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

Л. А. Глухова, Е. П. Фадеева, Е. Е. Фадеева

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 4-х частях

Часть 4

Рекомендовано УМО по образованию в области информатики и радиоэлектроники для специальностей, закрепленных за УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия

Библиотека БГУИР

Минск БГУИР 2013

УДК 004.421(076.5)
ББК 32.973.26-018.2я73
Г55

Р е ц е н з е н т ы:

кафедра программного обеспечения вычислительной техники и автоматизированных систем Белорусского национального технического университета (протокол №2 от 18.10.2012);

заведующий кафедрой информатики учреждения образования «Минский государственный высший радиотехнический колледж», кандидат технических наук, доцент Ю. А. Скудняков

Глухова, Л. А.

Г55 Основы алгоритмизации и программирования. Лабораторный практикум : учеб.-метод. пособие. В 4 ч. Ч. 4 / Л. А. Глухова, Е. П. Фадеева, Е. Е. Фадеева. – Минск : БГУИР, 2012. – 58 с. : ил.
ISBN 978-985-488-890-3 (ч. 4).

В четвертой части лабораторного практикума рассмотрены вопросы разработки программ с использованием среды программирования Borland Delphi 7. Описана среда Delphi. Приведены примеры разработки и отладки простейшей консольной программы и программы, имеющей модульную структуру. Даны варианты индивидуальных заданий для выполнения лабораторных работ по темам: множества; записи, типизированные файлы; динамические структуры.

Первая, вторая и третья части изданы в БГУИР соответственно в 2004, 2005 и 2007 гг.

УДК 004.421(076.5)
ББК 32.973.26-018.2я73

ISBN 978-985-488-890-3 (ч. 4)
ISBN 978-985-488-183-6
ISBN 985-444-616-6

© Глухова Л. А., Фадеева Е. П., Фадеева Е. Е., 2013
© УО «Белорусский государственный университет информатики и радиоэлектроники», 2013

Библиотека БГУИР

СОДЕРЖАНИЕ

Библиотека БГУИР

ВВЕДЕНИЕ

Данный лабораторный практикум представляет собой четвертую часть лабораторного практикума по дисциплине «Основы алгоритмизации и программирования» для студентов специальности I-40 01 01 «Программное обеспечение информационных технологий».

В первой части лабораторного практикума приведены сведения для выполнения лабораторной работы №1. Дано описание среды программирования Borland Pascal 7.0. Рассмотрены правила разработки и отладки программы, написанной на языке Pascal, на примере циклической программы с известным числом повторений. Приведены варианты индивидуальных заданий по лабораторной работе №1. Даны варианты заданий для самостоятельной работы.

Вторая часть лабораторного практикума посвящена методам и алгоритмам сортировки совокупностей однотипных объектов. Рассмотрены методы сортировки массивов и файлов. Приведены два вида вариантов индивидуальных заданий к лабораторным работам: по сортировке одномерных массивов и сортировке в двумерных массивах. Даны варианты заданий для самостоятельной работы.

Третья часть лабораторного практикума посвящена вопросам модульного построения программ с использованием процедур и функций. Рассмотрены различные механизмы передачи параметров в вызываемые подпрограммы. Рассмотрены особенности рекурсивной организации подпрограмм. Даны варианты индивидуальных заданий для выполнения лабораторных работ.

Библиотека БГУИР

1. СРЕДА BORLAND DELPHI 7

Среда программирования Borland Delphi 7 представляет собой интегрированную среду, включающую экранный текстовый редактор, компилятор, редактор связей, отладчик, а также ряд других инструментов, позволяющих упростить процесс разработки приложений, предназначенных для выполнения под управлением ОС Windows (дизайнер форм, палитра компонентов, инспектор объектов, менеджер проектов и т. д.).

После запуска Delphi 7 на экране появляется Splash-окно среды Delphi, и через несколько секунд после загрузки необходимых компонентов на экране отображаются собственно окна среды программирования (рис. 1.1).

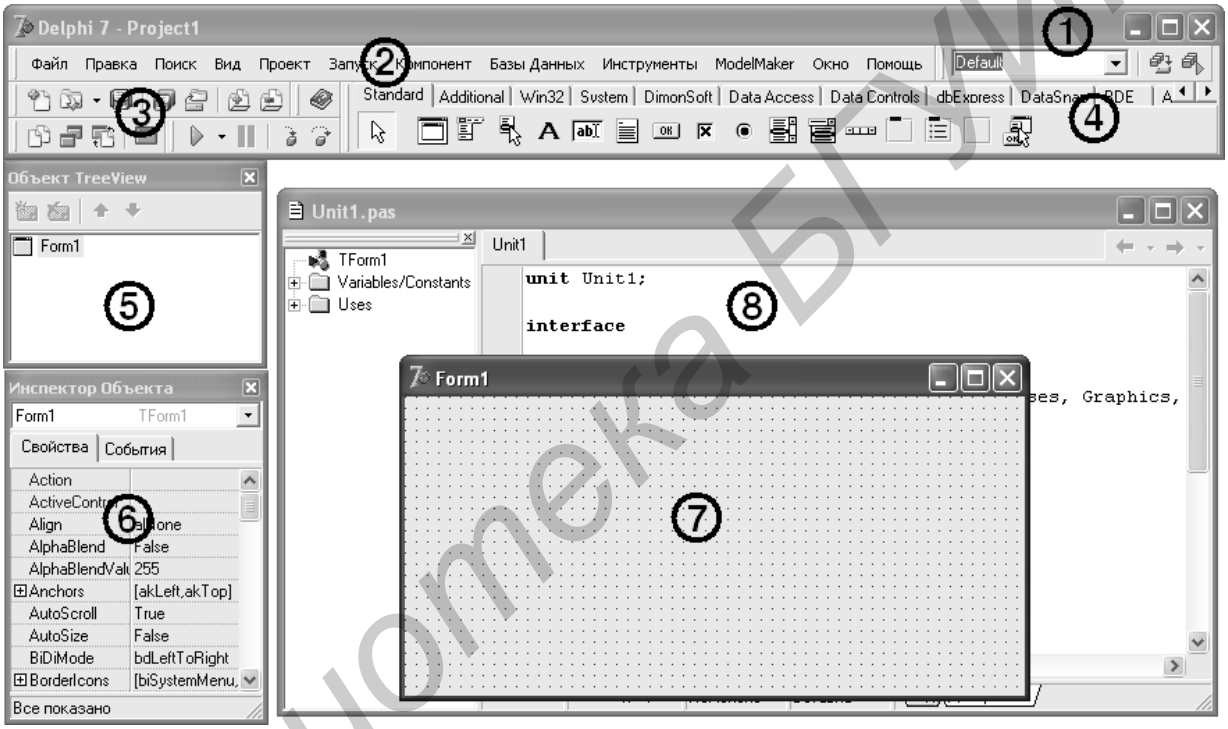


Рис. 1.1. Среда программирования Delphi 7

В верхней части экрана располагается главное окно Delphi (1). Закрытие этого окна приводит к завершению работы среды. Кроме того, здесь располагаются главное меню (2), некоторые пункты которого дублируются кнопками панели инструментов (3), а также палитра компонентов (4).

Ниже располагаются окна различных инструментов, входящих в состав среды программирования: дерева объектов (5), инспектора объектов (6), дизайнера форм (7), а также текстового редактора (8).

Посредством главного меню можно также отобразить ряд дополнительных окон, позволяющих эффективно решать задачи отладки приложений, настройки проектов. Описание пунктов главного меню и его подпунктов представлено в прил. 1.

Библиотека БГУИР

Среда программирования Delphi 7 позволяет работать с различными типами проектов, но чаще всего ее применяют для разработки приложений, работающих под управлением ОС Windows.

Все приложения в операционной системе Windows принято разделять на оконные и консольные. Оконные приложения выводят информацию в графическом виде, создавая окна с надписями и элементами управления, посредством которых пользователь управляет их выполнением, осуществляет ввод исходных данных и получает результаты работы программы. Консольные приложения предоставляют более примитивный пользовательский интерфейс: ввод команд осуществляется с клавиатуры, вывод данных, как правило, – на экран.

С точки зрения пользователя рассмотренное деление является достаточно условным: в случае необходимости консольное приложение может создавать графические окна, а оконное приложение может быть неотличимым от консольного. Тем не менее для программиста консольные приложения зачастую оказываются более удобными, поскольку позволяют перейти к реализации алгоритма решения поставленной задачи, не отвлекаясь на создание и настройку графического интерфейса. По этой причине для начального освоения языка программирования Delphi более предпочтительными являются консольные приложения.

Библиотека БГУИР

2. ПРИМЕР РАЗРАБОТКИ И ОТЛАДКИ ПРОГРАММЫ

2.1. Создание проекта консольного приложения

Разработка приложений в Delphi основана на понятии проекта. Создание нового приложения в этой среде программирования обычно начинается с создания нового проекта. Проект – это множество файлов, из которых будет создано приложение. Некоторые из них создаются средой программирования в автоматическом режиме, другие (например файлы с исходным кодом) – программистом.

Создадим проект консольного приложения. Для этого выберем пункт меню «Файл ➔ Новый ➔ Другое...» («File ➔ New ➔ Other…»). В результате откроется окно, в котором на нескольких вкладках будут перечислены типы проектов, создание которых поддерживается средой программирования (рис. 2.1).

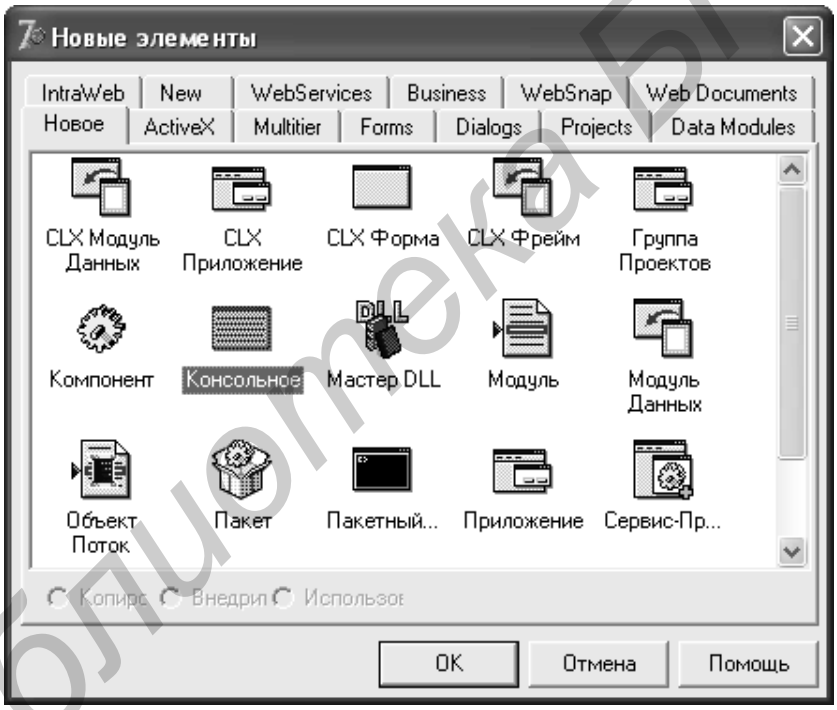


Рис. 2.1. Окно выбора типа проекта

Выберем в предложенном списке тип приложения «Консольное». В результате будет создан новый проект и открыто окно текстового редактора с шаблоном консольного приложения (рис. 2.2).

Окно текстового редактора разделено на две части. В правой части располагается текстовое поле для ввода текста программы. Вертикальная линия в этом поле отделяет первые 80 символов строки. Рекомендуется писать программу таким образом, чтобы ее строки были короче 80 символов. В результате текст программы будет находиться слева от этой линии.

Библиотека БГУИР

Левая часть окна – это так называемый «проводник кода». В нем для текущего файла отображаются по группам используемые вспомогательные модули, объявленные типы, переменные, константы, процедуры и функции и т. д. При работе с большими программными модулями «проводник кода» позволяет программисту быстрее находить интересующие его места в коде. На рис. 2.3 приведен вид «проводника кода» для случая, когда в текстовом редакторе открыт один из стандартных модулей – Dialogs.pas.

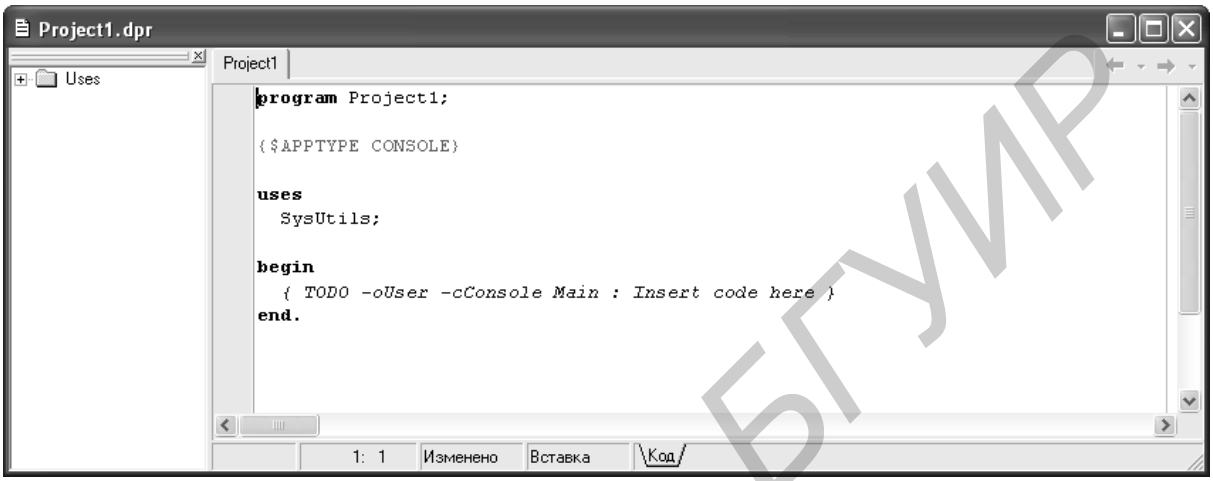


Рис. 2.2. Вид окна текстового редактора
при создании консольного приложения

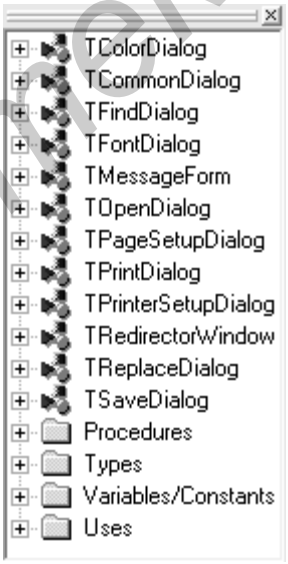


Рис. 2.3. Вид «проводника кода» для модуля Dialogs.pas

В нижней части окна текстового редактора также отображается информация о положении курсора (номер строки и столбца), режиме ввода (вставка или замена, переключается клавишей Insert). Имеется также набор вкладок, который в консольном приложении состоит из единственной вкладки «Код».

