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

Кафедра информационных технологий автоматизированных систем

А.В. Бушкевич, И.В. Лукьянова

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методическое пособие

для студентов специальности 40 02 01
«Вычислительные машины, системы и сети»
заочной формы обучения

В 2-х частях

Часть 1

Минск 2003

УДК 681.3.06 (075.8)
ББК 32.973-018 я 73
Б 90

Бушкевич А.В.

Б 90 Основы алгоритмизации и программирования: Метод. пособие для студ. спец. 40 02 01 «Вычислительные машины, системы и сети» заочной формы обучения: В 2 ч. Ч. 1. / А.В. Бушкевич, И.В. Лукьянова. – Мн.: БГУИР, 2003. – 30 с.: ил.

ISBN 985-444-499-6 (ч. 1).

В методическом пособии изложены значение и задачи изучаемого курса, требования к содержанию и оформлению контрольной работы, варианты контрольных заданий к контрольной работе, методика отладки программ в системе программирования Borland C++, а также приведен пример решения и оформления контрольного задания.

УДК 681.3.06 (075.8)
ББК 32.973-018я 73

ISBN 985-444-499-6 (ч. 1)
ISBN 985-444-500-3

Содержание

1. Значение и задачи курса
2. Методические рекомендации и варианты контрольных работ
3. Общие сведения о системе программирования Borland C++
4. Отладка программ в интегрированной среде программирования Borland C++
5. Пример решения контрольного задания
6. Варианты контрольных заданий

Литература

1. Значение и задачи курса

Основная цель курса «Основы алгоритмизации и программирования» заключается в освоении навыков алгоритмизации и программирования задач. Неотъемлемой чертой современного специалиста в области компьютерных технологий является умение разрабатывать алгоритмы и реализовывать их на компьютере, алгоритмически подходить к решению информационных задач, разбираться в терминологии программирования, представлять возможности современных языков разработки программного обеспечения. В прежнее время программирование было уделом математиков и системных специалистов. Сейчас приложения на всевозможных языках программирования необходимо разрабатывать специалистам различного уровня для самых разных областей: программы для Internet-технологий, макросы для документов Microsoft Office, несложные задачи по автоматизации офисных работ и т.д. В настоящее время существует большое количество разнообразных языков программирования, с помощью которых можно эффективно решать широкий круг задач. Но залогом успешной разработки программного обеспечения на любом языке программирования было и остается знание основных принципов алгоритмизации, понимание процесса работы программы, обработки компьютером машинных команд. Это является базисом для программиста любого профиля. Отдельное внимание на занятиях уделяется различным способам организации данных в программе, решению стандартных алгоритмических задач. Рассматриваемые вопросы являются основополагающими и для дисциплин «Конструирование программ и языки программирования», «Объектно-ориентированное программирование» и др., которые студенты будут изучать позже.

Итогом изучения курса должны стать:

– теоретические знания основ алгоритмизации задач и проектирования программ, создания программ на алгоритмических языках, основ алгоритмизации вычислительных процессов на ЭВМ;

– практические навыки программирования, отладки и выполнения на ЭВМ конкретных задач с использованием современных методов программирования.

В первой части методического пособия рассматриваются общие сведения об интегрированной среде программирования Borland C++, наиболее важные аспекты методики отладки программ в этой среде, а также приведены методические рекомендации и пример выполнения контрольного задания.

Темы, которые изучаются в первом семестре в рамках данного курса:

1. Основные характеристики языка Си. Алфавит языка. Идентификаторы. Базовые типы данных. Целый тип. Вещественный тип. Символьный тип, перечисления. Преобразование типов.
2. Операции языка Си. Приоритет операций. Операторы языка Си. Программирование разветвляющихся алгоритмов.
3. Программирование циклических структур алгоритмов. Операторы цикла.
4. Структурированные типы. Одномерные и многомерные массивы. Указатели. Операции над указателями.
5. Функции. Прототипы функций. Передача параметров в функции. Строковые данные. Локальные и глобальные переменные. Классы памяти. Внешние, статистические и регистровые переменные.

2. Методические рекомендации и варианты контрольных работ

Контрольная работа состоит из двух заданий:

1) реализация программы работы с матрицами;
2) реализация программы работы со строковыми типами данных.

Номер варианта заданий определяется по порядковому номеру фамилии в списке студентов группы. Например, номер фамилии студента в списке 23, значит, первое и второе задания будут с номером 23 (см. разд. 6 «Варианты контрольных заданий»).

Контрольная работа должна быть выполнена на листах формата А4, сброшюрованных в папку. Страницы в контрольной работе должны быть пронумерованы. Нумерация страниц в работе должна быть сквозная. На титульном листе приводится название кафедры, наименование дисциплины, факультета, номер курса, группы, фамилия, имя, отчество студента, номер варианта. Каждое задание должно содержать текст задачи, если необходимо пояснения к условию задачи, блок-схемы алгоритмов функций, разработанных в программе, листинг программы с комментариями и распечатку результатов работы программы (см. разд. 5).

