроса, должны быть указаны табельные номера, фамилии и разряды рабочих. Профессия и разряд, должны вводиться пользователем при выполнении запроса.

1 Создать обычный запрос, содержащий поля *Табельный номер*, *Фамилия* и *Профессия* (из таблицы *Рабочие*).

2 Для поля *Профессия* в строке **Условие отбора** ввести: [*Укажите профессию*]. Для поля *Разряд* в строке **Условие отбора** ввести >=[*Укажите разряд*]. Здесь *Укажите профессию* и *Укажите разряд* – подсказки, выводимые на экран при выполнении запроса. Они указываются в квадратных скобках.

3 Для поля *Профессия* снять флажок **Вывод на экран** (так как в результате запроса должен быть получен список рабочих одной профессии, выводить ее на экран для каждого рабочего не требуется). Описание запроса должно иметь примерно такой вид, как показано на рисунке 3.2.

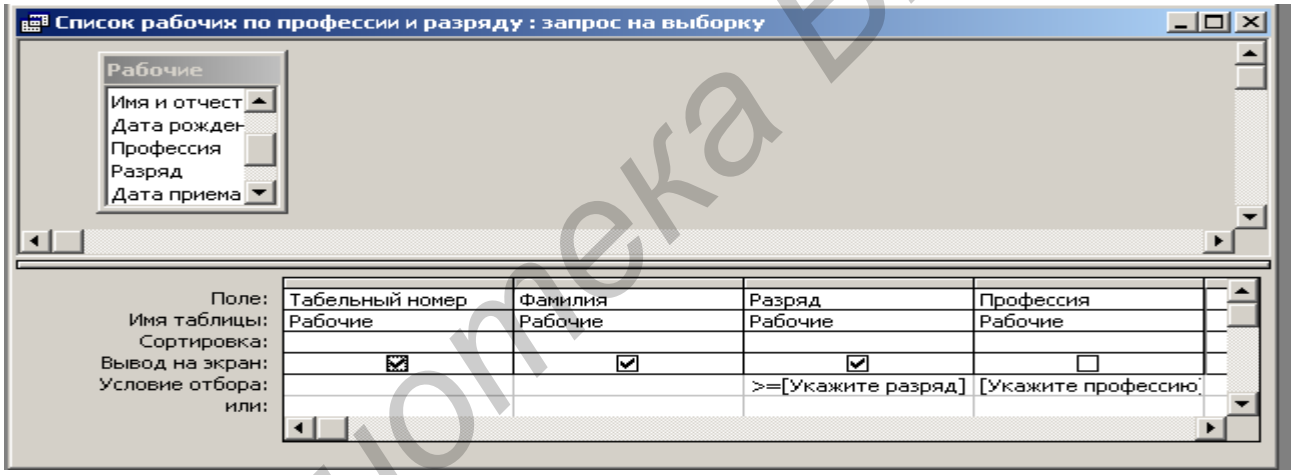

Рисунок 3.2 – Создание параметрического запроса

4 Сохранить запрос под именем *Список рабочих по профессии и разряду*. Выполнить запрос и убедиться в том, что он выполняется правильно.

3.2 Отчеты

3.2.1 Назначение, способы создания, виды отчетов.

Структура отчета

Отчеты предназначены для выборки информации из базы данных и ее представления в желаемой форме.

В качестве источников данных для отчетов могут использоваться таблицы или запросы. Отчеты могут содержать данные как из одной таблицы (или запроса), так и нескольких.

Имеются **три способа создания отчетов**:

1) автоматический (автоотчет) – отчет строится полностью автоматически и включает все поля выбранной таблицы (или отчета);

2) в режиме мастера – в процессе построения отчета пользователю предлагаются подсказки и возможности выбора;

3) в режиме конструктора – отчет строится пользователем самостоятельно.

Имеются следующие **основные виды отчетов**:

а) по способу размещения данных:

1) в столбец – все данные выводятся в один столбец;

2) ленточные (применяются чаще всего) – каждому полю базы данных соответствует столбец отчета;

б) по структуре:

1) простые – в отчет выводятся все данные подряд (возможно, с упорядочением по одному или нескольким полям);

2) группированные – данные в отчете группируются по значению какого-либо поля. Например, если в отчете содержатся данные о рабочих некоторой организации, то они могут быть сгруппированы по профессиям.

Большинство отчетов содержат групповые операции, обычно – суммирование по некоторым полям.

Кроме того, имеются специальные виды отчетов: почтовые наклейки, диаграммы.

Для создания простых отчетов, содержащих данные из одной таблицы, обычно проще использовать автоматический способ. В других случаях создавать отчеты удобнее в режиме мастера. В отчет, созданный автоматически или в режиме мастера, затем обычно вносятся изменения в режиме конструктора.

В **структуру отчета** входят следующие основные элементы (области отчета):

– заголовок отчета – располагается в начале отчета, обычно содержит название отчета;

– верхний и нижний колонтитулы – данные, размещаемые в начале и конце каждой страницы отчета (например номера страниц);

– заголовок и примечание группы – данные, помещаемые в начале и конце каждой группы. Например, если в отчете содержатся данные о рабочих, сгруппированные по профессии, то в заголовке группы можно указать название профессии, а в примечании группы – количество рабочих данной профессии. Если отчет простой (не разбит на группы), то он не содержит заголовков и примечаний групп;

– примечание отчета – данные, размещаемые в конце отчета (например итоги, подписи, дата и т. д.);

– область данных – основная часть отчета (данные из таблицы или запроса).

3.2.2 Автоматическое создание отчета

Пример 3.18 – Требуется создать отчет, содержащий список всех объектов и всю информацию о них.

1 Перейти на вкладку **Отчеты**. Нажать кнопку **Создать.**

2 Выбрать команду **Автоотчет: ленточный**. В качестве источника данных выбрать таблицу *Объекты*. Нажать **OK**. На экран выводится созданный отчет.

3 Просмотрев отчет, закрыть его. На запрос о сохранении изменений в отчете выбрать ответ **Да**. Указать имя отчета *Объекты*.

Чтобы снова просмотреть отчет, требуется выделить его и нажать **Открыть** или просто дважды щелкнуть по отметке отчета.

3.2.3 Внесение изменений в отчет. Оформление элементов отчета. Вычисляемые поля

Пример 3.19 – Пусть требуется внести в отчет *Объекты* следующие изменения: а) изменить заголовок отчета: он должен иметь вид «Заказы на строительство объектов»; б) предусмотреть нумерацию страниц в верхней части каждой страницы, кроме первой; в) предусмотреть в конце отчета вычисление суммы стоимости всех заказов, а также дату и подпись директора предприятия; г) изменить внешний вид заголовков колонок: заголовки из двух слов (например *Шифр объекта*) должны печататься в две строки; д) указать для каждого объекта количество дней, оставшихся до окончания строительства; е) предусмотреть сортировку отчета по количеству оставшихся дней (первыми должны указываться объекты с меньшим количеством оставшихся дней), а при одинаковом остатке – по стоимости заказа (первыми должны указываться объекты с большей стоимостью).

Для внесения любых изменений в отчет его необходимо открыть в режиме конструктора: выделить отчет и нажать кнопку **Конструктор**.

Изменение заголовка отчета. Для этого достаточно щелкнуть мышью по тексту заголовка и набрать новый заголовок.

Нумерация страниц. По умолчанию предусмотрена нумерация страниц в нижнем колонтитуле. Там имеется отметка «*="Страница " & [Page] & " из " & [Pages]*» (это значит, например, что номер страницы 2 в 8-страничном отчете будет иметь вид: *Страница 2 из 8*). Так как требуется нумерация в верхней части страницы, поле с этой отметкой следует удалить. Нумерация страниц задается следующим образом.

1 Выбрать команду **Вставка – Номера страниц**.

2 Установить переключатели **Страница N** и **Верхний колонтитул**. Выбрать **Выравнивание – по центру**. Снять флажок **Отображать номер на первой странице**. Нажать **OK**. Создается поле номера страницы.

3 Чтобы поле номера страницы не накладывалось на заголовки столбцов отчета, следует сдвинуть заголовки вниз. Это можно выполнить следующим образом:

– расширить область верхнего колонтитула. Для этого установить указатель мыши на его нижнюю границу (границу с областью данных) и, когда указатель примет вид двунаправленной стрелки, сдвинуть границу вниз с помощью мыши;

– выделить все поля заголовков столбцов. Для этого нажать клавишу **Shift** и, не отпуская ее, последовательно щелкнуть мышью по всем заголовкам столбцов (*Шифр объекта* и т. д.). Поле номера страницы не выделять. Выделив заголовки, отпустить клавишу **Shift;**

– с помощью мыши переместить заголовки таким образом, чтобы они не накладывались на номер страницы (при перемещении курсор имеет вид раскрытой ладони).

4 Внести изменения в формат номера страницы. Поле номера страницы содержит примерно следующее выражение: *=IIf([Page]>1;"Страница " & [Page];"")*. Чтобы указывался только номер (без слова **Страница**), изменить это выражение следующим образом: *=IIf([Page]>1;[Page];"")*. Изменить это выражение можно непосредственно в поле номера страницы или щелкнуть по этому полю правой кнопкой мыши, выбрать команду **Свойства**, перейти на вкладку **Данные** и внести изменения в строке **Данные**.

Оформление примечания отчета. Суммирование по столбцу отчета. Чтобы в отчете вычислялась сумма всех контрактов, необходимо вставить поле для ее вычисления.

1 Если примечание отчета отсутствует на экране, выбрать команду **Вид – Заголовок/примечание отчета**. Если в примечании отчета нет места для размещения данных (суммы, даты и подписи), то следует расширить примечание отчета, как показано выше. Кроме того, если на экране отсутствует панель элементов, выбрать команду **Вид – Панель элементов.**

2 Из панели элементов выбрать элемент **Поле** (с отметкой **аб**). Переместить курсор в примечание отчета и щелкнуть мышью в нужном месте. На экране появляются два новых элемента: новое поле (с отметкой **Свободный**) и надпись к нему (слово **Поле** и некоторый номер).

3 Вызвать окно свойств *надписи* нового поля. Для этого щелкнуть *по надписи* правой кнопкой мыши и выбрать команду **Свойства**. В появившемся окне свойств надписи перейти на вкладку **Макет** (или на вкладку **Все**). Для свойства **Подпись** ввести текст *Итого:*. Закрыть окно свойств.

4 Вызвать окно свойств *нового поля* (самого поля, а не надписи к нему). На вкладке **Данные** в строке свойства **Данные** ввести формулу: *=Sum([Стоимость контракта])*. На вкладке **Макет** в строке **Формат поля** выбрать **Денежный** или **С разделителями разрядов**. Закрыть окно свойств.

Примечание – Здесь *Стоимость контракта* – имя поля таблицы; оно должно быть заключено в квадратные скобки и абсолютно точно совпадать с именем поля, указанным в описании таблицы. Формула должна начинаться со знака «равно». Никаких знаков в конце формулы не ставится.

Чтобы ***поместить в примечание отчета дату***, следует выбрать команду **Вставка – Дата и время**. В появившемся окне выбрать желаемый формат даты, снять флажок **Формат времени**. Нажать **OK**. Поле даты помещается в отчет (по умолчанию – в заголовок отчета). Его следует переместить с помощью мыши в желаемое место (в данном примере – в примечание отчета).

Следует также удалить из нижнего колонтитула поле с отметкой **Now**; оно представляет собой отметку даты, вставляемую по умолчанию в нижнюю часть каждой страницы.

Чтобы ***поместить в примечание отчета подпись директора***, следует из панели элементов выбрать элемент **Надпись** (с отметкой ***Aa***). Переместить курсор (принявший вид креста с буквой ***A***) в область примечания отчета и щелкнуть мышью в нужном месте. В прямоугольнике, появившемся в области примечания отчета, ввести текст: *Директор*.

Печать заголовков столбцов в две строки. Чтобы заголовок столбца (например, *Шифр объекта*) печатался в отчете в две строки (слово *Шифр* – в одной строке, слово *объекта* – в другой), достаточно поместить курсор в желаемое место в заголовке (между двумя словами) и нажать **Ctrl-Enter**. Если часть заголовка колонки (например вторая строка) после этого не видна на экране, то следует расширить поле, в котором находится этот заголовок.

Если требуется изменить выравнивание текста в заголовке (например, выровнять его по левому краю или по центру), то необходимо вызвать окно свойств этого заголовка. Для этого щелкнуть по заголовку правой кнопкой мыши (при этом курсор *не должен находиться внутри заголовк*а) и выбрать команду **Свойства**. На вкладке **Макет** выбрать желаемую настройку для свойства **Выравнивание текста**. Можно также изменить шрифт и другие параметры текста. Затем закрыть окно свойств.

Вычисляемое поле. Рассмотрим создание вычисляемого поля, в котором будет указываться количество дней, оставшихся до конца строительства.

1 В верхнем колонтитуле (после заголовка *Дата окончания*) вставить надпись, как показано выше. Ввести надпись *Осталось дней*.

2 В области данных после поля *Дата окончания* вставить новое поле. Подпись к этому полю (со словом **Поле**) удалить, само поле (со словом **Свободный**) – оставить.

3 Перейти в окно свойств созданного поля. На вкладке **Данные** в строке **Данные** ввести: *=[Дата окончания]-date()*. На вкладке **Макет** выбрать **Выравнивание текста – По центру**. Закрыть окно свойств.

Сортировка отчета. Выполним сортировку отчета по оставшемуся количеству дней до конца строительства, а при одинаковом количестве дней – по стоимости контракта.

1 Выбрать команду **Вид – Сортировка и группировка**.

2 В колонке **Поле/выражение** в первой строке ввести выражение: *=[Дата окончания]-date()*, т. е. точно такое же выражение, как для вычисле-

ния остатка дней до конца строительства. В колонке **Порядок** сортировки выбрать **По возрастанию**.

3 Во второй строке в колонке **Поле/выражение** выбрать поле *Стоимость контракта*, а в колонке **Порядок сортировки – По убыванию**.

4 Закрыть окно **Сортировка и группировка**.

Окончательный вид отчета в режиме конструктора (после внесения всех изменений) показан на рисунке 3.3. Подготовленный отчет следует сохранить, затем закрыть (т. е. выйти из режима конструктора) и открыть для просмотра (нажатием кнопки **Открыть**). Отчет показан на рисунке 3.4.

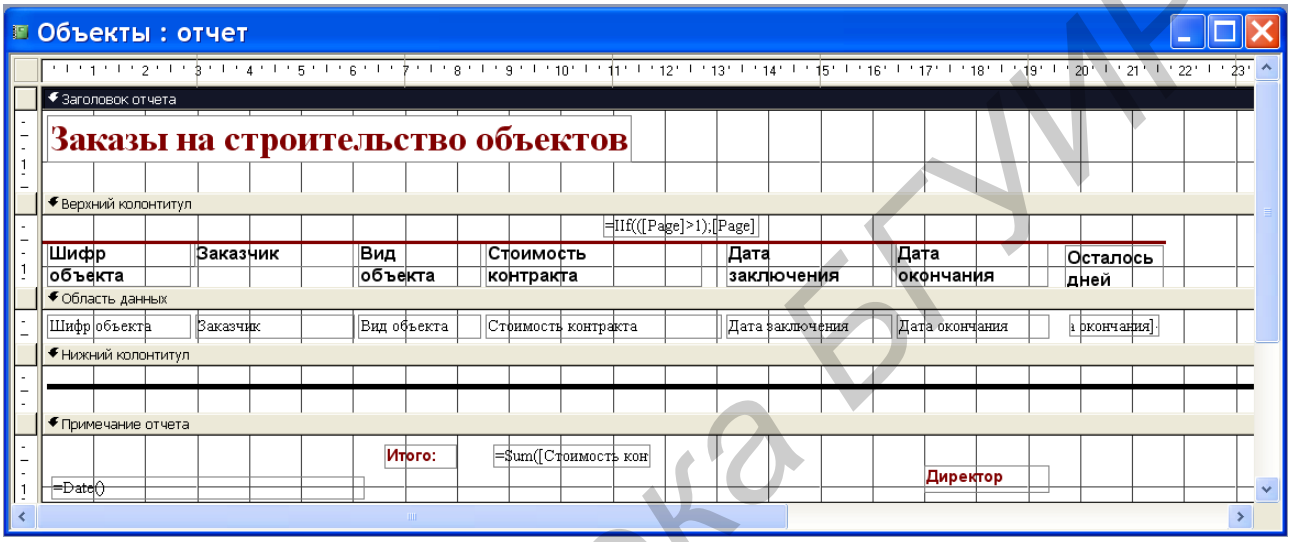

Рисунок 3.3 – Окончательный вид отчета в режиме конструктора

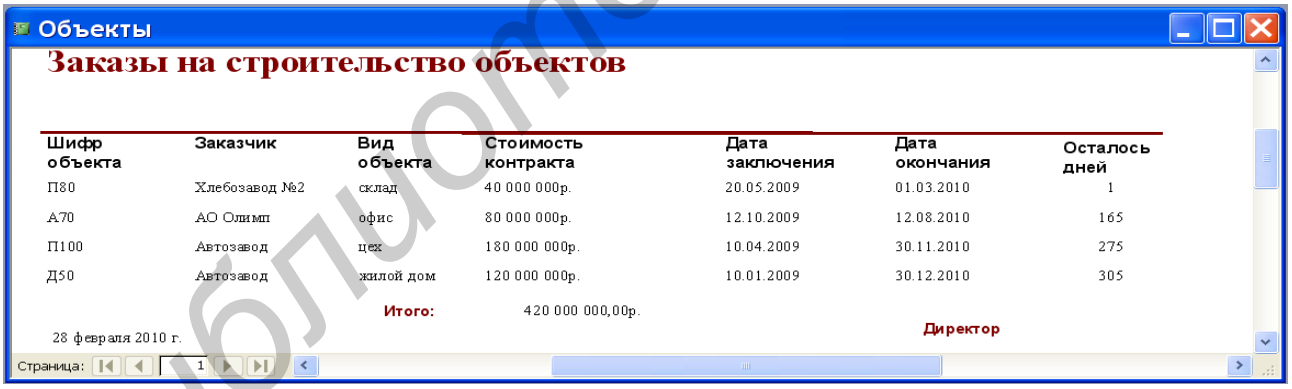

Рисунок 3.4 – Отчет в режиме просмотра

3.2.4 Группированные отчеты

Пример 3.20 – Требуется создать отчет, содержащий список рабочих. Для каждого рабочего указывается его табельный номер, фамилия, профессия, разряд. Список должен быть разбит по профессиям рабочих. Список рабочих одной профессии сортируется по табельному номеру. Кроме того, должно подсчитываться количество рабочих каждой профессии.

В данном случае требуется создать группированный отчет с одним уровнем группировки: по профессиям. Такие отчеты необходимо создавать в режиме мастера.

1 Перейти на вкладку **Отчеты**. Нажать кнопку **Создать**.

2 Выбрать команду **Мастер отчетов**. В качестве источника данных выбрать таблицу *Рабочие*. Нажать **OK**.

3 В очередном окне *Создание отчетов* из списка **Доступные поля** перенести в список **Выбранные поля** отметки полей *Табельный номер*, *Фамилия*, *Профессия*, *Разряд*. Нажать **Далее**.

4 В очередном окне **Создание отчетов** из списка **Добавить уровни группировки** выбрать поле *Профессия*. Нажать **Далее**.

5 В очередном окне **Создание отчетов** (выбор порядка сортировки) выбрать поле *Табельный номер*, чтобы списки рабочих (уже сгруппированные по профессиям) сортировались по табельному номеру. Нажать **Далее**.

6 Выбрать вид макета для отчета – **Структура 1**. Установить флажок **Настроить ширину полей для размещения на одной странице**. Нажать **Далее**.

7 Выбрать стиль оформления заголовка (по своему усмотрению). Нажать **Далее**.

8 Указать имя отчета *Список рабочих по профессиям*. Установить переключатель **Просмотреть отчет**. Нажать **Готово**. На экран выводится созданный отчет (см. рисунок 3.5).

Чтобы в отчете подсчитывалось количество рабочих каждой профессии, потребуется внести в отчет изменение: добавить после каждой группы (т. е. после списка рабочих одинаковой профессии) поле для подсчета количества рабочих. Это делается следующим образом.

1 Открыть отчет *Список рабочих по профессиям* в режиме конструктора.

2 Если в структуре отчета отсутствует примечание группы, выбрать команду **Вид – Сортировка и группировка**. Для поля *Профессия* (оно должно быть указано первым в колонке **Поле/выражение**) в строке **Примечание группы** выбрать значение **Да**. Закрыть окно **Сортировка и группировка**.

3 В примечание группы *Заказчик* вставить поле, как показано в пункте 3.2.3 (пример с суммированием по столбцу отчета). Ввести надпись для созданного поля: *Количество рабочих*. Для созданного поля вызвать окно свойств и в строке **Данные** ввести: *=Count([Табельный номер])*. Функция *Count* выполняет подсчет. Таким образом, будет подсчитываться количество табельных номеров (т. е. рабочих). Так как поле вставлено в примечании группы, подсчет будет выполняться для группы (т. е. для профессии).

4 Сохранить изменения в отчете. Закрыть и просмотреть его.

Пример 3.21 – Требуется создать отчет, содержащий информацию о распределении рабочих различных профессий по заказчикам. Для каждого заказчика должен выводиться список рабочих, занятых на объектах этого заказчика. Этот список должен быть разбит по профессиям рабочих. Для каждого рабочего указывается его табельный номер, фамилия, разряд, а также шифр объекта, на котором он работает. Список рабочих одной профессии, занятых у определенного заказчика, сортируется по табельному номеру.

Кроме того, должно подсчитываться количество рабочих, занятых у каждого заказчика.

В данном случае требуется создать отчет, содержащий данные из двух таблиц (*Объекты* и *Рабочие*). Отчет должен иметь два уровня группировки: по заказчикам и по профессиям.

1 Перейти на вкладку **Отчеты**. Нажать кнопку **Создать.**

2 Выбрать команду **Мастер отчетов**. Нажать **OK.**

3 В поле **Таблицы и запросы** выбрать таблицу *Рабочие*. Из списка **Доступные поля** перенести в список **Выбранные поля** отметки полей *Табельный номер*, *Фамилия*, *Профессия*, *Разряд*, *Шифр объекта*.

4 В поле **Таблицы и запросы** выбрать таблицу *Объекты*. Из списка **Доступные поля** перенести в список **Выбранные поля** отметку поля *Заказчик*. Нажать **Далее.**

5 В появившемся очередном окне **Создание отчетов** в поле **Выберите вид представления данных** выбрать такую таблицу, чтобы ни одно из полей в правой части окна не было выделено (необходимые поля будут выбраны на следующем шаге). В данном случае для этого требуется выделить таблицу **Рабочие**. Нажать **Далее.**

6 В очередном окне из списка **Добавить уровни группировки** выбрать поле *Заказчик*, затем – поле *Профессия* (именно в таком порядке, так как данные должны быть сгруппированы по заказчикам, а для каждого заказчика – по профессиям). Нажать **Далее.**

7 В очередном окне **Создание отчетов** (выбор порядка сортировки) выбрать поле *Табельный номер*, чтобы списки рабочих (уже сгруппированные по заказчикам и профессиям) сортировались по табельному номеру. Нажать **Далее**.

8 В последующих окнах выбрать вид макета для отчета (рекомендуемый вид – **Структура 1**), стиль оформления заголовка и другие параметры, задающие внешний вид отчета. Указать имя отчета *Распределение рабочих по объектам*. Созданный отчет должен иметь примерно такой вид, как показано на рисунке 3.6.

9 Чтобы подсчитать количество рабочих, занятых на объектах каждого заказчика, открыть отчет в режиме конструктора. Найти в отчете примечание группы *Заказчик* (если такого примечания нет, то установить его с помощью команды **Вид – Сортировка и группировка**) и установить в этом примечании поле для подсчета количества рабочих, как показано в примере 3.20. Сохранить изменения в отчете и просмотреть его.

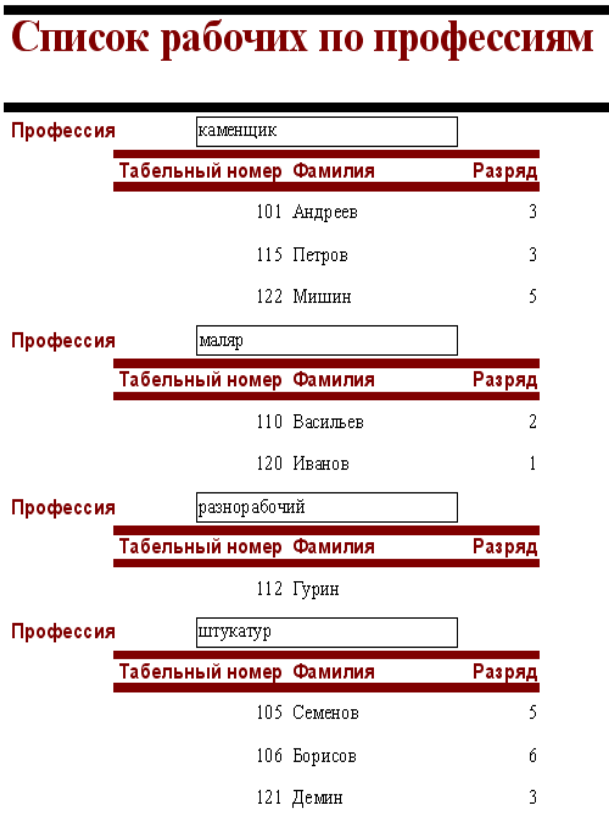

Рисунок 3.5 – Группированный отчет (один уровень группировки)

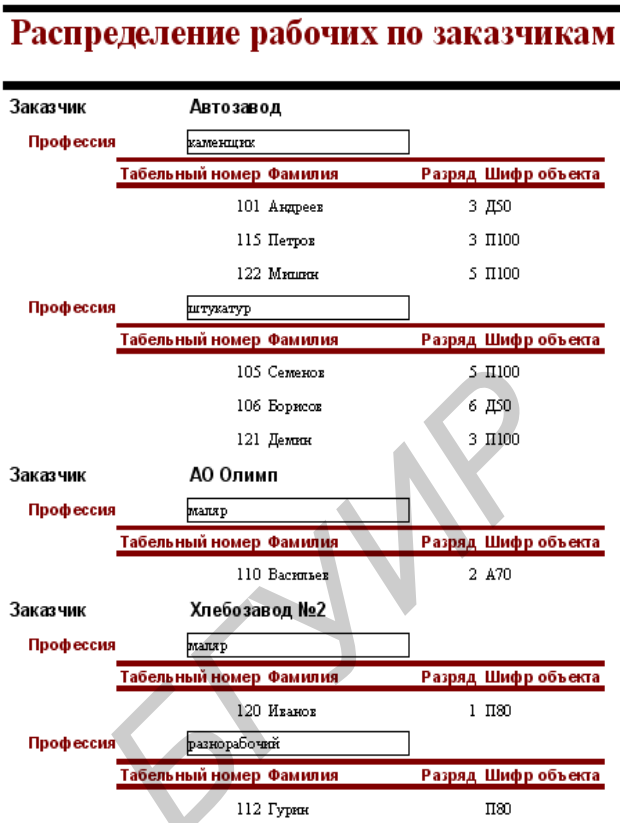

Рисунок 3.6 – Группированный отчет (два уровня группировки)

Пример 3.22 – Самостоятельно создать отчет, содержащий список объектов. Список должен быть сгруппирован по заказчикам объектов. Для каждого объекта указывается его шифр, вид объекта и стоимость контракта. Кроме того, должна подсчитываться сумма стоимостей контрактов каждого заказчика.

Указание – Для вычисления суммы стоимостей контрактов имеются два способа. Первый способ – можно, аналогично предыдущим примерам, создать группированный отчет в режиме мастера, затем открыть его в режиме конструктора и добавить поле для вычисления суммы в примечание группы *Заказчик*. Второй способ – в ходе создания отчета в одном из окон **Создание отчетов** нажать кнопку **Итоги** и установить для поля *Стоимость контракта* флажок **Sum**.