Библиотека БГУИР

Как будет показано далее, при компиляции проекта в нижней части окна текстового редактора может появляться еще одна область – для отображения обнаруженных ошибок, а также предупреждений и подсказок. Если компиляция завершается успешно, эта область автоматически исчезает.

Ошибкой многих программистов, как начинающих, так и опытных, является игнорирование выводимых средой предупреждений и подсказок.

Действительно, если анализ кода программы не выявил ошибок, компиляция программы успешно выполняется и полученный исполняемый модуль может быть запущен даже при наличии предупреждений. Тем не менее предупреждения компилятора указывают на имеющиеся в коде потенциальные проблемы, которые могут рано или поздно стать причиной появления в программе ошибок, отладка которых существенно затруднена. Анализ предупреждений компилятора и внесение соответствующих исправлений в код программы позволяет резко повысить надежность программы уже на начальных этапах разработки.

Подсказки компилятора обычно относятся к неиспользуемым переменным. Кроме того, подсказки могут выводиться, если компилятор обнаруживает фрагмент кода, удаление которого не приводит к изменениям в логике программы. Задача программиста – понять, действительно ли фрагмент кода лишний (в этом случае его нужно удалить из текста программы) или же была допущена логическая ошибка (например выражение в цикле с предусловием всегда равно False).

При компиляции хорошей программы количество ошибок, предупреждений и подсказок компилятора равно 0.

Следует обратить внимание на то, что автоматически созданный средой каркас проекта представляет собой полноценное приложение, которое может быть скомпилировано и запущено на выполнение.

Рассмотрим элементы каркаса проекта.

Program Project1 – строка, как и в языке Pascal, задает имя программы. При сохранении проекта в качестве имени программы будет автоматически подставлено имя файла проекта. Как и в Pascal, эта строка не является обязательной, а имя программы может быть произвольным идентификатором. Имя программы следует оставлять в том виде, в котором ее формирует среда программирования. При изменении имени программы следует контролировать его соответствие имени файла проекта, иначе компиляция завершится ошибкой.

{$APPTYPE CONSOLE} – данная конструкция представляет директиву компилятора. Она дает компилятору указание компилировать программу как консольное приложение. В частности, для консольного приложения системой автоматически создается консольное окно, в которое можно выводить информацию стандартными средствами языка Pascal (write/writeln). Если эту директиву убрать, попытка выполнить вывод на консоль завершится ошибкой «I/O Error 105».

Библиотека БГУИР

Uses SysUtils – этими строками подключаются используемые программой вспомогательные модули. По умолчанию к консольному приложению (кроме неявно используемого всеми программами модуля System) подключается модуль SysUtils, содержащий множество вспомогательных процедур для работы со строками, датами, файлами.

Между операторными скобками begin…end расположен комментарий { TODO -oUser -cConsole Main : Insert code here }, говорящий о том, что код программы должен быть расположен именно здесь. В отличие от директивы {$APPTYPE} эта запись является обычным комментарием и может быть удалена.

Необычный вид этого комментария объясняется использованием одной из мощных возможностей среды – TODO-списков. TODO-списки позволяют программисту поддерживать список изменений, которые требуется внести в программу. Например, если в ходе разработки какая-либо часть программы еще не готова и будет написана позже, к тому месту, где она будет использоваться, можно прикрепить «напоминание». Такое напоминание будет представлять собой комментарий, начинающийся со слова «TODO». Чтобы просмотреть все имеющиеся в проекте «напоминания», нужно выбрать пункт меню «Вид ➔ Список To-Do» («View ➔To-Do List»). При этом откроется окно, в котором в табличном виде будут перечислены все «напоминания» (рис. 2.4).

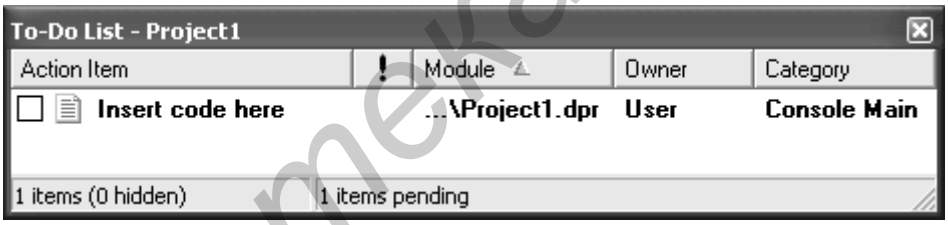


Рис. 2.4. Окно со списком TODO

Двойной щелчок по элементу списка открывает в текстовом редакторе соответствующий файл и помещает курсор в начало строки, к которой привязано «напоминание».

2.2. Разработка и отладка консольного приложения

Рассмотрим простейший пример разработки программы.

Пример. Для параметра A, изменяющегося от 100 до 1000 с шагом 10, и параметра B, изменяющегося от –10 до 10 с шагом 1, вычислить значение функции $Y = \frac{A}{B}$.

Исходный код консольного приложения был написан следующим образом:

Библиотека БГУИР

```
{ 1 } program Project1;
{ 2 }
{ 3 } {$APPTYPE CONSOLE}
{ 4 }
{ 5 }uses
{ 6 }  SysUtils;
{ 7 }
{ 8 }var
{ 9 }  a, b, y: Integer;
{10 }
{11 }begin
{12 }  a := 100;
{13 }  b := -10;
{14 }  repeat
{15 }    repeat
{16 }      y:=a/b;b:=b+1;WriteLn('a=',a:5,'b=',b:5,'y=',y:6:2);
{17 }      Slip(500);
{18 }    until b > 10;
{19 }    a := a + 10;
{20 }  until a > 1000;
{21 }end.
```

В данной программе намеренно допущен ряд ошибок, чтобы продемонстрировать некоторые принципы отладки программы.

Итак, при запуске программы на компиляцию будет выявлена первая ошибка (рис. 2.5).

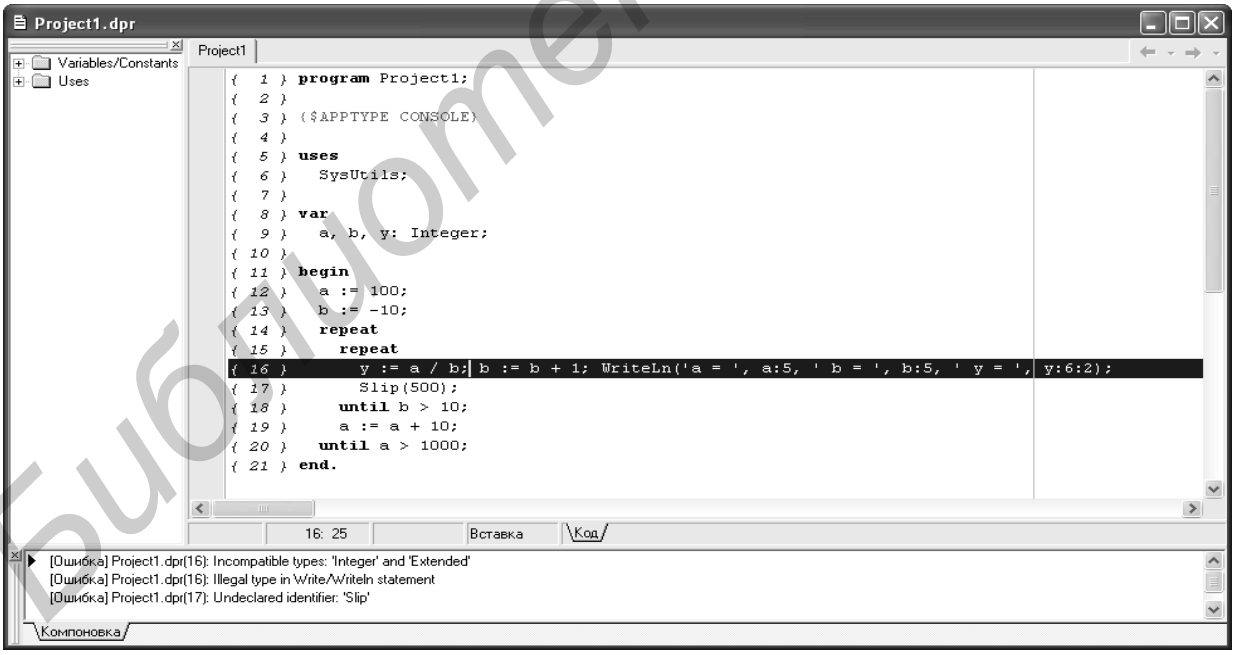


Рис. 2.5. Первый шаг отладки программы

Ошибка обнаружена в 16-й строке, о чем свидетельствует выделение этой строки цветом. В нижней части окна редактора кода на вкладке «Компоновка» отображается список обнаруженных при компиляции ошибок. Удобный способ

Библиотека БГУИР

исправления ошибок заключается в том, чтобы исправлять только ту ошибку, на которую непосредственно указывает среда программирования (в нашем случае – в 16-й строке), а затем заново выполнять компиляцию. При таком подходе, во-первых, можно убедиться в том, что ошибка действительно была исправлена, во-вторых, нет необходимости анализировать весь список ошибок. Например, в нашем случае вторая ошибка в списке перестанет быть актуальной, когда исправится первая.

Ошибка в 16-й строке свидетельствует о том, что тип результата операции не соответствует типу переменной, в которую заносится данный результат. Однако 16-я строка содержит 3 оператора. Для уточнения, в каком конкретно операторе допущена ошибка, необходимо расположить каждый оператор на отдельной строке. При следующем запуске программы на компиляцию станет ясно, что ошибка обнаружена в операторе: y:= a / b. Она возникла, т. к. результат деления a/b имеет вещественный тип и не может быть записан в целочисленную переменную y. Для переменной y необходимо указать тип Real.

После исправления этой ошибки, при очередном запуске программы на компиляцию возникла следующая ошибка – неизвестный идентификатор (рис. 2.6).

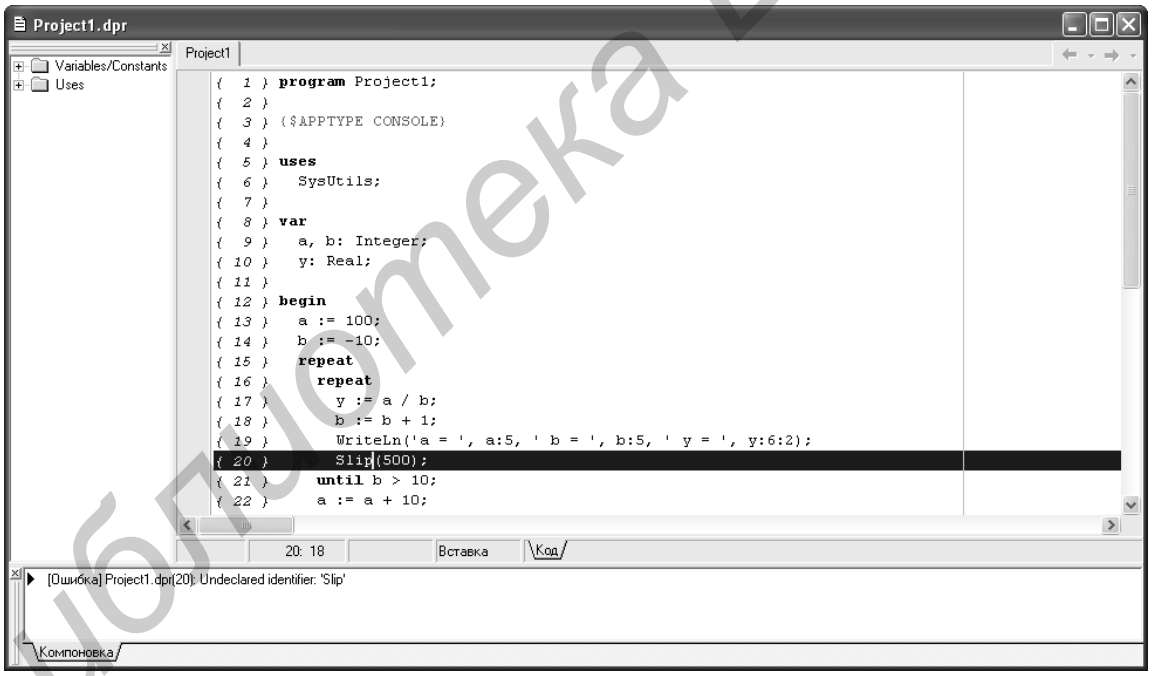


Рис. 2.6. Второй шаг отладки программы

Ошибка возникла в 20-й строке, где была предпринята попытка произвести задержку работы программы, воспользовавшись стандартной процедурой. Однако при написании имени процедуры была допущена орфографическая ошибка. Для проверки данного предположения необходимо открыть справочную систему Delphi и найти необходимую процедуру (рис. 2.7).

Библиотека БГУИР

Процедура Sleep предназначена для задержки выполнения программы на заданное количество миллисекунд. После исправления данной ошибки (имени Slip на Sleep) необходимо вновь запустить программу на компиляцию. На этот раз этап компиляции пройден успешно.

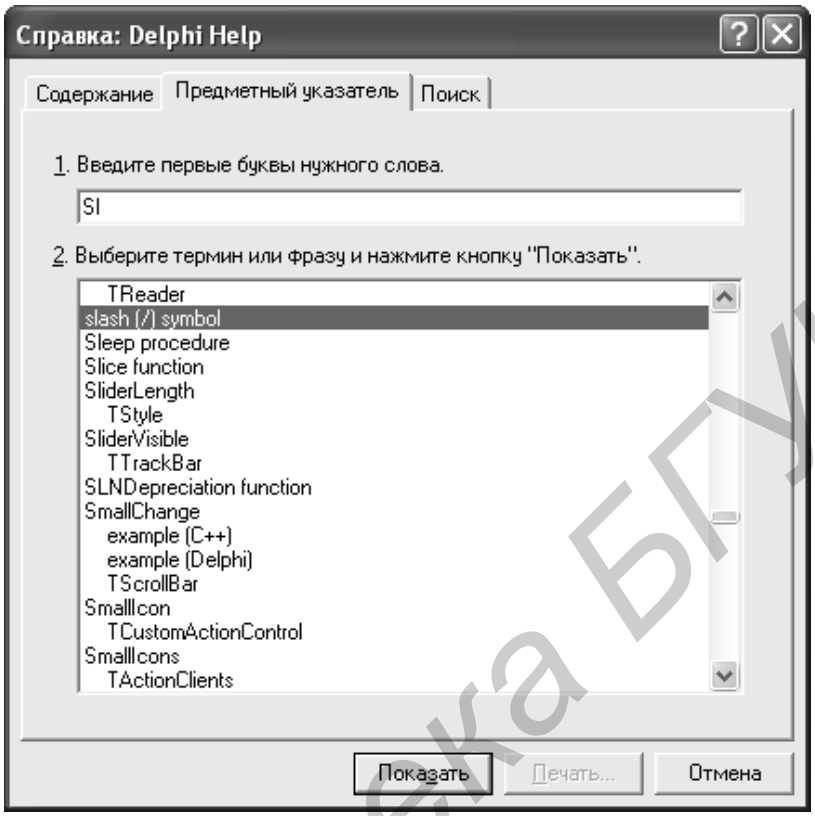


Рис. 2.7. Поиск имени процедуры в справочной системе

Однако при запуске программы на выполнение выведены не все результаты расчетов. Для локализации ошибки следует использовать пошаговый режим отладки, проверяя корректность значений переменных на каждом шаге. Для наблюдения за значениями переменных следует по очереди добавить их в список наблюдения. Для добавления переменной используется пункт меню «Запуск ➔ Добавить Наблюдателя» («Run ➔ Add Watch») или сочетание клавиш <Ctrl+F5> (рис. 2.8, 2.9).