3. Общие сведения о системе программирования Borland C++

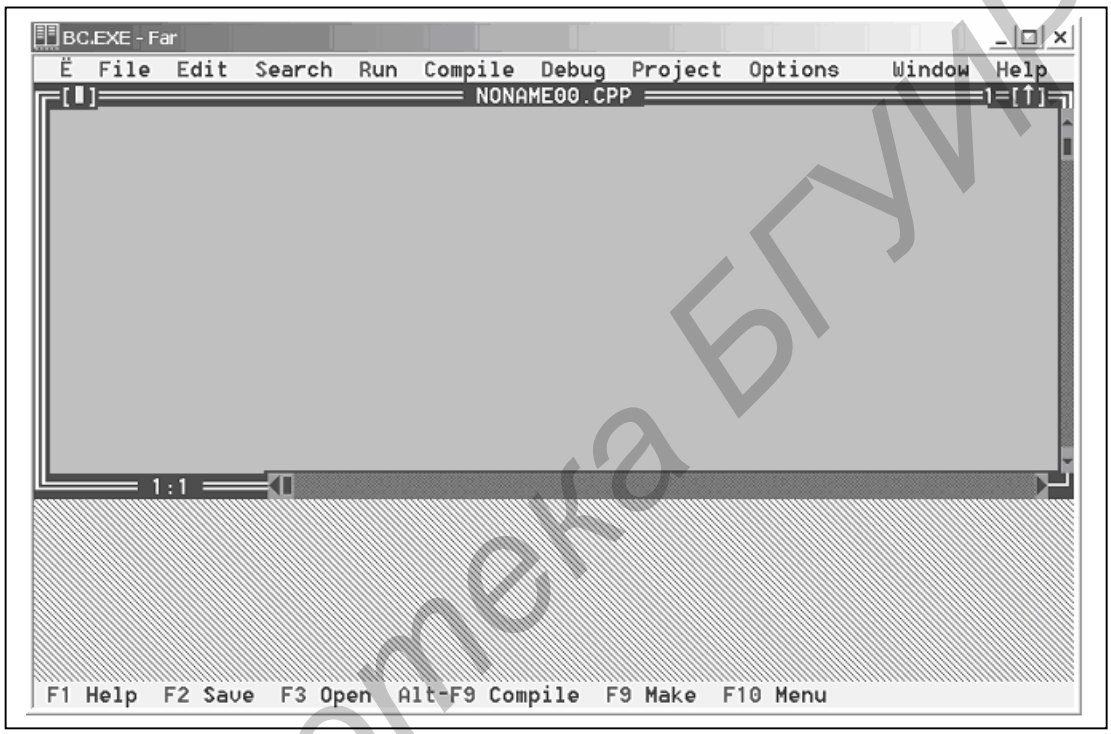
Система программирования на языке Си – Borland C++ включает:

1) интегрированную среду программирования (Integrated Development Environment – IDE);
2) компилятор исходного текста программы;
3) редактор связей (компоновщик);
4) библиотеки заголовочных файлов;
5) библиотеки функций;
6) программы-утилиты.

Интегрированная среда (IDE) – это программа, имеющая встроенный редактор текстов, подсистему работы с файлами, систему помощи, встроенный отладчик, подсистему управления компиляцией и редактированием связей, а

также компилятор и редактор связей. Таким образом, IDE дает возможность получить .EXE-файл, не используя другие программы. Borland C++ имеет два варианта интегрированной среды: BC.EXE для работы в реальном режиме; BCX.EXE для работы в защищенном режиме.

После запуска на исполнение файла IDE – BC.EXE на экране появится основное окно IDE (рисунок).



Данное окно поделено на три части. Верхняя строка – главное меню, рабочая область окна и нижняя строка экрана – строка состояния. Через главное меню можно обратиться к подсистемам интегрированной среды (подменю): системному меню (Ё), меню файловой системы (File), меню редактирования (Edit), меню поиска и замещения (Search), меню управления исполнением программ (Run), меню компиляции программ (Compile), меню управления проектом (Project), меню встроенного отладчика программ (Debug), меню опций (Options), меню управления окнами (Window) и меню помощи (Help). В строке состояния выделены назначения «горячих» клавиш, активных на данном этапе

работы. В рабочую область окна могут выводиться окна с пунктами меню, окна с редактируемым текстом программы, окна системы помощи и др.

Выбрать любую из команд главного меню можно одним из трех способов:

1) нажать клавишу F10 и с помощью клавиш со стрелками выбрать необходимую команду. Для выполнения команды нажать клавишу ENTER;
2) установить курсор манипулятора «мышь» на любое ключевое слово меню и нажать левую кнопку манипулятора;
3) использовать «горячие» клавиши. Одновременное нажатие клавиши Alt и «горячей» клавиши активизирует соответствующую команду.