3.2.5 Создание отчета на основе запроса

В качестве источника данных для отчета может использоваться не только таблица, но и запрос.

Пример 3.23 – Создать отчет, содержащий список объектов. Для каждого объекта указывается его шифр, заказчик, вид объекта, стоимость контракта, налог. Требуется также вычислить сумму налогов по всем контрактам.

Отчет строится в режиме мастера на основе запроса *Список объектов* (см. примеры 3.4, 3.5), так как в этом запросе содержатся все необходимые данные, включая налоги. Все действия, необходимые для построения отчета, рассмотрены в предыдущих примерах.

3.3 Порядок выполнения работы

Для базы данных, построенной в лабораторной работе №2, построить запросы согласно подразделу 3.1 и отчеты согласно подразделу 3.2.

3.4 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать титульный лист, цель работы и примеры, иллюстрирующие следующие возможности работы с СУБД MS Access (по одному на каждую рассматриваемую возможность):

а) создание запроса для выборки данных из двух таблиц;
б) добавление вычисляемого поля в запрос;
в) создание запроса с несколькими условиями (связка И);
г) создание запроса с несколькими условиями (связка ИЛИ);
д) создание запроса с групповой операцией;
е) создание перекрестного запроса;
ж) создание запроса на изменение базы данных;
з) создание параметрического запроса;
и) создание группированного отчета с суммированием по одному из полей.

По пунктам д), и) примеры должны быть приведены для заданий, указанных для самостоятельного выполнения.

Для каждого примера должно быть приведено следующее:

- постановка задачи;
- описание выполняемых операций и используемых элементов меню;
- копии экранов, иллюстрирующие ход решения задачи и ее результаты.

3.5 Контрольные вопросы

1 Назначение, виды и основные возможности запросов.
2 Условия отбора в запросах. Реализация связок И и ИЛИ.
3 Сортировка данных в запросах. Сортировка по нескольким полям.
4 Перекрестные запросы.
5 Запросы с групповыми операциями.
6 Перекрестные запросы.
7 Использование запросов для внесения изменений в таблицы.
8 Возможности использования запросов для внесения изменений в таблицы.
9 Назначение, виды и основные возможности отчетов.
10 Структура отчета.
11 Группированные отчеты.
12 Возможности использования отчетов для анализа данных в таблицах.

ЛАБОРАТОРНАЯ РАБОТА №4

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ VBA

Цель работы – Изучение основных операторов и реализации основных конструкций программирования в языке VBA.

4.1 Основные этапы работы с программами на VBA в Excel

Для подготовки и выполнения программы на языке VBA требуется выполнить следующее:

– в MS Excel выбрать из меню команду **Сервис – Макрос – Редактор Visual Basic**;

– в появившемся окне выбрать из меню команду **Insert – Module**. Создается модуль, т. е. открывается окно, в котором можно вводить текст программы.

В начале модуля может указываться инструкция **Option Explicit**. Если она указана, то все переменные, используемые в программе, необходимо будет объявлять в операторе **Dim** (подробнее об этом см. в подразделе 4.3).

В некоторых случаях, в зависимости от настройки среды VBA, инструкция **Option Explicit** указывается в начале модуля автоматически. Если программист желает использовать переменные, не объявляя их в операторе **Dim**, то инструкцию **Option Explicit** необходимо удалить.

Для запуска программы на выполнение необходимо выбрать из меню команду **Run – Run Sub/UserForm**.

4.2 Простейший пример программы на языке VBA

Пример 4.1 – Программа, возводящая указанное число **a** в указанную степень **b**.

```
Sub primer4_1()
‘Первый пример программы на VBA
Dim a As Single, b As Single
a = InputBox(“Введите основание: ”)
b = InputBox(“Введите показатель степени: ”)
x = a^b      ‘Возведение в степень
MsgBox(“Результат равен ” & x)
End Sub
```

Здесь слово **Sub** обозначает начало процедуры; ее имя в данном случае – **primer4_1**. Программа на языке VBA всегда состоит из одной или нескольких процедур (в данном случае – из одной).

Символ ‘ (одиночная кавычка) обозначает начало комментария. Текст комментария может быть любым.

Dim – оператор объявления переменных. В данном случае указано, что переменные **a** и **b** имеют тип **Single**, т. е. могут представлять собой как целые, так и дробные числа. Подробнее типы данных и объявление переменных будут рассмотрены в подразделе 4.3.

InputBox – функция для ввода значения переменной. Строка **a = InputBox (“Введите основание: ”)** означает, что вводится значение переменной **a**; при

этом на экран выводится сообщение **“Введите основание:”**. Строка **x = a^b** – оператор присваивания: вычисляется значение правой части (в данном случае переменная **a** возводится в степень **b**), и результат присваивается переменной, указанной в левой части (в данном случае – переменной **x**). Строка **MsgBox(“Результат равен ” & x)** означает, что на экран выводится сообщение **“Результат равен ”** и значение переменной **x.**

Примечание – Знак **&** в функции **MsgBox** предназначен для сцепления нескольких элементов данных, которые требуется вывести на экран, в данном примере – строки **“Результат равен”** и переменной **x**. Аналогично знак **&** может использоваться в функции **InputBox** (примеры такого использования будут приведены далее). Перед знаком **&** и после него обязательно должны быть указаны пробелы.

В одной строке можно разместить несколько операторов языка VBA. Для этого они разделяются символами «двоеточие». Так, программу из примера 4.1 можно было записать, например, следующим образом:

```
Sub primer4_1()
‘Первый пример программы на VBA
Dim a As Single, b As Single
a = InputBox(“Введите основание: ”) : b = InputBox(“Введите показатель степени: ”)
x = a^b : MsgBox(“Результат равен ” & x)        ‘Возведение в степень и вывод результата
End Sub
```

В рассмотренном примере использована процедура, называемая подпрограммой. Такая процедура начинается со слова **Sub**. В программе на VBA всегда имеется хотя бы одна процедура-подпрограмма. Кроме того, в языке VBA имеется еще один вид процедур – функции. Процедура-функция начинается со слова **Function**. Использование функций будет рассмотрено в подразделе 4.9.

4.3 Типы данных. Объявление переменных и констант

4.3.1 Типы данных

Тип данных, указанный для переменной, определяет, какие значения может принимать эта переменная. Основные типы данных, используемые в VBA, приведены в таблице 4.1.

Таблица 4.1 – Основные типы данных в VBA

Тип данных	Допустимые значения
Byte (байт)	от 0 до 255 (только целые)
Boolean (логический)	**True** или **False**
Integer (целое)	от -32768 до 32767 (только целые)
Long (длинное целое)	от -2147483648 до 2147483647 (только целые)
Single (с плавающей точкой, обычной точности)	от $-3{,}402823\cdot10^{38}$ до $3{,}402823\cdot10^{38}$
Double (с плавающей точкой, двойной точности)	от $-1{,}79769313486231\cdot10^{308}$ до $1{,}79769313486232\cdot10^{308}$
Date (дата)	от 01.01.100 до 31.12.9999
String (строка)	строки из любых символов, практически неограниченной длины

Примечание – Здесь приведены только основные типы данных. Более подробные сведения о типах данных имеются в литературе по VBA, а также в справочной системе.

4.3.2 Объявление переменных

Тип переменной указывается при ее объявлении. Переменные обычно объявляются в начале программы, до их первого использования, как правило – сразу после оператора начала процедуры (**Sub** или **Function**). Основной оператор языка VBA для объявления переменных – оператор **Dim**.

Пусть, например, требуется, чтобы переменная **a** могла принимать любые числовые значения (как целые, так и дробные), переменные **b** и **c** – только целые (причем как положительные, так и отрицательные), переменная **d** – только положительные целые значения (причем небольшие), переменная **x** – строковые значения, переменная **y** – булевы значения (т. е. только **True** или **False**). Эти переменные можно объявить следующим образом:

```
Dim a As Single, b As Integer, c As Integer, d As Byte, x As String, y As Boolean
```

После этого, например, попытка присвоить переменной **d** любое значение, превышающее 255, вызовет сообщение об ошибке из-за несоответствия типов.

Если переменная не объявлена в операторе **Dim**, то ей назначается тип **Variant**. В этом случае тип переменной определяется автоматически в зависимости от значения, присваиваемого ей. Например, если переменная **z** не указана в операторе **Dim**, то в одной части программы ей может быть присвоено значение **z=”Минск”** (т. е. строковое значение), а в другой части программы – значение **z=5** (т. е. числовое). Использование переменных типа **Variant** приводит к увеличению затрат памяти и времени выполнения программы, а в некоторых случаях может приводить к ошибкам. Поэтому при разработке сложных программ, как правило, желательно объявлять с помощью оператора **Dim** все переменные.

Объявление в операторе **Dim** обязательно для массивов (см. подраздел 4.7), а также для некоторых сложных типов данных, не рассматриваемых в данном пособии.

Рекомендуется объявлять в операторе **Dim** все переменные, вводимые с клавиатуры, т. е. с помощью функции **InputBox**. Если, например, переменная **z** не указана в операторе **Dim**, и для нее с клавиатуры вводится числовое значение (например 7), то оно может быть распознано программой не как число, а как строка символов (в данном случае – как строка “7”). Это приводит к ошибкам, например, в операциях сравнения.

Следует обратить внимание, что в операторе **Dim** тип должен указываться для *каждой* переменной *отдельно*. Так, в примере, приведенном выше, объявление **b, c As Integer** было бы неправильным: тип **Integer** в этом случае относится только к переменной **c**, а переменная **b** остается необъявленной, и ей назначается тип **Variant**.

4.3.3 Объявление констант

Константы обычно указываются в начале процедуры, как правило – сразу после оператора начала процедуры (**Sub** или **Function**) или операторов объявления переменных (**Dim**). Для объявления констант используется оператор **Const.**

Пусть, например, в процедуре требуется использовать цену некоторого изделия, равную 250, и вес изделия, равный 3,5. Эти величины удобно задать в начале процедуры в виде констант:

```
Const cena = 250, ves = 3.5
```

Изменять константу в программе нельзя. В рассмотренном примере попытка указать в программе оператор **cena = cena + 10** или **cena = 220** приведет к сообщению об ошибке.

4.3.4 Область видимости переменных

Сложные программы практически всегда состоят из нескольких процедур, а иногда и из нескольких модулей. В зависимости от области видимости, т. е. от того, в какой части программы может использоваться переменная, различают три вида переменных:

– переменные уровня процедуры, используемые только в пределах той процедуры, в которой они объявлены. Такие переменные объявляются оператором **Dim** в соответствующей процедуре;

– переменные уровня модуля, используемые во всех процедурах модуля. Такие переменные объявляются оператором **Dim** или **Private** *в начале модуля* перед процедурами, т. е. до первого оператора **Sub** или **Function**;

– переменные уровня проекта, используемые во всех модулях проекта (т. е. программы, состоящей из нескольких модулей). Такие переменные объявляются оператором **Public** в начале любого из модулей или в отдельном модуле.

Пример 4.2 – В модуле введена следующая программа:

```
Private x As Integer
Sub primer4_2a()
Dim y As Integer
x = 5
y = 10
Call primer4_2b
End Sub

Sub primer4_2b()
MsgBox ("x=" & x)
MsgBox ("y=" & y)
End Sub
```

Здесь **x** – переменная уровня модуля, используемая во всех процедурах модуля; **y** – переменная уровня процедуры, используемая только в той процедуре, где она объявлена, т. е. в процедуре **primer4_2a**.

Пусть запускается процедура **primer4_2a**. Для этого требуется расположить курсор в пределах текста этой процедуры и выбрать из меню команду **Run – Run Sub/UserForm**. Переменной **x** присваивается значение 5, переменной **y** – значение 10. Затем вызывается процедура **primer4_2b**. Для этого используется оператор **Call** (подробнее он будет рассмотрен в разделе 4.8). В этой процедуре на экран выводятся значения переменных **x** и **y**. Так как **x** – переменная уровня модуля, ее значение, присвоенное в любой из процедур (в данном случае – в процедуре **primer4_2a**), известно и во всех остальных процедурах модуля (в данном случае – в процедуре **primer4_2b**). Поэтому на экран будет выведено **x=5**. Переменная **y** – переменная уровня процедуры, поэтому ее значение, заданное в процедуре **primer4_2a**, неизвестно в процедуре **primer4_2b**. На экран будет выведено **y=**, так как переменной **y** в процедуре **primer4_2b** не присвоено никакого значения (т. е. она представляет собой пустую строку).

4.4 Оператор If

Общий вид условного оператора **If** следующий:

If условие1 **Then**
действия, выполняемые, если условие1 верно
Elseif условие2 **Then**
действия, выполняемые, если условие1 неверно, а условие2 верно
Else
действия, выполняемые, если и условие1, и условие2 неверны
End If

Части **Elseif** и **Else** необязательны, поэтому оператор **If** может иметь следующий вид:

If условие **Then**
действия, выполняемые, если условие верно
Else
действия, выполняемые, если условие неверно
End If

или следующий:

If условие **Then**
действия, выполняемые, если условие верно
End If

Если оператор **If** записывается в одну строку, то слова **End If** после него не указываются.

Пример 4.3 – Программа запрашивает число и умножает его на 2, если оно меньше 5, или на 3, если оно больше или равно 5.

```
Sub primer4_3()
Dim x As Single
x = InputBox(“Введите число: ”)
If x < 5 Then x=x*2 Else x=x*3
MsgBox(“Результат равен ” & x)
End Sub
```

Здесь оператор **If** записан в одну строку, поэтому слова **End If** не требуются.

Пример 4.4 – Программа для решения квадратного уравнения.

```
Sub primer4_4()
Dim a As Single, b As Single, c As Single
a = InputBox ("Введите коэффициент a")
b = InputBox ("Введите коэффициент b")
c = InputBox ("Введите коэффициент c")
d = b ^ 2 - 4 * a * c
If d > 0 Then
x1 = (-b - Sqr(d)) / (2 * a): x2 = (-b + Sqr(d)) / (2 * a)
MsgBox ("x1=" & x1)
MsgBox ("x2=" & x2)
Elseif d = 0 Then
x = -b / (2 * a)
MsgBox ("x=" & x)
Else
MsgBox ("Вещественных корней нет")
End If
End Sub
```

4.5 Безусловный переход. Оператор GoTo

Для безусловного перехода на некоторый оператор в программах на VBA используется оператор **GoTo**, за которым указывается метка оператора, к которому требуется перейти. Эта же метка указывается перед оператором, к которому требуется перейти. Между меткой и оператором указывается двоеточие.

Пример 4.5 – В программу вводится для некоторых вычислений число **x,** причем требуется, чтобы оно было не меньше 0 и не больше 100. Если вводится число меньше 0 или больше 100, то программа должна выводить сообщение об ошибке и завершать работу.

```
Sub primer4_5a()
Dim x As Single
x = InputBox("Введите число")
If (x < 0) Or (x > 100) Then GoTo oshibka
… ‘операторы вычислений с переменной x
GoTo konec
oshibka: MsgBox (“Введена недопустимая величина”)
konec: End Sub
```

Здесь **oshibka** и **konec** – метки. Если верно одно из условий **x < 0** или **x > 100**, то выполняется переход на метку **oshibka**. Если ни одно из этих условий не верно, то выполняются операторы вычислений, а затем – переход на метку **konec** (это требуется, чтобы не выводилось сообщение об ошибке).

Эту задачу можно решить и многими другими способами, в том числе и без операторов **GoTo**, например, так:

```
Sub primer4_5b()
Dim x As Single
x = InputBox("Введите число")
If (x >= 0) And (x <= 100) Then
```

```
… ‘операторы вычислений с переменной x
Else
MsgBox (“Введена недопустимая величина”)
End If
End Sub
```

4.6 Цикл ДО. Оператор For

Цикл ДО (т. е. цикл, повторяющийся заданное количество раз) реализуется в VBA оператором **For**, имеющим следующий вид:

For переменная_цикла = начальное_значение **To** конечное_значение **Step** шаг
операторы, составляющие цикл
Next переменная_цикла

При выполнении такого оператора переменная цикла сначала принимает указанное начальное значение. Затем выполняются операторы, составляющие цикл. После этого вычисляется следующее значение переменной цикла: к ее текущему значению прибавляется величина шага. Если слово **Step** (а значит, и величина шага) не указано, то переменная изменяется с шагом, равным 1. Снова выполняются операторы, составляющие цикл. Процесс завершается, когда переменная цикла превышает конечное значение.

Пример 4.6 – Программа, вычисляющая сумму всех четных чисел от 0 до 100, т. е. 0+2+4+6+…+100.

```
Sub primer4_6()
sum = 0
For x = 0 To 100 Step 2
sum = sum + x
Next x
MsgBox(“Сумма = ” & sum)
End Sub
```

Пример 4.7 – Программа, вычисляющая и выводящая на экран значения e^x для x=1, 4, 7, 10, 13, 16, 19.

```
Sub primer4_7()
For x = 1 To 19 Step 3
y = exp(x)
MsgBox(“x = ” & x & “  y = ” & y)
Next x
End Sub
```

4.7 Массивы

Массив в языке VBA, как и в других языках – объект программы, состоящий из нескольких элементов. Массивы необходимо объявлять в начале программы с помощью оператора **Dim**. Например, следующее объявление

```
Dim a(1 To 6) As Integer, b(1 To 3, 1 To 10) As Single, c (1 To 6, 1 To 3) As String
```

означает, что переменная **a** – одномерный массив, который может содержать не более шести элементов (целых чисел, так как указан тип **Integer**). Пере-

менная **b** – двумерный массив из трех строк и десяти столбцов; элементы этого массива – вещественные числа (тип **Single**). Переменная **c** – массив из шести строк и трех столбцов; его элементы – строки символов.

Использование переменных для указания размеров массивов в операторе **Dim** не допускается. Например, объявление **Dim a(1 To m) As Integer** недопустимо, даже если переменной **m** уже присвоено некоторое значение.

Во многих случаях удобно задавать размеры массива не в операторе **Dim** (т. е. не в начале программы), а позже, в ходе выполнения программы. Для этого используется оператор **ReDim**. Массив в этом случае называется динамическим.

Пусть требуется использовать в программе массив из целых чисел, но количество этих чисел сначала неизвестно (например, его нужно запрашивать у пользователя). Объявление массива в этом случае может быть следующим:

```
Dim a() As Integer, m As Byte
m = InputBox ("Введите количество элементов массива ")
ReDim a(1 To m)
```

Как видно из этого примера, для указания размеров динамического массива можно использовать переменные. Если массив объявлен как динамический, то изменять его размеры, используя оператор **ReDim**, можно неоднократно.

При обращении в программе к отдельным элементам массива номера элементов указываются в круглых скобках. Если массив двумерный, то сначала указывается номер *строки*, затем – номер *столбца*. Примеры:

```
a(2) = 15
b(2,7) = 8.3
c(1,4) = “Минск”
```

Здесь второму элементу массива **a** присвоено значение 15. Элементу массива **b**, расположенному во второй строке и седьмом столбце, присвоено значение 8,3. Элементу массива **c**, расположенному в первой строке и четвертом столбце, присвоено значение «Минск».

Пример 4.8 – Программа запрашивает количество элементов, которые должны содержаться в одномерном массиве (имя массива – **a**, максимально допустимое количество элементов – 20), затем – сами элементы массива **a.** Запрашивается также некоторое число **x.** Все элементы массива **a,** превосходящие **x,** умножаются на два. Подсчитывается также количество элементов, которые потребовалось умножить.

```
Sub primer4_8a()
Dim a(1 To 20) As Single, x As Single, m As Byte
m = InputBox("Введите количество элементов: ")
MsgBox(“Вводите элементы массива”)
For i = 1 To m
a(i) = InputBox("a(" & i & "): ")
Next i
x = InputBox("Введите число x: ")
```

```
For i = 1 To m
If a(i) > x Then
a(i) = a(i)*2
umnozheno = umnozheno + 1
End If
Next i
MsgBox(“Измененный массив ”)
For i = 1 To m
MsgBox("a(" & i & ") = " & a(i))
Next i
MsgBox ("Умножено: " & umnozheno)
End Sub
```

Здесь оператор **Dim** – объявление переменных: массива **a** и обычных (скалярных) переменных **x** и **m**. Запись **Dim a(1 To 20) As Single** означает, что переменная **a** – одномерный массив, который может содержать не более двадцати элементов. Эти элементы могут быть как целыми, так и дробными числами (тип данных **Single**).

Для обработки массива используются циклы. Например, для ввода элементов массива **a** применяется следующий цикл:

```
For i = 1 To m
a(i) = InputBox("a(" & i & "): ")
Next i
```

Здесь переменная **i** принимает сначала значение 1. При **i=1** выполняется оператор **InputBox,** т. е. вводится значение первого элемента массива **a(1)**. Затем выполняется оператор **Next i**, где переменная **i** принимает значение 2. Цикл повторяется: запрашивается элемент **a(2)**. Аналогично вводятся остальные элементы.

Затем в программе выполняются еще два цикла. В одном из них каждый элемент массива сравнивается с переменной **x,** и если выполняется условие **a(i) > x**, то умножается на два. В последнем цикле элементы измененного массива выводятся на экран.

Примечание – Приведенная в примере 4.8 реализация программы имеет ряд недостатков. Так, массив **a** в любом случае хранится в памяти компьютера как массив из двадцати элементов, даже если фактическое количество элементов массива (т. е. значение переменной **m**) меньше. В то же время ввести значение переменной **m**, превышающее 20, нельзя: при выполнении цикла произойдет выход за объявленную границу массива, и программа будет прервана с выдачей сообщения об ошибке. Разрешить эти проблемы можно, используя динамические массивы. В этом случае начало программы будет иметь следующий вид:

```
Sub primer4_8b()
Dim a() As Single, m As Byte
m = InputBox("Введите количество элементов: ")
ReDim a (1 To m)
```

Теперь размер массива **a** может быть любым, в зависимости от введенного значения переменной **m**.

Пример 4.9 – В программе вводится одномерный массив **a** и некоторое число **x** (аналогично примеру 4.8). Из элементов массива **a** должны состав-

ляться два новых массива: в один из них включаются все элементы, превышающие **x**, а в другой – все остальные.

```
Sub primer4_9()
Dim a() As Single, bol() As Single, men() As Single, m As Byte, x As Single
m = InputBox("Введите количество элементов: ")
ReDim a (1 To m), bol (1 To m), men (1 To m)
MsgBox(“Вводите элементы массива”)
For i = 1 To m
a(i) = InputBox("a(" & i & "): ")
Next i
x = InputBox("Введите число x: ")
For i = 1 To m
If a(i) > x Then
j = j+1 : bol(j) = a(i)
Else
k = k+1 : men(k) = a(i)
End If
Next i
MsgBox(“Элементы больше ” & x)
For i = 1 To j
MsgBox(bol(i))
Next i
MsgBox(“Элементы меньше ” & x)
For i = 1 To k
MsgBox(men(i))
Next i
End Sub
```

Здесь каждый элемент массива **a** сравнивается с переменной **x**. Если выполняется условие **a(i) > x**, то элемент **a(i)** включается в массив **bol**. Для этого увеличивается на единицу переменная **j** – номер очередного элемента массива **bol**. Затем создается новый (**j**-й) элемент массива **bol**: **bol(j) = a(i)**. Если же условие **a(i) > x** не выполняется, то элемент **a(i)** включается в массив **men**. Для определения номера очередного элемента массива **men** используется переменная **k.**

Пример 4.10 – В программе вводится двумерный массив (имя массива – **a**). Для этого запрашивается сначала количество строк и столбцов (переменные **m** и **n**), затем – сами элементы массива. Вычисляются суммы столбцов массива. Из них составляется новый массив – одномерный массив **asum.**

```
Sub primer4_10a()
Dim a(1 To 5, 1 To 10) As Single, asum(1 To 10) As Single
m = InputBox("Введите количество строк: ")
n = InputBox("Введите количество столбцов: ")
For i = 1 To m
For j = 1 To n
a(i, j) = InputBox("a(" & i & "," & j & "): ")
Next j
Next i
```

```
For j = 1 To n
For i = 1 To m
asum(j) = asum(j) + a(i, j)
Next i
Next j
For j = 1 To n
MsgBox ("Сумма " & j & "-го столбца = " & asum(j))
Next j
End Sub
```

Здесь оператор **Dim** – объявление переменных (в данном случае – массивов). Объявление **Dim a(1 To 5, 1 To 10) As Single** означает, что переменная **a** – двумерный массив, который может содержать не более пяти строк и не более десяти столбцов. Элементы массива – числа, которые могут быть как целыми, так и дробными (тип **Single**).

Для ввода массива **a** и для суммирования его столбцов используются вложенные циклы. Рассмотрим, например, вложенные циклы для суммирования столбцов массива:

```
For j = 1 To n
For i = 1 To m
asum(j) = asum(j) + a(i, j)
Next i
Next j
```

Здесь переменная **j** (номер столбца) принимает сначала значение 1. При **j=1** выполняется вложенный цикл:

```
For i = 1 To m
asum(j) = asum(j) + a(i, j)
Next i
```

т. е. переменная **i** (номер строки) принимает значения от 1 до **m**, и выполняется суммирование элементов **a(1,1), a(2,1),**…,**a(m,1)**. В результате вычисляется величина **asum(1)** – первый элемент нового массива **asum**, сумма элементов первого столбца массива **a.**

Затем переменная **j** (номер столбца) принимает значение 2. Снова выполняется вложенный цикл, т. е. переменная **i** (номер строки) принимает значения от 1 до **m**, и выполняется суммирование элементов **a(1,2), a(2,2)**,…,**a(m,2)**. В результате вычисляется величина **asum(2)**. Аналогичные действия выполняются для значений **j** = 3,…,**n**, т. е. для каждого столбца массива **a.**

Для вывода элементов массива **asum** на экран используется следующий цикл:

```
For j = 1 To n
MsgBox ("Сумма" & j & "-го столбца = " & asum(j))
Next j
```

Здесь переменная **j** изменяется от 1 до **n** с шагом 1, т. е. принимает значения 1, 2, 3, …, **n**. В каждом цикле (т. е. **n** раз) выполняется оператор

MsgBox ("Сумма" & j & "-го столбца = " & asum(j)), т. е. на экран выводится **j**-й элемент массива **asum.**

Примечание – Чтобы массивы **a** и **asum** могли иметь любые размеры в соответствии с введенными значениями переменных **m** и **n**, следует объявить их как динамические массивы. В этом случае начало программы будет иметь следующий вид:

```
Sub primer4_10b()
Dim a() As Single, asum() As Single, m As Byte, n As Byte
m = InputBox("Введите количество строк")
n = InputBox("Введите количество столбцов")
ReDim a (1 To m, 1 To n), asum(1 To n)
```

Пример 4.11 – Программа запрашивает элементы массива из пяти строк и трех столбцов (массив **a**), а также некоторое число (переменная **x**). Затем программа подсчитывает в каждой строке массива **a** количество элементов, равных переменной **x**. Если строка полностью состоит из чисел **x**, то номер строки выводится на экран.

```
Sub primer4_11a()
Dim a(1 To 5, 1 To 3) As Single, x As Single
m = 5 : n = 3
MsgBox(“Вводите массив”)
For i = 1 To m
For j = 1 To n
a(i, j) = InputBox("a(" & i & "," & j & "): ")
Next j
Next i
x = InputBox("Введите число x: ")
For i = 1 To m
kol = 0
For j = 1 To n
If a(i,j) = x Then kol = kol + 1
Next j
If kol = n Then MsgBox (i & "-я строка состоит из чисел " & x)
Next i
End Sub
```

Здесь для подсчета количества элементов строки, равных числу **x**, используется переменная **kol**. Цикл **For i = 1 To m** используется для перебора строк, цикл **For j = 1 To n** – для перебора элементов строки. В начале перебора очередной (**i**-й) строки выполняется оператор **kol=0**, т. е. переменная **kol** обнуляется. Затем каждый элемент **i**-й строки проверяется на равенство переменной **x**, и если равенство выполняется, то переменная **kol** увеличивается на единицу (**If a(i,j) = x Then kol = kol + 1**).