После добавления переменных a, b, y в список наблюдения следует начать пошаговую отладку. Для этого, используя клавишу <F8>, выполняют отдельные шаги программы. В процессе отладки окажется, что на очередной итерации делитель b оказывается равным 0. Это означает, что при $b = 0$ вычисление функции a/b завершается аварийно. Для устранения проблемы, следует вставить в код программы проверку деления на 0.

Библиотека БГУИР

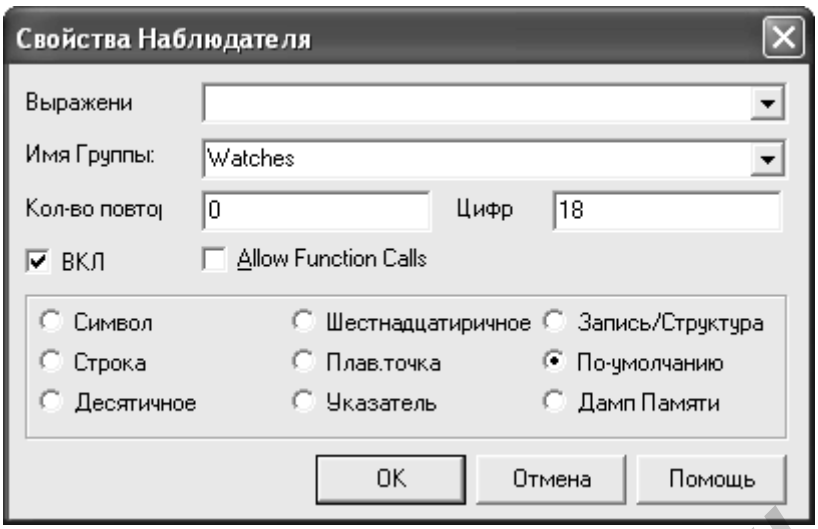


Рис. 2.8. Окно для изменения свойств режима наблюдения

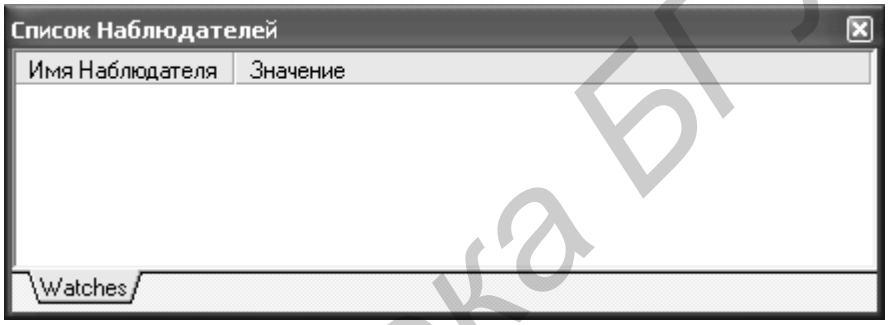


Рис. 2.9. Окно со списком наблюдения

Программный код был отредактирован следующим образом:

```
{ 1 }program Project1;
{ 2 }
{ 3 }{$APPTYPE CONSOLE}
{ 4 }
{ 5 }uses
{ 6 }  SysUtils;
{ 7 }
{ 8 }var
{ 9 }  a, b: Integer;
{10 }  y: Real;
{11 }
{12 }begin
{13 }  a := 100;
{14 }  b := -10;
{15 }  repeat
{16 }    repeat
{17 }      if  b = 0 then
{18 }        WriteLn('a =',a:5,'b =',b:5,'Error')
{19 }        else
{20 }        begin
```

Библиотека БГУИР

```
{21 }          y := a / b;
{22 }          b := b + 1;
{23 }          WriteLn('a =',a:5,'b =',b:5,'y =',y:6:2);
{24 }        end;
{25 }        Sleep(500);
{26 }    until b > 10;
{27 }    a := a + 10;
{28 }  until a > 1000;
{29 } end.
```

Если сейчас запустить программу, можно увидеть, что ситуация с делением на 0 обрабатывается корректно, но после достижения переменной b нулевого значения оно больше не изменяется, в результате чего программа зацикливается. Для того чтобы прервать выполнение программы, следует воспользоваться пунктом меню «Запуск ➔ Сброс программы» (Run ➔ Program Reset) или соответствующей ему комбинацией клавиш <Ctrl+F2>. Ошибка устраняется вынесением оператора с номером 22 b:=b+1 за пределы условного оператора If … then … else ….

Дальнейший запуск программы на выполнение показывает, что программа выдает результаты, не соответствующие ожидаемым: значение переменной y во многих случаях оказывается равным 10.00:

```
                    …
a= 800 b=  80 y= 10.00
a= 810 b=  81 y= 10.00
a= 820 b=  82 y= 10.00
a= 830 b=  83 y= 10.00
a= 840 b=  84 y= 10.00
a= 850 b=  85 y= 10.00
a= 860 b=  86 y= 10.00
a= 870 b=  87 y= 10.00
a= 880 b=  88 y= 10.00
                    …
```

Из примера видно, что значения, которые принимает переменная b, выходят за пределы отрезка [–10; 10]. Эта ошибка устраняется внесением оператора с номером 14 b:=−10 в тело внешнего цикла. После всех исправлений программа будет иметь следующий вид:

```
{ 1 }program Project1;
{ 2 }
{ 3 }{$APPTYPE CONSOLE}

{ 4 }
{ 5 }uses
{ 6 }  SysUtils;
{ 7 }
{ 8 } var
{ 9 }     a, b: Integer;
{10 }     y: Real;
```

Библиотека БГУИР

```
{11 }
{12 }begin
{13 }  a := 100;
{14 }  repeat
{15 }    b := -10;
{16 }      repeat
{17 }        if   b = 0   then
{18 }           WriteLn('a =',a:5,'b =',b:5,'Error')
{19 }        else
{20 }        begin
{21 }          y := a / b;
{22 }          WriteLn('a =',a:5,'b =',b:5,'y =',y:6:2);
{23 }        end;
{24 }        Sleep(500);
{25 }        b := b + 1;
{26 }      until b > 10;
{27 }      a := a + 10;
{28 }  until a > 1000;
{29 }end.
```

Рассмотрим еще некоторые настройки, которые могут быть полезными при отладке более сложных приложений:

– компилятор Delphi при генерации исполняемого модуля использует ряд приемов, позволяющих повысить быстродействие программы и эффективность использования ею вычислительных ресурсов. Эти оптимизации никак не влияют на поведение программы с точки зрения пользователя, но могут приводить к неожиданным для программиста изменениям логики при отладке (например, в простых циклах for порядок перебора значений переменной цикла может изменяться на противоположный). Для временного отключения оптимизаций следует в настройках проекта (меню «Проект ➔ Опции...» или «Project ➔ Options…») перейти на вкладку «Компилятор» («Compiler») и снять галочку «Оптимизация» («Optimization»). После завершения этапа отладки режим оптимизации следует включить для повышения быстродействия программы;

– в рассмотренном примере имела место ошибка времени выполнения (Run time error), а именно деление на 0, которая приводила к аварийному завершению выполнения программы без вывода сообщений об ошибках. Между тем, если бы программа была запущена не из среды программирования (выполнялась не «под отладчиком»), сообщение об ошибке было бы отображено. Для того чтобы ошибки времени выполнения были видны при отладке, следует выбрать пункт меню «Инструменты ➔ Опции Отладчика...» («Tools ➔ Debugger Options…»), в открывшемся окне перейти на вкладку «Исключения OS» («OS Exceptions») и для соответствующей ошибки вместо «User Program» выбрать «Debugger».

Библиотека БГУИР

2.3. Пример разработки программы, состоящей из нескольких модулей

При разработке сложных приложений и программных комплексов количество строк программы может исчисляться сотнями тысяч. Размещать весь исходный код таких программ в одном файле невыгодно: большой по объему программный модуль тяжело анализировать, логика взаимодействия отдельных частей кода в таком модуле имеет тенденцию к запутыванию.

Поэтому, как правило, сложные проекты разделяют на несколько логически независимых частей – модулей. Например, в компьютерной игре такими частями могут быть модули, отвечающие за отображение игровой сцены, сохранение/загрузку игры, управление ресурсами, звуком. Эти части взаимодействуют между собой, обмениваясь данными, но каждая из них остается независимой. Такой подход имеет ряд преимуществ.

Во-первых, программирование отдельных модулей может выполняться параллельно несколькими разработчиками: кто-то занимается отображением игровой сцены, кто-то продумывает логику подсчета игровых очков, ресурсов.

Во-вторых, модульная структура программы позволяет ускорить процесс компиляции программы: компиляция большого проекта «с нуля» может занимать десятки минут, тогда как возможность перекомпилировать только отдельные модули (и такая возможность предоставляется средой Delphi) позволяет существенно уменьшить этот показатель.

В-третьих, модульная структура позволяет более гибко управлять работой программы. Например, изначально игра для визуализации игровой сцены может использовать библиотеку OpenGL. Если в какой-то момент потребуется выполнить отображение игровой сцены средствами другой библиотеки (например Direct3D, входящей в состав DirectX), достаточно заменить лишь один модуль, отвечающий за отображение, остальные модули останутся нетронутыми.

Способ поддержки модульности программ в языке Pascal, положенном в основу языка Delphi, за многолетнюю историю промышленной разработки показал себя с наилучшей стороны и до сегодняшнего дня оказывается более эффективным, чем в ряде других языков, получивших широкое распространение (например C++).

Работу с модулями в среде Delphi рассмотрим на примере игровой программы «Отгадай число».

Правила игры: компьютер случайным образом выбирает число от 1 до 100 включительно. Задача игрока – отгадать это число не более, чем за 10 попыток. Для этого пользователь вводит свое предположение и получает в качестве ответа одно из сообщений:

- «Искомое число больше»;
- «Искомое число меньше»;
- «Вы отгадали!»;

Библиотека БГУИР

– «Вы не отгадали число XX с 10 попыток» (где XX – искомое число).

В начале игры игроку начисляется 1000 очков. За каждую неудачную попытку начисляются штрафные очки по формуле

$$Штраф = |Искомое\_число - Предположение\_игрока|$$

Кроме главного модуля в программе, реализующего логику программы, создадим два вспомогательных: модуль подсчета очков и модуль взаимодействия с пользователем.

Создадим новый проект консольного приложения и сохраним его в отдельном каталоге. Открытый в текстовом редакторе файл проекта – это главный модуль будущей программы.

Добавим в проект вспомогательные модули. Для этого следует выбрать пункт меню «Файл ➔ Новый ➔ Модуль» («File ➔ New ➔ Unit»). Вновь созданный модуль получит имя Unit1 (рис. 2.10). Сохраним его под именем Points.pas – это будет модуль подсчета очков. Аналогично создадим второй модуль и сохраним его под именем UI.pas (UI – это сокращение от User Interface, т. е. интерфейс пользователя).

Рис. 2.10. Вид окна текстового редактора после создания нового модуля

Создаваемые средой Delphi модули представляют собой каркас программного модуля. Рассмотрим его построчно.

Unit NameOfUnit – заголовок вспомогательного модуля. Вместо NameOfUnit среда автоматически подставляет идентификатор, совпадающий с именем файла. Если по какой-то причине имя файла и идентификатор после служебного слова unit окажутся различными, при компиляции возникнет ошибка: «Unit identifier ‘SomeUnit’ does not match file name».

Interface – зарезервированное слово interface начинает интерфейсную секцию вспомогательного модуля. Все переменные, константы, типы и процедуры, объявленные в этой секции, будут доступны из модулей, подключающих данный модуль, т. е. их можно будет использовать для взаимодействия (интерфейса) с данным модулем.

Библиотека БГУИР

Implementation – зарезервированное слово implementation начинает секцию реализации вспомогательного модуля. Объявленные в этой секции переменные, константы, типы и процедуры будут доступны только внутри данного модуля.

End. – слово end с точкой завершает модуль, все, что написано после него, будет проигнорировано компилятором.

Откроем в текстовом редакторе главный модуль и обратим внимание на то, что блок uses этого модуля был дополнен только что созданными вспомогательными модулями с указанием пути к ним. Такой способ записи позволяет размещать вспомогательные модули по совершенно произвольным путям на диске. Альтернативный вариант заключается в указании путей к модулям в настройках проекта.

Поскольку проект достаточно прост в реализации, воспользуемся методом восходящего проектирования, т. е. начнём разработку со вспомогательных процедур, которые затем задействуем в основной программе. В первую очередь нам понадобятся процедуры для запроса у пользователя числа-предположения и вывода всевозможных подсказок. Разместим их в модуле UI.pas. Примерный текст модуля приведен в прил. 2 (подразд. П.2.1).

Модуль UI.pas можно проверить на синтаксическую корректность, выполнив компиляцию проекта. Несмотря на то, что ни одна из процедур модуля программой пока не используется, часть ошибок в нем можно отловить уже на этом этапе.

Сейчас, когда уже написаны процедуры взаимодействия с пользователем, можно в первом приближении написать код главного модуля игры. Отказавшись пока от реализации подсчета очков, получим такой код:

```
  Program ConsoleGuess;
  {$APPTYPE CONSOLE}
  uses
      Points in 'Points.pas',
      UI in 'UI.pas';
  var
      Goal  : Integer;    // Искомое число
      Guess : Integer;    // Предположение пользователя
      Count : Integer;    // Номер попытки
  begin
//Инициализация генератора случайных чисел
     Randomize;
      repeat
         WriteHint(htSplash, 0);
//Выбираем случайное число от 1 до 100
         Goal := Random(100) + 1;
//Игровой цикл
         for Count := 1 to 10 do
         begin
             Guess := ReadGuess(Count);
             if Guess = Goal then
             begin
```

Библиотека БГУИР

```
                WriteHint(htEqual, 0);
                Break;
//Если число отгадано, прерываем игровой цикл
             end;
             if Goal > Guess then
                 WriteHint(htGreater, 0)
            else
            WriteHint(htLess, 0);
         end;
//Предположение и искомое число не равны. Число не отгадано
         if Guess <> Goal then
            WriteHint(htFailed, Goal);
    until not ReadPlayAgain();
  end.
```

Щелкнув правой кнопкой мыши в конце строки WriteHint(htEqual, 0), выберем в выпадающем меню пункт «Добавить Элемент To-Do…» («Add To-Do Item...»). В открывшемся окне добавления элемента TODO-списка укажем, что нужно передавать в качестве второго параметра WriteHint не 0, а количество набранных очков. Настройка остальных параметров может иметь смысл, когда над проектом работает несколько человек, в нашем случае достаточно указать текст и выбрать высокий приоритет. Подтвердим добавление элемента нажатием кнопки Ok, после этого в коде программы появится специальный TODO-комментарий (рис. 2.11).