Прежде чем начать работу в IDE, следует проверить и при необходимости выставить нужные опции. Настройка опций выполняется через меню Options. Меню Options включает команды Compiler, Transfer…, Make…, Linker…, Application…, Debugger…, Directories… . Как правило, при инсталляции среды программирования Borland C++ через стандартную программу инсталляции большинство необходимых опций установлено по умолчанию. Но Borland C++ можно установить, просто переписав необходимые каталоги с уже ранее установленного места. В этом случае особенно важно проверить выставленные опции. Прежде всего необходимо задать директории, используемые текстовым редактором «Output Directory», компилятором «Include Directories» и компоновщиком «Library Directories». Для этого используется команда Directories. В «Include Directories» задаются директории заголовочных файлов. В данном параметре разрешается использовать несколько директориев, разделенных знаком «;». В «Library Directories» задаются директории, содержащие объектный файл загрузчика (C0?.OBJ) и файлы библиотек функций (.LIB). В «Output Directory» задается директорий, в который помещаются файлы с расширениями .OBJ,

.EXE и .MAP. Если в данном параметре – пустая строка, значит, используется текущий директорий.

При выборе строки Compiler меню Options открывается ещё одно меню для настройки опций компилятора. Меню включает команды Code generation…, Entry/Exit Code…, C++ options…, Optimizations…, Source…, Messages…, Names… . Опция считается выбранной, если она помечена символом (•), и включенной, если она помечена символом [x]. Назначение каждой из опций описано в системе помощи интегрированной среды.

После настройки необходимых опций интегрированной среды Borland C++ можно приступать к написанию программы. Сначала надо ввести программу. Для этого необходимо войти в редактор, начав редактирование нового файла (выбирается Main Menu – File – New). Затем набирается текст программы. Прежде чем начинать компиляцию, следует сохранить файл под каким-нибудь именем (выбирается Main Menu – File – Save as…).

Не следует начинать компиляцию, компоновку или запускать программу без сохранения набранного текста программы! Запущенная на выполнение программа может привести к «повисанию» компьютера, и набранная программа будет потеряна.

Запуск программы на компиляцию выполняется либо через команду Compile to OBJ меню Compile, либо нажатием «горячей» клавиши Alt-F9. Команда Make EXE file меню Compile также запускает программу на компиляцию и при отсутствии синтаксических ошибок автоматически запускает компоновщик для получения .EXE-файла. Этой команде соответствует «горячая» клавиша F9. Еще одна возможность для запуска программы на компиляцию – команда Run меню Run («горячая» клавиша Ctrl-F9). После успешной компиляции и компоновки Run запускает полученный .EXE-файл на выполнение.

Все сообщения об ошибках и предупреждения IDE помещает в окно по имени Message. Это окно активно после завершения компиляции. Если в программе обнаружены ошибки или предупреждения, включаются средства трас-

сировки ошибки, которые связывают строки текста программы в окне редактора со строками окна Messages. Перемещение высвечивания клавишами со стрелками в окне Message синхронно сопровождается высвечиванием соответствующих ошибочных строк в тексте программы. При нажатии клавиши ENTER активизируется окно редактора, и курсор устанавливается на ошибочную строку. Нажатие клавиши F6 вновь делает активным окно Message.

После исправления всех синтаксических ошибок программа может быть запущена на выполнение (Ctrl-F9). Очень часто скомпилированная без синтаксических ошибок программа логически работает неверно. Причина неправильной работы программы не всегда очевидна. Для ее установления необходимо пользоваться отладочными средствами интегрированной среды Borland C++, которые рассматриваются в следующем разделе.

4. Отладка программ в интегрированной среде программирования Borland C++

Для создания программы без смысловых ошибок обычно пользуются отладчиками. Существуют разнообразные отладчики, позволяющие программистам отыскать любые смысловые ошибки в своих программах. В интегрированной среде Borland C++ имеется встроенный в среду отладчик. С его помощью можно проследить ход исполнения программы, текущие значения переменных и внутренних регистров процессора и при необходимости установить нужные значения.

Для того чтобы отладка программы в IDE стала возможной, необходимо перед выполнением компиляции и компоновки программы включить опцию Source Debugging, расположенную в меню опций отладчика (Main Menu-Options-Debugger).

Встроенный отладчик IDE позволяет:

1) выполнить программу по шагам, строка исходного текста за строкой;
2) выполнить программу до заданной строки, так называемой точки останова;
3) проследить изменение заданных переменных программы и при необходимости установить для них новые значения.

Команда Trace into из меню Run («горячая» клавиша F7) запускает программу на отладку. Интегрированная среда высвечивает строку программы, содержащую точку входа main(). После этого нажатием клавиши F7 вызывают выполнение кода, соответствующего одной строке текста программы на Си. Если в строке записана ссылка на функцию, начинается трассировка по тексту функции. Библиотечные функции выполняются без трассировки за одно нажатие клавиши F7. При необходимости выполнения строки Си-текста за один шаг (без трассировки по тексту функции) используется клавиша F8 (команда меню Run – Step over).