Таким образом, после завершения цикла **For j = 1 To n** (т. е. по окончании перебора элементов **i**-й строки) переменная **kol** оказывается равной количеству элементов строки, значение которых совпало с переменной **x**. В операторе **If kol = n Then MsgBox (i & "-я строка состоит из чисел " & x)** проверяется значение переменной **kol**. Если оно равно количеству столбцов массива (количеству элементов в строке), значит, вся строка состояла из пе-

ременных, равных числу **x**. В этом случае номер строки выводится на экран. Затем выполняется возврат к началу цикла **For i = 1 To m**, т. е. переменная **i** увеличивается на единицу, и проверяется очередная строка.

Следует обратить внимание, что переменная **x**, вводимая с клавиатуры, объявлена в операторе **Dim** с типом **Single**, т. е. с тем же типом, что и элементы массива. В данном случае такое объявление обязательно. Если не объявить переменную **x**, то при ее сравнении с элементом массива **a(i,j)** (в операторе **If a(i,j) = x Then …**) эти величины *всегда* будут распознаваться как разные, так как элементы массива имеют тип **Single** (т. е. представляют собой десятичные числа), а переменная **x** будет рассматриваться как строка символов, а не как число (например, значение 7 будет распознано как строка “7”).

Рассмотрим еще один способ решения данной задачи. Чтобы определить, состоит ли вся строка массива из чисел **x**, воспользуемся логической переменной **vse.**

```
Sub primer4_11b()
... См. программу primer4_11a …
For i = 1 To m
vse = True
For j = 1 To n
If a(i,j) <> x Then vse = False
Next j
If vse = True Then MsgBox (i & "-я строка состоит из чисел " & x)
Next i
End Sub
```

Здесь в начале перебора каждой строки переменной **vse** присваивается значение **True**. Затем каждый элемент строки проверяется на равенство переменной **x**, и если равенство *не* выполняется, то переменная **vse** получает значение **False**. Таким образом, по окончании перебора элементов строки переменная **vse** будет иметь значение **False**, если хотя бы один элемент строки будет иметь значение, отличное от **x** (и останется равной **True**, если все элементы строки будут равны переменной **x**). В операторе **If vse = True Then …** проверяется значение переменной **vse**. Если эта переменная равна **True**, значит, вся строка состояла из переменных, равных числу **x**. В этом случае номер строки выводится на экран.

Пример 4.12 – В программу вводится двумерный массив. Программа определяет в каждом столбце массива максимальное число и выводит его на экран.

```
Sub primer4_12()
Dim a(1 To 3, 1 To 5) As Single
m = 3 : n = 5
MsgBox(“Вводите массив”)
For i = 1 To m
For j = 1 To n
a(i, j) = InputBox("a(" & i & "," & j & ")")
Next j
```

```
Next i
For j = 1 To n
maximum = a(1,j)
For i = 1 To m
If a(i,j) > maximum Then maximum = a(i,j)
Next i
MsgBox ("В " & j & "-м столбце максимальный элемент равен " & maximum)
Next j
End Sub
```

Здесь цикл **For j = 1 To n** используется для перебора столбцов, цикл **For i = 1 To m** – для перебора элементов столбца. Переменная **maximum** используется для запоминания максимального элемента столбца. Сначала она принимается равной первому элементу столбца: **maximum = a(1,j)**. Если в ходе перебора столбца обнаруживается элемент, превышающий текущее значение переменной **maximum**, то он присваивается этой переменной. В результате по окончании перебора столбца переменная **maximum** будет равна его максимальному элементу.

Пример 4.13 – В программу вводится двумерный массив. Программа определяет в каждом столбце массива максимальное число и меняет его местами с первым элементом данного столбца. Измененный массив выводится на экран.

```
Sub primer4_13()
Dim a(1 To 3, 1 To 5) As Single
m = 3 : n = 5
MsgBox(“Вводите массив”)
For i = 1 To m
For j = 1 To n
a(i, j) = InputBox("a(" & i & "," & j & ")")
Next j
Next i
For j = 1 To n
maximum = a(1,j)
nomer = 1
For i = 1 To m
If a(i,j) > maximum Then
maximum = a(i,j)
nomer = i
End If
Next i
x = a(1,j)
a(1,j) = a(nomer,j)
a(nomer,j) = x
Next j
For i = 1 To m
For j = 1 To n
MsgBox("a(" & i & "," & j & ") = " & a(i,j))
Next j
Next i
End Sub
```

Поиск максимального элемента столбца выполняется аналогично предыдущему примеру. В переменной **maximum** запоминается максимальный элемент столбца, а в переменной **nomer** – номер этого элемента (другими словами, номер строки, в которой находится максимальный элемент данного столбца). В следующей группе операторов первый и максимальный элемент **j**-го столбца меняются местами:

```
x = a(1,j)
a(1,j) = a(nomer,j)
a(nomer,j) = x
```

Здесь **x** – вспомогательная переменная, используемая для промежуточного хранения первого элемента столбца.

Пример 4.14 – В программу вводится двумерный массив. По каждой строке вычисляется среднее значение. Составляется массив из средних значений, превышающих некоторую заданную величину (эта величина вводится с клавиатуры и обозначается как переменная **predel**). Этот массив выводится на экран.

```
Sub primer4_14()
Dim a() As Single, sred() As Single, predel As Single, m As Byte, n As Byte
m = InputBox("Введите количество строк ")
n = InputBox("Введите количество столбцов ")
ReDim a(1 To m, 1 To n), sred (1 To m)
For i = 1 To m
For j = 1 To n
a(i, j) = InputBox("a(" & i & "," & j & ")")
Next j
Next i
predel = InputBox(“Введите минимально допустимое среднее”)
k = 0
For i = 1 To m
sum = 0
For j = 1 To n
sum = sum + a(i,j)
Next j
sred_stroki = sum/n
If sred_stroki > predel Then
k=k+1          ‘Вычисляется номер нового элемента массива sred
sred(k) = sred_stroki
End If
Next i
For i = 1 To k
MsgBox (sred(i))
Next i
End Sub
```

Здесь **sred** – массив из средних значений, превышающих заданную величину **predel**. Каждый раз, когда среднее значение строки превышает переменную **predel** (т. е. выполняется условие **sred_stroki > predel**), вычисляется новое значение переменной **k** – номер очередного элемента массива **sred**.

Для этого переменная **k** увеличивается на единицу (**k=k+1**). Затем создается новый (*k*-й) элемент массива **sred**: **sred(k) = sred_stroki.**

Примечание – Здесь для переменной **k** обнуление необязательно, так как все переменные по умолчанию сначала равны нулю. Оператор **k = 0** приведен в программе только для наглядности. Для переменной **sum** обнуление обязательно, так как она вычисляется для каждой строки заново.

Пример 4.15 – В программу вводится двумерный массив, и по каждой его строке вычисляется среднее значение. Составляется новый двумерный массив (обозначенный в программе как **nov**): в него включаются строки исходного массива, для которых среднее значение превышает некоторую заданную величину (переменная **predel**).

```
Sub primer4_15()
Dim a() As Single, nov() As Single, predel As Single, m As Byte, n As Byte
m = InputBox("Введите количество строк ")
n = InputBox("Введите количество столбцов ")
ReDim a(1 To m, 1 To n), nov(1 To m, 1 To n)
…       ‘Ввод исходного массива (имя массива – a), как в примере 4.14
predel = InputBox(“Введите минимально допустимое среднее”)
k = 0
For i = 1 To m
sum = 0
For j = 1 To n
sum = sum + a(i,j)
Next j
sred_stroki = sum/n
If sred_stroki > predel Then
k=k+1           ‘Вычисляется номер новой строки массива nov
For j = 1 To n
nov(k,j) = a(i,j)          ‘i-я строка исходного массива, в которой среднее превышает
Next j                     ‘предел, копируется в очередную (k-ю) строку нового массива
End If
Next i

MsgBox("Составлен новый массив”)
For i = 1 To k
For j = 1 To n
MsgBox("nov(" & i & "," & j & ") = " & nov(i,j))
Next j
Next i
End Sub
```

4.8 Цикл ПОКА. Оператор Do While

Кроме цикла ДО, который повторяется заданное количество раз, применяется также цикл ПОКА, который повторяется до тех пор, пока верно некоторое заданное условие (или, наоборот, пока некоторое условие не станет верным). В VBA имеется несколько операторов для реализации цикла ПОКА. Простейший из них – оператор **Do While**:

Do While условие
операторы, выполняемые в цикле
Loop

Цикл повторяется, пока заданное в нем условие верно, и прекращается, когда оно становится неверным. Условие проверяется перед началом цикла. Если условие сразу оказывается неверным, то цикл не выполняется ни разу.

Примечание – В VBA имеется также оператор цикла **Do Until**, по структуре аналогичный оператору **Do While**. Цикл, заданный оператором **Do Until**, повторяется, пока заданное в нем условие неверно, и прекращается, когда оно становится верным.

Пример 4.16 – В программе вводится одномерный массив из восьми чисел, и выполняется суммирование его элементов (начиная с первого), пока сумма не превысит 20.

```
Sub primer4_16a()
Dim a(1 To 8) As Integer
MsgBox(“Вводите массив”)
For i = 1 To 8
a(i) = InputBox("a(" & i & ")")
Next i
sum = 0 : i = 0
Do While sum <= 20
i = i + 1
sum = sum + a(i)
Loop
MsgBox ("Сумма достигнута на " & i & "-м элементе и равна" & sum)
End Sub
```

Здесь переменная **sum** сначала равна нулю, поэтому условие **sum<=20** верно. Выполняются операторы цикла:

```
i = i + 1
sum = sum + a(i)
```

Таким образом, переменная **sum** становится равной первому элементу массива **a**. Если он превышает 20, то цикл завершается, так как условие **sum<=20** оказывается неверным. Выполняется оператор, следующий за циклом, т. е. выводится сообщение о найденной сумме. Если же элемент **a(1)** не превышает 20, то условие **sum<=20** остается верным, и цикл повторяется снова: переменная **i** принимает значение 2, и к переменной **sum** прибавляется величина **a(2)**. Цикл повторяется, пока переменная **sum** не превысит 20.

Следует обратить внимание, что рассмотренная программа не вполне правильная: в ней не предусмотрено, что сумма элементов массива может оказаться меньше 20. В этом случае программа завершится с выдачей сообщения об ошибке, когда переменная **i** превысит объявленное количество элементов массива (в данном примере оно равно 8). Чтобы исправить эту ошибку, можно изменить программу, как показано ниже.

```
Sub primer4_16b()
Dim a(1 To 8) As Integer
MsgBox ("Вводите массив")
For i = 1 To 8
a(i) = InputBox("a(" & i & ")")
Next i
sum = 0: i = 0
Do While (sum <= 20) And (i < 8)
i = i + 1
sum = sum + a(i)
Loop
If sum > 20 Then
MsgBox ("Сумма достигнута на " & i & "- м элементе и равна " & Sum)
Else
MsgBox ("Сумма не достигнута")
End If
End Sub
```

Здесь цикл **Do While** выполняется до тех пор, пока верны два условия: переменная **sum** не превышает 20, и переменная **i** меньше 8. Если хотя бы одно из этих условий нарушается, то цикл прекращается. Поэтому, если в ходе выполнения цикла переменная **sum** достигает значения, превышающего 20, то цикл прекращается, так как нарушается условие **sum <= 20**. Если же оказывается, что сумма всех элементов массива меньше 20, то цикл прекращается из-за нарушения условия **i < 8** (так как переменная **i** увеличивается на единицу в каждом цикле и достигает значения 8, когда вычислена сумма всех элементов массива). После прекращения цикла в операторе **If** проверяется достигнутое значение переменной **sum**, и выводится соответствующее сообщение.

Пример 4.17 – Вычислить сумму ряда $\sum_{n=0}^{\infty} \frac{(-1)^n}{n!(2n+1)}$ с точностью до 0,0001.

```
Sub primer4_17()
n=0 : element = 1 : sum = 0
Do While Abs (element) >= 0.0001
sum = sum + element
n=n+1
factorial = 1
For i=1 To n
factorial = factorial * i
Next i
element = ((-1)^n)/(factorial*(2*n+1))
Loop
MsgBox(“Сумма ” & sum & “достигнута при n = ” & n-1)
End Sub
```

Здесь **Abs** – встроенная (стандартная) функция языка VBA, вычисляющая модуль. Для вычисления факториала использован цикл **For**, рассмотренный ранее. Вычитание при выводе результата (**n–1**) требуется из-за того, что в цикле переменная **n** увеличивается, вычисляется очередной (**n**-й) элемент

ряда, и только после этого проверяется условие окончания цикла (превышает ли вычисленный элемент ряда заданную величину 0,0001). Цикл завершится, когда вычисленный элемент ряда окажется меньше 0,0001. Этот элемент уже не прибавляется к вычисляемой сумме. Поэтому на момент окончания цикла переменная **n** будет иметь значение, на единицу превышающее количество элементов вычисленной суммы.

4.8 Подпрограммы

Во многих случаях удобно выделить часть программы в отдельную процедуру (подпрограмму) и вызывать ее для выполнения из другой процедуры. Подпрограмма может иметь входные и выходные параметры. Входные параметры – это величины, передаваемые в подпрограмму из другой (вызывающей) процедуры и являющиеся для вызываемой подпрограммы «исходными данными». Этими величинами могут быть как переменные, так и константы (например конкретные числа). Выходные параметры – это переменные, являющиеся результатами выполнения вызываемой подпрограммы.

Пример 4.18 – Требуется разработать программу для составления массива из элементов арифметической прогрессии. Первый элемент прогрессии a_1, шаг прогрессии d и необходимое количество элементов n указываются пользователем. Составление массива требуется реализовать как подпрограмму.

```
Sub primer4_18()
Dim a1 As Single, d As Single, n As Byte, a() as Single
a1 = InputBox("Введите первый элемент прогрессии ")
d = InputBox("Введите шаг прогрессии ")
n = InputBox("Укажите, сколько элементов требуется ")
ReDim a (1 To n)       ‘a – массив из элементов прогрессии, который требуется составить
Call arif_progr(a1, d, n, a)
For i = 1 To n
MsgBox(i & “-й элемент прогрессии равен “ & a(i))
Next i
End Sub

Sub arif_progr(nach, shag, kol, elem)
elem(1) = nach
For i=2 To kol
elem(i) = elem(i-1)+shag
Next i
End Sub
```

Здесь подпрограмма **primer4_18** является вызывающей (или основной) подпрограммой, а **arif_progr** – вызываемой подпрограммой. Таким образом, данная программа в целом состоит из двух подпрограмм. При ее запуске курсор должен находиться в пределах текста *основной* подпрограммы.

В начале основной подпрограммы выполняется ввод данных, а также объявление массива **a**, который должен быть составлен в результате выполнения программы.

Оператор **Call** – вызов подпрограммы. В данном примере он имеет следующий вид: **Call arif_progr(a1, d, n, a)**. Здесь **arif_progr** – имя вызываемой подпрограммы. Переменные **a1**, **d** и **n** – входные параметры, так как при вызове подпрограммы их значения уже заданы. Переменная **a** – выходной параметр, так как она определяется в результате выполнения подпрограммы.

Следующая строка представляет собой заголовок подпрограммы:

```
Sub arif_progr(nach, shag, kol, elem)
```

Количество параметров в вызове подпрограммы (т. е. в операторе **Call**) и в ее заголовке должно совпадать, и порядок параметров в этих операторах должен соответствовать друг другу.

В данном примере при вызове подпрограммы **arif_progr** переменная **nach** получает значение переменной **a1**, переменная **shag** – значение **d**, переменная **kol** – значение **n**. Переменная **elem** получает значение переменной **a**, т. е. она будет представлять собой массив из **n** элементов. Так как этим элементам в основной программе не было присвоено никакого значения, они равны нулю.

Затем выполняется подпрограмма **arif_progr.** В результате ее выполнения вычисляются элементы массива **elem**. Эти же значения получают и соответствующие элементы массива **a** в основной программе.

По окончании выполнения вызванной подпрограммы происходит возврат в основную подпрограмму. Выполняется оператор, следующий за оператором **Call**. В данном случае это оператор цикла, в котором элементы полученного массива выводятся на экран.

Примечание – В данном примере имена переменных в основной программе (**a1**, **d**, **n** и **a**) отличаются от имен соответствующих переменных в вызываемой подпрограмме (**nach**, **shag**, **kol** и **elem**). Эти имена могут и совпадать.

В качестве входных параметров могут использоваться не только переменные, но и константы. Пусть, например, в рассматриваемой задаче не требуется запрашивать у пользователя начальный элемент прогрессии и ее шаг: задано, что $a_1 = 5$, $d = 2$. У пользователя запрашивается только количество элементов прогрессии, которые требуется вычислить. Тогда вызов подпрограммы мог бы иметь следующий вид: **Call arif_progr(5, 2, n, a)**. Никаких изменений в самой подпрограмме **arif_progr** при этом не требуется. При вызове подпрограммы переменная **nach** получит значение 5, а переменная **shag** – значение 2. Все остальные действия выполняются точно так же, как описано выше.

Входные и выходные параметры подпрограммы могут и совпадать: другими словами, величины, передаваемые в подпрограмму, могут изменяться в ней и возвращаться измененными в вызывающую подпрограмму. Это показано в следующем примере.

Пример 4.19 – Требуется разработать подпрограмму, которая бы меняла местами два столбца в двумерном массиве.

```
Private m As Byte
Sub primer4_19()
```

```
Dim a() As Single, k1 As Byte, k2 As Byte, n As Byte
‘a – двумерный массив из m строк и n столбцов
…        ‘Объявление и ввод массива (см. примеры, рассмотренные выше).
 ‘k1 и k2 – номера столбцов, которые требуется поменять местами
k1 = InputBox("Введите первый номер столбца ")
k2 = InputBox("Введите второй номер столбца ")
Call obmen_stolbcov(a, k1, k2)
For i = 1 To m
For j = 1 To n
MsgBox("a(" & i & "," & j & ") = " & a(i,j))  ‘Вывод измененного массива на экран
Next j
Next i
End Sub

Sub obmen_stolbcov(massiv, k1, k2)
For i=1 To m
x = massiv(i, k1)
massiv(i, k1) = massiv(i, k2)
massiv(i, k2) = x
Next i
End Sub
```

Здесь переменная **m** (количество строк в массиве) используется в обеих процедурах (**primer4_19** и **obmen_stolbcov**), поэтому ее требуется объявить как переменную уровня модуля, т. е. в операторе **Private** перед текстом процедур.

В подпрограмму **obmen_stolbcov** передаются три входных параметра: массив **a** (в подпрограмме он получает имя **massiv**) и номера столбцов **k1** и **k2** (в подпрограмме они получают такие же имена – **k1** и **k2**). В подпрограмме массив изменяется: столбцы с номерами **k1** и **k2** меняются местами. Измененный массив возвращается в основную подпрограмму.

Примечание – Вместо того, чтобы объявлять переменную **m** в операторе **Private** (т. е. как переменную уровня модуля), можно было передать ее в процедуру **obmen_stolbcov** в качестве одного из параметров.

4.9 Функции

Функция, как и подпрограмма – это отдельная процедура, вызываемая из другой процедуры. Отличие функции от подпрограммы состоит в том, что в результате выполнения функции определяется значение одной переменной, в то время как подпрограмма может определять значения нескольких переменных. Имя переменной, определяемой в результате выполнения функции, должно совпадать с именем самой функции.

Функция, как и подпрограмма, имеет параметры, т. е. величины, передаваемые в функцию из вызывающей процедуры. Этими величинами могут быть как переменные, так и константы. Если параметрами являются переменные, то они могут изменяться при выполнении функции, но такие изменения, как правило, нежелательны, так как усложняют понимание программы.

Следует обратить внимание, что способы вызова подпрограммы и функции существенно различаются. Подпрограмма вызывается с помощью специ-

ального оператора **Call**. Функция обычно вызывается в правой части оператора присваивания.

Пример 4.20 – Требуется разработать функцию для вычисления среднего значения заданного столбца в двумерном массиве.

```
Private m As Byte
Sub primer4_20()
Dim a() As Single, n As Byte, k As Byte
‘a – двумерный массив из m строк и n столбцов
…       ‘Объявление и ввод массива (см. примеры, рассмотренные выше).
k = InputBox("Введите номер столбца, для которого требуется вычислить среднее ")
s = sred_stolb(a, k)
MsgBox ("Среднее по " & k & “-му столбцу равно ” & s)
End Sub

Function sred_stolb(massiv, nomer)
sum = 0
For i = 1 To m
sum = sum + massiv(i, nomer)
Next i
sred_stolb = sum / m
End Function
```

Здесь **sred_stolb** – функция. Она вызывается в операторе присваивания **s = sred_stolb(a, k)**. Таким образом, переменные **a** и **k** (т. е. массив и номер столбца, для которого требуется вычислить среднее) являются входными параметрами функции **sred_stolb**. В самой функции **sred_stolb** значения этих переменных присваиваются переменным **massiv** и **nomer** соответственно (можно было использовать и любые другие имена переменных). Функция вычисляет среднее по столбцу с номером **nomer**. Результат присваивается переменной **sred_stolb**. Следует обратить внимание, что имя этой переменной обязательно должно совпадать с именем самой функции. По окончании выполнения функции **sred_stolb** происходит возврат в основную программу: выполняется оператор, где была вызвана функция, т. е. оператор **s = sred_stolb(a, k)**. Таким образом, значение функции **sred_stolb** присваивается переменной **s**. Затем эта переменная выводится на экран оператором **MsgBox.**

Примечание – В данном примере, как и в примере 4.19, переменная **m** объявлена как переменная уровня модуля, чтобы ее можно было использовать во всем модуле (т. е. и в подпрограмме **primer4_20**, и в функции **sred_stolb**). Вместо такого объявления можно было передать эту переменную в функцию **sred_stolb** в качестве одного из параметров.

Пример 4.21 – Требуется разработать программу для решения следующей задачи: найти в двумерном массиве столбец с максимальным средним значением и поменять его местами с первым столбцом.

Для решения этой задачи средние значения по столбцам массива будут вычисляться с использованием функции, разработанной в примере 4.20. Столбец с максимальным средним значением будет определяться в основной программе. Чтобы поменять местами первый столбец массива и столбец с

максимальным средним значением, воспользуемся процедурой, разработанной в примере 4.19.

```
Private m As Byte
Sub primer4_21()
Dim a() As Single, n As Byte
‘a – двумерный массив из m строк и n столбцов
…       ‘Объявление и ввод массива.
max_sred = -10000
For j=1 To n
s = sred_stolb(a, j)
If s > max_sred Then
jmax = j
max_sred = s
End If
Next j
call obmen_stolbcov(a, 1, jmax)
MsgBox(“Измененный массив”)
For i = 1 To m
For j = 1 To n
MsgBox("a(" & i & "," & j & ") = " & a(i,j))  ‘Вывод измененного массива на экран
Next j
Next i
End Sub

Function sred_stolb(massiv, j)
…       ‘См. пример 4.20.
End Function

Sub obmen_stolbcov(massiv, k1, k2)
…       ‘См. пример 4.19.
End Sub
```

Здесь переменной **max_sred** сначала присваивается очень большое отрицательное число. Затем в этой переменной сохраняется максимальное из вычисленных средних значений по столбцам. С этой целью перебираются все столбцы, для каждого столбца вычисляется среднее (с помощью функции **sred_stolb**), оно сравнивается с текущим значением переменной **max_sred**. Если выполняется условие **s > max_sred** (найденное среднее превышает текущее значение переменной **max_sred**), то значение среднего сохраняется в переменной **max_sred**, а номер столбца, где найдено это среднее – в переменной **jmax**. В результате в переменной **jmax** сохраняется номер столбца с максимальным средним. В подпрограмме **obmen_stolbcov** столбцы с номерами 1 и **jmax** меняются местами.

4.10 Область видимости процедур

В начале заголовка процедуры, т. е. перед словом **Sub** или **Function**, может указываться слово **Private** или **Public**, задающее область видимости процедуры:

– **Private**: процедура может вызываться только из процедур того модуля, в котором она находится;

– **Public**: процедура может вызываться из процедур как того модуля, в котором она находится, так и других модулей.

Если область видимости не указана, то по умолчанию предполагается видимость процедуры во всех модулях (т. е. область видимости, соответствующая слову **Public**). Если программа состоит только из одного модуля, то область видимости процедур этой программы никак не влияет на ее работу.

4.11 Варианты заданий

Вариант 1 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива; ввод некоторого числа **x**; вывод на экран номеров строк массива, где число **x** встречается хотя бы один раз.

Вариант 2 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива; ввод некоторого числа **x**; ввод некоторого целого числа **i**; замена всех чисел в **i**-й строке, превышающих введенное число **x**, на это число **x** (например, если введено **x** = 5, **i** = 2, то требуется во второй строке массива заменить на 5 все числа, превышающие 5); вывод измененного массива на экран.

Вариант 3 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива; ввод некоторого числа **x**; вычисление суммы каждой строки массива; вывод на экран номеров тех строк, суммы которых превышают **x**.

Вариант 4 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива **a**; ввод одномерного числового массива **b** (количество элементов в массиве **b** должно быть равно количеству столбцов в массиве **a**); ввод некоторого целого числа **i**; сравнение каждого элемента **i**-й строки массива **a** с соответствующим элементом массива **b** и подсчет количества случаев, когда элемент массива **a** превышал соответствующий элемент массива **b**; вывод подсчитанного количества случаев на экран.

Вариант 5 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива; ввод некоторого числа **x**; для каждого столбца массива – подсчет и вывод на экран количества вхождений числа **x**.

Вариант 6 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива **a**; ввод одномерного числового массива **b** (количество элементов в массиве **b** должно быть равно количеству столбцов в массиве **a**); в каждой строке массива **a** – замена всех элементов, превышающих соответствующий элемент массива **b**, на этот элемент массива **b**; вывод измененного массива **a** на экран.

Вариант 7 – Составить программу, выполняющую следующие действия: ввод двумерного числового массива **a**; ввод одномерного числового массива **b** (количество элементов в массиве **b** должно быть равно количеству строк в массиве **a**); ввод некоторого числа **j**; сложение каждого элемента **j**-го столбца

массива **a** с соответствующим элементом массива **b**; вывод на экран измененного массива **a.**

Вариант 8 – В программе вводятся два двумерных массива (**a** и **b**) одинаковой размерности. Сравниваются суммы соответствующих строк массивов: сумма первой строки массива **a** – с суммой первой строки массива **b**, сумма второй строки **a** – с суммой второй строки **b**, и т. д. Если сумма строки массива **a** оказывается меньше суммы соответствующей строки массива **b**, то строки меняются местами (строка в массиве **a** заменяется на соответствующую строку из массива **b**, и наоборот). Обмен строками должен быть реализован в виде отдельной процедуры.

Вариант 9 – В программе вводятся два массива: двумерный (обозначим его как **a**) и одномерный (**b**), причем количество элементов в одномерном массиве должно быть равно количеству строк в двумерном массиве. Для каждого элемента массива **b** определяется ближайший к нему элемент в соответствующей строке массива **a** (т. е. элемент, разность с которым минимальна). Поиск ближайшего элемента должен быть реализован в виде отдельной процедуры. Из номеров этих ближайших элементов составляется одномерный массив.

Вариант 10 – В программе вводится двумерный массив. Из него должен составляться новый массив, включающий все строки исходного массива, кроме тех, которые состоят из одинаковых чисел. Проверка строк (чтобы выяснить, состоит ли данная строка из одинаковых чисел) должна быть реализована в виде отдельной процедуры.

Вариант 11 – В программе вводится двумерный массив. В каждой строке определяется максимальное и минимальное число и вычисляется их разность. Строка, где эта разность максимальна, должна меняться местами со строкой, где разность минимальна. Определение минимального и максимального значения в строке должно быть реализовано в виде отдельной процедуры.

Вариант 12 – В программе вводится двумерный массив. Из него составляются два новых массива: в первый из них включаются те строки исходного массива, где нет ни одного отрицательного числа, во второй – те строки, где есть хотя бы одно отрицательное число. Проверка строки на наличие отрицательного числа должна быть реализована в виде отдельной процедуры.