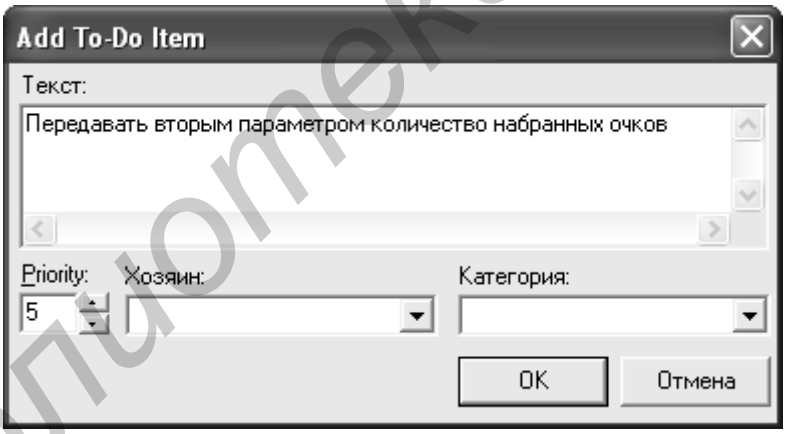


Рис. 2.11. Окно добавления элемента TODO-списка

Следует обратить внимание на то, что главный модуль программы не использует возможностей, предоставляемых модулем SysUtils, – поэтому данный модуль исключен из uses-списка. Ещё одна причина отказаться от его использования заключается в его влиянии на размер получаемого после компиляции исполняемого файла программы. Если после успешной компиляции программы выбрать пункт меню «Проект ➔ Инфо по ИмяПроекта» («Project ➔ Information for ProjectName»), среда Delphi отобразит окно, в котором будет краткая информация о результатах компиляции, в том

Библиотека БГУИР

числе и о размере EXE-файла. Скомпилировав программу с подключенным модулем SysUtils и без его подключения, можно заметить, что использование SysUtils добавляет около 25 КБ к размеру программы, что составляет более 50 % от ее общего размера.

Если сейчас запустить программу, мы увидим, что выводимые сообщения отображаются некорректно. Причина кроется в том, что консоль в операционной системе Windows изначально разрабатывалась так, чтобы быть обратно совместимой с приложениями MS-DOS. Поэтому по умолчанию консоль интерпретирует выводимые в нее сообщения как записанные в кодировке OEM (кодовая страница 866), в то время как текстовый редактор Delphi использует кодировку ANSI (как правило, это кодовая страница 1251). Коды символов кириллицы в этих кодировках отличаются. Существует несколько решений этой проблемы.

Во-первых, можно создавать исходные тексты программ в текстовом редакторе, поддерживающем сохранения документов в OEM-кодировке (компилятор Delphi умеет их компилировать), но в этом случае требуются дополнительные инструменты, а значит, теряются преимущества использования интегрированной среды.

Во-вторых, можно все строковые данные перед выводом в консоль преобразовывать специальными функциями Windows API (например CharToOEM() и ее аналогами). Это решение лучше предыдущего, но не идеально, поскольку нужно следить, чтобы весь вывод в консоль проходил эту обработку. Кроме того, это означает отказ от такой возможности, как вывод нескольких значений одним вызовом write/writeln, т. к. придется заранее преобразовывать в строки всю выводимую информацию, что крайне неудобно.

Самый удобный вариант решения проблемы заключается в использовании модуля EsConsole.pas, текст которого приведен в прил. 2 (подразд. П.2.2). Он перехватывает все выводимые на консоль текстовые данные и обрабатывает их вторым способом. При этом перехват осуществляется прозрачно для программиста: все, что нужно сделать, чтобы перехват осуществлялся, это подключить модуль к проекту.

Воспользуемся готовым модулем или наберем его самостоятельно в любом текстовом редакторе, кроме встроенного в среду программирования Delphi. Полученный текстовый файл следут сохранить с расширением .pas в папку проекта. Теперь в среде Delphi выберем пункт меню «Проект ➔ Добавить в Проект...» («Project ➔ Add to Project…») и укажем созданный файл. Этот способ добавления файлов в проект используется, когда на диске уже есть готовый модуль.

Далее скомпилируем и запустим программу на выполнение, чтобы убедиться, что она корректно обрабатывает все игровые ситуации. Единственное, что еще не реализовано в программе, это подсчет количества набранных очков, оно пока равно 0.

Библиотека БГУИР

Теперь, когда программа работает корректно, примем решение о дальнейшей разработке проекта. Если полученный на данном этапе результат удовлетворяет, можно вовсе отказаться от подсчета очков. В этом случае остается только изменить реализацию процедуры WriteHint в модуле UI.pas, чтобы она не выводила количество набранных очков, и удалить из проекта модуль Points.pas. Для удаления модуля предназначен пункт меню «Проект ➔ Удалить из Проекта...» («Project ➔ Remove from Project…»), при выборе которого открывается окно со списком модулей проекта. Выбрав подлежащий удалению модуль, нажмем кнопку Ok и подтвердим удаление модуля.

Обратим внимание на то, что при удалении модуля из проекта соответствующий ему файл на диске не удаляется и его можно будет снова добавить в проект, если это потребуется.

Теперь предположим, что разработка программы продолжается и нужно добавить функцию подсчета очков. Как и предполагалось в самом начале разработки программы, вынесем логику подсчета очков в модуль Points.pas. Для данного примера модуль Points.pas имеет вид, представленный в прил. 2 (подразд. П.2.3).

Дополнив логику главного модуля программы вызовами процедур из Points.pas, получим следующий код:

```
  program ConsoleGuess;
  {$APPTYPE CONSOLE}
  uses
       UI in 'UI.pas',
       EsConsole in 'EsConsole.pas',
       Points in 'Points.pas';
  var
       Goal  : Integer;    // Искомое число
       Guess : Integer;    // Предположение пользователя
       Count : Integer;    // Номер попытки
       Pts   : Integer;    // Количество очков
  begin
      Randomize;
//Инициализация генератора случайных чисел
       repeat
          WriteHint(htSplash, 0);
//Выбираем случайное число от 1 до 100
          Goal := Random(100) + 1;
          InitPoints(Pts);
 //Игровой цикл
        for Count := 1 to 10 do
        begin
            Guess := ReadGuess(Count);
            if Guess = Goal then
           begin
              { TODO 5 : Передавать вторым параметром количество
набранных очков }
             WriteHint(htEqual, 0);
//Если число отгадано, прерываем игровой цикл
```


Библиотека БГУИР

```
            Break;
          end;
           if Goal > Guess then
               WriteHint(htGreater, 0)
          else
              WriteHint(htLess, 0);
              UpdatePoints(Pts, Goal, Guess);
        end;
//Предположение и искомое число не равны. Число не отгадано
        if Guess <> Goal then
           WriteHint(htFailed, Goal);
     until not ReadPlayAgain();
  end.
```

Если сейчас запустить программу, ее поведение не изменится: количество заработанных очков будет выводиться равным 0. Откроем TODO-список (пункт меню «Вид ➔ Список To-Do» или «View ➔ To-Do List») и обнаружим там пометку о необходимости передавать в WriteHint(htEqual, …) корректное количество набранных очков. Двойным щелчком по элементу TODO-списка перейдем к нужной строке и изменим ее:

```
WriteHint(htEqual, Pts);
```

Теперь в TODO-списке можно поставить слева от соответствующего элемента галочку. В результате соответствующий комментарий из

```
{TODO 5 : Передавать вторым параметром количество набранных очков}
```

превратится в

```
{DONE 5 : Передавать вторым параметром количество набранных очков}.
```

При коллективной разработке данный элемент TODO-списка мог быть создан другим разработчиком. В этом случае пометка «DONE» означала бы, что теперь он должен убедиться в том, что внесенные изменения соответствуют его ожиданиям. В случае же, когда над проектом работает один программист, элемент можно либо сразу удалить, либо оставить, если есть вероятность, что к нему еще придется вернуться.

Скомпилируем и запустим программу. Убедимся в том, что она работает корректно.

Покажем некоторые дополнительные преимущества модульного подхода. Пусть правила начисления очков изменились: первоначальное количество очков уменьшается на 100 с каждой новой игрой; когда количество очков станет равным 200, уменьшение очков следует прекратить; штрафные очки за неудачную попытку начислять, используя формулу

Библиотека БГУИР

$$Штраф = 8 \cdot \left| Искомое\_число - Предположение\_игрока \right|.$$

Кроме того, в связи с этими изменениями нужно учесть, что количество очков может становиться на очередном шаге отрицательным. Чтобы избежать этого, при достижении количеством очков нулевого значения, оно не должно меняться и должно оставаться равным 0. Изменения затронут только один модуль, а именно Points.pas. В результате, модуль Points.pas примет вид, представленный в прил. 2 (подразд. П.2.4). Нетрудно убедиться, что этих изменений действительно достаточно для перечисленных изменений в логике подсчета очков.

Обратим внимание на зарезервированное слово initialization. С него начинается необязательная секция инициализации модуля. Код, записанный в этой секции, будет выполнен при запуске использующей его программы. Если модуль подключается в нескольких модулях программы, секция инициализации будет выполнена только 1 раз. Это удобно для инициализации глобальных переменных, выделения памяти для внутренних структур модуля.

В данном примере в секции инициализации задается начальное значение глобальной переменной InitialPoints, используемой модулем и недоступной за его пределами. Секция initialization есть и в рассмотренном ранее модуле EsConsole.pas:

```
  initialization
     Rewrite(Output);
//Проводим инициализацию файла
     TTextRec(Output).InOutFunc := @ConOutFunc;
     TTextRec(Output).FlushFunc := @ConOutFunc;
```

Благодаря этому при запуске программы, использующей данный модуль, весь вывод в консоль перенаправляется в функцию, определенную в данном модуле.

Помимо секции initialization модуль может содержать секцию finalization. Она оформляется аналогично секции инициализации, но предназначена для освобождения ресурсов, захваченных модулем: выделенной памяти, установленных обработчиков. Секция финализации автоматически выполняется однократно при завершении работы программы. Схематично можно представить использование секций initialization и finalization так:

```
unit SomeUnit;
interface
...
implementation
...
initialization
     GetMem(P, 1024);
finalization
     FreeMem(P, 1024);
end.
```

Библиотека БГУИР

В качестве примера дополним модуль UI.pas секциями инициализации и финализации. В секции инициализации сообщим пользователю о том, что программа использует данный модуль и отобразим краткую информацию о нем; в секцию финализации поместим вывод сообщения «До встречи!». В обоих случаях после вывода информации добавим задержку функцией Sleep().

Приведем модуль UI.pas в общем виде с учетом этих изменений:

```
unit UI;
interface
...
implementation
...
initialization
    writeln('Программа использует модуль UI.pas.');
    writeln('Данный модуль организовывает взаимодействие с
пользователем.');
    Sleep(10000);
finalization
    writeln ('До встречи!');
    Sleep(5000);
end.
```

2.4. Контрольные вопросы

1. Опишите назначение консольного режима среды Delphi.
2. Каковы ограничения работы в консольном режиме?
3. Перечислите последовательность действий при создании консольного приложения в среде Delphi.
4. Перечислите разделы текстового редактора среды Delphi и укажите назначение каждого из них.
5. Перечислите отладочные возможности среды Delphi.
6. Для чего служит пошаговый режим выполнения программы и как его активировать?
7. Каково назначение списка TODO?
8. Для чего служит список наблюдения (Watch) и как добавить переменную в этот список?
9. Что такое программный модуль? Каково его назначение?
10. Перечислите последовательность действий при создании программного модуля.
11. Какова структура программного модуля?
12. Каково назначение каждой секции программного модуля?

Библиотека БГУИР

3. ЗАДАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

3.1. Лабораторная работа по теме «Записи. Типизированные файлы»

Цель работы: разработка, отладка и выполнение программы обработки записей. Организация работы с типизированными файлами.

Для заданного преподавателем варианта необходимо разработать программу на языке Delphi. Информация, обрабатываемая программой, должна храниться в файле записей. Программа должна содержать необходимые комментарии. Следует предусмотреть простейший вывод на экран входных и выходных данных.

В каждом варианте задания реализовать функции добавления, удаления, корректировки и просмотра записей файла.

Варианты заданий

1. Магазин имеет список товаров на своем складе. Каждая запись списка содержит следующую информацию о товаре: группа товаров, код товара, наименование продукции, наименование модели, цена, количество. Требуется:

- отсортировать товары внутри группы по одному из признаков: код товара, наименование продукции, цена;
- вывести товары, количество которых на складе меньше заданного значения;
- осуществлять поиск товара по наименованию продукции и модели.

2. Написать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: наименование специальности, номер группы, ФИО студента, оценки, полученные им за последнюю сессию, средний балл (вычисляется в программе). Требуется:

- отсортировать весь список студентов в порядке убывания среднего балла;
- отсортировать студентов в группе в порядке убывания среднего балла;
- вывести список студентов, не получивших в сессию неудовлетворительных оценок;
- осуществлять поиск студента по ФИО.

3. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая запись содержит следующую информацию: наименование группы изделий (телевизор, радиотелефон и т. п.), марка изделия, дата приемки в ремонт, дата исполнения заказа, состояние готовности заказа (выполнен или не выполнен). Требуется:

- выдать информацию о состоянии готовности заказов на текущие сутки по группам изделий;
- выдать информацию о заказах, не выполненных в срок;
- отсортировать заказы в группах по убыванию даты исполнения заказа;

Библиотека БГУИР

− осуществлять поиск изделия по наименованию, дате приемки, дате исполнения заказа.

4. Разработать программу («электронную сваху») для службы знакомств. Имеется 2 списка: список женихов и список невест. В каждом списке кандидат (жених или невеста) характеризуются записью, содержащей следующие поля: порядковый номер кандидата, данные о кандидате (имя, возраст, рост, вес, привычки, хобби), требования к партнеру (в виде диапазона Min-Max для возраста, роста, веса). Требуется:
− выявить все возможные пары с учетом требований кандидатов;
− при согласии сторон пара считается сформированной и кандидаты в списке помечаются как удаленные.

5. В фирме имеется информация о комплектующих изделиях компьютерной техники с указанием типа комплектующего изделия, фирмы-изготовителя, модели, параметров, цены, информации о наличие. Требуется:
− подобрать все возможные варианты комплектации компьютера в заданном ценовом диапазоне;
− осуществлять поиск изделия по типу изделия, фирме-изготовителю, модели;
− отсортировать изделия для каждого типа по убыванию цены.

6. В магазине имеется список поступивших в продажу CD/DVD дисков. Каждая запись списка содержит: тип хранимой информации (фильм, музыка, СОФТ и т. п.), наименование, автора, цену и примечание. Требуется:
− сортировать внутри каждого типа информацию по наименованию либо по автору;
− осуществлять поиск диска по автору, по наименованию;
− осуществлять выбор информации по типу, по автору.