Для ускорения процесса отладки используется команда Go to cursor меню Run («горячая» клавиша F4). Программа выполняется до строки, в которой в данный момент располагается текстовый курсор. Например, если программа попадает в цикл и нет необходимости выполнять по шагам все его итерации, необходимо переместить курсор на следующую после тела цикла строку текста и нажать клавишу F4. Точно так же можно пропустить ту часть отлаживаемой программы, детальный анализ которой не требуется.

Другая возможность ускоренного выполнения отлаживаемой программы – использование точек останова. Включение точки выполняет команда Toggle breakpoint меню Debug («горячая» клавиша Ctrl-F8). Точка останова помещается в той строке программы, где располагается текстовый курсор. Повторное выполнение команды Toggle breakpoint удаляет точку останова. Строка Си-текста с точкой останова высвечивается на экране другим цветом.

Когда исполняющаяся программа достигает точки останова, IDE проверяет условие, которое может быть задано для нее. Если условие не задано, эта точка считается точкой безусловного останова. Здесь выполнение программы всегда останавливается и может быть продолжено либо по шагам (клавиша F7), либо до курсора (клавиша F4), либо до следующей точки останова (клавиша Ctrl-F9).

Если же с точкой останова связано какое-то условие, она называется условной. Выполнение программы останавливается здесь, если только в этот момент выполненное условие истинно. Задание условий выполняется через окно диалога, открываемое при выполнении команды Breakpoints… меню Debug.

При отладке программы важно иметь возможность наблюдать за изменением значений переменных в ходе выполнения программы. Интегрированная среда Borland C++ использует для этого специальное окно с именем Watch. Оно появляется сразу после нажатия клавиши F7. Задание имен переменных или выражений, называемых далее точками наблюдения (Watches), выполняется командой Watches меню Debug. Ее выполнение открывает подменю, содержащее команды управления точками наблюдения: Add watch…, Delete watch, Edit watch…, Remove all watches.

Команда Add Watch («горячая» клавиша Ctrl-F7 в случае, когда активно окно текстового редактора) добавляет в окно Watch новую точку наблюдения. При выполнении данной команды открывается окно диалога, с использованием которого задается любое допустимое выражение языка Си, в том числе имя переменной, и, если необходимо, через запятую специфицируется формат представления переменной. По умолчанию выражением является слово в текущей позиции курсора активного окна редактирования, а формат выбирается по типу переменной.

После того как завершен ввод выражения, нажимается клавиша ENTER и в окне Watch появляется новая строка, показывающая текущее значение вычисленного выражения.

Окно Watch, когда оно активно, позволяет выполнить добавление, удаление и редактирование точки наблюдения без использования команды подменю Watch. Высвеченная в окне Watch строка соответствует текущей точке наблюдения. Ее редактирование осуществляется простым нажатием клавиши Enter. Для удаления текущей точки наблюдения нажимается клавиша Del. Добавление новой точки наблюдения происходит при нажатии клавиши Ins.

Кроме описанного выше во встроенном отладчике IDE имеются также и другие возможности. Прежде всего, это возможность просмотра и изменения значений переменных в процессе отладки программы. Команда Evaluate/modify… меню Debug («горячая» клавиша Ctrl-F4) открывает окно диалога, содержащее три поля: поле Expression для задания выражения, поле Result, где отображается результат выполнения выражения, и поле New Value для задания, если это возможно, нового значения выражения.

В поле ввода Expression задается любое выражение языка Си, включая константы, имена переменных и символы, специфицирующие формат представления переменной. В этом поле помещается имя переменной, на которую указывал курсор в текстовом окне в момент нажатия клавиши Ctrl-F4, или выражение текущей точки наблюдения, если активным является окно Watch. Когда завершен ввод в поле Expression, нажатие клавиши ENTER позволяет вычислить выражение. Результат отображается в поле Result. Команда Evaluate/modify… часто используется программистами вместо калькулятора для вычислений. Например, задав выражение (234+0x1ac)/0xf4, можно получить результат сложения и деления констант в шестнадцатеричной системе счисления.

Команда Inspect меню Debug позволяет исследовать и модифицировать ячейки памяти, отведенные под переменные. Окно Inspect может быть открыто нажатием «горячей» клавиши Alt-F4.