Вариант 13 – В программе вводится двумерный массив. В каждой строке вычисляется среднее значение. Составляется новый массив такой же размерности, что и исходный массив. Строки нового массива составляются из элементов соответствующей строки исходного массива, превосходящих среднее значение по своей строке. Так как количество элементов, отбираемых из каждой строки, оказывается при этом разным, недостающие элементы в конце строк нового массива остаются равными нулю. Вычисление среднего по строке массива должно быть реализовано в виде отдельной процедуры.

Вариант 14 – В программе вводятся два двумерных массива (**a** и **b**). Количество строк в этих массивах должно быть одинаковым. Сравниваются суммы

соответствующих строк массивов: сумма первой строки массива **a** – с суммой первой строки массива **b** сумма второй строки **a** – с суммой второй строки **b**, и т. д. Составляется новый массив из двух строк. В первой строке нового массива содержатся номера тех строк исходных массивов, где сумма строки массива **a** оказалась больше суммы соответствующей строки массива **b**. Во второй строке нового массива содержатся номера всех остальных строк (где сумма строки массива **a** не превысила сумму строки массива **b**). Суммирование строки массива должно быть реализовано в виде отдельной процедуры.

Вариант 15 – В программе вычисляется значение функции *sh x* (гиперболический синус) на основе ее представления в виде ряда Тейлора: $sh\,x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$. Значение переменной *x*, а также требуемая точность вводятся с клавиатуры. Вычисление ряда Тейлора должно быть реализовано в виде функции (вызывающая процедура должна содержать только ввод исходных данных, вызов функции и вывод результата).

Вариант 16 – В программе вводится квадратная матрица (обозначим ее как **a**) и некоторое число ***s***. Элемент матрицы **a(1,1)** сравнивается с числом **s**. Если **a(1,1) > s**, то программа завершает работу. В противном случае вычисляется сумма элементов квадратной матрицы 2x2 из левого верхнего угла матрицы **a**. Если эта сумма снова оказывается меньше **s**, то вычисляется сумма матрицы 3x3, и т. д., пока не будет достигнута сумма, превышающая **s.** Вычисление суммы квадратной матрицы, выделенной из исходной матрицы **a**, должно быть реализовано в виде функции. Программа должна выводить на экран размерность квадратной матрицы, сумма которой превысила **s**. Если число **s** не превышено при суммировании всей матрицы **a**, то должно выводиться сообщение об ошибке.

Вариант 17 – В программе вводится одномерный массив **a**, состоящий из целых чисел, не превышающих 100. Вводятся также два числа: **g** (вводится значение 100) и **k** (некоторое произвольное целое число). В программе подсчитывается количество элементов массива **a**, превышающих величину **g**. Если это количество составляет не меньше **k**, то программа завершает работу; при этом из чисел, превышающих **g**, составляется новый массив. В противном случае величина **g** уменьшается на 0,5, и проверка повторяется. Подсчет количества элементов массива, превышающих заданную границу **g**, должен быть реализован в виде функции.

Вариант 18 – В программе вводится одномерный массив **a**, состоящий из положительных чисел, и некоторое число **s**. Вычисляется среднее значение элементов массива. Если оно оказывается меньше **s**, то программа завершает работу; при этом из ненулевых элементов массива **a** составляется новый массив. В противном случае в массиве определяется минимальное ненулевое число, оно заменяется на ноль, снова вычисляется среднее, и проверка повторяется. Определение минимального элемента массива должно быть реализовано в виде функции.

Вариант 19 – В программе вычисляется интеграл от некоторой функции $y = f(x)$ на отрезке $[a, b]$ методом левых прямоугольников. Для этого отрезок $[a, b]$ разбивается на k участков одинаковой длины $d = (b–a)/k$, т. е. определяются точки $x_0 = a$, $x_1 = a + d$, $x_2 = a + 2d$, …, $x_{k–1} = a + (k–1)d$. Вычисляются значения функции y в этих точках: $y_0 = f(x_0)$, $y_1 = f(x_1)$, …, $y_{k–1} = f(x_{k–1})$. Оценка интеграла вычисляется как $(y_0 + y_1 + \ldots + y_{k–1})\cdot d$. Затем k увеличивается на единицу, и вычисление повторяется. Процесс завершается, когда разность оценок интеграла при двух последовательных значениях k оказывается меньше заданной точности. Требуется разработать программу для вычисления интеграла от функции $y = e^{x^2}$ этим методом. В программе должны запрашиваться границы отрезка (a и b), а также требуемая точность. Начальное значение k задается в программе (например, $k = 2$). Непосредственно алгоритм вычисления интеграла должен быть реализован в виде функции.

Вариант 20 – В программе определяется решение некоторого уравнения $f(x) = 0$ на отрезке $[a, b]$ методом дихотомии. Для применения этого метода необходимо выбрать точки a и b таким образом, чтобы величины $f(a)$ и $f(b)$ имели разные знаки. Определяется середина этого отрезка $c = (b+a)/2$, и в этой точке вычисляется значение $f(c)$. Если это значение отличается от нуля на величину, не превышающую заданной точности, то задача решена (c – решение). В противном случае отрезок $[a, b]$ разбивается пополам. Выбирается тот из двух полученных отрезков, на концах которого исследуемая функция $f(x)$ принимает разные знаки, и для этого отрезка процесс повторяется. Требуется разработать программу для решения уравнения $\sin x / x = 0$ этим методом. В программе должны запрашиваться границы отрезка (a и b), а также требуемая точность. Непосредственно алгоритм решения уравнения должен быть реализован в виде функции.

Вариант 21 – В программе вводятся два одномерных массива (обозначим их как **a** и **b**), состоящих из положительных чисел, причем известно, что сумма массива **a** меньше суммы массива **b** (если это не так, то должно выводиться сообщение об ошибке). Составляется новый одномерный массив **c**. Сначала в него включаются все элементы массива **a**; затем к нему добавляются элементы массива **b**, начиная с **b(1)**, до тех пор, пока сумма массива **c** не превысит сумму оставшихся элементов массива **b**. Суммирование элементов массива должно быть реализовано в виде функции. На экран выводится полученный массив **c**.

4.12 Порядок выполнения работы

Для выполнения данной работы рекомендуется разработать и отладить три программы: по одной согласно вариантам 1–7, 8–14 и 15–21.

Примечание – Предполагается, что для решения задач из вариантов 15–21 будет использоваться (наряду с другими конструкциями VBA) оператор цикла **DO WHILE**.

4.13 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать:

– титульный лист;
– цель работы;
– листинги разработанных программ;
– копии экранов, иллюстрирующие выполнение программ (ввод исходных данных, вывод результатов);
– пояснения по использованию операторов цикла, массивов, процедур и функций (в виде отдельного текста или комментариев к программам).

4.14 Контрольные вопросы

1 Структура программы на VBA.
2 Объявление переменных и констант в VBA.
3 Область видимости переменных в VBA.
4 Объявление массивов в VBA. Динамические массивы.
5 Циклы ДО в VBA.
6 Циклы ПОКА в VBA.
7 Процедуры в VBA: подпрограммы.
8 Процедуры в VBA: функции.

Ответы на все контрольные вопросы должны сопровождаться примерами в виде фрагментов программ.

ЛАБОРАТОРНАЯ РАБОТА №5

ОПЕРАЦИИ С ЯЧЕЙКАМИ И РАБОЧИМИ ЛИСТАМИ MS EXCEL В ПРОГРАММАХ НА VBA

Цель работы – Освоение разработки программ на VBA для обработки данных, размещенных в рабочих листах табличного процессора MS Excel.

5.1 Основные способы ссылок на ячейки рабочего листа Excel

Простейшие способы ссылки на значения отдельных ячеек – **Range(**ячейка**).Value** или **Cells**(строка, столбец).**Value**.

Пример 5.1 – Программа считывает число из ячейки A2, извлекает из него квадратный корень и выводит результат в ячейку A4.

```
Sub primer5_1()
x = Range("A2").Value
y = Sqr(x)
Range("A4").Value = y
End Sub
```

Это же можно реализовать и многими другими способами, например:

```
Sub primer5_1()
x = Cells(2, 1).Value
y = Sqr(x)
Cells(4, 1).Value = y
End Sub
```

или

```
Sub primer5_1()
Cells(4, 1).Value = Sqr((Cells(2, 1).Value))
End Sub
```

В качестве адресов ячеек могут использоваться не только конкретные значения, но и переменные. Например, в операторе **x = Cells(i,j).Value** переменная **x** получит значение, равное содержимому ячейки, расположенной в **i**-й строке и **j**-м столбце текущего рабочего листа. При этом переменные **i** и **j** должны иметь некоторые значения, причем они должны представлять собой целые положительные числа (если это не так, то программа прерывается с выводом сообщения об ошибке).

Можно также ссылаться на ячейки, отсчитывая их не от левого верхнего угла рабочего листа (т. е. не от ячейки A1), а *от некоторой заданной ячейки*. Например, ссылка **Range("E7").Cells(3,2).Value** означает ссылку на значение ячейки F9, так как, если отсчитывать ячейки от E7, то ячейка F9 находится в третьей строке и втором столбце от заданной ячейки. Эту же ссылку можно указать как **Cells(7,5).Cells(3,2).Value**. Вместо конкретных номеров строк и столбцов могут указываться переменные. Например, ссылка

Cells(7,5).Cells(i,j).Value – этот ссылка на ячейку, расположенную в **i**-й строке и **j**-м столбце относительно ячейки E7.

Ссылки на ячейки могут также указываться относительно *ячейки, выделенной с помощью мыши*. Например, ссылка **Selection.Cells(3,2).Value** означает ссылку на значение ячейки, находящейся в третьей строке и втором столбце от выделенной ячейки. Если при этом выделено несколько ячеек, то ссылка определяется относительно *левого верхнего угла* выделенного диапазона.

Во многих случаях удобно указать диапазон ячеек, связав с ним некоторую переменную, а затем ссылаться на отдельные ячейки этого диапазона. Для связывания переменных с диапазонами ячеек используется оператор **Set.** Основные способы такого связывания следующие:

– ссылка на прямоугольный диапазон ячеек, заданный явно:

Set переменная = **Range**(левый_верхний_угол, правый_нижний_угол)

– ссылка на диапазон ячеек, выделенный на рабочем листе с помощью мыши:

Set переменная = **Selection**

– ссылка на прямоугольный диапазон ячеек, заполненный данными (например числами) и отделенный от других данных хотя бы одной свободной строкой и столбцом:

Set переменная = **ячейка.CurrentRegion**

В последнем случае **ячейка** – это ссылка (в форме **Range**, **Cells** или **Selection**) на любую ячейку в заполненном диапазоне.

Примеры операторов **Set**, связывающих переменные с диапазонами ячеек, приведены в таблице 5.1.

Таблица 5.1 – Примеры описания диапазонов ячеек в операторе Set

Пример	Описание
Set d = Range(“A2:E8”)	Переменная **d** связывается с диапазоном ячеек A2:E8.
Set d = Range(Cells(2,1),Cells(8,5))	То же, что и в предыдущем примере.
Set x = Selection	Переменная **x** связывается с диапазоном ячеек, выделенным с помощью мыши (это может быть как одна ячейка, так и несколько).
Set y = Range(“E2”).CurrentRegion	Переменная **y** связывается с заполненным диапазоном ячеек, содержащим ячейку E2.
Set y = Cells(2,5).CurrentRegion	То же, что и в предыдущем примере (так как ячейка во второй строке и пятом столбце – это ячейка E2).
Set z = Selection.CurrentRegion	Переменная **z** связывается с заполненным диапазоном ячеек, содержащим выделенную ячейку.

Пусть переменная связана с некоторым диапазоном ячеек с помощью оператора **Set**. Тогда ссылка **переменная.Cells(i,j).Value** – это ссылка на ячейку, расположенную в **i**-й строке и **j**-м столбце относительно *левого верхнего угла* диапазона, с которым связана указанная переменная.

Пример 5.2 – Пусть имеется следующий фрагмент программы:

```
Set d = Range(“A2:E7”)
Set f = Selection
x = d.Cells(1,3).Value
y = d.Cells(1,8).Value
z = Cells(1,3).Value
k = f.Cells(2,4).Value
```

В данном случае переменной **x** будет присвоено значение ячейки C2, так как эта ячейка находится в первой строке и третьем столбце относительно диапазона, связанного с переменной **d**. Аналогичным образом, переменная **y** получит значение ячейки H2. Переменная **z** получит значение ячейки C1, т. е. ячейки из первой строки и третьего столбца рабочего листа, так как ссылка на другой диапазон ячеек (например на переменную **d**) в данном случае отсутствует, и ячейка определяется относительно рабочего листа в целом. Переменной **k** будет присвоено значение ячейки из второй строки и четвертого столбца относительно диапазона, выделенного на рабочем листе с помощью мыши (точнее, левого верхнего угла этого диапазона).

Для диапазона, связанного с переменной, легко определить его размеры, т. е. количество строк и столбцов. Для этого используются стандартные свойства **Rows.Count** и **Columns.Count**.

Пусть к фрагменту программы, приведенному в примере 5.2, добавлены следующие строки:

```
m = d.Rows.Count
n = d.Columns.Count
```

Переменная **m** в этом случае получит значение 6 (количество строк в заданном диапазоне), а переменная **n** – значение 5 (количество столбцов).

5.2 Примеры обработки данных в ячейках рабочего листа MS Excel

Пример 5.3 – Некоторый прямоугольный диапазон ячеек на рабочем листе Excel заполнен числами. Требуется вычислить средние значения каждой строки этого диапазона ячеек и вывести их справа от диапазона. Если, например, числами заполнен диапазон B1:E5, то требуется сначала вычислить среднее значение ячеек B1:E1 и вывести его в ячейку F1, затем вычислить среднее значение ячеек B2:E2 и вывести его в ячейку F2 и т. д.

Рассмотрим несколько случаев решения этой задачи в зависимости от того, как задан диапазон ячеек, для которого вычисляются средние значения строк:

а) вычисления выполняются для диапазона ячеек B1:E5:

```
Sub primer5_3a_1()
Set d = Range("B1:E5")
m = d.Rows.Count
n = d.Columns.Count
For i = 1 To m
sum = 0
For j = 1 To n
```

```
sum = sum + d.Cells(i, j).Value
Next j
srednee = sum / n
d.Cells(i, n + 1).Value = srednee
Next i
End Sub
```

Здесь переменная **d** связывается с диапазоном ячеек B1:E5. Переменные **m** и **n** получают значения, равные количеству строк и столбцов этого диапазона. Затем вычисляется среднее для каждой строки этого диапазона. Важно обратить внимание, что ссылка **Cells(i, j)** – это ссылка на ячейку с номером строки **i** и номером столбца **j**, причем номера строк и столбцов отсчитываются от левого верхнего угла заданного диапазона (в данном случае – от ячейки B1).

Для вычисления среднего значения каждой строки используются вложенные циклы **For** (см. лабораторную работу 4).

В операторе **d.Cells(i, n + 1).Value = srednee** вычисленное среднее значение выводится в ячейку с номером строки **i** и номером столбца **n + 1** относительно диапазона **d**, т. е. в столбец справа от этого диапазона.

Примечание – Если бы в программе отсутствовал оператор **Set d = Range("B1:E5")**, то для ссылки на диапазон ячеек каждый раз требовалось бы указывать его. Например, для определения количества строк потребовалось бы указать: **m = Range("B1:E5").Rows.Count**.

Конечно, рассмотренную задачу можно было решить и многими другими способами. Например, можно было воспользоваться ссылкой на ячейки в форме **Cells**:

```
Sub primer5_3a_1()
Set d = Range(Cells(1, 2), Cells(5, 5))
…
```

Здесь вместо обозначения ячейки B1 использовано обозначение **Cells(1, 2)**, а вместо E5 – **Cells(5, 5)**.

Рассмотрим еще один способ решения рассмотренной задачи: содержимое ячеек вводится в массив, который затем обрабатывается.

```
Sub primer5_3a_2()
Dim a() As Single, srednie() As Single
Set d = Range("B1:E5")
m = d.Rows.Count
n = d.Columns.Count
ReDim a(1 To m, 1 To n), srednie (1 To m)
For i = 1 To m
For j = 1 To n
a(i, j) = d.Cells(i, j).Value
Next j
Next i

For i = 1 To m
sum = 0
For j = 1 To n
```

```
sum = sum + a(i, j)
Next j
srednie(i) = sum / n
Next i

For i = 1 To m
d.Cells(i, n + 1).Value = srednie(i)
Next i
End Sub
```

Здесь значения ячеек диапазона B1:E5 считываются в двумерный массив **a**. Затем вычисляются средние значения строк этого массива. Эти средние значения сохраняются в одномерном массиве **srednie**. Элементы этого массива затем выводятся в ячейки справа от диапазона B1:E5;

б) вычисления выполняются для произвольного диапазона ячеек, выделенного с помощью мыши.

Программа для решения этой задачи отличается от приведенной в примере 5.3, а) только тем, что оператор **Set d = Range("B1:E5")** требуется заменить на **Set d = Selection**. Здесь переменной **d** присваивается диапазон ячеек, выделенный с помощью мыши. В операторах **m = d.Rows.Count** и **n = d.Columns.Count** переменные **m** и **n** получают значения, равные количеству строк и столбцов этого диапазона. Дальнейшие действия выполняются аналогично примерам, рассмотренным выше;

в) вычисления выполняются для произвольного прямоугольного диапазона ячеек. Известно, что одна из ячеек этого диапазона – B1.

В программе, приведенной в примере 5.3, а), требуется заменить оператор **Set d = Range("B1:E5")** на **Set d = Range("B1").CurrentRegion**. Здесь переменной **d** присваивается заполненный данными прямоугольный диапазон ячеек, одна из которых – B1;

г) вычисления выполняются для произвольного прямоугольного диапазона ячеек, одна из которых выделена с помощью мыши.

В программе, приведенной в примере 5.3, а), требуется заменить оператор **Set d = Range("B1:E5")** на **Set d = Selection.CurrentRegion**.

Пример 5.4 – На рабочем листе Excel в столбце C, начиная с ячейки C2 (т. е. в ячейках C2, C3 и т. д.), введены фамилии студентов, а в столбцах D, E, F, G – их оценки по четырем предметам. Требуется получить в столбце L (начиная с ячейки L1) список студентов, имеющих средний балл не ниже минимально допустимого. В столбце M рядом с фамилиями студентов должны выводиться их средние баллы. Минимально допустимый средний балл вводится с клавиатуры.

```
Sub primer5_4()
Dim min_ball As Single
min_ball = InputBox("Введите минимально допустимый средний балл: ")
Set d = Range("C2").CurrentRegion
```

```
m = d.Rows.Count
n = d.Columns.Count
k = 0
For i = 1 To m
sum = 0
For j = 2 To n
sum = sum + d.Cells(i, j).Value
Next j
srednee = sum / (n-1)
If srednee >= min_ball Then
k = k + 1
Cells(k, 12).Value = d.Cells(i,1).Value
Cells(k, 13).Value = srednee
End If
Next i
End Sub
```

Здесь в операторе **Set d = Range("C2").CurrentRegion** переменной **d** присваивается диапазон заполненных ячеек, заполненных данными (фамилиями и оценками); одна из этих ячеек – C2. Переменная **m** получает значение – количество строк диапазона **d** (в данном примере – количество студентов, для которых введены данные). Переменная **n** – количество столбцов диапазона **d**. Если данные введены в соответствии с постановкой задачи, то переменная **n** должна получить значение 5, так как диапазон данных в этом примере содержит пять столбцов: в столбце C – фамилии студентов, в столбцах D–G – их оценки.

Цикл **For i = 1 To m** предназначен для перебора строк (каждая строка содержит данные об одном студенте). Для каждого студента вычисляется средний балл. Цикл **For j = 2 To n** предназначен для перебора оценок студента (т. е. столбцов). Начальное значение переменной **j**, используемой в качестве номера столбца, равно двум (а не одному), так как оценки, по которым вычисляется среднее, начинаются со второго столбца в диапазоне данных (в первом столбце находятся фамилии, а не оценки). Величина **d.Cells(i, j).Value** – это значение ячейки с оценкой студента. Следует еще раз обратить внимание, что номера ячеек (значения **i** и **j**) отсчитываются для диапазона **d**, т. е. от ячейки C2.

Если вычисленный средний балл оказывается не ниже заданной минимальной величины (т. е. выполняется условие **srednee >= min_ball**), то фамилия студента выводится в столбец L (в 12-й столбец рабочего листа), а его средний балл – в столбец M (13-й столбец). Переменная **k** – номер строки рабочего листа, куда выводится фамилия студента; при выводе каждой фамилии она увеличивается на единицу.

Следует обратить внимание на строку **Cells(k,12).Value=d.Cells(i,1).Value**. Здесь **Cells(k, 12).Value** – ячейка, расположенная в **k**-й строке и 12-м столбце (т. е. столбце L) *рабочего листа Excel*. Этой ячейке присваивается значение **d.Cells(i, 1).Value**, т. е. содержимое ячейки, расположенной в **i**-й строке и первом столбце *диапазона* **d** (в этой ячейке находится фамилия студента).

Пример 5.5 – На рабочем листе Excel введены данные о студентах (см. исходные данные для примера 5.4). Требуется удалить данные обо всех студентах, имеющих средний балл ниже минимально допустимого.

```
Sub primer5_5()
Dim min_ball As Single
min_ball = InputBox("Введите минимально допустимый средний балл: ")
Set d = Range("C2").CurrentRegion
m = d.Rows.Count
n = d.Columns.Count
i = 1
Do While i <= m
sum = 0
For j = 2 To n
sum = sum + d.Cells(i, j).Value
Next j
srednee = sum / (n-1)
If srednee < min_ball Then
For k = i to m-1
For j = 1 to n
d.Cells(k, j).Value = d.Cells(k+1, j).Value
Next j
Next k
For j = 1 to n
d.Cells(m, j).Value = Empty
Next j
m = m-1
Else
i = i+1
End If
Loop
End Sub
```

В данном случае количество строк в диапазоне с данными (переменная **m**) не является постоянной величиной, а может изменяться, так как некоторые строки (данные о студентах) будут удаляться. Поэтому для перебора строк использован цикл **Do While**.

Если для **i**-го студента (т. е. в **i**-й строке) вычисленный средний балл оказывается ниже заданной минимальной величины, то данные об этом студенте требуется удалить. Следующая группа операторов смещает все последующие данные, начиная с **i** + 1-й строки, на одну строку выше:

```
For k = i to m-1
For j = 1 to n
d.Cells(k, j).Value = d.Cells(k+1, j).Value
Next j
Next k
```

Другими словами, в **i**-ю строку записываются данные из **i** + 1-й строки, в **i** + 1-ю – из **i** + 2-й строки, …, в **m –** 1-ю – из **m**-й (т. е. из последней) строки.

Затем последняя (**m**-я) строка очищается: ее ячейкам присваивается значение **Empty**.

После этого общее количество строк в диапазоне (или, другими словами, номер последней строки) уменьшается на единицу: **m = m – 1.**

Если же для **i**-го студента средний балл оказался не ниже минимально необходимого, то просто выполняется переход к данным о следующем студенте (к следующей строке диапазона данных): **i = i + 1.**

Примечание – В этом примере показано, как *очистить* ячейку. Если требуется *проверить*, пуста ли ячейка, то используется функция **IsEmpty**. Например, функция **IsEmpty(Cells(1,5).Value)** возвращает значение **True**, если ячейка E1 пуста, и значение **False**, если в этой ячейке есть какая-либо величина.

Пример 5.6 – На рабочем листе Excel введены данные о контрактах на поставку некоторых товаров (по каждому контракту поставляется один товар). Для каждого контракта в столбце A указан его номер, в столбце B – название поставляемого товара, в столбце C – количество товара. Один и тот же товар может поставляться по нескольким контрактам, но цена товара во всех контрактах одинакова. В столбце F перечислены названия товаров, в столбце G – цены на них. Фрагмент исходных данных для задачи приведен на рисунке 5.1. Требуется вычислить и вывести в столбец D стоимость каждого контракта. В конце столбца стоимостей требуется вывести сумму стоимостей всех контрактов.

	A	B	C	D	E	F	G	H
1	101	Насос	20			Двигатель	1200	
2	105	Колесо	100			Колесо	40	
3	108	Панель	60			Насос	60	
4	110	Колесо	40			Панель	25	
5	112	Насос	15					
6	117	Двигатель	12					
7	124	Колесо	30					

Рисунок 5.1 – Фрагмент исходных данных для примера 5.6

```
Sub primer5_6()
Set d1 = Range("A1").CurrentRegion ‘d1 – диапазон ячеек с данными о контрактах
Set d2 = Range("F1").CurrentRegion ‘d2 – диапазон ячеек с данными о товарах
m1 = d1.Rows.Count ‘ Количество контрактов
m2 = d2.Rows.Count ‘ Количество товаров
For i = 1 To m1          ‘ Перебор всех контрактов
nazv = d1.Cells(i,2).Value ‘ Переменной nazv присваивается название товара, поставляемого
                           ‘ по контракту
kol = d1.Cells(i,3).Value   ‘ Переменной kol присваивается количество поставляемого товара
For j = 1 To m2             ‘ Перебор данных о товарах (поиск цены на товар)
If nazv = d2.Cells(j,1).Value then ‘Если товар найден
cena = d2.Cells(j,2).Value          ‘Цена на товар присваивается переменной cena
stoimost = cena*kol                 ‘ Вычисление стоимости контракта
d1.Cells(i,4).Value = stoimost      ‘ Стоимость выводится в i-ю строку, столбец D
```

```
sum_stoimost = sum_stoimost + stoimost     ‘Суммируется стоимость всех контрактов
End If
Next j
Next i
d1.Cells(m1+1,4).Value = sum_stoimost      ‘В строку m1+1 (в конце столбца стоимостей)
                                           ‘выводится сумма стоимостей всех контрактов
End Sub
```

В данном примере используются два диапазона: **d1** – диапазон ячеек с данными о контрактах, **d2** – с данными о товарах и ценах на них. Если считать, что на рисунке 5.1 показаны все данные, то диапазон **d1** содержит ячейки A1:C7, а **d2** – ячейки F1:G4. Переменная **i** используется в качестве номера строки в диапазоне **d1**, переменная **j** – также номер строки, но в диапазоне **d2**.

5.3 Примеры операций с рабочими листами MS Excel

Во многих случаях данные, требующие обработки, размещаются на нескольких рабочих листах MS Excel. Если требуется указать рабочий лист, на котором находится ячейка, то перед ссылкой на нее (т. е. перед словом **Range** или **Cells**) указывается слово **Worksheets("имя_листа")**.

Пример 5.7 – На рабочем листе **Лист1** в столбец введено несколько чисел. Курсор находится в одной из ячеек с этими числами. Требуется скопировать на другой рабочий лист (имя листа – **Лист5**) все числа, превышающие 25. Числа должны копироваться в столбец E, начиная с ячейки E2. Между числами не должно быть пустых ячеек. Другими словами, если из десяти заданных чисел только три числа превысят 25, то они должны быть выведены в ячейки E2, E3 и E4 на рабочем листе **Лист5**.

```
Sub primer5_7()
Set d=Selection.CurrentRegion
m=d.Rows.Count
Set vyvod = Worksheets("Лист5").Range("E2")
j = 0
For i = 1 To m
x = d.Cells(i, 1).Value
If x > 25 Then
j = j + 1
vyvod.Cells(j, 1) = x
End If
Next i
End Sub
```

В операторе **Set d=Selection.CurrentRegion** переменная **d** связывается с заполненным диапазоном ячеек, содержащим выделенную ячейку. Затем в операторе **m=d.Rows.Count** переменной **m** присваивается количество строк в заполненном диапазоне, т. е. количество чисел в столбце. В операторе **Set vyvod = Worksheets("Лист5").Range("E2")** переменная **vyvod** связывается с ячейкой E2 на рабочем листе **Лист5**. Эта переменная будет затем использоваться для ссылок на ячейки, в которые будут выводиться отобранные числа. Переменная **j** – номер

строки на рабочем листе **Лист5** (начиная с ячейки E2), куда будет копироваться число из исходного набора данных. Сначала эта переменная принимается равной нулю. Цикл **For i = 1 To m** используется для перебора ячеек с числами. В операторе **x = d.Cells(i, 1).Value** значение ячейки из **i**-й строки текущего столбца присваивается переменной **x**. При этом не требуется указывать, что ячейка находится на рабочем листе **Лист1**, так как этот рабочий лист – текущий. Если значение **x** превышает 25, то переменная **j** увеличивается на единицу, и переменная **x** выводится в соответствующую ячейку рабочего листа **Лист5**.