7. В деканате имеется список студентов. Каждая запись этого списка содержит: номер группы, ФИО студента, оценки, полученные им за учебный год (две сессии: зимнюю и летнюю), средний балл по каждой сессии (вычисляется в программе). Требуется:
− сформировать список задолженников по результатам зимней сессии;
− сортировать списки по ФИО студентов внутри каждой группы;
− сформировать список студентов для отчисления (получивших две и более неудовлетворительные оценки за летнюю сессию и имевших задолженности (хотя бы одну неудовлетворительную оценку) за зимнюю сессию);
− осуществлять поиск студента в исходном списке по ФИО.

Библиотека БГУИР

8. В больнице имеется общий список больных. Каждая запись списка содержит: ФИО больного, пол больного, возраст, диагноз, номер палаты (при добавлении поле пустое). Требуется:

– разместить больных по палатам так, чтобы больные с одинаковым диагнозом располагались, по возможности, в одной палате. При этом разнополых больных в одну палату размещать нельзя. После размещения выдать список больных, которых не удалось разместить в палаты, с указанием ФИО больного, пола больного, возраста, диагноза;

– осуществлять поиск больных по номеру палаты, диагнозу, ФИО.

9. В проектной фирме есть список работ, закрепленный за сотрудниками. Каждая запись списка содержит: название проекта, задание в рамках данного проекта, ФИО исполнителя, ФИО руководителя, дату выдачи задания, срок выполнения, дату сдачи задания сотрудником. Требуется:

– выдать список всех проектов руководителя;

– выдать список всех задач сотрудника;

– выдать список всех сотрудников проекта;

– выдать список всех сотрудников, не справившихся с заданием в срок.

10. В проектной фирме есть список работ, выполняемых сотрудником. Каждая запись списка содержит: ФИО исполнителя, название проекта, задание в рамках данного проекта, дату выполнения задания, даты начала и окончания работы. Требуется:

– выдать список всех проектов фирмы, отсортированных по дате завершения работы;

– выдать список всех задач, работа над которыми велась одним сотрудником за текущие сутки;

– выдать список сотрудников с указанием суммарного времени работы каждого за прошедший месяц.

11. В проектной фирме есть список работ, закрепленный за сотрудниками. Каждая запись списка содержит: название проекта, задание в рамках данного проекта, ФИО исполнителя, ФИО руководителя, дату выдачи задания, срок выполнения. Требуется:

– выдать список всех задач по конкретному проекту и их исполнителей;

– выдать список всех задач, срок завершения выполнения которых – ближайший месяц.

12. Дана рейтинговая таблица футболистов, содержащая ФИО футболиста, клуб, амплуа, количество забитых мячей, сумму штрафных очков. Требуется:

– выдать список из 10 самых успешных игроков текущего сезона, имеющих максимальное количество забитых мячей и минимальное количество штрафных очков;

Библиотека БГУИР

- выдать список из 10 игроков с максимальной суммой штрафных очков;
- отсортировать записи по убыванию количества забитых мячей.

13. Разработать программу «Биржа труда». Имеется список фирм с вакансиями. Каждая запись списка содержит: название фирмы, должность, оклад, количество дней отпуска, требования к нанимаемому (наличие высшего образования (да/нет), возрастной диапазон (min/max); работа в данной должности не менее N-лет). Также имеется список кандидатов, каждая запись которого содержит: ФИО кандидата, дату рождения, наличие высшего образования (да/нет), желаемую должность, минимальный оклад, список должностей, занимаемых ранее, с периодом работы. Требуется:
- для каждой фирмы подобрать возможных кандидатов по каждой вакансии;
- выдать список дефицитных должностей (кандидаты отсутствуют);
- выдать список кандидатов, для которых не найдена вакансия.

14. Разработать программу «Биржа труда». Имеется список фирм с вакансиями. Каждая запись списка содержит: название фирмы, наименование специальности, должность, оклад, количество дней отпуска, требования к нанимаемому (наличие высшего образования (да/нет), возрастной диапазон (min/max)). Также имеется список кандидатов, каждая запись которого содержит: ФИО кандидата, дату рождения, специальность, наличие высшего образования (да/нет), желаемую должность, минимальный оклад. Требуется:
- для каждого кандидата подобрать список возможных вакансий;
- выдать список дефицитных вакансий (кандидаты отсутствуют);
- выдать список фирм, для которых не подобран ни один кандидат.

15. В ресторане имеется меню с указанием названия блюда, его описания и цены. Также имеется список заказов за текущие сутки. Каждая запись списка содержит: номер заказа, номер столика, название блюда, количество порций. Требуется:
- оформить счет по каждому заказу;
- выдать список блюд ресторана, пользующихся максимальным спросом;
- отсортировать список блюд по возрастанию цены.

16. В ресторане имеется меню с указанием названия блюда, его категории (холодные закуски, супы и т. п.) и цены. Также имеется список заказов за текущие сутки. Каждая запись списка содержит: номер заказа, номер столика, название блюда, количество порций. Требуется:
- определить самое «популярное» блюдо в категории;
- определить самый «прибыльный» заказ;
- выдать список заказов по убыванию суммы заказа.

Библиотека БГУИР

17. На фирме имеется список заказов на покупку товаров на следующие сутки. Каждая запись списка содержит: номер заказа, адрес доставки, дата и время (от – до) доставки, вес груза в килограммах. Также имеется список курьеров, каждая запись которого содержит: номер курьера, ФИО курьера, время работы (от – до), грузоподъемность автомобиля. Требуется:

- распределить заказы между курьерами;
- выдать список всех заказов курьера в последовательности их выполнения;
- выдать список всех заказов, которые не могут быть исполнены в срок.

18. На фирме формируются списки заказов на покупку товаров. Каждая запись списка заказов содержит: номер заказа, дату заказа, реквизиты заказчика. Каждый заказ ссылается на список товаров в заказе, который в свою очередь содержит: номер заказа, код товара, количество товара. В прайс-листе хранятся код товара, цена за единицу товара, наименование товара. Требуется сформировать накладную по каждому заказу. Накладная содержит список товаров конкретного заказа с указанием количества и цены, а также суммарную стоимость заказа. Номер заказа и суммарная стоимость указывается в «шапке» накладной.

19. В поликлинике генерируется список талонов к врачу. Каждая запись списка содержит: дату, время, номер очереди, ФИО больного (изначально поле пустое), номер кабинета, ФИО врача. Генерация талонов происходит на неделю, начиная с введенной даты, в соответствии с графиком работ врачей. График работы врача содержит: специализацию врача, ФИО врача, временной диапазон работы на каждый день с понедельника по субботу (длительность приема врачом больного – 15 минут). Требуется:

- сформировать списки талонов к врачу;
- осуществлять поиск всех записей к врачу на конкретную дату;
- осуществлять поиск записей о больном по ФИО.

20. В библиотеке имеется список книг. Каждая запись списка содержит: фамилию автора (или авторов), название книги, год издания, издательство, количество страниц. Требуется:

- определить, имеются ли в данном списке книги, в названии которых встречается некоторое ключевое слово (например «программирование»);
- осуществлять поиск книги по ФИО автора либо по названию;
- отсортировать все книги библиотеки по году издания;
- выдать все книги одного издательства.

21. В магазине имеется список поступивших в продажу автомобилей. Каждая запись списка содержит: страну-изготовитель, марку автомобиля, параметры автомобиля (тип двигателя, стоимость, расход бензина на 100 км, надежность (число лет безотказной работы), комфортность (в баллах).

Библиотека БГУИР

Покупатель, в свою очередь, имеет ряд требований по каждому из этих параметров. Эти требования могут задаваться в виде некоторого интервала (например стоимость – 10…30 тыс. ($)) либо выбираться из перечисленных списков. Необходимо:

− осуществлять поиск автомобилей, удовлетворяющих требованиям покупателя;

− отсортировать автомобили по маркам, а внутри списка одной марки по любому выбранному покупателем параметру.

22. В предвыборной кампании производится регистрация кандидатов в депутаты. Каждый кандидат при регистрации, указывает номер округа, в котором он собирается баллотироваться, ФИО, наименование партии, которую он представляет, возраст, профессию, доход за прошедший год. Требуется:

− сформировать информационный бюллетень, в котором приводится следующая информация по кандидатам от каждой политической партии: число поданных заявлений на регистрацию, средний возраст кандидатов, наиболее часто встречающаяся профессия и средний доход кандидатов;

− осуществлять поиск полной информации по кандидатам каждой партии.

23. В технической службе аэропорта имеется справочник, содержащий записи следующей структуры: тип самолета, год выпуска, расход горючего на 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов на следующие сутки. Каждая запись расписания содержит: номер рейса, пункт назначения, дальность полета, тип самолета, время вылета, время приземления в пункте назначения. Требуется:

− рассчитать суммарное количество горючего, необходимое для обеспечения полетов на следующие сутки;

− осуществлять поиск всех рейсов в конкретный пункт назначения;

− составить документ-расписание полетов.

24. В библиотеке имеется список книг. Каждая запись списка содержит: код книги, фамилию автора, название книги, год издания, язык издания. Также имеется список читателей, каждая запись которого содержит: код читателя, ФИО читателя, домашний адрес, контактный телефон. Книги, взятые каждым читателем, заносятся в отдельный список, содержащий код читателя, код книги, дату выдачи, требуемый срок возврата и реальную дату возврата книги. Требуется:

− создать список книг, находящихся у читателей;

− создать список должников (книгу не вернули в течение десяти дней после срока возврата);

− осуществлять поиск книг по ФИО автора либо по названию;

− осуществлять поиск читателя по ФИО.

Библиотека БГУИР

25. У администратора железнодорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата отправления, номер рейса, конечный пункт назначения, время отправления, число купейных и плацкартных мест, число свободных купейных и плацкартных мест. Требуется:

– выдать информацию об имеющихся свободных местах по каждому рейсу и каждому типу мест;

– выдать документ-расписание движения поездов по каждому дню недели;

– выдать список всех рейсов в пункт назначения с указанием общего числа свободных мест и числа проданных билетов.

26. Имеется ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке этой ведомости записана ФИО абитуриента, специальность, на которую он поступает, средний балл аттестата, полученные оценки по отдельным дисциплинам (например, физика, математика, русский язык). Требуется:

– выдать информацию по каждой специальности, содержащую ФИО абитуриента, суммарный балл каждого из них;

– отсортировать записи по каждой специальности по убыванию суммарного балла;

– осуществлять поиск абитуриента по ФИО.

27. В больнице имеется общий список больных. Каждая запись списка содержит: ФИО больного, пол больного, возраст, диагноз, номер палаты, дату поступления, дату выписки. Требуется:

– указать номера палат, в которых лежат больные с более чем тремя разными диагнозами;

– сортировать списки невыписавшихся больных по убыванию даты поступления;

– осуществлять поиск больных по номеру палаты, полу, диагнозу, ФИО, возрасту (в режиме диапазона min – max).

28. Разработать программу («электронную сваху») для службы знакомств. Имеется один список женихов и невест. В списке кандидат (жених или невеста) характеризуются записью, содержащей следующие поля: порядковый номер кандидата, данные о кандидате (пол, ФИО, возраст, рост, вес), требования к партнеру (в виде диапазона min – max для возраста, роста, веса). Требуется:

– выявить все возможные пары;

– осуществлять поиск всех возможных кандидатов;

– вывести отсортированный по убыванию возраста список для кандидатов-женихов и кандидатов-невест.

Библиотека БГУИР

29. Разработать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: наименование специальности, номер группы, ФИО студента, форму обучения (бюджетная, платная), оценки, полученные им в сессию, средний балл, вычисляемый программно. Требуется:

- отсортировать в группах студентов по ФИО;
- вывести список студентов, которые учатся на «отлично» (имеют 9- и 10-балльные оценки) с формой обучения «платная»;
- вывести списки студентов в разрезе формы обучения в порядке убывания среднего балла;
- осуществлять поиск студентов по ФИО, номеру группы, форме обучения.

30. На складе комплектующих имеется список деталей, каждая из которых характеризуется принадлежностью к группе товаров, наименованием товара, количеством, рядом признаков: признак_1 (числовой), признак_2 (строковый), признак_3 (логический). Требуется:

- отсортировать товары в группе по убыванию признака_1;
- выдавать информацию по запросу о количестве имеющихся деталей по каждому наименованию, о количестве деталей указанного наименования с заданными признаками, о количестве деталей различных наименований, имеющих одинаковые один, два или три признака.

3.2. Лабораторная работа по теме «Множество»

Цель работы: разработка, отладка и выполнение программы обработки множеств.

Для заданного преподавателем варианта необходимо разработать программу на языке Delphi. Программа должна содержать необходимые комментарии. Следует предусмотреть простейший вывод на экран входных и выходных данных.

Варианты заданий

1. Заданы:

```
Type
  ПРОДУКТ = (хлеб, масло, молоко, мясо, рыба, соль, сыр, колбаса, сахар,
             чай, кофе);
  АССОРТИМЕНТ = Set of  ПРОДУКТ;
  МАГАЗИНЫ = Array [1..20] of АССОРТИМЕНТ
```

Библиотека БГУИР

Описать процедуру *Наличие (Маг,A,B,C)*, которая по информации из массива Маг типа МАГАЗИНЫ (Маг[i] – это множество продуктов, имеющихся в i-том магазине) присваивает параметрам A, B, C типа АССОРТИМЕНТ следующие значения: A – множество продуктов, которые есть во всех магазинах; B – множество продуктов, каждый из которых есть хотя бы в одном магазине.

2. В соответствии с определенными в задании 1 типами описать процедуру *Наличие (Маг,A,B,C)*, которая по информации из массива Маг типа МАГАЗИНЫ (Маг[i] – это множество продуктов, имеющихся в i-том магазине) присваивает параметрам A, B, C типа АССОРТИМЕНТ следующие значения: C – множество продуктов, которых нет ни одном магазине; D – множество продуктов, которые есть только в одном магазине.

3. Заданы:

```
Type
   ИМЯ = (ВАСЯ, ВОЛОДЯ, ИРА, ЛИДА, МАРИНА, МИША, НАТАША,
                  ОЛЕГ,  ОЛЯ, СВЕТА, ЮЛЯ);
   ГОСТИ = SET of  ИМЯ;
   ГРУППА = Array [ИМЯ] of  ГОСТИ
```

Описать логическую функцию *Везде (ГР)*, определяющую, есть ли в группе *ГР* типа ГРУППА хотя бы один человек, побывавший в гостях у всех остальных из группы, и указать его имя (ГР [x] – множество людей, бывших в гостях у человека с именем X; X не принадлежит множеству ГР [X]).

4. В соответствии с определенными в задании 3 типами описать логическую функцию *Нигде (ГР)*, определяющую, есть ли в группе *ГР* типа ГРУППА хотя бы один человек, не побывавший в гостях ни у одного человека из группы, и указать его имя (ГР [x] – множество людей, бывших в гостях у человека с именем X; X не принадлежит множеству ГР [X]).

5. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. Вывести в алфавитном порядке все гласные буквы, которые входят в каждое слово.
Примечание: гласные буквы – а, е, и, о, у, ы, э, ю, я;
согласные буквы – все остальные буквы, кроме й, ь, ъ;
звонкие согласные – б, в, г, д, ж, з, л, м, н, р;
глухие согласные – к, п, с, т, ф, х, ц, ч, ш, щ.

6. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все глухие согласные буквы, которые не входят только в одно слово.