В ходе отладки программы можно сразу вносить изменения в ее текст. После каждого изменения отладчик запрашивает подтверждение на продолжение сеанса отладки или предлагает повторную компиляцию программы. Если

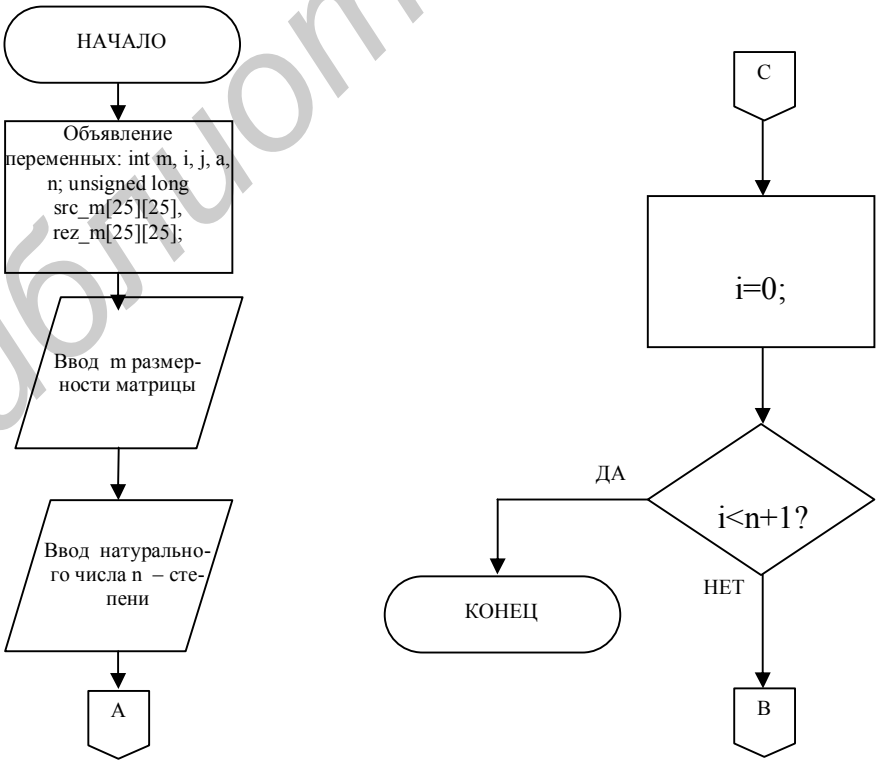
необходимо начать выполнение программы сначала, не дожидаясь ее завершения, используется клавиша Ctrl-F2.

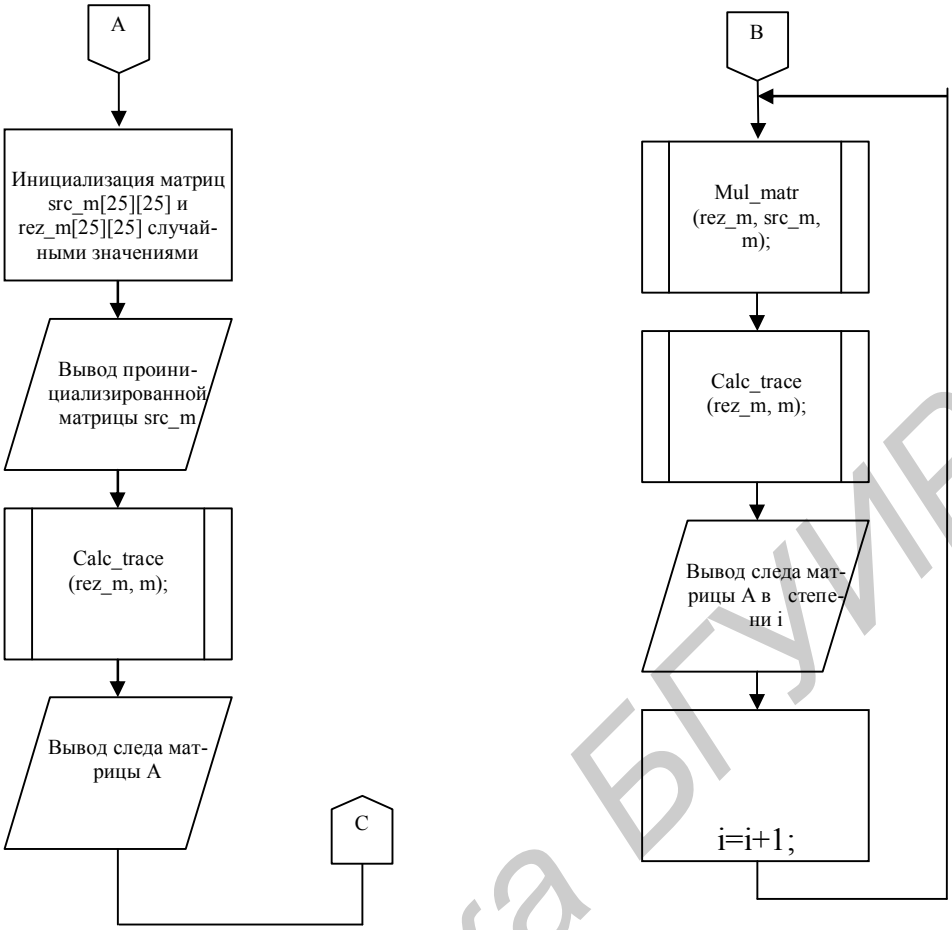
5. Пример решения контрольного задания

Задание. Даны квадратная матрица порядка m и натуральное число n. Вычислить следы матрицы A, A^2, …, A^n.