Примечание – Можно было не использовать переменную **vyvod**, а указать оператор вывода отобранных чисел в ячейку следующим образом: **Worksheets("Лист5").Range("E2"). Cells(j, 1) = x**.

Пример 5.8 – Пусть в условиях примера 5.7 рабочий лист **Лист5**, на который требуется выводить отобранные числа, еще не существует. Программа в этом случае будет следующей:

```
Sub primer5_8()
Set d=Selection.CurrentRegion
m=d.Rows.Count
Set NewSheet = Worksheets.Add
NewSheet.Name = "Лист5"
Worksheets("Лист1").Activate
Set vyvod = Worksheets("Лист5").Range("E2")
j = 0
…
…                ‘См. пример 5.7
…
End Sub
```

Здесь в операторе **Set NewSheet = Worksheets.Add** создается новый рабочий лист. При этом он становится текущим. В операторе **NewSheet.Name = "Лист5"** ему присваивается имя **Лист5**. В операторе **Worksheets("Лист1").Activate** текущим становится рабочий лист **Лист1**. Последующий текст программы такой же, как в примере 5.7.

В данном случае важно, чтобы оператор **Set d=Selection.CurrentRegion** был указан до оператора **Set NewSheet = Worksheets.Add**. Это необходимо, чтобы программа определила текущий диапазон ячеек (и «запомнила» его в переменной **d**), прежде чем текущим станет другой (новый) рабочий лист.

Пример 5.9 – В условиях примера 5.4 требуется вывести перечень отобранных студентов на новый рабочий лист с именем **Отобранные**. На этом листе фамилии студентов должны выводиться в столбец A (начиная с ячейки A1), а средние баллы – в столбец B.

Для этого необходимо предусмотреть в программе создание рабочего листа с именем **Отобранные** (аналогично примеру 5.8). Операторы вывода результатов в ячейки рабочего листа будут иметь следующий вид:

```
Worksheets("Отобранные").Cells(k, 1).Value = d.Cells(i, 1).Value
Worksheets("Отобранные").Cells(k, 2).Value = srednee
```

Пример 5.10 – Пусть в условиях примера 5.6 информация о ценах на товары хранится на отдельном рабочем листе с именем **Цены**. Названия товаров указаны в столбце A, а цены – в столбце B.

В программе, приведенной в примере 5.6, достаточно заменить оператор, задающий диапазон ячеек с ценами, на следующий:

```
Set d2 = Worksheets("Цены").Range("A1").CurrentRegion
```

5.4 Варианты заданий

Вариант 1 – На рабочем листе имеется заполненная числами прямоугольная область. Программа должна определять минимальный элемент в первом столбце этой области, а затем менять местами строку, где этот элемент находится, с последней строкой области. Кроме того, программа должна выводить на экран номера строк, которые поменялись местами.

Вариант 2 – На рабочем листе имеется заполненная числами прямоугольная область (будем называть ее исходной). Программа должна скопировать в другую область рабочего листа (область результатов) все строки исходной области, кроме строк, заполненных некоторым числом. Это число запрашивается с клавиатуры. Пример исходных данных и результатов программы приведен на рисунке 5.2 (копируются все строки, кроме заполненных числом 7). Кроме того, программа должна выводить на экран количество скопированных строк.

5	7	2	3	5	7	2	3
6	7	7	7	6	7	7	7
7	7	7	7	4	2	1	9
4	2	1	9	2	3	6	2
7	7	7	7				
2	3	6	2				

Рисунок 5.2 – Исходные данные и результаты для варианта 2

Вариант 3 – На рабочем листе имеется некоторая прямоугольная область, заполненная числами (будем называть ее исходной). Программа должна из каждой строки исходной области скопировать в другую область рабочего листа (область результатов) все числа, превышающие некоторое заданное число. Это число запрашивается с клавиатуры. Пример исходных данных и результатов программы приведен на рисунке 5.3 (копируются все числа, превышающие 4). Кроме того, программа должна выводить на экран общее количество скопированных чисел.

5	7	2	8	5	7	8	
1	2	9	4	9			
3	2	3	1				
6	9	5	8	6	9	5	8

Рисунок 5.3 – Исходные данные и результаты для варианта 3

Вариант 4 – На рабочем листе расположены две прямоугольные области, заполненные числами (будем называть их первой и второй областью). Размеры областей одинаковы. Программа должна сравнивать суммы соответ-

ствующих строк этих областей. Если сумма строки из первой области оказывается меньше, чем сумма соответствующей строки из второй области, то эти строки меняются местами (т. е. строка с большей суммой переносится из второй области в первую, а строка с меньшей суммой – из первой области во вторую). Кроме того, программа должна выводить на экран количество случаев, когда строки менялись местами.

Вариант 5 – На рабочем листе имеется прямоугольная область, заполненная числами. Пользователь вводит с клавиатуры некоторое число. Программа определяет в каждой строке выделенной области число, ближайшее к введенному. Номера этих чисел выводятся на рабочий лист в отдельный столбец (например, если введено число 5, а в некоторой строке есть числа 8, 12, 4 и 7, то для этой строки должен быть выведен номер 3).

Вариант 6 – На рабочем листе имеется прямоугольная область, заполненная числами. Программа должна вычислить для каждого столбца области среднее значение и поменять местами столбцы с максимальным и минимальным средним значением. Кроме того, номера столбцов, которые поменялись местами, должны выводиться на экран.

Вариант 7 – На рабочем листе имеется заполненная числами прямоугольная область из m строк и n столбцов (будем называть эту область ячеек первой). Каждая строка представляет собой координаты некоторой точки в n-мерном пространстве. На этом же рабочем листе в другом месте расположена строка из n чисел – координаты еще одной точки в n-мерном пространстве (будем называть эту область ячеек второй). Программа должна находить среди точек n-мерного пространства, указанных в первой области, точку, ближайшую к указанной во второй области. Номер найденной ближайшей точки должен выводиться на экран.

Указание – Расстояние между точками вычисляется как квадратный корень из суммы квадратов разностей их координат. Квадратный корень в VBA вычисляется функцией **Sqr**.

Вариант 8 – На рабочем листе **Цены** в столбце A, начиная с ячейки A1, введены названия товаров, в столбце B – цены на эти товары, в столбце C – названия валют, в которых указаны цены. Может быть указано несколько товаров, цены которых выражены в одной и той же валюте.

На рабочем листе **Курсы** в столбце A, начиная с A1, перечислены названия валют, в столбце B – их курсы в долларах. Курс каждой валюты указывается один раз.

Программа должна вычислять цены товаров в долларах (если исходная цена была указана в долларах, то она не должна изменяться). Вычисленные цены должны выводиться в столбец D рабочего листа **Цены**. Должно также подсчитываться и выводиться на экран количество товаров, для которых потребовался пересчет цены в доллары (т. е. количество товаров, для которых исходная цена была указана не в долларах).

Вариант 9 – На рабочем листе **Работники** в столбце А (начиная с ячейки А1) введены фамилии работников, в столбце В – номера отделов, где они работают, в столбце С – зарплаты. Может быть указано несколько работников, работающих в одном и том же отделе.

На рабочем листе **Повышение** в столбце А, начиная с А1, перечислены номера отделов, для которых повышается зарплата, в столбце В – коэффициенты повышения зарплаты для работников данного отдела. Для каждого отдела коэффициент повышения зарплаты указывается только один раз (т. е. он одинаков для всех работников отдела). Некоторые отделы могут быть не указаны на листе **Повышение** (для них зарплата не повышается).

Программа должна вычислять новые зарплаты (путем умножения старой зарплаты на коэффициент повышения). Новые зарплаты должны указываться в столбце С рабочего листа **Работники** вместо старых зарплат. Должно также подсчитываться и выводиться на экран количество всех работников, для которых повышается зарплата.

Вариант 10 – На рабочем листе **Контракты** в столбце А, начиная с ячейки А1, введены номера контрактов, в столбце В – названия товаров, продаваемых по этим контрактам (по каждому контракту – один товар), в столбце С – количество товара, в столбце D – цены, по которым продаются товары. Может быть указано несколько контрактов на продажу одного и того же товара.

На рабочем листе **Итоги** в столбце А, начиная с А1, перечислены названия товаров. Каждый товар указывается один раз.

Для каждого товара программа должна вычислять количество контрактов на поставку данного товара и суммарное количество данного товара, поставляемое по всем контрактам. Количество контрактов на поставку товара должно выводиться в столбец В рабочего листа **Итоги**, а количество поставляемого товара по всем контрактам – в столбец С этого рабочего листа. Должна также подсчитываться и выводиться на экран суммарная стоимость всех контрактов.

Вариант 11 – На рабочем листе **Контракты** в столбце А, начиная с ячейки А1, введены номера контрактов, в столбце В – названия товаров, проданных по этим контрактам (по каждому контракту – один товар), в столбце С – цены, по которым проданы товары. Может быть указано несколько контрактов на продажу одного и того же товара.

На рабочем листе **Ограничения** в столбце А, начиная с А1, перечислены названия товаров, в столбце В – предельные цены на них (т. е. товары запрещается продавать по ценам, превышающим эти величины). Предельная цена на каждый товар указывается только один раз. Некоторые товары не указываются на листе **Ограничение** (на них цена не ограничивается).

Для каждого контракта, где цена превышает предельную, программа должна заменять ее (в рабочем листе **Контракты**) на соответствующую предельную цену. Должно также подсчитываться и выводиться на экран количество контрактов, для которых потребовалось снижение цены.

Вариант 12 – На рабочем листе **Контракты** в столбце A, начиная с ячейки A1, введены номера контрактов, в столбце B – названия товаров, проданных по этим контрактам (по каждому контракту – один товар), в столбце C – количество товара, проданного по каждому контракту, в столбце D – цены, по которым проданы товары. Может быть указано несколько контрактов на продажу одного и того же товара.

На рабочем листе **Ставки** в столбце A, начиная с A1, перечислены названия товаров, в столбце B – ставки налогов по контрактам на эти товары. Ставка налога для каждого товара указывается только один раз. Например, если в ячейке A1 указано название товара – компьютер, а в ячейке B1 – ставка 12 %, это означает, что с каждого контракта на поставку компьютеров выплачивается налог в размере 12 % от его полной стоимости.

Для каждого контракта программа должна вычислять выплачиваемый за него налог. Величины налогов должны выводиться в столбце E рабочего листа **Контракты**. Должна также подсчитываться и выводиться на экран сумма налогов по всем контрактам.

Вариант 13 – На рабочем листе **Контракты** в столбце A, начиная с ячейки A1, введены номера контрактов, в столбце B – названия товаров, продаваемых по этим контрактам (по каждому контракту – один товар), в столбце C – количество товара, в столбце D – цены, по которым продаются товары. Может быть указано несколько контрактов на продажу одного и того же товара.

На рабочем листе **Повышение** в столбце A, начиная с A1, перечислены названия товаров, в столбце B – коэффициенты повышения цены на товары. Коэффициент повышения цены для каждого товара указывается только один раз. Например, если в ячейке A1 указано название товара – компьютер, а в ячейке B1 – коэффициент 1,2, это означает, что цена компьютеров во всех контрактах должна быть повышена в 1,2 раза. Некоторые товары не указываются на листе **Повышение** (на них цена не повышается).

Для каждого контракта программа должна вычислять новую цену (путем умножения старой цены на коэффициент повышения). Новые цены должны указываться в столбце D рабочего листа **Контракты** (вместо старых цен). Должно также подсчитываться и выводиться на экран количество контрактов, для которых потребовалось повышение цены.

Вариант 14 – На рабочем листе **Контракты** в столбце A, начиная с ячейки A1, введены номера контрактов, в столбце B – названия товаров, продаваемых по этим контрактам (по каждому контракту – один товар), в столбце C – количество товара, в столбце D – цены, в столбце E – названия заказчиков (у каждого контракта один заказчик).

На рабочем листе **Заказчики** в столбце A, начиная с A1, перечислены названия заказчиков. Каждый заказчик указывается один раз.

Для каждого заказчика программа должна подсчитывать количество контрактов и их общую стоимость. Эти величины должны выводиться в столбцы

B и C рабочего листа Заказчики. Должно также подсчитываться и выводиться на экран количество заказчиков, для которых нет ни одного контракта.

5.5 Порядок выполнения работы

Для выполнения данной работы требуется разработка и отладка двух программ: по одной согласно вариантам 1–7 и 8–14.

5.6 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать:

- титульный лист;
- цель работы;
- листинги разработанных программ;
- копии экранов, иллюстрирующие выполнение программ (размещение исходных данных в рабочем листе MS Excel, вывод результатов);
- пояснения по использованию ссылок на ячейки рабочего листа и по операциям с рабочими листами MS Excel (в виде отдельного текста или комментариев к программам).

5.7 Контрольные вопросы

1 Способы ссылок на ячейки рабочего листа MS Excel.
2 Способы ссылок на диапазоны ячеек рабочего листа MS Excel.
3 Операции с выделенным диапазоном ячеек рабочего листа MS Excel.
4 Начало отсчета в ссылках на ячейки рабочего листа MS Excel.
5 Операции с рабочими листами MS Excel.

Ответы на все контрольные вопросы должны сопровождаться примерами в виде фрагментов программ.

ЛАБОРАТОРНАЯ РАБОТА №6

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ В ПРОГРАММАХ НА VBA

Цель работы – Изучение возможностей использования элементов управления на рабочем листе MS Excel в программах на VBA.

6.1 Размещение элементов управления на рабочем листе Excel

Чтобы разместить элемент управления (кнопку, переключатель, флажок и т. д.) на рабочем листе Excel и сделать его работоспособным, требуется выполнить следующее:

– вызвать на экран панель инструментов **Элементы управления** (команда **Вид – Панели инструментов – Элементы управления**);

– перейти в режим конструктора. Для этого в панели инструментов **Элементы управления** нажать кнопку **Режим конструктора**;

– выбрать из панели инструментов желаемый элемент управления и разместить его на рабочем листе Excel;

– вызвать на экран меню свойств элементов управления (меню **Properties**). Для этого в панели инструментов **Элементы управления** нажать кнопку **Свойства**;

– используя меню свойств элементов управления, установить желаемые свойства элемента управления. Для этого выделить элемент управления на рабочем листе Excel или выбрать его из списка в верхней части меню **Properties.** После этого установить желаемые свойства;

– разработать программу на языке VBA для работы с элементом управления. Для этого выбрать элемент управления; в панели инструментов **Элементы управления** нажать кнопку **Исходный текст**. Вызывается редактор VBA, и открывается модуль текущего рабочего листа, в котором следует ввести текст программы для обработки событий, связанных с элементом управления. Событием может быть, например, нажатие кнопки, изменение значения счетчика, выбор значения переключателя и т. д.;

– вернуться из редактора VBA на рабочий лист Excel. Выйти из режима конструктора, нажав кнопку **Выход** из режима конструктора в панели инструментов **Элементы управления**.

6.2 Пример использования элементов управления: кнопки, переключатели, счетчики, флажки, текстовые поля

Пример 6.1 – Пусть требуется разместить на рабочем листе Excel следующие элементы управления:

– кнопка **Выполнить**;

– переключатели **Меньше** и **Больше**;

– счетчик;

– флажок **Сумма**;

– текстовое поле.

При нажатии кнопки **Выполнить** в диапазоне ячеек, выделенном с помощью мыши, должны выполняться следующие действия: если установлен переключатель **Меньше**, то значения всех ячеек, меньшие, чем некоторая предельная величина, должны заменяться на эту величину. Если же установлен переключатель **Больше**, то заменяться должны все значения, превышающие предельную величину. Сама предельная величина устанавливается в одной из ячеек Excel с помощью счетчика. Кроме того, если установлен флажок **Сумма**, то должна вычисляться сумма выделенного диапазона ячеек. Эта сумма должна выводиться в созданное на рабочем листе текстовое поле.

Создание элементов управления

Чтобы создать желаемые элементы управления, необходимо вызвать на экран панель инструментов **Элементы управления** (как показано в подразделе 6.1) и с помощью мыши разместить на рабочем листе необходимые элементы управления: кнопку, два переключателя, счетчик, флажок, текстовое поле.

Указание свойств элементов управления

Чтобы установить свойства элементов управления, необходимо сначала вызвать на экран меню **Properties** (см. выше).

Чтобы установить свойства кнопки, следует выделить ее с помощью мыши на рабочем листе Excel или выбрать ее из списка элементов управления, имеющегося в верхней части меню **Properties** (кнопка по умолчанию имеет имя **CommandButton1**). Для кнопки требуется установить следующие свойства:

- **Name: Obrabotka**;
- **Caption: Выполнить**.

Свойство **Name** – имя, под которым кнопка будет использоваться в программе на VBA. Свойство **Caption** – подпись кнопки, т. е. текст, который будет указан на кнопке.

Для одного из переключателей (имена переключателей по умолчанию – **OptionButton1** и **OptionButton2**) установим следующие свойства:

- **Name: Bolshe;**
- **Caption: Больше;**
- **GroupName: Bol_men.**

Здесь свойство **Name** – имя для ссылок на переключатель; **Caption** – подпись переключателя, т. е. текст, который будет указан рядом с ним; **GroupName** – имя группы переключателей. Из всех переключателей, для которых указано одинаковое имя группы, в любой момент может быть установлен только один, остальные – сброшены.

Аналогично установим свойства другого переключателя:

- **Name: Menshe;**
- **Caption: Меньше;**
- **GroupName: Bol_men.**

Для счетчика (имя по умолчанию – **SpinButton1**) установим следующие свойства:

- **Name: Predel;**
- **LinkedCell: A10.**

Свойство **LinkedCell** задает ячейку, в которой будет выводиться значение, установленное с помощью счетчика (ячейка A10 выбрана произвольно).

Для флажка (имя по умолчанию – **CheckBox1**) установим следующие свойства:

- **Name: Summa;**
- **Caption: Сумма.**

Для текстового поля (имя по умолчанию – **TextBox1**) требуется установить только свойство **Name: Summa_diapazona.**

Разработка программы

Чтобы приступить к написанию программы для работы с созданными элементами управления, следует выбрать на рабочем листе кнопку **Выполнить** (так как программа должна выполняться именно при нажатии кнопки) и в панели инструментов **Элементы управления** нажать кнопку **Исходный текст**. Вызывается редактор VBA, и открывается модуль текущего рабочего листа. В нем автоматически создается заголовок процедуры:

```
Private Sub Obrabotka_Click()
```

Здесь **Obrabotka** – имя элемента управления, для которого создается программа (в данном случае – имя кнопки). **Click** – имя события, для обработки которого создается программа (в данном случае – щелчок мыши по кнопке). Слово **Private** обозначает область видимости процедуры (см. лабораторную работу №4). Так как в данном случае вся программа располагается в одном модуле, указание области видимости процедуры никак не влияет на ее работу.

Программа, вызываемая при щелчке мышью по кнопке **Obrabotka**, имеет следующий вид.

```
Private Sub Obrabotka_Click()
Set d = Selection
m = d.Rows.Count
n = d.Columns.Count

granitsa = Predel.Value
If Bolshe.Value = True Then
For i = 1 To m
For j = 1 To n
If d.Cells(i, j).Value > granitsa Then d.Cells(i, j).Value = granitsa
Next j
Next i
Else
For i = 1 To m
For j = 1 To n
If d.Cells(i, j).Value < granitsa Then d.Cells(i, j).Value = granitsa
```

```
Next j
Next i
End If

If Summa.Value = True Then
For i = 1 To m
For j = 1 To n
s = s + d.Cells(i, j).Value
Next j
Next i
Summa_diapazona.Value = s
End If
End Sub
```

Основные действия, выполняемые программой, следующие. В начале программы определяются размеры области, выделенной на рабочем листе Excel (см. лабораторную работу №5). Затем в операторе **granitsa = Predel.Value** переменной **granitsa** присваивается значение счетчика **Predel.Value**, т. е. значение, установленное с помощью этого счетчика и отображающееся (в данном примере) в ячейке A10.

Проверяется значение переключателя **Bolshe** (свойство **Value**). Если переключатель установлен, то в выделенной области всем ячейкам, значение которых превышает переменную **granitsa**, присваивается значение этой переменной. Если переключатель **Bolshe** не установлен (значит, установлен переключатель **Menshe**), то, наоборот, изменяется значение ячеек в выделенной области, меньших, чем заданная величина.

В операторе **If Summa.Value = True** проверяется значение флажка **Summa**. Если оно равно **True** (флажок установлен), то вычисляется сумма элементов выделенного диапазона. Эта сумма присваивается свойству **Value** текстового поля **Summa_diapazona** (т. е. выводится в это поле).

6.3 Пример использования элементов управления: списки

Пример 6.2 – Пусть в столбце A рабочего листа Excel, начиная с ячейки A1, введены номера контрактов, в столбце B (начиная с B1) – их стоимости. Требуется разработать программу для отбора номеров контрактов, стоимость которых составляет не менее или, наоборот, не более (по выбору пользователя) некоторой заданной величины.

Пусть для этого предполагается разместить на рабочем листе Excel следующие элементы управления:

- кнопка **Отобрать;**
- список, содержащий два элемента: **Не менее** и **Не более;**
- текстовое поле.

При нажатии кнопки **Отобрать** должны выполняться следующие действия: если в списке выбран элемент **Не менее**, то должны отбираться номера контрактов, стоимость которых составляет не менее величины, указанной в текстовом поле. Если же выбран элемент **Не более**, то должны отбираться

номера контрактов, стоимость которых не превышает величины, указанной в текстовом поле. Номера отобранных контрактов должны выводиться в столбец E, начиная с ячейки E1.

Пусть на рабочем листе Excel создана кнопка со свойствами **Name – Vybor, Caption – Отобрать**, а также текстовое поле со свойством **Name – Granitsa**.

Рассмотрим более подробно создание списка. Требуется в каком-либо месте рабочего листа Excel ввести элементы создаваемого списка. Пусть в ячейке M1 введен текст **Не менее**, а в ячейке M2 – **Не более**.

Примечание – Элементы списка могут и не перечисляться на рабочем листе Excel, а задаваться в программе. Этот способ задания элементов списка будет рассмотрен в лабораторной работе №7.

После того, как на рабочем листе Excel создан список (имя по умолчанию – **ListBox1**), для него необходимо указать следующие свойства: **Name – Bol_men** (или любое другое имя), **ListFillRange** – **M1:M2** (т. е. диапазон ячеек, где указаны элементы списка).

Введем следующую программу, которая должна выполняться при щелчке мышью по кнопке **Отобрать**:

```
Private Sub Vybor_Click()
Set d = Range("A1").CurrentRegion
m = d.Rows.Count
x = CSng(granitsa.Value)
Set rez = Range("E1")
k = 0

If Bol_men.ListIndex = 0 Then
For i = 1 To m
If d.Cells(i, 2) >= x Then
k = k + 1
rez.Cells(k, 1) = d.Cells(i, 1)
End If
Next i
End If

If bol_men.ListIndex = 1 Then
For i = 1 To m
If d.Cells(i, 2) <= x Then
k = k + 1
rez.Cells(k, 1) = d.Cells(i, 1)
End If
Next i
End If

End Sub
```

Приведем некоторые пояснения по программе. Переменной **x** присваивается значение текстового поля **granitsa**; функция **CSng** требуется для преобразования значения этого поля (по умолчанию имеющего тип **String**) в тип **Single**, т. е. в число.

ListIndex – свойство элемента управления типа «Список» (в данном случае – списка **Bol_men**), представляющее собой номер текущего (выбранного) элемента списка, причем элементы списка нумеруются *с нуля*. Таким образом, условие **bol_men.ListIndex = 0** означает, что выбран первый элемент списка (в данном случае – элемент **Не менее**); условие **bol_men.ListIndex = 1** означает, что выбран второй элемент (**Не более**).

Вместо свойства **ListIndex** можно было бы использовать свойство **Text**, представляющее собой значение текущего (выбранного) элемента списка. В этом случае в программе вместо условия **If bol_men.ListIndex = 0** было бы указано условие **If bol_men.Text = "Не менее"**, а вместо условия **If bol_men.ListIndex = 1** – условие **If bol_men.Text = "Не более"**.

6.4 Варианты заданий

Вариант 1 – Установить на рабочем листе следующие элементы управления: два переключателя (**Число** и **Среднее**), счетчик, флажок **Сумма**, кнопку **Подсчитать**, два текстовых поля.

Разработать программу на VBA для обработки данных в диапазоне ячеек Excel, заполненном числами и выделенном с помощью мыши. При нажатии кнопки **Подсчитать** в заданной строке выделенного диапазона подсчитывается количество чисел, превышающих число, заданное в первом текстовом поле (если установлен переключатель **Число**), или среднее по этой строке (если установлен переключатель **Среднее**). Номер строки, для которой выполняется подсчет, задается с помощью счетчика. Подсчитанное количество отображается в ячейке справа от выбранной строки. Кроме того, если установлен флажок **Сумма**, то вычисляется сумма выбранной строки; она выводится во второе текстовое поле.

Вариант 2 – Установить на рабочем листе следующие элементы управления: два переключателя (**Среднее арифметическое** и **Среднее геометрическое**), счетчик, два флажка (**Строка** и **Столбец**), кнопку **Вычислить**.

Разработать программу на VBA для обработки данных в диапазоне ячеек Excel, заполненном числами и выделенном с помощью мыши. При нажатии кнопки **Вычислить** вычисляется среднее арифметическое или среднее геометрическое (в зависимости от установленного переключателя) по строке или по столбцу выделенной области. Если установлен флажок **Строка**, то вычисление выполняется по строке, если установлен флажок **Столбец** – по столбцу; если установлены оба флажка, то вычисление выполняется и по строке, и по столбцу. Если не установлен ни один из флажков, то вычисления не выполняются. Номер строки (или столбца), для которой выполняются вычисления, задается с помощью счетчика. Результат вычисления выводится в ячейку рядом с соответствующей строкой или столбцом.

Указание – Среднее геометрическое из n чисел – это их произведение, из которого взят корень степени n.

Вариант 3 – Установить на рабочем листе следующие элементы управления: два переключателя (**Выбранный** и **Все**), счетчик, флажок **Максимальный**, кнопку **Вычислить**, текстовое поле.

Ввести в рабочий лист исходные данные: в столбец A (начиная с ячейки A1) – фамилии работников, в столбцы B–F – их доходы за пять лет.

Разработать программу на VBA для вычисления среднего дохода каждого работника. Вычисление выполняется при нажатии кнопки **Вычислить**. Если при этом установлен переключатель **Выбранный**, то средний доход вычисляется для одного работника (его номер указывается с помощью счетчика), если установлен переключатель **Все** – то для всех работников. Вычисленные средние доходы выводятся в столбец G. Кроме того, если доходы вычисляются для всех работников, и при этом установлен флажок **Максимальный**, то в текстовом поле отображается фамилия человека с максимальным средним доходом.

Вариант 4 – Установить на рабочем листе следующие элементы управления: два переключателя (**По строке** и **По всему диапазону**), счетчик, флажок **Подсчет**, три текстовых поля, кнопку **Замена**.

Разработать программу на VBA для обработки данных в диапазоне ячеек Excel, заполненном числами и выделенном с помощью мыши. При нажатии кнопки **Замена** выполняется замена числа, указанного в первом текстовом поле, на число, указанное во втором текстовом поле. В зависимости от состояния переключателей замена выполняется в одной строке выделенного диапазона или по всему диапазону. Если замена выполняется в одной строке, то ее номер задается с помощью счетчика. Кроме того, если установлен флажок **Подсчет**, то в третье текстовое поле выводится количество выполненных замен.

Вариант 5 – Установить на рабочем листе следующие элементы управления: два переключателя (**За последние** и **За все**), счетчик, флажок **Максимальный**, кнопку **Средний доход**, текстовое поле.