Библиотека БГУИР

7. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все согласные буквы, которые не входят ни в одно слово.

8. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все звонкие согласные буквы, которые входят более чем в одно слово.

9. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке звонкие согласные буквы, которые входят хотя бы в одно слово.

10. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.

11. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все глухие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово.

12. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все глухие согласные буквы, которые не входят хотя бы в одно слово.

13. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все гласные буквы, которые не входят более чем в одно слово.

14. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все звонкие согласные буквы, которые не входят ни в одно нечетное слово и входят в каждое четное слово.

Библиотека БГУИР

15. Дана непустая последовательность слов из строчных русских букв; между соседними словами – запятая, за последним словом точка. В соответствии с примечанием к заданию 5 вывести в алфавитном порядке все согласные буквы, которые входят только в одно слово.

16. Решить задачу-ребус.

```
   V O L V O
+  F I A T
   M O T O R
```

Каждая буква – это цифра, разным буквам соответствуют разные цифры. Необходимо заменить буквы цифрами так, чтобы получилось верное равенство. Найти все решения (если есть несколько решений).

17. Решить задачу-ребус.

$$ABC = AB + BC + CA$$

Каждая буква – это цифра, разным буквам соответствуют разные цифры. Необходимо заменить буквы цифрами так, чтобы получилось верное равенство. Найти все решения (если есть несколько решений).

18. Решить задачу-ребус.

```
   И К С
+  И С К
   К С И
```

Каждая буква – это цифра, разным буквам соответствуют разные цифры. Необходимо заменить буквы цифрами так, чтобы получилось верное равенство. Найти все решения (если есть несколько решений).

19. Решить задачу-ребус

```
    Т О Ч К А
+   К Р У Г
    К О Н У С
```

Каждая буква – это цифра, разным буквам соответствуют разные цифры. Необходимо заменить буквы цифрами так, чтобы получилось верное равенство. Найти все решения (если есть несколько решений).

20. Написать программу для ввода латинских букв алфавита. Если введенный символ не является буквой латинского алфавита, заменить его знаком «?». Заменить все введенные прописные буквы латинского алфавита строчными. Следить за тем, чтобы буквы не повторялись (введенная повторно буква заменяется символом «*»). Вывести на экран исходный и скорректированный текст.

Библиотека БГУИР

21. Разработать игровую программу для тренировки памяти. В ее основу положить следующие правила игры. Программа выдает пользователю множество чисел длины, соответствующей уровню сложности. Через некоторое время числа исчезают с экрана, и игрок должен воспроизвести увиденное им множество. На экран выводить количество правильно введенных чисел.

22. Дано натуральное число N. Составить программу, печатающую все цифры, не входящие в десятичную запись данного натурального числа в порядке возрастания.

23. Даны три множества X1, X2, X3, содержащие целые числа из диапазона 1…100. Известно, что мощность каждого множества равна 10. Сформировать новое множество Y:

$$Y = (X1 + X2) * (X2 – X3).$$

На экран вывести исходные множества, а также полученное множество.

24. Даны три множества X1, X2, X3, содержащие целые числа из диапазона 1…100. Известно, что мощность каждого множества равна 10. Сформировать новое множество Y:

$$Y = (X1 - X2) + (X2 * X3).$$

На экран вывести исходные множества, а также полученное множество.

25. Дан некоторый текст, за которым следует точка (в сам текст точка не входит). Определить, является ли этот текст правильной записью Формулы:

<Формула>::= <Терм> | (< Формула > <Знак> < Формула >).
<Знак>::= + | - | *.
<Терм>::=<Имя> | <Целое>.
<Имя>::=<Буква> | <Имя><Буква> |<Имя><Цифра>.
<Целое>::=<Цифра> | <Целое><Цифра>.
<Буква>::=а | б | в | г | д | е | ж.
<Цифра>::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 .

26. Заданы:

```
Type
    ГОРОД = (A, B, C, D, E, F, G, H);
    ГОРОДА = SET of  ГОРОД;
    РЕЙСЫ = Array [ГОРОД] of  ГОРОДА
```

Библиотека БГУИР

Описать процедуру *МожноПопасть (P,H,K)*, которая по рейсам P (P[x] – множество городов, в которые можно за один рейс доехать автобусом из города «x») определяет K – множество городов, в которые можно попасть автобусом за один рейс или через другие города из города H.

27. В соответствии с определенными в задании 26 типами описать процедуру *МожноПопасть (P,H,K)*, которая по рейсам P (P[x] – множество городов, в которые можно за один рейс доехать автобусом из города «x») определяет K – множество городов, в которые можно попасть автобусом только через другие города из города H.

28. Заданы:

Type

ВЕРШИНА = (b1, b2, b3, b4, b5, b6, b7, b8);
СОСЕДИ = SET of Вершина;
ГРАФ = *Array [Вершина] of Соседи*

Описать логическую функцию *Path (G,N,K)*, которая определяет, есть ли в ориентированном графе G путь из вершины N в вершину K.

29. В соответствии с определенными в задании 28 типами описать логическую функцию *Path (G,N,K,D)*, которая определяет, есть ли в ориентированном графе G путь из вершины N в вершину K, и, если есть, присваивает параметру D длину (число дуг) кратчайшего пути из N в K.

30. В соответствии с определенными в задании 28 типами описать процедуру *Path (G,N,K,D)*, которая определяет, есть ли в ориентированном графе G путь из вершины N в вершину K, и, если есть, присваивает параметру D длину (число дуг) максимального пути из N в K; в случае отсутствия пути параметру D присваивается значение –1.

3.3. Лабораторная работа по теме «Динамические структуры»

Цель работы: разработка, отладка и выполнение программы обработки динамических структур данных.

Для заданного преподавателем варианта необходимо разработать программу на языке Delphi. Программа должна содержать необходимые комментарии. Следует предусмотреть простейший вывод на экран входных и выходных данных.

Во всех заданиях должны быть описаны функции и процедуры для работы с соответствующим типом данных (например для стека: добавить элемент в стек, удалить элемент из стека).

Библиотека БГУИР

Варианты заданий

1. Даны натуральное число N, действительные числа $X_1 ... X_n$. Вычислить, используя однонаправленный список с дисциплиной обслуживания «стек»:

$$X_1 * X_n + X_2 * X_{n-1} + \ldots + X_n * X_1 .$$

2. Даны натуральное число N, действительные числа $X_1 ... X_n$. Вычислить, используя однонаправленный список с дисциплиной обслуживания «очередь»:

$$(X_1 + X_n) * (X_2 + X_{n-1}) * \ldots * (X_n + X_1) .$$

3. Даны натуральное число N, действительные числа $X_1 ... X_n$. Вычислить, используя двунаправленный список:

$$(X_1 + X_2 + 2 * X_n) * (X_2 + X_3 + 2 * X_{n-1}) * \ldots * (X_{n-1} + X_n + 2 * X_2) .$$

4. Даны натуральное число N, действительные числа $A_1 ... A_n$. Выяснить, используя однонаправленный список с дисциплиной обслуживания «стек», имеются ли среди чисел $A_1 \ldots A_n$ совпадающие.

5. Даны целые числа $A_1 \ldots A_i$. Известно, что среди $A_1 \ldots A_i$ есть хотя бы одно отрицательное число. Пусть $A_1 \ldots A_n$ – члены данной последовательности, предшествующие первому отрицательному члену (n – заранее неизвестно). Получить, используя однонаправленный список с дисциплиной обслуживания «очередь»:

$$\text{Min} (A_1 + A_2, A_2 + A_3, \ldots, A_{n-1} + A_n) .$$

6. Даны натуральное число N, действительные числа $A_1 ... A_{2n}$. Вычислить, используя двунаправленный список:

$$(A_1 - A_{2n}) * (A_3 - A_{2n-2}) * (A_5 - A_{2n-4}) * \ldots * (A_{2n-1} - A_2) .$$

7. Даны натуральное число N, действительные числа $A_1 ... A_{2n}$. Вычислить, используя однонаправленный список с дисциплиной обслуживания «стек»:

$$A_1 * A_{2n} + A_2 * A_{2n-1} + \ldots + A_n * A_{n+1} .$$

8. Даны натуральное число N, действительные числа $A_1 ... A_{2n}$. Вычислить, используя однонаправленный список с дисциплиной обслуживания «очередь»:

$$\text{Min} (A_1 + A_{n+1}, A_2 + A_{n+2}, \ldots, A_n + A_{2n}) .$$

9. Даны целые числа $A_1 \ldots A_i$. Известно, что среди $A_1 \ldots A_i$ есть хотя бы одно отрицательное число. Пусть $A_1 \ldots A_n$ - члены данной последовательности, предшествующие первому отрицательному члену (n – заранее неизвестно). Вычислить, используя двунаправленный список:

Библиотека БГУИР

$$\text{Max}\ (A_1,\ A_1{*}A_2,\ A_1{*}A_2{*}A_3,\ \ldots\ ,\ A_1{*}A_2{*}A_3{*}\ldots{*}A_n)\ .$$

10. Даны целые числа $A_1 \ldots A_i$. Известно, что среди $A_1 \ldots A_i$ есть хотя бы одно отрицательное число. Пусть $A_1 \ldots A_n$ – члены данной последовательности, предшествующие первому отрицательному члену (n – заранее неизвестно). Получить количество полных квадратов среди чисел $A_1 \ldots A_n$, используя однонаправленный список с дисциплиной обслуживания «стек».

11. Даны натуральное число N, действительные числа $A_1 \ldots A_{2n}$. Вычислить, используя однонаправленный список с дисциплиной обслуживания «очередь»:

$$\text{Max}\ (\text{Min}\ (A_1,\ A_{2n}),\ \text{Min}(A_2,\ A_{2n-1}),\ \text{Min}\ (A_n,\ A_{n+1}))\ .$$

12. Даны натуральное число N, действительные числа $A_1 \ldots A_{2n}$. Выяснить, используя двунаправленный список, что для $i = 1, \ldots, N$ выполнено:

$$A_i = -A_{n+i}\ .$$

13. Даны натуральное число N, действительные числа $A_1 \ldots A_{2n}$. Выяснить, используя однонаправленный список с дисциплиной обслуживания «стек», что для $i = 1, \ldots, N$ выполнено:

$$A_i = 2{*}A_{n-i} + A_{2{*}n-i+1}\ .$$

14. Даны натуральное число N, действительные числа $A_1 \ldots A_{2n}$. Выяснить, используя однонаправленный список с дисциплиной обслуживания «очередь», что для $i = 1, \ldots, N$ выполнено:

$$A_i + A_{2{*}n-i+1} > 17\ .$$

15. Даны натуральное число N, действительные числа $A_1 \ldots A_{2n}$. Выяснить, используя двунаправленный список, что для $i = 1, \ldots, N$ выполнено:

$$A_{2{*}n-i+1} < A_i <= A_{2{*}n-I}\ .$$

16. Даны целые числа $A_1 \ldots A_i$. Известно, что среди $A_1 \ldots A_i$ есть хотя бы одно отрицательное число. Пусть $A_1 \ldots A_n$ – члены данной последовательности, предшествующие первому отрицательному члену (n – заранее неизвестно). Вычислить, используя однонаправленный список с дисциплиной обслуживания «стек»:

$$\text{Min}\ (A_1,\ 2{*}A_2,\ 3{*}A_3,\ n{*}A_n)\ .$$

17. Одно из возможных представлений «длинного» текста – разделение его на участки (строки) равной длины. Используя представление текста в виде двунаправленного списка, описать:

Библиотека БГУИР

1) функцию *Числострок (T)* для подсчета числа строк в тексте T;

2) процедуру *Добавить (T,I,J)*, добавляющую после I-той строки текста T копию J-той строки;

3) процедуру *Перестановка (T,I,J),* меняющую местами I-тую и J-тую строки текста T.

18. Одно из возможных представлений «длинного» текста – разделение его на участки (строки) равной длины. Используя представление текста в виде однонаправленного списка, описать:

1) логическую функцию *Элем (T,I,J,C),* проверяющую, есть ли в тексте T строка с номером I, и если есть, присваивающую значение J-го символа этой строки параметру C;

2) процедуру *Удалить (T,I)*, удаляющую I-тую строку из текста T;

3) процедуру *Замена (T,I,J),* заменяющую I-тую строку текста T на копию J-той строки.

19. Одно из возможных представлений «длинного» текста – разделение его на участки (строки) равной длины. Используя представление текста в виде двунаправленного списка, описать:

1) логическую функцию *Поиск (T,I,J,C),* определяющую, входит ли символ C в текст T, и, если входит, присваивающую параметрам I и J координаты» первого вхождения этого символа: I – номер строки, а J – номер позиции в этой строке;

2) процедуру *Удалить (T,I)*, удаляющую I-тую строку из текста T;

3) процедуру *Замена (T,I,J),* заменяющую I-тую строку текста T на копию J-той строки.

20. В текстовом файле записан текст, сбалансированный по круглым скобкам (использовать очередь или стек). Требуется для каждой пары соответствующих открывающих и закрывающих скобок вывести номера их позиций в тексте, упорядочив пары номеров в порядке возрастания номеров позиций закрывающих скобок.

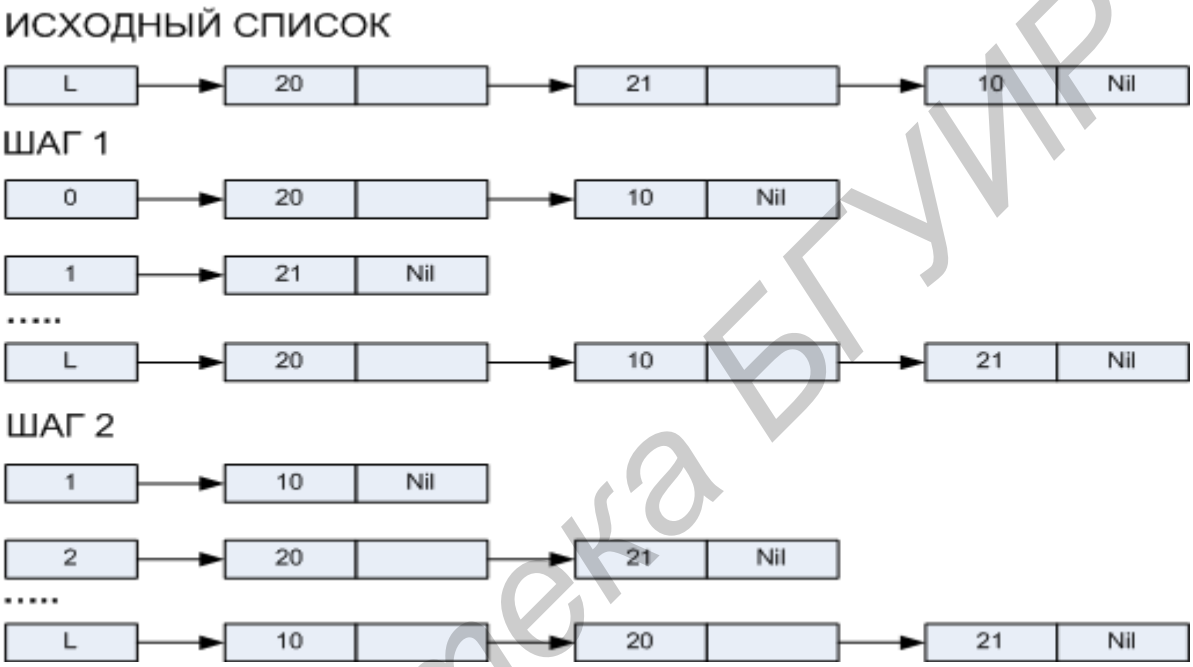
21. В текстовом файле записан текст, сбалансированный по круглым скобкам (использовать очередь или стек). Требуется для каждой пары соответствующих открывающих и закрывающих скобок напечатать номера их позиций в тексте, упорядочив пары номеров в порядке возрастания номеров позиций открывающих скобок.

22. Описать процедуру, упорядочивающую по неубыванию числа непустого списка L с помощью следующего алгоритма: создать 10 пустых подсписков (по количеству цифр), а затем, просматривая числа исходного списка, занести в K-й подсписок все числа, оканчивающиеся цифрой K, после

Библиотека БГУИР

чего эти подсписки объединить в один список L, записав в последнее звено К-го подсписка ссылку на начало (К+1)-го подсписка. Далее аналогичный метод применяется по отношению к предпоследней цифре чисел (не нарушая при этом упорядоченность по последней цифре), затем – по отношению к третьей от конца цифре и т. д.

Например:
Const N=2;
Type Число=string [N]



23. Под «выражением» будем понимать следующую конструкцию:

<Выражение>::=<Терм> | <Терм><Знак+-><Выражение> .
<Знак +->::= + | - .
<Терм>::=<Множитель> | <Множитель> * <Терм> .
<Множитель>::=<Число> | <Переменная> | (<Выражение>) | <Множитель> @ <Число> .
<Число>::=<Цифра> .
<Переменная>::=<Буква>. ,

где знак @ – обозначает операцию возведения в степень.

Постфиксной формой записи выражения называется запись, в которой знак операции размещен за операндами. Например:

а+в	ав+
а*в+с	ав*с+
а*(в+с)	авс+*
а+в@с@d*e	abc@d@e*+

Библиотека БГУИР

Описать функцию, которая вычисляет как целое число значение выражения (без переменных), записанного в постфиксной форме в текстовый файл.

Для решения задачи использовать следующий алгоритм: выражение просматривается слева направо; если встречается операнд (число), то его значение (как целое) заносится в стек, а если встречается знак операции, то из стека извлекаются два последних элемента (это операнды данной операции); над ними выполняется операция, и ее результат записывается в стек. В конце концов в стеке останется только одно число – значение всего выражения.

24. С учетом определения выражения и его записи в постфиксной форме, приведенных в задании 23, описать процедуру, которая переводит выражение, записанное в обычной (инфиксной) форме в текстовом файле infix, в постфиксную форму и в таком виде записывает его в текстовый файл postfix.

Для решения задачи использовать следующий алгоритм: в стек записывается открывающая скобка, и выражение просматривается слева направо; если встречается операнд (число или переменная), то он сразу переносится в файл postfix; если встречается открывающая скобка, то она заносится в стек, а если встречается закрывающая скобка, то из стека извлекаются находящиеся там знаки операций до ближайшей открывающей скобки, которая также удаляется из стека, и все эти знаки (в порядке их извлечения) записываются в файл postfix; когда же встречается знак операции, то из конца стека извлекаются (до ближайшей скобки, которая сохраняется в стеке) знаки операций, старшинство которых больше или равно старшинству данной операции, и они записываются в файл postfix; после чего рассматриваемый знак заносится в стек; в заключение выполняются такие же действия, как если бы встретилась закрывающая скобка.

25. Пусть L обозначает кольцевой (циклический) двунаправленный список с заглавным звеном. Описать процедуры или функции для работы с таким списком:

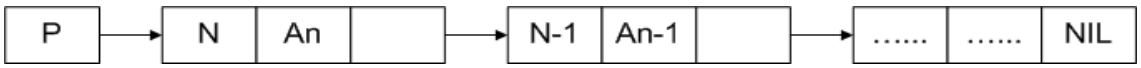
1) подсчитывающую количество элементов списка L, у которых равные «соседи»;

2) определяющую, есть ли в списке L хотя бы один элемент, который равен следующему за ним (по кругу) элементу;

3) в списке L переставляющую в обратном порядке все элементы между первым и последним вхождениями элемента E, если E входит в L не менее двух раз.

26. Степенной многочлен N-й степени с целыми коэффициентами Ai можно представить в виде списка, причем если коэффициент Ai равен 0, то соответствующее звено отсутствует в списке:

Библиотека БГУИР



Описать тип данных, соответствующий такому представлению, и определить следующие функции и процедуры для работы с этими списками-многочленами:

1) процедуру, которая считывает из входного файла безошибочную запись двух многочленов и формирует соответствующие списки-многочлены P и Q;

2) логическую функцию, проверяющую на равенство многочлены P и Q;

3) функцию, вычисляющую значение многочлена P в целой точке X.

27. С учетом представления многочлена N-й степени, приведенного в задании 26, описать соответствующий тип данных и определить следующие функции и процедуры для работы с этим списком-многочленом:

1) процедуру, которая считывает из входного файла безошибочную запись многочлена и формирует соответствующий список-многочлен P;

2) функцию, вычисляющую значение многочлена P в целой точке X;

3) процедуру, которая строит многочлен Q – производную многочлена P.

28. Используя стек, решить следующую задачу (решение описать в виде функции или процедуры). Проверить, является ли содержимое текстового файла T правильной записью формулы следующего вида:

<Формула> ::= <Терм> | <Терм> + <Формула> | <Терм> - <Формула>.
<Терм> ::= <Имя> | (<Формула>) | {<Формула>} | [<Формула>] .
<Имя> ::= <X> | <Y> | <Z>.

29. Используя стек, решить следующую задачу (решение описать в виде функции или процедуры). В текстовом файле LOG записано без ошибок логическое выражение (ЛВ) в следующей форме:

<ЛВ>::=true | false | (!<ЛВ>) | (<ЛВ> ^ <ЛВ>) | (<ЛВ> & <ЛВ>) . ,

где знаки !, &, ^ обозначают соответственно отрицание, дизъюнкцию конъюнкцию.

Вычислить значение этого выражения типа boolean.

30. Используя стек, решить следующую задачу (решение описать в виде функции или процедуры). В текстовом файле F записана без ошибок формула следующего вида:

<Формула>::=<Цифра>|M(<Формула>,<Формула>)|м(<Формула>,<Формула>),
где M – обозначает функцию max, а м – функцию min;

<Цифра>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 .

Вычислить (как целое число) значение данной формулы (например для M(5,м(6,8)) будет получен результат 6).

Библиотека БГУИР

31. Назовем иерархическим списком заключенную в круглые скобки последовательность элементов, разделенных запятыми; элементы списка – это атомы или снова списки:

<Список>::=( ) | (<Элементы>).
<Элементы>::=<Элемент> | <Элемент>, <Элемент>.
<Элемент>::=<Атом> | <Список>.

Под атомом понимается последовательность, содержащая от 1 до N букв и цифр, где N – заранее известное натуральное число. Пример подобного списка: (AD75,(3,(),(7H))).

Предложить и описать представление таких списков и определить следующие рекурсивные функции и процедуры для работы с ними:

1) процедуру, которая считывает из входного файла записанный без ошибок список и строит L – соответствующее представление этого списка;
2) логическую функцию, проверяющую, входит ли атом A в список L;
3) процедуру, печатающую все атомы, входящие в список.

32. С учетом определения иерархического списка и атома, приведенных в задании 31, предложить и описать представление таких списков и разработать следующие рекурсивные функции и процедуры для работы с ними:

1) процедуру, которая считывает из входного файла записанный без ошибок список и строит L – соответствующее представление этого списка;
2) логическую функцию, проверяющую на равенство списки L1 и L2;
3) процедуру, печатающую список L в том виде, как он определен указанными в задании 31 металингвистическими формулами.

3.4. Содержание отчета по лабораторным работам

Отчет по лабораторной работе должен содержать следующие разделы:

1) постановка задачи;
2) выбор метода решения и его обоснование;
3) алгоритмы решения задачи;
4) выбор и обоснование структур данных;
5) список идентификаторов программы и подпрограмм с указанием назначения и типа идентификатора, представленный в виде таблицы;
6) перечень тестов и наборы данных для тестирования программы;
7) листинг структурированной программы с комментариями (комментарии должны содержать вводный комментарий к программе, комментарии- заголовки к подпрограммам и модулям, построчные комментарии из расчета один комментарий на 2–3 строчки исходного текста).

Библиотека БГУИР

ЛИТЕРАТУРА

1. ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992–01–01. – М. : Изд-во стандартов, 1991.

2. Глухова, Л. А. Основы алгоритмизации и программирования : лаб. практикум для студ. спец. «Программное обеспечение информационных технологий» днев. формы обуч. В 4 ч. Ч 1 / Л. А. Глухова, Е. П. Фадеева, Е. Е. Фадеева, С. В. Болтак. – Минск : БГУИР, 2004. – 41 с.

3. Глухова, Л. А. Основы алгоритмизации и программирования : лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» днев. формы обуч. В 4 ч. Ч 2 / Л. А. Глухова, Е. П. Фадеева, Е. Е. Фадеева. – Минск : БГУИР, 2005. – 52 с.

4. Глухова, Л. А. Основы алгоритмизации и программирования: Процедуры и функции языка Pascal : Лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» днев. формы обуч. В 4 ч. Ч 3 / Л. А. Глухова, Е. П. Фадеева, Е. Е. Фадеева. – Минск : БГУИР, 2007. – 51 с.

5. Фаронов, В. В. DELPHI. Программирование на языке высокого уровня : учебник для вузов / В. В. Фаронов. – СПб. : Питер, 2010. – 640 с.

6. Тюкачев, Н. А. Программирование в Delphi для начинающих / Н. А. Тюкачев , К. С. Рыбак, Е. Е. Михайлова. – СПб. : BHV, 2007. – 762 с.

7. Рубанцев, В. Delphi в примерах, играх и программах: Дополнительные материалы / В. Рубанцев. – М. : Я+Я, 2011. – 408 с.

8. Учебник по Delphi 7 для начинающих [Электронный ресурс]. – 2012. – Режим доступа : http://udelphi7.gym5cheb.ru/.

9. Иллюстрированный онлайн учебник по Borland Delphi 7 с примерами [Электронный ресурс]. – 2012. – Режим доступа: http://delphi.support.uz/.

Библиотека БГУИР

ПРИЛОЖЕНИЕ 1

ОПИСАНИЕ ПУНКТОВ И ПОДПУНКТОВ МЕНЮ

Таблица П.1.1.

Пункты главного меню

Пункт меню	Комментарии
1	2
Файл (File)	Управление проектами и отдельными файлами
Правка (Edit)	Операции, связанные с редактированием текста, а также предоставляемые дизайнером форм
Поиск (Search)	Поиск и замена текстовых строк, а также получение информации об именах переменных, констант и типов
Вид (View)	Отображение/скрытие окон среды и встроенного отладчика
Проект (Project)	Настройка свойств проекта, добавление/удаление модулей, компиляция и сборка приложений
Запуск (Run)	Запуск приложений, инструменты встроенного отладчика
Компонент (Component)	Управление компонентами приложений, элементами управления ActiveX, пакетами и палитрой компонентов
Базы Данных (Database)	Инструменты для создания и управления базами данных
Инструменты (Tools)	Управление настройками среды, текстового редактора, отладчика, инструментов перевода и другими вспомогательными инструментами
Окно (Window)	Управление окнами среды
Помощь (Help)	Получение справочной информации по среде, языку программирования Delphi

Таблица П.1.2.

Подпункты пункта меню Файл (File)

Подпункт меню	Комментарии
1	2
Новый (New)	Позволяет создавать новые проекты, модули и документы
Открыть (Open)	Открывает для редактирования ранее созданные и сохраненные файлы (в том числе и целые проекты)
Открыть Проект (Open Project)	Открывает ранее созданный проект, загружая все связанные с ним файлы
Переоткрыть (Reopen)	Позволяет быстро открыть файлы, которые уже открывались ранее
Сохранить (Save)	Сохраняет редактируемый файл под его текущим именем
Сохранить Как (Save As)	Сохраняет редактируемый файл под новым именем
Сохранить Проект Как (Save Project As)	Сохраняет текущий проект под новым именем
Сохранить Все (Save All)	Сохраняет все открытые файлы
Закрыть (Close)	Закрывает текущий проект и все связанные с ним модули и формы
Закрыть Все (Close All)	Закрывает все открытые файлы

Библиотека БГУИР

Окончание таблицы П.1.2.

1	2
Использовать Модуль (Use Unit)	Позволяет подключить выбранный модуль к текущему модулю
Печать (Print)	Позволяет распечатать редактируемый модуль, форму или документ
Выход (Exit)	

Таблица П.1.3.

Подпункты пункта меню Правка (Edit)

Подпункт меню	Комментарии
1	2
Восстановить (Undo/Undelete)	Позволяет отменять последние изменения, внесенные в код программы
Восстановить (Redo)	Восстанавливает изменения, отмененные предыдущим пунктом меню
Вырезать (Cut)	Перемещает выделенный фрагмент кода в буфер обмена, удаляя его из текста программы
Копировать (Copy)	Копирует выделенный фрагмент кода в буфер обмена, не удаляя его из текста программы
Вставить (Paste)	Копирует содержимое буфера обмена в текущую позицию курсора
Удалить (Delete)	Удаляет выделенный фрагмент кода
Выбрать Все (Select All)	Выделяет весь текст редактируемого файла
Привязать к Сетке (Align to Grid)	Размещает выбранные в дизайнере форм компоненты, привязывая к сетке
Перенести Вперед (Bring to Front)	Размещает выбранный в дизайнере форм компонент поверх всех остальных
Послать Назад (Send to Back)	Размещает выбранный в дизайнере форм компонент под всеми остальными
Привязка (Align)	Позволяет настроить тип привязки, используемый дизайнером форм
Размер (Size)	Позволяет изменять размеры компонента, выбранного в дизайнере форм
Масштаб (Scale)	Позволяет масштабировать компоненты в дизайнере форм
Порядок Tab (Tab Order)	Позволяет настроить порядок переключения между элементами управления
Порядок Создания (Creation Order)	Позволяет настроить порядок создания компонентов
Отразить Детские (Flip Children)	Позволяет горизонтально отразить расположение элементов управления текущей формы (при написании приложений для стран, где текст читается справа налево)
Блокировать Управления (Lock Controls)	Фиксирует текущие положения и размеры элементов управления
Добавить в Интерфейс (Add to interface)	Позволяет добавить новый метод или свойство к ActiveX-компоненту

Библиотека БГУИР

Таблица П.1.4.