Ввести в рабочий лист исходные данные: в столбец A (начиная с ячейки A1) – фамилии работников, в столбцы B–F – их доходы за пять лет.

Разработать программу на VBA для вычисления среднего дохода каждого работника. Вычисление выполняется при нажатии кнопки **Средний доход**. Если при этом установлен переключатель **За последние**, то средний доход вычисляется за последние годы (количество лет указывается с помощью счетчика). Если установлен переключатель **За все**, то средний доход вычисляется за пять лет. Вычисленные средние доходы выводятся в столбец G. Кроме того, если установлен флажок **Максимальный**, то в текстовом поле отображается фамилия работника с максимальным средним доходом.

Вариант 6 – Установить на рабочем листе следующие элементы управления: два переключателя (**Строки** и **Столбцы**), два счетчика, флажок **Сумма**, текстовое поле, кнопку **Переставить**.

Разработать программу на VBA для обработки данных в диапазоне ячеек Excel, заполненном числами и выделенном с помощью мыши. При нажатии кнопки **Переставить** в выделенном диапазоне меняются местами две строки

или два столбца (в зависимости от настройки переключателей). Номера строк (или столбцов), которые требуется поменять местами, задаются с помощью счетчиков. Кроме того, если установлен флажок **Сумма**, то вычисляется сумма строк (или столбцов), которые поменялись местами. Эта сумма выводится в текстовое поле.

Вариант 7 – Установить на рабочем листе следующие элементы управления: два переключателя (**В начало** и **В конец**), счетчик, флажок **По всему диапазону**, кнопку **Переставить**, текстовое поле.

Разработать программу на VBA для обработки данных в диапазоне ячеек Excel, заполненном числами и выделенном с помощью мыши. При нажатии кнопки **Переставить** в заданной строке определяется максимальный элемент, и он меняется местами с первым или с последним элементом этой строки (в зависимости от настройки переключателей). Номер строки, для которой выполняется перестановка, задается с помощью счетчика. Кроме того, найденный максимальный элемент отображается в текстовом поле. Если установлен флажок **По всему диапазону**, то перестановка выполняется для всех строк выделенного диапазона (независимо от настройки счетчика).

6.5 Порядок выполнения работы

1 Разработать и отладить одну из программ по приведенным вариантам заданий. Сохранить отлаженную программу и сделать ее копию.

2 Внести в разработанную программу следующие изменения: выбор, реализованный в исходной программе с помощью переключателей, должен быть реализован в виде списка.

6.6 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать:

– титульный лист;
– цель работы;
– листинги разработанных программ;
– копии экранов, иллюстрирующие выполнение программ (размещение элементов управления, ввод исходных данных, вывод результатов);
– пояснения по использованию элементов управления, размещенных на рабочем листе MS Excel (в виде отдельного текста или комментариев к программам).

6.7 Контрольные вопросы

1 Основные этапы разработки программы с использованием элементов управления.

2 Элементы управления в программах на VBA: кнопки.

3 Элементы управления в программах на VBA: флажки.

4 Элементы управления в программах на VBA: текстовые поля.

5 Элементы управления в программах на VBA: счетчики. Отображение значения счетчика.

6 Переключатели. Использование нескольких групп переключателей.

7 Текстовые поля и их использование для ввода и вывода данных.

8 Элементы управления в программах на VBA: списки.

Ответы на все контрольные вопросы должны сопровождаться примерами в виде фрагментов программ.

ЛАБОРАТОРНАЯ РАБОТА №7

ПОЛЬЗОВАТЕЛЬСКИЕ ФОРМЫ В ПРОГРАММАХ НА VBA

Цель работы – Изучение возможностей построения интерфейса программ на VBA с применением пользовательских форм.

7.1 Создание пользовательской формы в Excel

Пользовательская форма – диалоговое окно, в котором можно размещать элементы управления. Пользовательская форма позволяет создавать желаемый интерфейс, не ограничиваясь возможностями стандартного интерфейса Excel или других программ.

Чтобы создать пользовательскую форму в среде Excel и сделать ее работоспособной, требуется выполнить следующее:

– выбрать команду **Сервис – Макрос – Редактор Visual Basi**c;

– в среде программирования Visual Basic выбрать команду **Insert – UserForm**. Создается пользовательская форма, т. е. пустое окно, где можно размещать элементы управления;

– вызвать на экран панель элементов управления (если она не вызвана автоматически). Для этого выбрать команду **View – Toolbox** (в других версиях – **View – Controls**);

– разместить в окне пользовательской формы желаемые элементы управления;

– вызвать на экран меню свойств элементов управления (меню **Properties**), если оно не вызвано автоматически. Для этого выбрать команду **View – Properties Window**;

– используя меню свойств элементов управления, установить желаемые свойства пользовательской формы или размещенных на ней элементов управления. Для этого выбрать из списка в верхней части меню **Properties** желаемый элемент управления или пользовательскую форму (**UserForm**); после этого установить желаемые свойства;

– разработать программу на языке VBA для работы с пользовательской формой. Для этого выбрать команду **View – Code**. Вызывается редактор VBA, и открывается модуль пользовательской формы, в котором следует ввести текст программы для обработки событий, связанных с пользовательской формой (инициализация пользовательской формы, нажатие кнопок и т. д.);

– вернуться из режима разработки программы в режим работы с окном пользовательской формы. Для этого убедиться, что в окне проекта (в левой части экрана) выбрана отметка пользовательской формы, и выбрать команду **View – Object**;

– для начала работы с пользовательской формой (т. е. для ее «запуска») выбрать команду **Run/Run Sub/UserForm**.

Рассмотрим примеры разработки программ с пользовательскими формами, содержащими различные элементы управления.

7.2 Кнопки, текстовые поля, списки

Пример 7.1 – Требуется разработать пользовательскую форму (рисунок 7.1), которая будет запрашивать у пользователя диапазон ячеек рабочего листа Excel (левый верхний и правый нижний угол) и вычислять по выбору пользователя сумму или произведение чисел в ячейках этого диапазона. Для ввода границ диапазона (т. е. левого верхнего и правого нижнего угла) и для вывода результата (суммы или произведения) должны использоваться текстовые поля пользовательской формы. Для выбора операции (сумма или произведение) должен использоваться список. Вычисление должно выполняться при нажатии кнопки, размещенной на пользовательской форме. Кроме того, требуется создать кнопку для закрытия пользовательской формы.

Создание элементов управления и указание их свойств

Требуется создать пользовательскую форму, как показано в подразделе 7.1. В ее окне требуется разместить элементы управления: три надписи (**Label**), три текстовых поля (**TextBox**), раскрывающийся список (**ComboBox**), две кнопки (**CommandButton**). В результате окно пользовательской формы должно иметь примерно такой вид, как показано на рисунке 7.2.

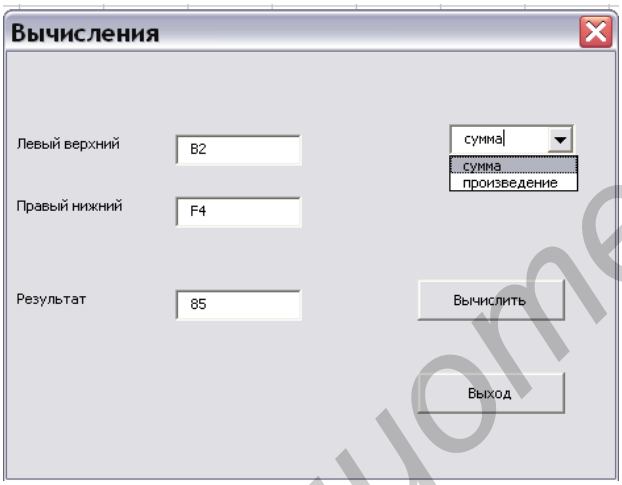

Рисунок 7.1 – Пользовательская форма (пример 7.1) во время работы программы

Рисунок 7.2 – Пользовательская форма (пример 7.1) в режиме редактирования

Чтобы форма имела такой вид, как показано на рисунке 7.2, требуется с помощью меню **Properties** задать для нее следующие свойства:

– форма в целом (**UserForm**): **Name – Sum_proizv** (имя формы, используемое для ссылок на нее); **Caption – Вычисления** (заголовок формы);

– надписи: для первой надписи (на рисунке 7.2 – верхняя) указать свойство **Caption – Левый верхний**. Для второй надписи указать свойство **Caption – Правый нижний**, для третьей указать **Caption – Результат**;

– текстовые поля: для верхнего поля указать свойство **Name – Nach**, для следующего – **Kon**, для третьего – **Rez**;

– раскрывающийся список: **Name – Operacii**;

– кнопка для вычисления: **Name – Schet**, **Caption – Вычислить**;

– кнопка для закрытия формы: **Name – Vyhod**, **Caption – Выход**.

Некоторые другие свойства элементов формы (например, перечень элементов списка и номер его текущего элемента) будут заданы в программе для работы с формой.

Разработка программы

Чтобы приступить к написанию программы для работы с созданной пользовательской формой, требуется после создания формы, приведенной на рисунке 7.2, выбрать команду **View – Code**. Вызывается редактор VBA, и открывается модуль пользовательской формы. В нем необходимо ввести следующий текст программы (некоторые заголовки подпрограмм будут созданы автоматически):

```
Private Sub UserForm_Initialize()
Operacii.AddItem "сумма"
Operacii.AddItem "произведение"
Operacii.ListIndex = 0
End Sub

Private Sub Schet_Click()
diap = Nach.Value + ":" + Kon.Value
Set d = Range(diap)
m = d.Rows.Count
n = d.Columns.Count
If Operacii.ListIndex = 0 Then
Sum = 0
For i = 1 To m
For j = 1 To n
Sum = Sum + d.Cells(i, j).Value
Next j
Next i
Rez.Value = Sum
End If
If Operacii.ListIndex = 1 Then
proizv = 1
For i = 1 To m
For j = 1 To n
proizv = proizv * d.Cells(i, j).Value
Next j
Next i
rez.Value = proizv
End If
End Sub

Private Sub Vyhod_click()
Unload Sum_proizv
End Sub
```

Процедура **UserForm_Initialize()** выполняется в начале работы с пользовательской формой, т. е. выполняет ее инициализацию. Оператор **Operacii.AddItem “сумма”** означает, что к списку **Operacii** применяется

действие (метод) **AddItem**, т. е. к списку добавляется заданный элемент – слово **“сумма”**. Затем в список аналогично включается элемент **“произведение”**. В операторе **operacii.ListIndex = 0** свойству **ListIndex** списка **Operacii** присваивается значение 0; это означает, что по умолчанию в списке будет выбран **первый по порядку** элемент, т. е. элемент **“сумма”** (элементы списка имеют номера, начиная *с нуля*).

Процедура **Schet_Click()** выполняется при нажатии кнопки с именем **Schet** (т. е. кнопки **Вычислить**). В операторе **diap = Nach.Value + ":" + Kon.Value** составляется символьная строка из значений текстовых полей **Nach** и **Kon**, между которыми добавляется двоеточие; таким образом, строка **diap** будет представлять собой диапазон ячеек, заданный в текстовых полях. Например, если в текстовом поле **Nach** пользователь введет значение **B2**, а в текстовом поле **Kon** – значение **F4**, то переменная **diap** будет иметь значение **“B2:F4”**.

В операторе **Set d = Range(diap)** переменная **d** связывается с диапазоном, заданным переменной **diap**. В двух следующих операторах определяется количество строк и столбцов в заданном диапазоне.

В операторе **If Operacii.ListIndex = 0 Then …** проверяется, чему равно свойство **ListIndex** списка **Operacii**, т. е. какой элемент выбран в списке **Operacii**. Если выбран элемент с номером 0 (т. е. первый элемент – **“сумма”**), то выполняется суммирование значений ячеек в заданном диапазоне. Результат вычислений (переменная **Sum**) выводится в текстовое поле **Rez:** для этого выполняется присваивание **Rez.Value = Sum**.

Аналогично: если выбран элемент списка с номером 1 (**“произведение”**), то вычисляется произведение ячеек в заданном диапазоне, и результат выводится в текстовое поле **Rez**.

Процедура **Vyhod_click()** выполняется при нажатии кнопки с именем **Vyhod** (т. е. кнопки **Выход**). Оператор **Unload Sum_proizv** закрывает форму.

7.3 Флажки, счетчики

Пример 7.2 – Требуется разработать пользовательскую форму (рисунок 7.3) для возведения чисел в заданную степень и для извлечения корней заданной степени (или выполнения обеих этих операций). Для ввода числа, которое требуется возвести в степень (извлечь из него корень), должно использоваться текстовое поле. Показатель степени (для возведения или извлечения корня) задается с помощью счетчика. Для выбора операции (возведение в степень или извлечение корня) используются флажки. Результаты должны выводиться в текстовые поля. Вычисление должно выполняться при нажатии кнопки, размещенной на пользовательской форме. Кроме того, требуется создать кнопку для закрытия пользовательской формы.

Создание элементов управления и указание их свойств

Требуется создать пользовательскую форму и разместить на ней элементы управления, как показано на рисунке 7.4. Для элементов управления необ-

ходимо задать следующие свойства (конечно, имена элементов управления могут быть и другими):

– форма в целом (**UserForm**): **Name – Stepen_koren** (имя формы, используемое для ссылок на нее); **Caption – Степени и корни** (заголовок формы);

– надписи (**Label**): две надписи со свойствами **Caption** – **Число** и **Показатель**;

– текстовые поля (**TextBox**): четыре текстовых поля со свойствами **Name – Osnovanie, Pokazatel, Stepen, Koren** (на рисунке 7.4 эти текстовые поля приведены сверху вниз);

– счетчик (**SpinButton**): свойство **Name – Pokaz**;

– флажки (**CheckBox**): два флажка со свойствами **Name – Vozved** и **Izvlech**, **Caption – Возведение** и **Извлечение;**

– кнопки: кнопка для вычисления результата должна иметь свойства **Name – Schet**, **Caption – Вычислить**, а кнопка для закрытия формы – свойства **Name – Vyhod**, **Caption – Выход**.

Примечание – Важно понимать, что все связи между элементами управления (например, отображение значения счетчика, вывод результатов операций и т. д.) должны задаваться в программе для работы с формой.

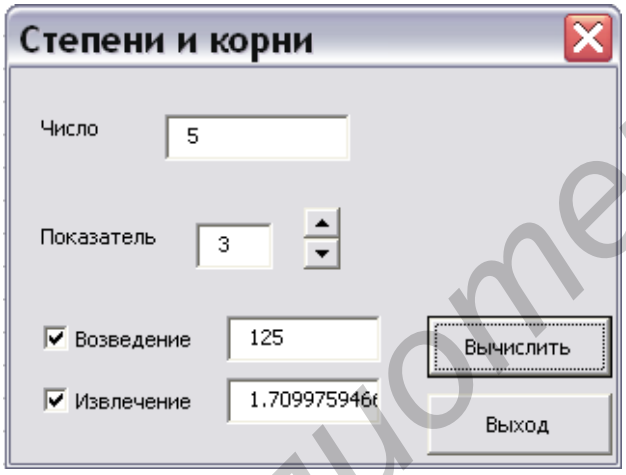

Рисунок 7.3 – Пользовательская форма (пример 7.2) во время работы программы

Рисунок 7.4 – Пользовательская форма (пример 7.2) в режиме редактирования

Разработка программы

Программа для работы с формой для данной задачи должна иметь примерно следующий вид:

```
Private Sub UserForm_Initialize()
Vozved.Value = True
Izvlech.Value = True
End Sub

Private Sub Pokaz_change()
pokazatel.Value = Pokaz.Value
End Sub

Private Sub Raschet_click()
```

```
x = CSng(Osnovanie.Value)
y = CInt(Pokazatel.Value)
If vozved.Value = True Then
z = x ^ y
Stepen.Value = z
Else
Stepen.Value = ""
End If
If Izvlech.Value = True Then
z = x ^ (1 / y)
Koren.Value = z
Else
Koren.Value = ""
End If
End Sub

Private Sub Vyhod_click()
Unload Stepen_koren
End Sub
```

Процедура **UserForm_Initialize()** выполняется в начале работы с пользовательской формой. Операторы **Vozved.Value = True** и **Izvlech.Value = True** означают, что в начале работы с формой оба флажка (**Vozved** и **Izvlech**) должны быть установлены.

Процедура **Pokaz_change()** выполняется при каждом изменении значения счетчика **Pokaz**. Оператор **Pokazatel.Value = Pokaz.Value** означает, что значение счетчика **Pokaz** выводится в текстовое поле **Pokazatel** (т. е. присваивается его свойству **Value**). Это необходимо, чтобы видеть на экране значение счетчика (т. е. заданную степень).

Процедура **Schet_Click()** выполняется при каждом нажатии кнопки с именем **Schet** (кнопки **Вычислить**). В операторе **x = CSng(Osnovanie.Value)** переменной **x** присваивается значение текстового поля **Osnovanie** (т. е. число, которое требуется возвести в степень и/или извлечь из него корень); функция **CSng** преобразует значение этого поля в тип **Single** (вещественное число). В операторе **y = CInt(Pokazatel.Value)** переменной **y** присваивается значение поля **Pokazatel** (показатель степени), преобразованное в тип **Integer**.

Если выполняется условие **vozved.Value = True** (флажок **Vozved** установлен), то выполняется заданное возведение в степень, и результат выводится в текстовое поле **Stepen**. Если условие **Vozved.Value = True** не выполняется (флажок **Vozved** сброшен), то текстовое поле **Stepen** очищается (в него выводится пустая строка).

Аналогично, если установлен переключатель **Izvlech**, то выполняется извлечение корня заданной степени, и результат выводится в текстовое поле **Koren**.

Процедура **Vyhod_Click()**, выполняемая при нажатии кнопки с именем **Vyhod**, предназначена для закрытия формы. Она аналогична процедуре для этой цели, приведенной в предыдущем примере.

7.4 Список из нескольких колонок

Пример 7.3 – В рабочем листе Excel введен список сотрудников некоторой организации (столбец А) и их доходы (столбец В). Эти данные должны отображаться в списке, содержащемся в пользовательской форме (рисунок 7.5).

Требуется, чтобы при щелчке по элементу списка (т. е. при выборе сотрудника) вычислялся налог с дохода выбранного сотрудника. Налог должен выводиться в текстовое поле **Налог** на форме, а также на рабочий лист Excel в столбец С (рядом с доходом соответствующего сотрудника). При нажатии на кнопку **Выход** форма должна закрываться.

Для решения этой задачи требуется создать пользовательскую форму. Установить для нее свойства: **Name = Nalogi**, **Caption = Расчет налогов**.

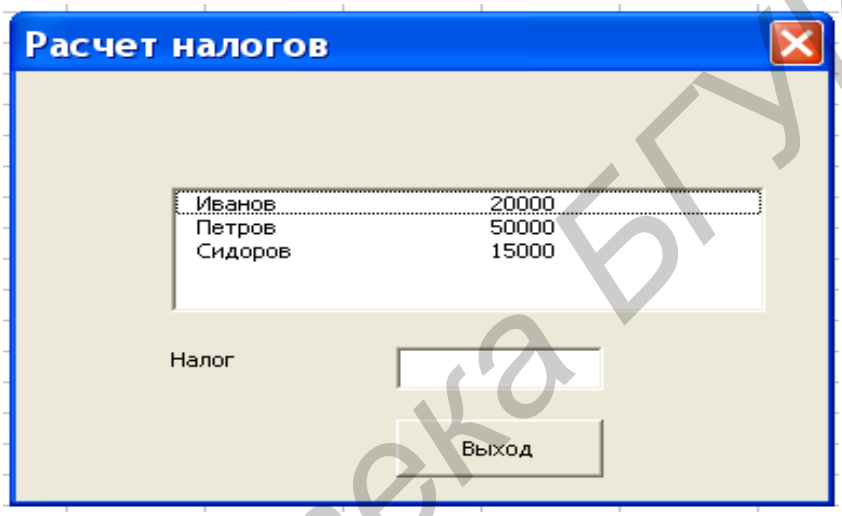

Рисунок 7.5 – Пользовательская форма со списком из двух колонок (пример 7.3)

Требуется разместить на форме элементы управления и задать им свойства, как показано в таблице 7.1.

Таблица 7.1 – Элементы управления для примера 7.3

Элемент управления	Свойства
Список (**Listbox**)	**Name = Spisok_sotr**
Надпись (**Label**)	**Caption = Налог**
Текстовое поле (**TextBox**)	**Name = Pole_nalog**
Кнопка (**CommandButton**)	**Name = Vyhod, Caption = Выход**

Введем следующую процедуру инициализации пользовательской формы:

```
Private Sub Userform_Initialize()
Set d = Range("A1").CurrentRegion
With spisok_sotr
.ColumnCount = 2
.RowSource = d.Address
End With
End Sub
```

Здесь в операторе **Set d = Range("A1").CurrentRegion** переменная **d** связывается с диапазоном, заполненным данными (в данном примере – фамилиями сотрудников и их доходами). Оператор **With Spisok_sotr** указывает, что дальнейшие действия выполняются с объектом **Spisok_sotr** (где **Spisok_sotr** – имя списка, размещенного на форме). Область действия оператора **With** завершается строкой **End With**. Использование оператора **With** позволяет сократить ссылки на объект (в данном случае – на список **Spisok_sotr**). В операторе **.ColumnCount = 2** устанавливается свойство списка **Spisok_sotr**, задающее количество его колонок (имя списка уже задано в операторе **With**). В операторе **.RowSource = d.Address** свойству списка **RowSource**, задающему источник данных для заполнения списка, присваивается адрес объекта **d** (т. е. диапазона с данными).

Введем процедуру, которая должна выполняться при щелчке по элементу списка:

```
Private Sub Spisok_sotr_Click()
With spisok_sotr
i = .ListIndex
dohod = .List(i, 1)
Call raschet_naloga(dohod, nalog)
Pole_nalog.Value = nalog
Cells(i + 1, 3).Value = nalog
End With
End Sub
```

Здесь в операторе **i = .ListIndex** переменной **i** присваивается номер текущего элемента (точнее, текущей строки) списка **Spisok_sotr**; следует обратить внимание, что строки (как и столбцы) списка нумеруются *с нуля*. В операторе **dohod = .List(i, 1)** переменной **dohod** присваивается величина из **i**-й строки и первого столбца списка **Spisok_sotr**, т. е. величина дохода выбранного сотрудника. Следует еще раз обратить внимание, что столбцы списка нумеруются с нуля, т. е. в нулевом столбце находятся фамилии, а в первом – доходы.

В операторе **Call raschet_naloga(dohod, nalog)** вызывается процедура, реализующая алгоритм расчета налога. Результатом ее выполнения является переменная **dohod**. В двух следующих операторах эта величина выводится в текстовое поле пользовательской формы и на рабочий лист Excel.

Процедура расчета налога может быть, например, следующей:

```
Sub raschet_naloga(ByVal summa_dohoda, summa_naloga)
If summa_dohoda < 20000 Then
summa_naloga = summa_dohoda * 0.09
Else
summa_naloga = 20000 * 0.09 + (summa_dohoda - 20000) * 0.12
End If
End Sub
```

Здесь обозначение **ByVal** указывает, что в переменную **summa_dohoda** передается только значение соответствующей переменной, указанной при

вызове процедуры (в данном случае – значение переменной **dohod**). Тем самым исключается случайное изменение переменной **dohod** в процедуре. Следует обратить внимание, что обозначение **ByVal** относится только к переменной **summa_dohoda**; таким образом, изменение переменной **summa_naloga** вызывает изменение соответствующей переменной **nalog** в вызывающей процедуре.

Приведем также процедуру для закрытия формы нажатием кнопки **Vyhod**:

```
Private Sub Vyhod_Click()
Unload Nalogi
End Sub
```

7.5 Список с возможностью выбора нескольких элементов

Пример 7.4 – Фамилии и доходы сотрудников некоторой организации, введенные на рабочем листе Excel, отображаются в списке, содержащемся в пользовательской форме (рисунок 7.6). При этом должна быть предусмотрена возможность выбрать из списка несколько сотрудников. При нажатии на кнопку **Средний доход** должен вычисляться средний доход выбранных сотрудников.

Следует разместить на форме элементы управления, как показано на рисунке 7.6. Процедура инициализации формы аналогична приведенной в примере 7.3. Кроме того, в этой процедуре для списка, содержащегося в форме, необходимо задать свойство **.MultiSelect = 2**. Это означает, что из списка можно будет выбирать несколько элементов, используя клавиши **Ctrl** и **Shift**.

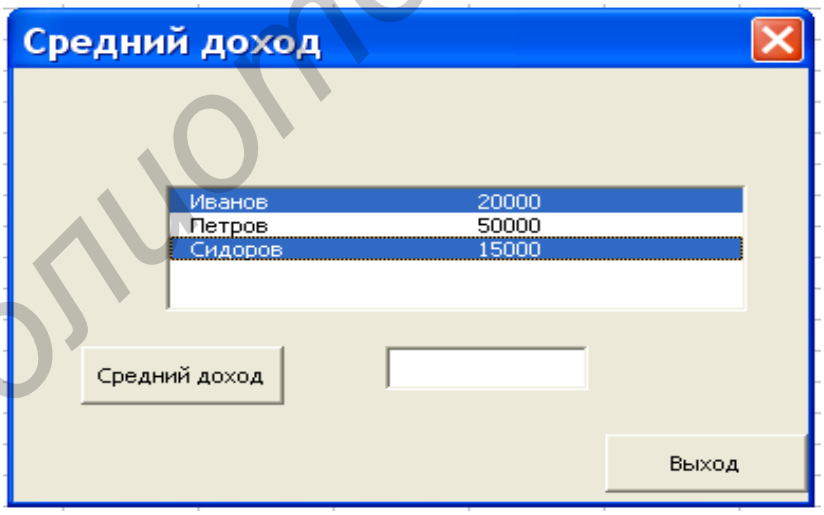

Рисунок 7.6 – Пользовательская форма, содержащая список с возможностью выбора нескольких элементов (пример 7.4)

Процедура, выполняемая при нажатии кнопки **Средний доход**, может быть следующей:

```
Private Sub Sreddohod_Click()
vybrano = 0
summa = 0
With Spisok_sotr
```

```
For i=0 To .ListCount-1
If .Selected(i) = True Then
vybrano = vybrano + 1
summa = summa + .List(i,1)
End If
Next i
If vybrano >0 Then
srednee = summa/vybrano
Pole_sred.Value = srednee
Else
MsgBox(“Ничего не выбрано”)
End If
End With
End Sub
```

Здесь оператор **With Spisok_sotr** указывает, что дальнейшие действия выполняются с объектом **Spisok_sotr** (где **Spisok_sotr** – имя списка, размещенного на форме). Свойство **.ListCount** – количество элементов этого списка. В цикле **For i=0 To .ListCount-1** выполняется перебор элементов списка (начиная с нулевого элемента). Свойство **.Selected(i)** принимает значение **True**, если **i**-й элемент списка выбран. Выполняется подсчет выбранных элементов списка (в переменной **vybrano**), а также суммирование величин, находящихся во втором (по номеру – первом) столбце списка, т. е. доходов. Затем вычисляется среднее и оно выводится в текстовое поле **Pole_sred**.

7.6 Составление списка в программе

Кроме считывания элементов списка из ячеек, заданных свойством **RowSource**, имеется еще один способ добавления элементов в список – использование метода **AddItem**.

Пример 7.5 – На рабочем листе в столбце А указаны номера контрактов, в столбце В – названия городов, где выполняется данный контракт, в столбце С – стоимости контрактов. Каждый контракт выполняется в одном городе; при этом в одном городе может выполняться несколько контрактов. Требуется, чтобы названия городов отображались в списке на пользовательской форме (рисунок 7.7). Каждый город должен быть указан в списке только один раз. При щелчке по названию города должна вычисляться сумма стоимостей контрактов, выполняемых в этом городе.

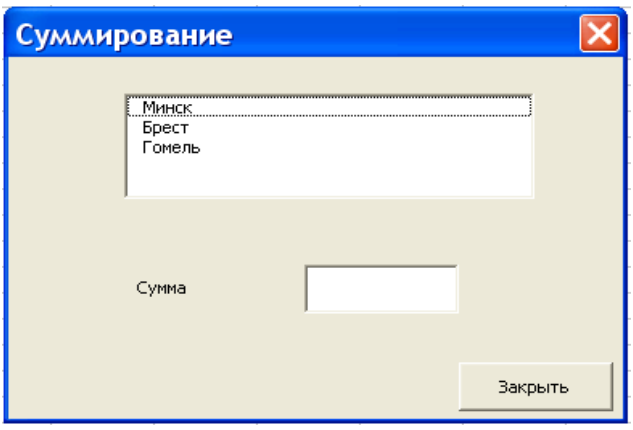

Рисунок 7.7 – Пользовательская форма со списком, составленным по данным с рабочего листа (пример 7.5)

На пользовательской форме необходимо разместить список (с именем **Spisok_gorodov**), надпись **Сумма**, текстовое поле (с именем **Pole_summa**), а также кнопку для закрытия формы. Для инициализации формы можно использовать следующую процедуру:

```
Private Sub Userform_Initialize()
Dim nazv() As String
Set d = Range("A1").CurrentRegion
m = d.Rows.Count
ReDim nazv(1 To m)
kol = 1: nazv(1) = d.Cells(1, 2).Value
For i = 2 To m
nazvanie = d.Cells(i, 2).Value
vkluchenо = False
For j = 1 To kol
If nazvanie = nazv(j) Then vkluchenо = True
Next j
If vkluchenо = False Then
kol = kol + 1
nazv(kol) = nazvanie
End If
Next i
For k = 1 To kol
Spisok_gorodov.AddItem (nazv(k))
Next k
End Sub
```

Здесь массив **nazv** будет использоваться для хранения названий городов. Так как их количество заранее не известно, массив **nazv** объявлен как динамический: его размерность будет указана в операторе **ReDim** после того, как будет определено возможное количество городов. В операторе **Set d = Range("A1"). CurrentRegion** переменная **d** связывается с диапазоном ячеек, где размещаются данные о контрактах. В операторе **m = d.Rows.Count** определяется количество строк в диапазоне ячеек с данными, то есть количество контрактов. В операторе **ReDim nazv(1 To m)** размер массива **nazv** устанавливается в соответствии с количеством контрактов (максимально возможным количеством городов).

Группа операторов **kol = 1** … **Next i** реализует сбор списка названий городов в массиве **nazv**. Сначала в массив включается название города, указанного в первой строке (для первого контракта). Для каждого последующего города выполняется проверка на его наличие в массиве **nazv**; для этого его название (переменная **nazvanie**) сравнивается со всеми названиями, уже имеющимися в массиве **nazv**. Если город еще не включен в этот массив, то он добавляется в него, и количество городов в массиве (переменная **kol**) увеличивается на 1.

В цикле **For k = 1 To kol** … **Next k** элементы массива **nazv** последовательно добавляются в список **Spisok_gorodov.** Для добавления элементов в список используется метод **AddItem.**

Процедура для обработки щелчка по списку городов (т. е. для вычисления суммы стоимостей контрактов по выбранному городу) может быть следующей:

```
Private Sub Spisok_gorodov_Click()
nazvanie = Spisok_gorodov.Text
Set d = Range("A1").CurrentRegion
m = d.Rows.Count
sum = 0
For i = 1 To m
If d.Cells(i, 2).Value = nazvanie Then sum = sum + d.Cells(i, 3).Value
Next i
Pole_summa.Value = sum
End Sub
```

Здесь в операторе **nazvanie = Spisok_gorodov.Text** переменной **nazvanie** присваивается выбранный элемент списка. Затем в диапазоне ячеек, заполненном данными, выполняется суммирование величин из столбца C (т. е. стоимостей контрактов), если соответствующая величина в столбце B (название города) совпадает со значением переменной **nazvanie**. Вычисленная сумма выводится в текстовое поле **Pole_summa**.

7.7 Поля выбора ячеек

Поле выбора ячеек (**RefEdit**) – элемент управления, позволяющий выбирать ячейки на рабочем листе Excel (аналогично тому, как это делается, например, при использовании встроенных функций в Excel).

Пример 7.6 – Требуется разработать программу для сортировки наборов чисел. В программе используется форма, приведенная на рисунке 7.8. В поле выбора ячеек, имеющем подпись **Диапазон**, выбирается диапазон ячеек, заполненных числами, из произвольного числа строк и одного столбца. В поле выбора ячеек **Вывод** указывается ячейка, с которой должен начинаться вывод результатов (отсортированного набора чисел). Сортировка выполняется при нажатии кнопки **Сортировать**. На странице **Параметры** размещаются два переключателя для выбора способа сортировки: по возрастанию или по убыванию.

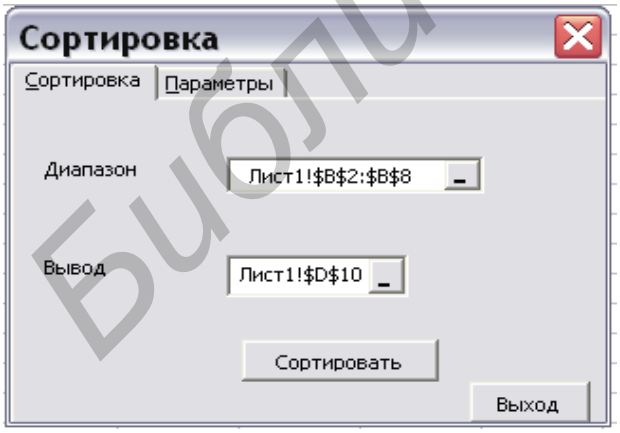

Рисунок 7.8 – Пользовательская форма с полями выбора ячеек (пример 7.6)

Для решения этой задачи следует разместить на пользовательской форме необходимые элементы управления. Сначала следует разместить элемент управления «Набор страниц» (**MultiPage**). Щелкнув правой кнопкой мыши по заголовку первой страницы (**Page1**), выбрать команду **Rename** и указать новое имя (в данном случае – **Сортировка**) в поле **Caption**. Аналогично переименовать вторую страницу (**Page2**).

Разместить остальные элементы управления. На странице **Сортировка** разместить две надписи (**Диапазон** и **Выход**), два поля выбора ячеек (с именами **Diap_sort** и **Diap_vyvod**), две кнопки (с именами **sort** и **vyhod**). На

странице **Параметры** разместить два переключателя (с именами **Sort_po_vozr** и **Sort_po_ub**) для выбора способа сортировки – по возрастанию или по убыванию.

В процедуре инициализации формы предусмотреть, чтобы один из переключателей на странице **Параметры** был установлен, а другой – сброшен.

Для обработки щелчка по кнопке **Сортировать** ввести следующую процедуру (здесь предполагается, что для этой кнопки установлено свойство **Name – Sort**):

```
Private Sub Sort_click()
Dim massiv() As Single
Set d = Range(Diap_sort.Value)
m = d.Rows.Count
n = d.Columns.Count
If n <> 1 Then
MsgBox ("Неправильно выбран диапазон данных")
Exit Sub
End If
Set dv = Range(diap_vyvod.Value)
mm = dv.Rows.Count
nn = dv.Columns.Count
If (mm <> 1) Or (nn <> 1) Then
MsgBox ("Неправильно указан адрес вывода")
Exit Sub
End If
ReDim massiv(1 To m)
For i = 1 To m
massiv(i) = d.Cells(i).Value
Next i
If Sort_po_vozr.Value = True Then
Call sortirovka_v(massiv, m)
Else
Call sortirovka_ub(massiv, m)
End If
For i = 1 To m
dv.Cells(i, 1).Value = massiv(i)
Next i
End Sub
```

Здесь в строке **Dim massiv() As Single** объявлен динамический массив. В операторе **Set d = Range(Diap_sort.Value)** переменная **d** связывается с диапазоном ячеек, заданном в поле выбора ячеек **Diap_sort**. В операторах **m = d.Rows.Count** и **n = d.Columns.Count** определяются размеры выбранного диапазона ячеек. Затем проверяется количество столбцов в выбранном диапазоне: оно должно быть равно 1. Если это не так, выводится сообщение о неправильно выбранном диапазоне данных, и выполняется выход из процедуры (**Exit Sub**).

В операторе **Set dv = Range(diap_vyvod.Value)** переменная **dv** связывается с диапазоном, указанным в поле выбора ячеек **diap_vyvod**. Затем прове-

ряются размеры этого диапазона. Он должен представлять собой в точности одну ячейку (с нее будет начинаться вывод отсортированного массива). Если это не так, то выводится сообщение о неправильно заданном адресе вывода.

В операторе **ReDim massiv(1 To m)** размер массива **massiv** устанавливается в соответствии с выбранным диапазоном. Затем в этот массив передается содержимое ячеек выбранного диапазона.

В зависимости от того, какой из переключателей на странице **Параметры** установлен, вызывается одна из процедур сортировки: по возрастанию (**Sortirovka_v**) или по убыванию (**Sortirovka_ub**). В процедуру передается массив **massiv**, а также его размер (**m**).

Затем в цикле содержимое отсортированного массива **massiv** выводится в столбец рабочего листа Excel, начиная с ячейки, указанной в поле **diap_vyvod**.

Процедура сортировки по возрастанию может быть следующей:

```
Sub sortirovka_v(a, m)
For i = 1 To m – 1 ‘ i-й элемент массива будет сравниваться со всеми последующими
For j = i + 1 To m
If a(i) > a(j) Then
x = a(i): a(i) = a(j): a(j) = x ‘ Если i-й элемент оказался больше j-го, то они меняются местами
End If
Next j
Next i
End Sub
```

Процедура сортировки по убыванию полностью аналогична приведенной процедуре и отличается от нее только знаком операции сравнения.

7.8 Варианты заданий

Вариант 1 – Создать пользовательскую форму, содержащую следующие элементы управления: два поля выбора ячеек с надписями **Диапазон** и **Копия**; текстовое поле с надписью **Столбец**; счетчик; флажок **Сохранить копию**; список из двух элементов (**Поменять местами** и **Заменить**); кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** в диапазоне, заданном в поле выбора ячеек **Диапазон**, первый столбец меняется местами со столбцом, номер которого указан в поле **Столбец** (если в списке выбран элемент **Поменять местами**), или заменяется на этот столбец (если выбран элемент **Заменить**). Номер столбца в текстовом поле **Столбец** задается с помощью счетчика. Если выполняется замена, и при этом установлен флажок **Сохранить копию**, то элементы первого столбца выбранной области копируются в ячейки рабочего листа, начиная с ячейки, выбранной в поле **Копия**. При нажатии кнопки **Выход** форма закрывается.

Вариант 2 – Создать пользовательскую форму, содержащую следующие элементы управления: поле выбора ячеек с надписью **Диапазон**; два тексто-

вых поля с надписями **Число** и **Заменить на**; два счетчика; список из двух элементов (**Подсчет** и **Замена**); кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** в диапазоне, заданном в поле выбора ячеек **Диапазон**, выполняется подсчет количества вхождений числа, введенного в поле **Число**, или его замена на число, указанное в поле **Заменить на**. Числа в поля **Число** и **Заменить на** могут вводиться с клавиатуры или с помощью счетчиков. Выполняемая операция (подсчет или замена) выбирается из списка. Количество найденных или замененных чисел выводится на экран командой **MsgBox**. При нажатии кнопки **Выход** форма закрывается.

Вариант 3 – Создать пользовательскую форму, содержащую следующие элементы управления: поле выбора ячеек с надписью **Диапазон**; два текстовых поля с надписями **Строка 1** и **Строка 2**; два счетчика; список из двух элементов (**Номера** и **Мин-Макс**); флажок **Вывод номеров**; кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** в диапазоне, заданном в поле выбора ячеек **Диапазон**, меняются местами две строки. Если выбран элемент списка **Номера**, то меняются местами строки, номера которых указаны в текстовых полях. Номера в текстовых полях устанавливаются с помощью счетчиков. Если выбран элемент **Мин-Макс**, то меняются местами строки, содержащие минимальный и максимальный элемент выбранной области. Если установлен флажок **Вывод номеров**, но номера строк, которые поменялись местами, выводятся на экран командой **MsgBox**. При нажатии кнопки **Выход** форма закрывается.

Вариант 4 – Создать пользовательскую форму, содержащую следующие элементы управления: два поля выбора ячеек с надписями **Исходные данные** и **Результаты**; текстовое поле с надписью **Число**; счетчик; список из двух элементов (**Строки** и **Всего**); флажок **Вывод номеров**; кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** в диапазоне, заданном в поле выбора ячеек **Исходные данные**, подсчитывается количество строк, содержащих число, указанное в поле **Число**, или общее количество вхождений этого числа. Значение поля **Число** вводится с клавиатуры или с помощью счетчика. Режим подсчета (подсчет строк или общего количества вхождений) выбирается из списка. Результат подсчета выводится на экран командой **MsgBox**. Кроме того, если выполняется подсчет строк, и при этом установлен флажок **Вывод номеров**, то в рабочий лист (начиная с ячейки, указанной в поле **Результаты**) выводятся номера строк, содержащих заданное число. При нажатии кнопки **Выход** форма закрывается.

Вариант 5 – Создать пользовательскую форму, содержащую следующие элементы управления: поле выбора ячеек с надписью **Диапазон**; текстовое поле с надписью **Строка**; счетчик; флажок **Вывести**; список из двух элементов (**По номеру** и **По максимуму**); кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** в диапазоне, заданном в поле выбора ячеек **Диапазон**, первая строка меняется местами со строкой, определяемой в зависимости от выбранного элемента списка: если выбран элемент **По номе-**

ру, то используется строка, номер которой указан в поле **Строка**, а если выбран элемент **По максимуму**, то определяется строка, содержащая максимальный элемент выбранного диапазона. Номер строки в текстовом поле **Строка** задается с помощью счетчика. Если установлен флажок **Вывести**, то на экран выводятся номера и суммы строк, которые поменялись местами. При нажатии кнопки **Выход** форма закрывается.

Вариант 6 – Создать пользовательскую форму, содержащую следующие элементы управления: два поля выбора ячеек с надписями **Исходные данные** и **Результаты**; текстовое поле с надписью **Граница**; счетчик; флажок **Вывести средние**; список из двух элементов (**Граница** и **Общее среднее**); кнопки **Выполнить** и **Выход**.

При нажатии кнопки **Выполнить** по каждой строке вычисляется среднее. В рабочий лист (начиная с ячейки, выбранной в поле **Результаты**) выводятся номера строк, для которых среднее превышает заданную величину. В качестве такой величины используется число, заданное в поле **Граница** (если в списке выбран элемент **Граница**) или среднее, вычисленное по всему выбранному диапазону (если в списке выбран элемент **Общее среднее**). Значение поля **Граница** вводится с клавиатуры или с помощью счетчика. Если при этом установлен флажок **Вывести средние**, то вместе с номерами строк выводятся и их средние (в соседний столбец). При нажатии кнопки **Выход** форма закрывается.

Вариант 7 – На рабочем листе Excel в столбце А введены фамилии студентов, в столбцах В–Е – их оценки по четырем экзаменам. Эти данные отображаются в списке на пользовательской форме (рисунок 7.9).

Должна быть предусмотрена возможность выбора нескольких студентов из списка. При нажатии на кнопку **Вывести** на рабочий лист Excel должны выводиться фамилии студентов, выбранные в списке, или фамилии студентов с максимальными значениями среднего балла (в зависимости от состояния переключателя). Если выбран вывод лучших студентов, то их количество задается в текстовом поле с помощью счетчика. Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод**.

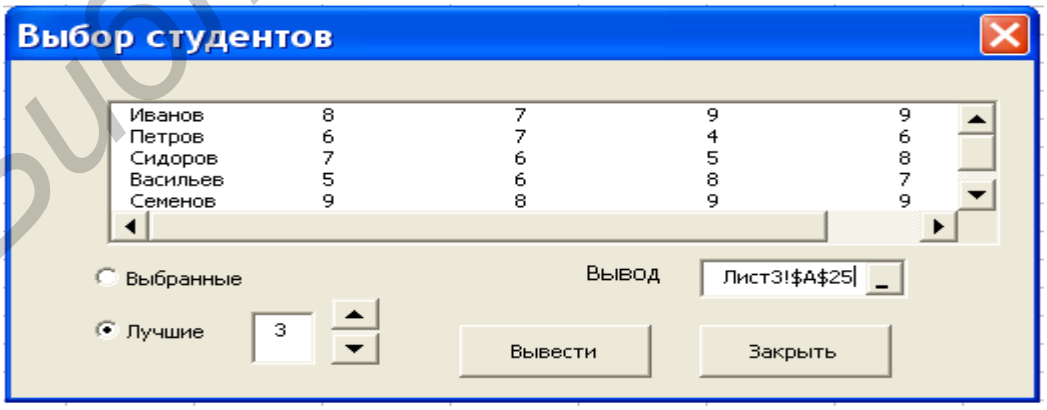

Рисунок 7.9 – Пользовательская форма для варианта задания 7

Вариант 8 – На рабочем листе Excel в столбце А введены номера контрактов, в столбце В – названия товаров (поставляемых по контрактам), в

столбце C – количество товара, в столбце D – цена товара (за единицу). В списке на пользовательской форме (рисунок 7.10) должен отображаться перечень товаров, поставляемых по контрактам, причем каждый товар должен быть указан в списке только один раз.

Должна быть предусмотрена возможность выбора нескольких товаров из списка. При нажатии на кнопку **Вывести** на рабочий лист Excel должны выводиться суммы стоимостей контрактов по каждому из выбранных товаров, а также (в зависимости от настройки флажков на странице **Настройки**) количество контрактов и количество единиц товара (по каждому из выбранных товаров). Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод**.

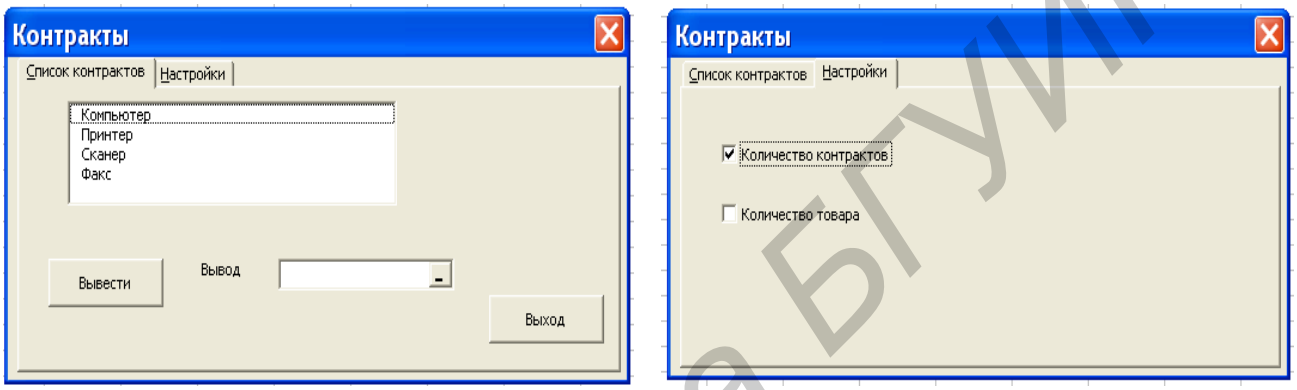

Рисунок 7.10 – Пользовательская форма для варианта задания 8

Вариант 9 – На рабочем листе Excel в столбце A введены номера контрактов, в столбце B – названия поставляемых по контрактам товаров, в столбце C – стоимость контрактов. В списке на пользовательской форме (рисунок 7.11) должен отображаться перечень товаров, поставляемых по контрактам, причем каждый товар должен быть указан в списке только один раз.

Должна быть предусмотрена возможность выбора нескольких товаров из списка. При нажатии на кнопку **Вывести** на рабочий лист Excel должны выводиться суммы стоимостей контрактов по каждому из выбранных товаров или перечень этих контрактов (в зависимости от состояния переключателя). Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод.**

Вариант 10 – На рабочем листе Excel в столбце A введены фамилии студентов, в столбцах B–E – их оценки по четырем экзаменам. Эти данные отображаются в списке на пользовательской форме (рисунок 7.12).

Должна быть предусмотрена возможность выбора нескольких студентов из списка. При нажатии на кнопку **Вывести** на рабочий лист Excel должен выводиться перечень студентов, выбранных в списке, или студентов со средним баллом, превышающим величину в поле **Минимальный балл** (в зависимости от состояния переключателя). Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод**. Для размещения

переключателей и поля выбора ячеек использовать элемент управления **Рамка (Frame)**.

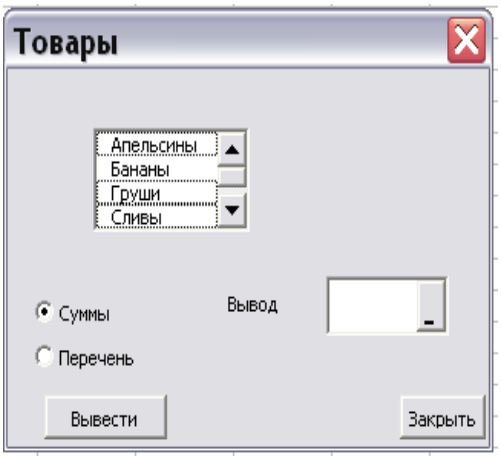

Рисунок 7.11 – Пользовательская форма для варианта задания 9

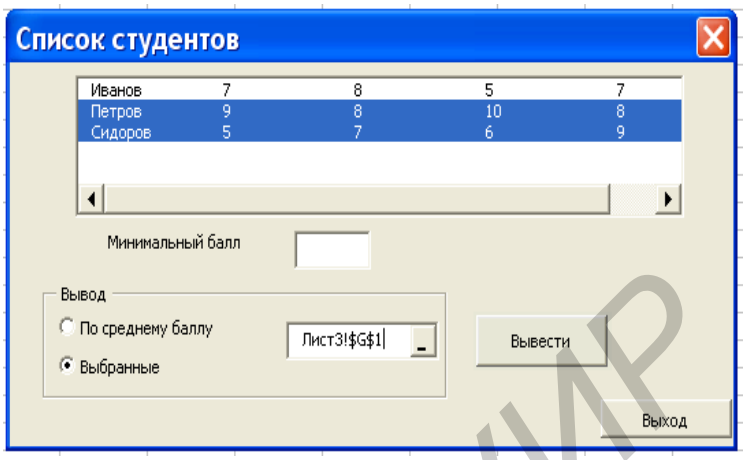

Рисунок 7.12 – Пользовательская форма для варианта задания 10

Вариант 11 – На рабочем листе Excel в столбце А введены номера контрактов, в столбце В – названия заказчиков (в каждом контракте – один заказчик), в столбце С – стоимости контрактов. В списке на пользовательской форме (рисунок 7.13) должен отображаться перечень заказчиков, причем каждый заказчик должен быть указан только один раз.

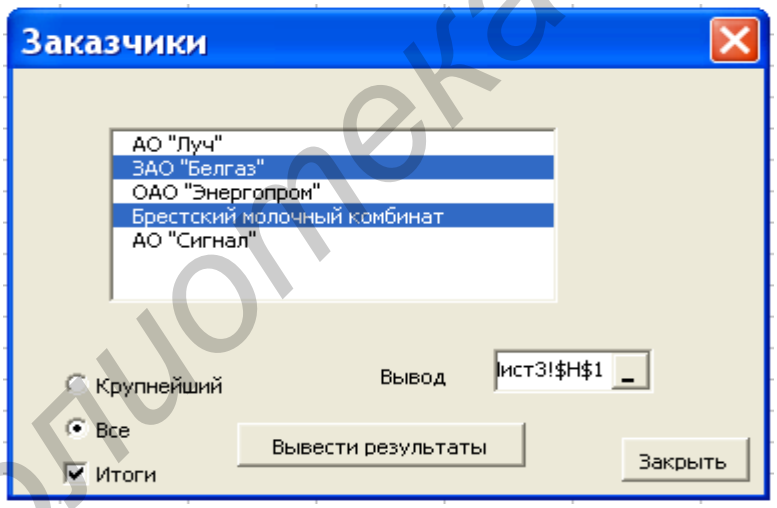

Рисунок 7.13 – Пользовательская форма для варианта задания 11

Должна быть предусмотрена возможность выбора нескольких заказчиков из списка. При нажатии на кнопку **Вывести результаты** на рабочий лист Excel должен выводиться перечень всех заказов выбранных заказчиков или крупнейший (по стоимости) заказ каждого из этих заказчиков (в зависимости от состояния переключателя). Кроме того, если установлен флажок **Итоги**, то для каждого из выбранных заказчиков должно выводиться общее количество заказов и их общая стоимость. Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод**.

Вариант 12 – На рабочем листе Excel в столбце А введены фамилии работников, в столбце В – названия подразделений предприятия, в которых они

работают, в столбце С – заработные платы. В списке на пользовательской форме (рисунок 7.14) должен отображаться перечень подразделений предприятия, причем каждое подразделение указывается только один раз.

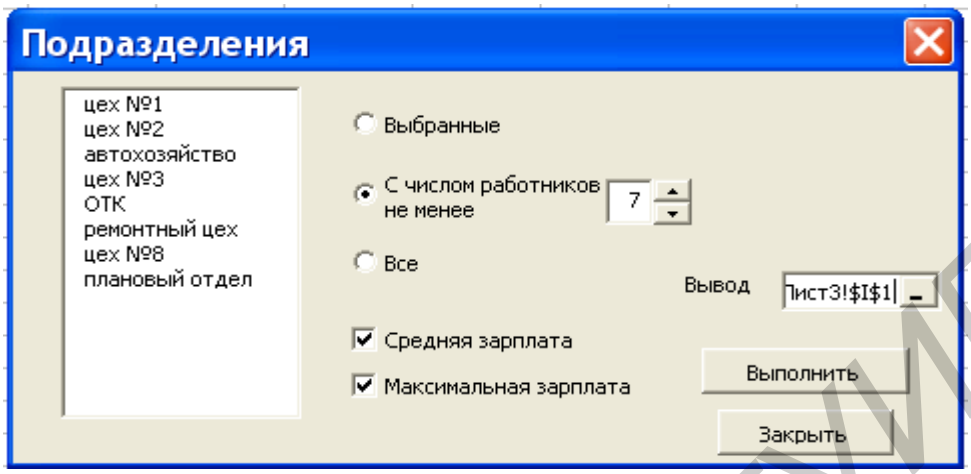

Рисунок 7.14 – Пользовательская форма для варианта задания 12

Должна быть предусмотрена возможность выбора нескольких подразделений из списка. При нажатии на кнопку **Выполнить** на рабочий лист Excel должны выводиться данные о персонале подразделений предприятия: количество работников, а также средняя и максимальная зарплата по подразделению (если установлены соответствующие флажки). В зависимости от состояния переключателя выводятся данные по подразделениям, выбранным в списке, по подразделениям с количеством работников не меньше заданного (это количество устанавливается с помощью счетчика) или по всем подразделениям. Вывод выполняется на рабочий лист, начиная с ячейки, указанной в поле выбора ячеек **Вывод**.

7.9 Порядок выполнения работы

Для выполнения данной работы требуется разработка и отладка двух программ: по одной согласно вариантам 1–6 и 7–12.

Примечание – Так как задания по вариантам 7–12 достаточно сложные, по согласованию с преподавателем они могут выполняться частично: реализуются функции нескольких элементов управления. В этом случае рекомендуется самостоятельно реализовать остальные элементы управления.

7.10 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать:

– титульный лист;
– цель работы;
– листинги разработанных программ;
– копии экранов, иллюстрирующие выполнение программ (пользовательская форма с элементами управления, ввод исходных данных, вывод результатов);

– пояснения по использованию элементов управления, размещенных на пользовательской форме (в виде отдельного текста или комментариев к программам).

7.11 Контрольные вопросы

1 Основные этапы разработки программы с пользовательской формой.

2 Простые элементы управления на пользовательской форме: флажки, переключатели, счетчики, текстовые поля.

3 Списки. Способы заполнения списков.

4 Списки с возможностью выбора нескольких элементов.