Подпункты пункта меню Поиск (Search)

Подпункт меню	Комментарии
1	2
Найти (Find)	Осуществляет поиск первого вхождения заданной подстроки в тексте программы
Поиск в файлах (Find in Files)	Осуществляет поиск всех вхождений заданной подстроки и выводит их в нижней части окна редактора кода
Заменить (Replace)	Осуществляет замену заданной подстроки заданным текстом
Искать дальше (Search Again)	Повторяет поиск ранее заданного фрагмента, начиная с текущей позиции курсора
Инкрементальный Поиск (Incremental Search)	Инкрементальный Поиск (Incremental Search) – ищет заданную подстроку в тексте программы, отображая найденные вхождения по мере ввода пользователем подстроки для поиска
Идти к строке номер (Go to Line Number)	Перемещает курсор к строке с заданным порядковым номером
Поиск Ошибки (Find Error)	Перемещает курсор к месту возникновения последней ошибки времени выполнения
Обзор Символа (Browse Symbol)	Осуществляет поиск всех вхождений заданного идентификатора

Таблица П.1.5.

Подпункты пункта меню Вид (View)

Подпункт меню	Комментарии
1	2
Менеджер Проекта (Project Manager)	Отображает и делает активным окно менеджера проектов
Менеджер Перевода (Translation Manager)	Отображает и делает активным окно менеджера перевода
Инспектор Объектов (Object Inspector)	Отображает и делает активным окно инспектора объектов
Вид Дерева Объектов (Object TreeView)	Отображает и делает активным окно, содержащее графическое изображение иерархии компонентов
Список To-Do (To-Do List)	Отображает и делает активным окно, содержащее список To-Do для текущего проекта
Палитра Привязки (Alignment Palette)	Отображает плавающую панель с инструментами управления привязкой элементов управления
Обозреватель (Browser)	Отображает и делает активным окно обозревателя проекта
Проводник Кода (Code Explorer)	Отображает и делает активным окно проводника кода
Список Компонентов (Component List)	Отображает и делает активным окно со списком доступных компонентов
Список Окон (Window List)	Отображает список открытых в среде окон
Доп. Инфо Сообщений (Additional Message Info)	Отображает окно с дополнительной информацией о сообщениях компилятора

Библиотека БГУИР

Окончание таблицы П.1.5.

1	2
Окна Отладки (Debug Windows)	Содержит дополнительные пункты меню для вызова средств отладки
Рабочие столы (Desktops)	Позволяет управлять, применять, сохранять и удалять настройки размещения окон среды программирования
Переключить Форму/Модуль (Toggle Form/Unit)	Поочередно переключает среду между редактированием формы и соответствующим ей модулем
Модули (Units)	Отображает список модулей проекта
Формы (Forms)	Отображает список форм проекта
Библиотека Типов (Type Library)	Отображает окно редактора библиотеки типов (используется при разработке ActiveX-компонентов)
Новое Окно Редактора (New Edit Window)	Создает новое окно редактора кода
Панели Инструментов (Toolbars)	Позволяет управлять видимостью панелей инструментов среды

Таблица П.1.6.

Подпункты пункта меню Проект (Project)

Подпункт меню	Комментарии
1	2
Добавить в Проект (Add to Project)	Позволяет добавить модули в проект
Удалить из Проекта (Remove from Project)	Позволяет удалить один или несколько модулей, включенных в проект
Импорт Библиотеки Типов (Import Type Library)	Позволяет импортировать в палитру компонентов ActiveX-компоненты, установленные в системе
Добавить в Хранилище (Add to Repository)	Позволяет добавить файлы проекта в репозиторий
Показать Исходник (View Source)	Отображает в редакторе кода главный модуль проекта
Языки (Languages)	Предоставляет инструменты для создания многоязычных приложений
Добавить Новый Проект (Add New Project)	Выводит диалоговое окно создания нового проекта
Добавить существующий Проект (Add Existing Project)	Позволяет добавить ранее созданный проект в менеджер проектов
Компилировать ИмяПроекта (Compile/Make ProjectName)	Выполняет перекомпиляцию изменившихся файлов текущего проекта
Компоновать ИмяПроекта (Build ProjectName)	Выполняет полную перекомпиляцию проекта

Библиотека БГУИР

Окончание таблицы П.1.6.

1	2
Проверка синтаксиса ИмяПроекта (Syntax Check ProjectName)	Выполняет проверку синтаксиса в пределах текущего проекта
Инфо про ИмяПроекта (Information for ProjectName)	Выводит информацию о текущем состоянии проекта и результатах его последней компиляции
Компилировать Все Проекты (Compile/Make All Projects)	Выполняет перекомпиляцию изменившихся файлов всех проектов
Компоновать Все Проекты (Build All Projects)	Выполняет полную перекомпиляцию всех проектов
Опции Web распространения (Web Deployment Options)	Предоставляет возможность настройки опций распространения готовых ActiveX-компонентов
web распространение (Web Deploy)	Подготавливает готовый ActiveX-компонент к распространению с учетом соответствующих настроек
Опции (Options)	Отображает окно управления настройками проекта

Таблица П.1.7.

Подпункты пункта меню Запуск (Run)

Подпункт меню	Комментарии
1	2
Запуск (Run)	Компилирует и запускает на выполнение текущий проект
Присоединить к Процессу (Attach to Process)	Позволяет подключиться к отладке запущенного в системе процесса
Параметры (Parameters)	Позволяет указать параметры командной строки для запуска
Регистрация ActiveX сервера (Register ActiveX Server)	Регистрирует в системе разрабатываемый ActiveX-компонент
Отмена регистрации ActiveX сервера (Unregister ActiveX Server)	Отменяет регистрацию в системе разрабатываемого ActiveX-компонента
Установить COM+ объекты (Install COM+ Objects)	Позволяет объектам приложения использоваться при помощи COM+
Пропуск блока (Step Over)	Используется для пошаговой отладки программ без входа в вызываемые процедуры
Трассировка (Trace Into)	Используется для пошаговой отладки программ с входом в вызываемые процедуры
Трассировка на след. строку (Trace To Next Source Line)	Выполняет трассировку кода, останавливаясь на строке, следующей за текущей строкой

Библиотека БГУИР

Окончание таблицы П.1.7.

1	2
Запуск до Курсора (Run To Cursor)	Запускает программу на выполнение с остановкой при достижении текущей строки
Запуск до возврата (Run Until Return)	Выполняет программу, останавливаясь на выходе из текущей процедуры
Показать Точку Выполнения (Show Execution Point)	Перемещает курсор в редакторе кода к текущему месту выполнения программы
Пауза программы (Program Pause)	Временно приостанавливает выполнение программы
Сброс программы (Program Reset)	Принудительно завершает выполнение отлаживаемой программы
Проверить (Inspect)	Позволяет просмотреть информацию о состоянии переменной, на имени которой установлен курсор, или некоторого выражения
Вычислить/Изменить (Evaluate/Modify)	Открывает диалоговое окно, позволяющее производить вычисление значений произвольных выражений и изменять значения переменных
Добавить Наблюдателя (Add Watch)	Позволяет наблюдать за значениями переменных во время отладки
Добавить СтопТочку (Add Breakpoint)	Позволяет добавить точку останова

Библиотека БГУИР

МОДУЛИ ПРОГРАММЫ

П.2.1. Текст модуля UI.pas

```
////////////////////////////////////////////////////////////
/
//Модуль UI.pas
//Реализация функций взаимодействия с пользователем
////////////////////////////////////////////////////////////
    unit UI;
    interface
    type
//Тип выводимой пользователю подсказки (см.проц. WriteHint())
    THintType=(htSplash,htInput,htError,htGreater,htLess,
htEqual,htFailed,htPlayAgain);
    procedure WriteHint(HintType:THintType;Param:Integer);
    function ReadGuess(GuessNumber:Integer):Integer;
    function ReadPlayAgain:Boolean;
    implementation
    uses
       Windows;
//Вспомогательная процедура ClearConsole()
//Очищает консоль и помещает курсор в верхний левый угол окна
    procedure ClearConsole;
    var
       hConsole     : THandle;
       ConsoleInfo  : TConsoleScreenBufferInfo;
       Coord        : TCoord;
       WrittenChars : LongWord;
    begin
       FillChar(ConsoleInfo,SizeOf(ConsoleInfo),0);
       FillChar(Coord, SizeOf(Coord),0);
       hConsole := GetStdHandle(STD_OUTPUT_HANDLE);
       GetConsoleScreenBufferInfo(hConsole,ConsoleInfo);
       FillConsoleOutputCharacter(hConsole,' ',
           ConsoleInfo.dwSize.X * ConsoleInfo.dwSize.Y,
           Coord, WrittenChars);
       SetConsoleCursorPosition(hConsole,Coord);
    end;
//Вспомогательная процедура ShowSplash()
//Отображает текст приветствия программы.
    procedure ShowSplash;
    begin
      ClearConsole;
writeln('==============================================
===');
      writeln('=         Игра         «Отгадай         число»
=');
      writeln('=
=');
```

Библиотека БГУИР

```
      writeln('= Цель игры: отгадать число от 1 до 100 с 10
попыток                                                    =');
writeln('==========================================================
===');
      writeln;
    end;
//Процедура WriteHint()
//Отображает   текстовую   подсказку,   заданную   параметром
HintType:
//htSplash  Текст приветствия программы
//htInput   Запрос на ввод числа-предположения
//htError      Сообщение  о  выходе  числа-предположения  за
диапазон //htGreater Сообщение "Искомое число больше"
//htLess    Сообщение "Искомое число меньше"
//htEqual   Сообщение "Вы отгадали!"
//htFailed  Сообщение "Вы не отгадали число XX с 10 попыток"
//htPlayAgain Запрос о повторении игры
//Параметр Param зависит от значения HintType:
//htSplash  Игнорируется
//htInput   Должен содержать номер текущей попытки
//htGreater Игнорируется
//htLess    Игнорируется
//htEqual   Содержит количество заработанных игроком очков
//htFailed  Содержит искомое число
//htPlayAgain Игнорируется
    procedure WriteHint(HintType: THintType; Param: Integer);
    begin
        case HintType of
            htSplash:
               ShowSplash;
            htInput:
               write('Попытка     №',Param,'.Введите     Ваше
предположение: ');
            htError:
               writeln('Ошибка! Число должно быть в диапазоне
от 1 до 100.');
            htGreater:
              writeln('Искомое число больше');
            htLess:
              writeln('Искомое число меньше');
            htEqual:
            begin
              writeln('Вы отгадали!');
              writeln('Количество очков: ', Param);
            end;
            htFailed:
              writeln('Вы не отгадали число ', Param, ' с 10
попыток');
            htPlayAgain:
              writeln('Хотите  сыграть  ещё  раз?  (Введите  N,
чтобы отказаться)');
        end;
    end;
```

Библиотека БГУИР

```
//Функция ReadGuess()
//Запрос на ввод числа от 1 до 100. При попытке ввести число,
//выходящее за диапазон,выводит подсказку,повторяет запрос.
    function ReadGuess(GuessNumber: Integer): Integer;
    begin
       Result := 0;
       while True do
       begin
           WriteHint(htInput, GuessNumber);
           readln(Result);
           if (Result >= 1) and (Result <= 100) then
             Break;
           WriteHint(htError, 0);
       end;
    end;

//Функция ReadPlayAgain
//Запрос на повторение игры. Возвращает True, если
//пользователь готов сыграть ещё раз, False - если отказался.
    function ReadPlayAgain: Boolean;
    var
      Answer : String;
    begin
       WriteHint(htPlayAgain, 0);
       readln(Answer);
       Result := (Answer <> 'N') and (Answer <> 'n');
    end;
    end.
```

П.2.2. Текст модуля EsConsole.pas

```
////////////////////////////////////////////////////
/
//Модуль EsConsole.pas
//Преобразование символов-кириллицы
////////////////////////////////////////////////////
/
    unit EsConsole;
    interface
    implementation
    uses
       Windows;
    type
       TTextRec = record
            Handle: Integer;
            Mode: Integer;
            BufSize: Cardinal;
            BufPos: Cardinal;
            BufEnd: Cardinal;
            BufPtr: PChar;
            OpenFunc: Pointer;
            InOutFunc: Pointer;
            FlushFunc: Pointer;
            CloseFunc: Pointer;
```

Библиотека БГУИР

```
            UserData: array[1..32] of Byte;
            name: array[0..259] of Char;
            Buffer: array[0..127] of Char;
        end;
    function ConOutFunc(var Text: TTextRec): Integer;
    var
        Dummy: Cardinal;
        SavePos: Integer;
    begin
       SavePos := Text.BufPos;
       if  SavePos > 0 then
       begin
          Text.BufPos := 0;
          CharToOemBuff(Text.BufPtr, Text.BufPtr, SavePos);
          if  WriteFile(Text.Handle,  Text.BufPtr^,  SavePos,
Dummy,nil) then
             Result := 0
          else
            Result := GetLastError;
       end
       else
         Result := 0;
    end;
    initialization
       Rewrite(Output);    // Проводим инициализацию файла
       TTextRec(Output).InOutFunc := @ConOutFunc;
       TTextRec(Output).FlushFunc := @ConOutFunc;
     end.
```

П.2.3. Текст модуля Points.pas

```
////////////////////////////////////////////////////////
/
//Модуль Points.pas
//Подсчет очков
////////////////////////////////////////////////////////
/
    unit Points;
    interface
    procedure InitPoints(var Points:Integer);
    procedure UpdatePoints(var Points: Integer; Goal, Guess:
Integer);
    implementation
//Процедура InitPoints()
//Устанавливает начальное количество очков пользователя.
    procedure InitPoints(var Points: Integer);
    begin
      Points:=1000;
    end;
//Процедура UpdatePoints()
//Изменяет количество очков пользователя с учётом значений
//искомого числа и числа-попытки.
    procedure UpdatePoints(var Points: Integer; Goal, Guess:
Integer);
```


Библиотека БГУИР

```
    begin
       Points:= Points-Abs(Goal-Guess);
    end;
  end.
```

П.2.4. Текст модуля Points.pas

```
////////////////////////////////////////////////////////////
/
//Модуль Points.pas
//Подсчет очков
////////////////////////////////////////////////////////////
/
    unit Points;
    interface
    procedure InitPoints(var Points: Integer);
    procedure UpdatePoints(var Points: Integer; Goal, Guess:
Integer);
    var
       InitialPoints : Integer;
    implementation
    procedure InitPoints(var Points: Integer);
    begin
        Points := InitialPoints;
        Dec(InitialPoints, 100);
        if   InitialPoints < 200 then
            InitialPoints := 200;
    end;
    procedure UpdatePoints(var Points: Integer; Goal, Guess:
Integer);
    begin
       Points := Points - 8 * Abs(Goal - Guess);
       if  Points < 0 then
           Points := 0;
    end;
    initialization
        InitialPoints := 1000;
    end.
```

Библиотека БГУИР

Св. план 2012, поз. 31

Учебное издание

Глухова Лилия Александровна
Фадеева Елена Павловна
Фадеева Елена Евгеньевна

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 4-х частях

Часть 4

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *Н. В. Гриневич, Е. С. Чайковская*
Корректор *А. В. Бас*
Компьютерная правка, оригинал-макет *А. А. Лысеня*

Подписано в печать 27.05.2013. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,6. Уч.-изд. л. 3,1. Тираж 100 экз. Заказ 276.

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

Библиотека БГУИР