5 Списки из нескольких колонок.

6 Возможности пользовательских форм для указания диапазонов ячеек с данными.

Ответы на все контрольные вопросы должны сопровождаться примерами в виде фрагментов программ.

ЛАБОРАТОРНАЯ РАБОТА №8

ПРОГРАММЫ НА VBA ДЛЯ РАБОТЫ С ТЕКСТОВЫМИ ДАННЫМИ

Цель работы – Освоение разработки программ на VBA для работы со строковыми данными и текстовыми файлами.

8.1 Подготовка среды VBA для работы с файлами

Для работы с файлами (и ряда других операций) в VBA используется библиотека **Microsoft Scripting Runtime**. Прежде чем выполнять любую программу, использующую операции с файлами, необходимо подключить эту библиотеку. Для этого необходимо в среде VBA выбрать команду меню **Tools - References** и установить флажок **Microsoft Scripting Runtime**.

Основной объект файловой системы VBA, позволяющий выполнять операции с файлами – объект **filesystemobject**. В начале любой программы, работающей с файлами, его необходимо создать:

```
Set fso = CreateObject("scripting.filesystemobject")
```

8.2 Ввод данных из файла

Пример 8.1 – Имеется текстовый файл (например, подготовленный в программе Блокнот), содержащий некоторые числа. В каждой строке файла содержится одно число. Требуется разработать программу для ввода всех чисел из файла в столбец С рабочего листа Excel, начиная с ячейки С1. Кроме того, требуется предусмотреть вывод сообщения об ошибке и прерывание программы в случае, если в файле будут введены нечисловые данные. Файл должен выбираться пользователем из стандартного окна открытия файла; если пользователь отказывается от загрузки файла (нажатием клавиши ECS), то программа должна прекращать работу.

```
Sub primer8_1()
Set fso = CreateObject("scripting.filesystemobject")
ChDrive("D")
ChDir("D:\User")
otkr_file = Application.GetOpenFilename("Text Files (*.txt), *.txt")
If otkr_file = False Then Exit Sub
Set dan = fso.OpenTextFile(otkr_file, ForReading)
i=0
Do While Not dan.AtEndOfStream
stroka = dan.ReadLine
stroka = Trim(stroka)
If Not (IsNumeric(stroka)) Then
MsgBox ("Ошибка в файле")
Exit Sub
End If
x = CSng(stroka)
i = i + 1
```

```
Cells(i, 3).Value = x
Loop
dan.Close
End Sub
```

Здесь оператор **Set fso = CreateObject("scripting.filesystemobject")** создает объект с именем **fso**, который будет использоваться для операций с файлами (все имена в этом операторе – зарезервированные).

Оператор **ChDrive("D")** устанавливает в качестве текущего диск **D**; оператор **ChDir("D:\User")** задает текущий путь.

Примечание – Оператор **ChDir** делает текущим указанный каталог на указанном диске, но не изменяет текущий диск. Поэтому перед оператором **ChDir** в данном примере указан оператор **ChDrive**, задающий текущий диск.

Оператор **otkr_file = Application.GetOpenFilename("Text Files (*.txt), *.txt")** открывает стандартное окно загрузки файла. В нем отображается перечень файлов, содержащихся в текущем каталоге и соответствующих заданному шаблону (в данном случае – файлов с расширением .TXT). Пользователь выбирает один из файлов. Имя выбранного файла присваивается заданной переменной, в данном случае – переменной **otkr_file**; эта переменная будет строковой (т. е. имеет тип **String**). Если пользователь отказывается от загрузки файла (например, нажатием клавиши ESC), то переменная **otkr_file** становится логической (тип **Boolean**) и принимает значение **False**.

Примечание – Шаблон файлов требуется указывать именно так, как показано в данном примере, т. е. указывать сначала описание файлов (**Text Files (*.txt)**), затем – указание расширения (***.txt**). Например, если бы потребовалось, чтобы в окне загрузки перечислялись не TXT-, а XLS-файлы, то оператор открытия окна загрузки файла имел бы следующий вид: **otkr_file = Application.GetOpenFilename ("Excel Files (*.xls), *.xls")**. Если бы требовалось перечислить в окне загрузки все файлы, то оператор был бы указан в следующей форме: **otkr_file = Application.GetOpenFilename().**

Оператор **If otkr_file = False Then Exit Sub** проверяет, не отказался ли пользователь от загрузки файла. В случае отказа от загрузки работа программы завершается (оператор **Exit Sub**).

В операторе **Set dan = fso.OpenTextFile(otkr_file, ForReading)** файл с заданным именем (хранящимся в переменной **otkr_file**) открывается для чтения (**ForReading**), т. е. из него можно будет только читать данные. С этим файлом связывается объект **dan**, который будет использоваться для последующих операций с этим файлом.

В операторе **i = 0** задается начальное значение переменной **i**; она будет затем использоваться в качестве номера строки на рабочем листе Excel и при необходимости увеличиваться.

Оператор **Do While Not dan.AtEndOfStream** – начало цикла, который будет повторяться, пока не будет достигнут конец файла, связанного с объектом **dan**. В данном случае это файл, выбранный пользователем и открытый ранее для чтения (см. выше).

В операторе **stroka = dan.ReadLine** вводится одна строка из заданного файла; эта строка присваивается переменной **stroka** (здесь **ReadLine** – операция ввода). В операторе **stroka = Trim(stroka)** функция **Trim** удаляет из переменной **stroka** начальные и концевые пробелы, т. е. пробелы в начале и в конце строки (если они есть).

В операторе **If Not (IsNumeric(stroka)) Then …** проверяется, является ли переменная **stroka** числом. Для этого используется функция **IsNumeric**, принимающая значение **True**, если ее аргумент – число, и **False** – если это не число (например, если в переменной **stroka** содержится буква или пробел). Если оказывается, что **stroka** – не число, то выводится сообщение об ошибке **MsgBox ("Ошибка в файле")** и программа завершается (**Exit Sub**).

Если оказывается, что переменная **stroka** содержит число, то в операторе **x = CSng(stroka)** эта переменная преобразуется в переменную типа **Single** (функция **CSng**). Затем номер текущей строки на рабочем листе Excel увеличивается на единицу (**i = i + 1**), и переменная **x** выводится в очередную ячейку столбца С, т. е. третьего столбца (**Cells(i, 3).Value = x**).

По окончании цикла, т. е. после достижения конца файла (**dan.AtEndOfStream**), файл закрывается (оператор **dan.Close**).

8.3 Вывод данных в файл

Пример 8.2 – На рабочем листе Excel в столбце А (начиная с ячейки А1) хранятся номера контрактов, в столбце В – количество поставляемого товара по соответствующему контракту, в столбце С – цена за одно изделие. Требуется вывести в текстовый файл номера и стоимости контрактов, стоимость которых превышает 10 000 ден. ед. В текстовом файле номер и стоимость контракта должны располагаться в одну строку и разделяться пробелом. На рабочем листе Excel имеется также текстовое поле **Imya_faila**, где пользователь указывает имя файла для вывода данных. Если файл с указанным именем уже существует, то данные должны добавляться в него.

```
Sub primer8_2()
Set fso = CreateObject("scripting.filesystemobject")
rez_file = Imya_faila.Value
Set vyvod = fso.OpenTextFile(rez_file, ForAppending, True)
Set d = Cells(1, 1).CurrentRegion
m = d.Rows.Count
For i = 1 To m
stoimost = d.Cells(i, 2).Value * d.Cells(i, 3).Value
If stoimost > 10000 Then
nomer = d.Cells(i, 1).Value
stroka = CStr (nomer) + " " + CStr(dohod)
vyvod.WriteLine (stroka)
End If
Next i
vyvod.Close
End Sub
```

Здесь, как и в предыдущем примере, оператор **Set fso = CreateObject("scripting.filesystemobject")** создает объект с именем **fso**, который будет использоваться для операций с файлами (все имена в этом операторе – зарезервированные).

В операторе **rez_file = Imya_faila.Value** переменной **rez_file** присваивается значение текстового поля **Imya_faila**, где пользователем должно быть указано имя файла для вывода данных.

В операторе **Set vyvod = fso.OpenTextFile(rez_file, ForAppending, True)** файл с заданным именем (хранящимся в переменной **rez_file**) открывается для добавления данных (**ForAppending**). Значение **True** указывает, что в случае, если указанный файл не существует, он будет создан (и также открыт для добавления данных). С открываемым файлом связывается объект **vyvod**.

Затем определяется заполненная данными область рабочего листа, начиная с ячейки **Cells(1,1)** (т. е. с A1); этой области будет соответствовать объект **d**. Определяется количество строк (**m = d.Rows.Count**).

Для каждой строки вычисляется стоимость указанного в ней контракта (переменная **stoimost**). Если она превышает 10 000, то из номера контракта и его стоимости составляется строка: **stroka = CStr (nomer) + " " + CStr(stoimost)**. Здесь **CStr** – функция преобразования из числовых данных в строковые, знак + обозначает операцию сцепления строк. В операторе **vyvod.WriteLine(stroka)** полученная строка выводится в файл, соответствующий объекту **vyvod**. Здесь **WriteLine** – операция вывода строки в файл.

По окончании цикла, т. е. после вывода в файл всех необходимых строк, файл закрывается (оператор **vyvod.Close**).

8.4 Обмен данными между двумя файлами

Пример 8.3 – Имеется некоторый текстовый файл, где в каждой строке находится или некоторое число, или некоторый текст. Требуется вывести все числа из этого файла, превышающие 10, в другой файл. Имя файла с исходными данными должно выбираться пользователем. Имя файла для вывода результатов – D:\User\chisla.txt. Если этот файл уже существует, то его прежнее содержимое должно удаляться.

```
Sub primer8_3()
Set fso = CreateObject("scripting.filesystemobject")
ChDrive "D"
ChDir "D:\User"
otkr_file = Application.GetOpenFilename("Text Files (*.txt), *.txt")
rez_file = "D:\User\chisla.txt"
Set ishod = fso.OpenTextFile(otkr_file, ForReading)
Set rezult = fso.OpenTextFile(rez_file, ForWriting, True)
Do While Not ishod.AtEndOfStream
stroka = ishod.ReadLine
If IsNumeric(stroka) Then
x = CSng(stroka)
If x > 10 Then rezult.WriteLine (x)
```

```
End If
Loop
ishod.Close
rezult.Close
End Sub
```

В операторе **Set rezult = fso.OpenTextFile(rez_file, ForWriting, True)** файл с заданным именем (хранящимся в переменной **rez_file**) открывается для записи данных с потерей прежних данных (**ForWriting**). Значение **True** указывает, что в случае, если указанный файл не существует, он будет создан. Если бы требовалось добавлять данные в этот файл (с сохранением прежних данных), то вместо **ForWriting** было бы указано **ForAppending**. С открываемым файлом связывается объект **rezult**.

Следует обратить внимание, что в конце работы программы необходимо закрыть все использовавшиеся в ней файлы (в данном примере – два файла, с которыми были связаны объекты **ishod** и **result**).

8.5 Функции обработки строк

В таблице 8.1 приведены некоторые встроенные функции языка VBA для обработки строк.

Таблица 8.1 – Основные функции обработки строк в VBA

Функция и ее аргументы	Возвращаемое значение
Len (строка)	Длина строки
Left (строка, длина)	Подстрока заданной длины, выделенная из исходной строки слева
Right (строка, длина)	Подстрока заданной длины, выделенная из исходной строки справа
Mid (строка, начало, длина)	Подстрока заданной длины, выделенная из исходной строки, начиная с символа с номером «начало»
Trim (строка)	Заданная строка без начальных и концевых пробелов
LTrim (строка)	Заданная строка без начальных пробелов
RTrim (строка)	Заданная строка без концевых пробелов
LCase (строка)	Заданная строка, преобразованная в строчные буквы
UCase (строка)	Заданная строка, преобразованная в заглавные буквы
InStr (начало, исходная строка, искомая подстрока)	Позиция искомой подстроки в исходной строке, начиная с позиции с номером «начало». Если «начало» не указано, то поиск выполняется с первой позиции. Если искомая подстрока отсутствует, возвращается значение 0
Replace (исходная строка, заменяемая подстрока, заменяющая подстрока, начало, количество замен)	В исходной строке одна подстрока заменяется на другую. Замена выполняется, начиная с позиции с номером «начало». Если «начало» не указано, то замена выполняется с первой позиции. Если не указано «количество замен», то заменяются все вхождения заменяемой подстроки
StrComp (строка1, строка2)	Выполняется сравнение заданных строк. Возвращается значение 1, если «строка1» больше «строки2», значение 0, если строки равны, и значение –1, если «строка1» меньше «строки2»

Приведем примеры программ с использованием этих функций.

Пример 8.4 – Имеется текстовый файл, где указаны фамилии людей и их доходы в следующей форме:

Иванов 200
Петров 500
Сидоров 120

Требуется вывести эти данные на рабочий лист Excel: фамилии – в столбец А, доходы – в столбец В. Имя файла должно запрашиваться у пользователя в стандартном окне открытия файла.

```
Sub primer8_4()
Set fso = CreateObject(“Scripting.FileSystemObject”)
ChDrive (“D”)
ChDir(“D:\User”)
otkr_file = Application.GetOpenFilename()
If otkr_file = False Then Exit Sub
Set vvod = fso.OpenTextFile(otkr_file, ForReading)
i = 0
Do While Not vvod.AtEndOfStream
stroka = vvod.ReadLine
stroka = Trim(stroka)
k = InStr(stroka,” “)
If k = 0 Then
MsgBox(“Ошибка в файле данных”)
Exit Sub
End If
familia = Left(stroka, k - 1)
dlina = Len(stroka)
dohod_stroka = Right(stroka, dlina - k)
dohod_stroka = Trim(dohod_stroka)
If IsNumeric(dohod_stroka) = False Then
MsgBox(“Ошибка в файле данных”)
Exit Sub
End If
dohod=CSng(dohod_stroka)
i = i + 1
Cells(i,1).value = familia
Cells(i,2).value = dohod
Loop
Vvod.Close
End Sub
```

Начальные действия, выполняемые в этой программе (выбор и открытие файла, а также организация ввода данных до конца файла с помощью оператора **Do While**), рассмотрены ранее.

В операторе **stroka = vvod.ReadLine** вводится одна строка из заданного файла. Эта строка присваивается переменной **stroka**. В операторе **stroke = Trim(stroka)** функция **Trim** удаляет из переменной **stroka** начальные и концевые пробелы, т. е. пробелы в начале и в конце строки (если они есть).

В операторе **k=InStr(stroka,” “)** определяется позиция первого пробела в переменной **stroka**. Например, если переменная **stroka** имеет значение «Иванов 200», то переменная **k** получит значение 7. Если пробел в переменной **stroka** отсутствует (**k = 0**), значит, данные в файле заданы неправильно, и программа завершается с выдачей сообщения об ошибке в файле данных.

В операторе **familia = Left(stroka, k – 1)** из переменной **stroka** выделяются ее левые символы до первого пробела, т. е. фамилия.

Оператор **dlina = Len(stroka)** определяет длину строки, введенной из файла (эта длина включает фамилию, доход, а также пробелы между ними). В операторе **dohod_stroka = Right(stroka, dlina - k)** выделяется правая часть строки. Например, для строки «Иванов 200» переменные получат значение **k = 7**, **dlina**=10 (если между фамилией и доходом указан один пробел). Таким образом, функция **Right(stroka, dlina - k)** выделит из переменной **stroka** три правых символа, т. е. подстроку «200».

Оператор **dohod_stroka = Trim(dohod_stroka)** приведен на случай, если между фамилией и доходом будет указано несколько пробелов. Функция **Trim** удалит их.

Затем с помощью функции **IsNumeric** проверяется, действительно ли выделенная часть строки (переменная **dohod_stroka**) представляет собой число. Если это не так (функция **IsNumeric** возвращает значение **False**), значит, данные в файле заданы неправильно. В этом случае выводится сообщение об ошибке, и программа прерывается (оператором **Exit Sub**).

После проверки переменная **dohod_stroka** преобразуется в число (переменная **dohod**) с помощью функции **CSng**. В операторе **i = i + 1** вычисляется номер очередной строки на рабочем листе Excel, в которую требуется вывести фамилию и доход. Затем эти величины выводятся в заданные ячейки.

Примечание – Функции для работы со строками, как и другие функции, могут быть «вложены» друг в друга. Например, вместо операторов **dohod_stroka = Right(stroka, dlina-k)** и **dohod_stroka = Trim(dohod_stroka)** можно было использовать один оператор: **dohod_stroka = Trim(Right(stroka, dlina - k))**.

Пример 8.5 – В столбце А рабочего листа **Лист1** указаны фамилии, имена и отчества работников некоторой организации. В каждой ячейке указаны фамилия, имя и отчество (полностью) одного работника. Между фамилией и именем, а также между именем и отчеством указано в точности по одному пробелу. Требуется получить на рабочем листе **Лист2** список работников, содержащий их фамилии и инициалы. Список должен быть упорядочен по алфавиту.

```
Sub primer8_5()
Dim spisok() As String                  ‘Объявление массива для списка работников
Set d = Range(“A1”).CurrentRegion       ‘Определение диапазона с данными
m = d.Rows.Count                        ‘Определение количества строк (работников)
ReDim spisok(1 To m)                    ‘Объявление массива необходимого размера
```

```
For i = 1 To m                                  ‘Перебор всех работников
fam_im_ot = Trim(Cells(i,1))                    ‘Содержимое ячейки присваивается
                                                ‘переменной fam_im_ot. Если есть начальные
                                                ‘или конечные пробелы, то они удаляются
probel1 = InStr(fam_im_ot,” “)                  ‘Определяется позиция первого пробела (между
                                                ‘фамилией и именем)
fam_i = Left(fam_im_ot, probel1 + 1)            ‘Выделяется фамилия и первая буква имени
probel2 = InStr(probel1 + 1, fam_im_ot, ” “)    ‘Определяется позиция второго пробела (между
                                                ‘именем и отчеством). Поиск начинается
                                                ‘с позиции, следующей за первым пробелом
init_ot = mid(fam_im_ot, probel2 + 1, 1)        ‘Выделяется первая буква отчества
fam_io = fam_i+”.“+init_ot+”.”                  ‘Фамилия и инициалы соединяются
spisok(i) = fam_io                              ‘Присваивание элементу массива
Next i                                          ‘Переход к следующему работнику

‘ Сортировка списка работников (массива spisok)
For i = 1 To m – 1                              ‘ФИО каждого работника будет сравниваться
For j = i + 1 To m                              ‘со всеми последующими
If StrComp(spisok(i), spisok(j)) = 1 Then       ‘Если ФИО i-го работника «больше», чем j-го,
x = spisok(i)                                   ‘то они меняются местами
spisok(i) = spisok(j)
spisok(j) = x
End If
Next j
Next i

‘ Вывод отсортированного массива на рабочий лист Лист2
Worksheets("Лист2").Activate                    ‘Рабочий лист становится текущим
For i = 1 To m
Cells(i,1).Value = spisok(i)                    ‘Вывод в ячейки рабочего листа, в столбец А
Next i
End Sub
```

Функции, используемые в этой программе, описаны в таблице 8.1. Приведем лишь пояснения относительно функции **StrComp**, используемой для «сравнения» строк при их сортировке по алфавиту. При сравнении строк «большей» считается строка, расположенная «ниже» по алфавиту. Например, если в приведенной программе некоторый (**i**-й) элемент массива **spisok** будет содержать значение «Иванов Ю. С.», а **j**-й элемент этого массива – значение «Яшин А. П.», то функция **StrComp(spisok(i), spisok(j))** вернет значение –1, так как строка «Иванов Ю. С.» считается «меньшей», чем строка «Яшин А. П.». В этом случае элементы массива не будут меняться местами. Если же, например, **i**-й элемент массива **spisok** содержит значение «Антонов С. К.», а **j**-й – значение «Андреев Ф. Р.», то функция **StrComp(spisok(i), spisok(j))** вернет значение 1, и элементы массива поменяются местами.

Примечание – В данном примере (в отличие от всех предыдущих примеров в этой лабораторной работе) все данные размещаются в рабочих листах Excel, а текстовые файлы не используются. Поэтому действия, описанные в разделе 8.1, в данном случае не требуются.

8.6 Варианты заданий

Вариант 1 – На рабочем листе **Лист1** в столбце А указаны фамилии людей, в столбце В – их адреса (адрес включает почтовый индекс, город, улицу, дом и квартиру). Имеется также текстовый файл со списком городов (в каждой строке – один город). Программа должна выводить в новый текстовый файл и на рабочий лист **Лист2** перечень людей, проживающих в городах, названия которых указаны в исходном текстовом файле. Этот перечень должен быть упорядочен по названиям городов.

Вариант 2 – На рабочем листе в столбце А расположены фамилии работников некоторой организации, в столбце В – их доходы. Имеется также текстовый файл, где указаны дополнительные доходы некоторых из этих работников: в каждой строке файла – фамилия и дополнительный доход, разделенные пробелами (одним или несколькими). Порядок фамилий в файле и на рабочем листе может не совпадать. Программа должна прибавлять дополнительные доходы, указанные в текстовом файле, к доходам, содержащимся в столбце В. Кроме того, перечень работников и их доходов, упорядоченный по алфавиту, должен выводиться в новый текстовый файл.

Вариант 3 – На рабочем листе Excel в столбце А расположены номера контрактов, в столбце В – названия товаров, в столбце С – цены на товары, в столбце D – количества товаров. Цена конкретного товара во всех контрактах одинакова. Имеется также текстовый файл, где указаны названия некоторых из товаров и коэффициенты повышения цен на них: в каждой строке файла – название товара и коэффициент повышения цены, разделенные пробелами. Программа должна вычислять новые цены и выводить их в столбец С взамен старых. Кроме того, данные обо всех контрактах, для которых была изменена цена товара, должны выводиться в новый текстовый файл. Эти данные должны быть упорядочены по названиям товаров.

Вариант 4 – На рабочем листе в столбце А введены фамилии работников, в столбце В – номера отделов, где они работают (в одном отделе может быть несколько работников), в столбце С – их зарплаты. Имеется также текстовый файл, где указаны номера отделов и коэффициенты повышения зарплаты: в каждой строке файла – номер отдела и коэффициент повышения зарплаты для его работников, разделенные пробелами. Программа должна вычислять новые зарплаты и выводить их в столбец С взамен старых. Кроме того, программа должна вычислять для каждого отдела количество работников и сумму их зарплат (считать, что в файле с коэффициентами повышения зарплат перечислены все отделы). Эти данные должны выводиться в новый текстовый файл: каждая строка в этом файле должна содержать данные по одному отделу. Данные в новом файле должны быть упорядочены по номеру отдела.

Вариант 5 – На рабочем листе **Лист1** в столбце А перечислены названия валют, в столбце В – их курсы в долларах. Имеется также текстовый файл, где приведены названия товаров, цены этих товаров и названия валют, в ко-

торых указаны цены. В каждой строке файла имеются данные по одному товару, разделенные пробелами. Программа должна выводить в новый текстовый файл названия товаров и их цены в долларах. Перечень должен быть упорядочен по названиям товаров. Кроме того, данные о товарах (название товара, исходная цена, валюта, цена в долларах) должны выводиться на рабочий лист **Лист2.**

Вариант 6 – На рабочем листе в столбце A расположены фамилии студентов, в столбцах B–D – результаты сдачи ими трех экзаменов. Имеется также текстовый файл, где указаны фамилии этих же студентов и результаты сдачи ими четвертого экзамена (в каждой строке файла – фамилия и оценка, разделенные пробелами). Порядок фамилий в файле и на рабочем листе может не совпадать. Программа должна выводить оценки по четвертому экзамену в столбец E. Программа должна также создавать два текстовых файла: в один из них должны выводиться фамилии студентов со средним баллом выше 8 (с указанием среднего балла), во второй – фамилии остальных студентов. Оба файла должны быть упорядочены по фамилиям.

Вариант 7 – Имеется текстовый файл. Программа должна подсчитывать количество строк в этом файле, содержащих хотя бы одну цифру.

Вариант 8 – Имеется текстовый файл, в каждой строке которого несколько слов, разделенных пробелами. Программа должна по каждой строке файла составлять аббревиатуру из первых букв слов, составляющих эту строку. Аббревиатуры выводятся в столбец рабочего листа Excel.

Вариант 9 – Имеется текстовый файл, в каждой строке которого несколько слов, разделенных пробелами. Программа должна подсчитывать количество слов в каждой строке. Результаты выводятся в столбец рабочего листа Excel.

Вариант 10 – Программа должна запрашивать с клавиатуры строковую переменную. В качестве этой переменной вводится строка из нескольких слов, разделенных пробелами. Программа должна выводить эти слова в обратном порядке.

Вариант 11 – Имеется текстовый файл, в каждой строке которого – несколько слов, разделенных пробелами (одним или несколькими). Программа должна создавать новый текстовый файл, состоящий из тех же строк, но интервалы между словами должны состоять только из одиночных пробелов.

Вариант 12 – Имеется текстовый файл, в каждой строке которого – несколько слов, разделенных пробелами. Из каждой строки этого файла программа должна выделять первое и последнее слово. Из полученных строк должен создаваться новый текстовый файл.

8.7 Порядок выполнения работы

Для выполнения данной работы требуется разработка и отладка двух программ: по одной согласно вариантам 1–6 и 7–12.

Примечание – Варианты 1–6 представляют собой задачи на операции с файлами, варианты 7–12 – задачи на обработку строк.

8.8 Содержание отчета

Отчет оформляется в формате .DOC. Отчет должен содержать:

– титульный лист;
– цель работы;
– листинги разработанных программ;
– копии экранов, иллюстрирующие выполнение программ (выбор файлов для ввода–вывода данных, ввод исходных данных, вывод результатов);
– пояснения по операциям с файлами и строковыми данными (в виде отдельного текста или комментариев к программам).

8.9 Контрольные вопросы

1 Возможности выбора файла для ввода и вывода данных.
2 Ввод данных из файла.
3 Вывод данных в существующий файл.
4 Преобразование числовых данных в строковые и строковых – в числовые.
5 Примеры функций для работы со строковыми данными.
6 Операции со строковыми данными: поиск заданной подстроки в строке.
7 Операции со строковыми данными: составление новой строки.
8 Операции со строковыми данными: выделение подстроки.

Ответы на все контрольные вопросы должны сопровождаться примерами в виде фрагментов программ.

Литература

1 Шерстев, В. Л. Компьютерные информационные технологии / В. Л. Шерстнев. – Витебск : ВГТУ, 2006. – 350 с.

2 Кеттелл, Дж. Microsoft Office 2003. Полное руководство / Дж. Кеттелл, Г. Харт-Дэвис, К. Симмонс. – М. : ЭКОМ, 2006. – 832 с.

3 Меженный, О. А. Microsoft Office 2003. Краткое руководство. – М. : Издательский дом «Вильямс», 2005. – 368 с.

4 Гарнаев, А. Ю. Excel, VBA, Internet в экономике и финансах. – СПб. : БХВ–Петербург, 2002. – 816 с.

5 Слепцова, Л. Д. Программирование на VBA: Самоучитель. – М. : Издательский дом «Вильямс», 2004. – 384 с.

6 Программирование в пакетах Microsoft Office / С. В. Назаров [и др.]. М. : Финансы и статистика, 2007. – 656 с.

7 Ананьев, А. И. Самоучитель Visual Basic 6.0 / А. И. Ананьев, А. Ф. Федоров. – СПб. : БХВ–Петербург, 2005. – 624 с.

СОДЕРЖАНИЕ

Св. план 2010, поз. 38

Учебное издание

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 1

Батин Николай Владимирович
Хаджинова Наталья Владимировна

ПРИМЕНЕНИЕ ПАКЕТА MS OFFICE ДЛЯ ОБРАБОТКИ ИНФОРМАЦИИ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *Т. Н. Крюкова*, *Е. С. Чайковская*
Корректор *А. В. Бас*
Компьютерная правка, оригинал-макет *Ю. Ч. Клочкевич*

Подписано в печать 24.01.2012. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л 8,25. Уч.-изд. л. 7,4. Тираж 100 экз. Заказ 792.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6