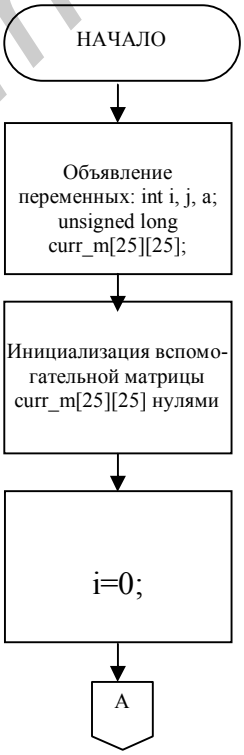
Метод вычисления. Следом матрицы называется сумма элементов ее главной диагонали. След матрицы в степени n определяется путем умножения матрицы на саму себя n раз и вычисления следа от полученной матрицы. Программа состоит из функции main(), функции вычисления следа матрицы calc_trace() и функции умножения двух матриц порядка m mul_matr(). Умножение двух матриц порядка m производится по формуле $C_{ij} = A_{i1} \cdot B_{1j} + A_{i2} \cdot B_{2j} + \ldots + A_{im} \cdot B_{mj}$.

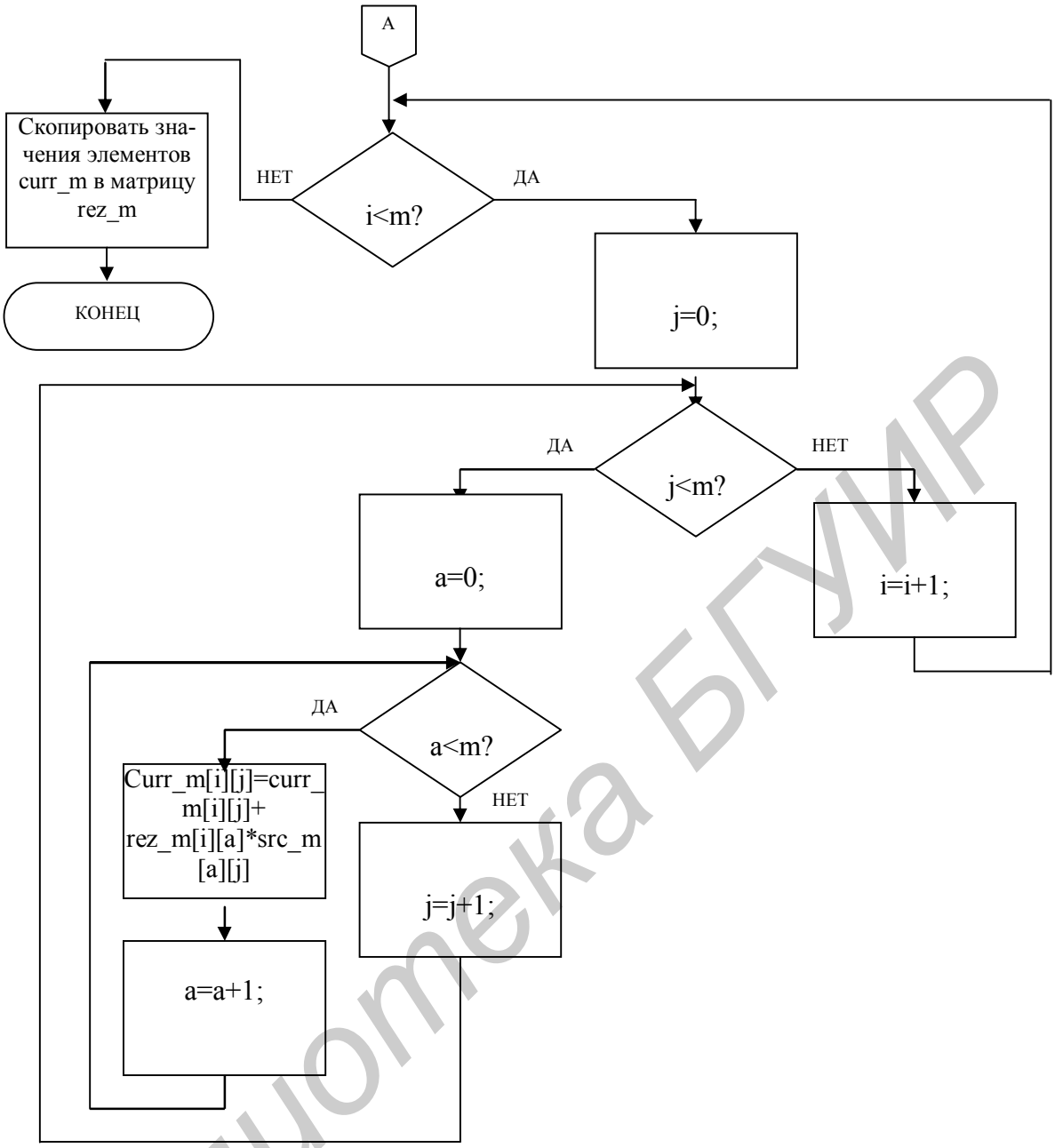
Блок-схема алгоритма функции main:



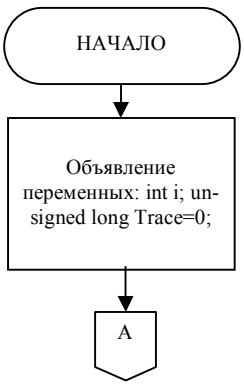


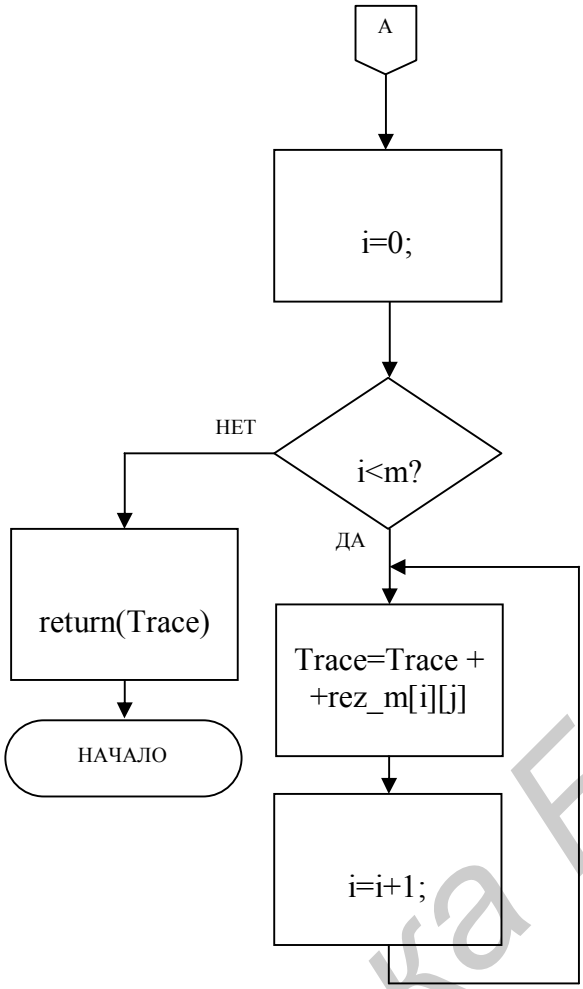
Блок-схема алгоритма функции mul_matr:





Блок-схема алгоритма функции calc_trace:





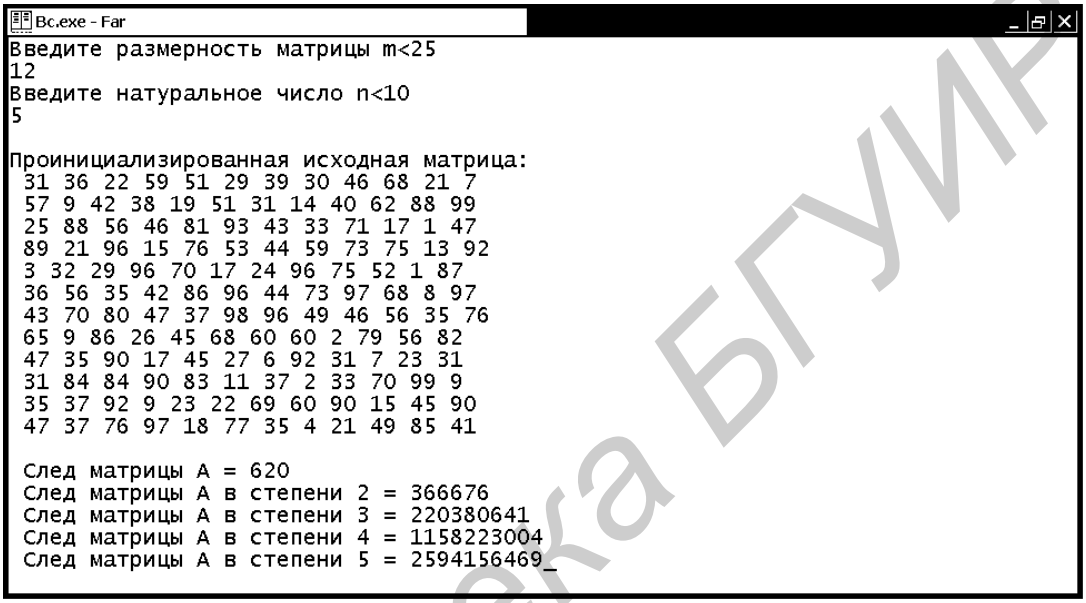
Листинг программы:

```
#include <stdio.h>
#include <stdlib.h> //srand(), rand()
#include <bios.h>   //bioskey
#include <conio.h>  //clrscr()

//Умножение двух матриц
void mul_matr(unsigned long rez[25][25], unsigned long src[25][25], int);
//Вычисление следа матрицы
unsigned long calc_trace(unsigned long rez[25][25],int);

void main(void)
{
 int m, i, j, a,n;
 unsigned long src_m[25][25], rez_m[25][25];
 clrscr();
 puts("Введите размерность матрицы m<25");
 scanf("%d",&m);
 puts("Введите натуральное число n<10");
 scanf("%d",&n);
```

```
 srand(n);
 for (i=0; i<m; i++)
  for (j=0; j<m; j++)
   {
    src_m[i][j]=rand() % 100;
    rez_m[i][j]=src_m[i][j];
   }
   printf("\nПроинициализированная исходная матрица: ");
 for (i=0; i<m; i++)
  {
   printf("\n");
   for (j=0; j<m; j++)
    {
     printf(" %d",src_m[i][j]);
    }
  }
 printf("\n\n След матрицы A = %lu",calc_trace(rez_m,m));
 for (i=2; i<n+1; i++)
  {
   mul_matr(rez_m,src_m,m);
   printf("\n След матрицы A в степени %d = %lu",i,calc_trace(rez_m,m));
  }
  bioskey(0);
}

void mul_matr(unsigned long rez_m[25][25], unsigned long src_m[25][25], int m)
{
 int i, j, a;
 unsigned long curr_m[25][25];

 for (i=0; i<m; i++)
  for (j=0; j<m; j++)
   {
    curr_m[i][j]=0;
   }

 for (i=0; i<m; i++)
  for (j=0; j<m; j++)
   for (a=0; a<m; a++)
    curr_m[i][j]=curr_m[i][j]+rez_m[i][a]*src_m[a][j];

 for (i=0; i<m; i++)
   for (j=0; j<m; j++)
    {
     rez_m[i][j]=curr_m[i][j];
    }
}

unsigned long calc_trace(unsigned long rez_m[25][25], int m)
{
 int i;
```

```
 unsigned long Trace=0;

for (i=0; i<m; i++)
  {
   Trace=Trace+rez_m[i][i];
  }
 return(Trace);
}
```

Результат работы программы:



6. Варианты контрольных заданий

Задачи с матрицами.

1. Массив длиной N в случайном порядке заполнен целыми числами из диапазона от 0 до N. Каждое число встречается в массиве не более одного раза. Найти отсутствующее число (дырку). Использование дополнительной памяти, пропорциональной длине массива, не допускается.
2. Сегментом называется непрерывная последовательность элементов массива. Массив состоит из двух сегментов неодинаковой длины. Не используя дополнительную память, пропорциональную N, поменять сегменты местами.

3. Массив длиной N заполнен в случайном порядке числами из диапазона от 1 до $k < N$. Не используя других массивов, подсчитать количество различных чисел. Число действий должно быть порядка N+k.
4. Даны два упорядоченных массива A и B (необязательно одинаковой длины). В каждом из массивов могут быть совпадающие элементы. Не используя дополнительной памяти, найти количество совпадающих значений элементов A и B (т.е. количество t, для которых t=A[i]=B[j]).
5. В массиве длиной N в случайном порядке находятся элементы «к» (красный), «б» (белый) и «с» (синий). Переставить их в порядке следования цветов голландского национального флага, чтобы вначале были все красные, затем белые и после них — синие. Допустимые операции — проверка цвета и обмен местами значений двух элементов. Иными словами, надо запрограммировать примитивного робота, который даже считать не умеет.
6. Сегментом называется непрерывная последовательность элементов массива. В неупорядоченном массиве целых чисел (как положительных, так и отрицательных) найти сегмент с максимальной суммой элементов.
7. Имеется массив целых чисел размерностью $M \times N$, $M \leq N$. Среди квадратов со стороной $k < M$ найти такой, сумма элементов которого максимальна.
8. Квадратный участок земли размером $N \times N$ метров ($N \leq 200$) разбит на клетки со стороной 1 метр. Клетка считается занятой, если на ней растет дерево. Деревья рубить нельзя. Найти для строительства дома квадрат максимальной площади, свободный от деревьев.
9. Элементами массива a[1..n] являются неубывающие массивы [1..m] целых чисел (a: array [1..n] of array [1..m] of integer; a[1][1] <= ... <= a[1][m], ..., a[n][1] <= ... <= a[n][m]). Известно, что существует число, входящее во все массивы a[i] (существует такое x, что для всякого i из [1..n] найдётся j из [1..m], для которого a[i][j]=x). Найти одно из таких чисел x.

10. Задан массив чисел A[1:N,1:M],упорядоченный по возрастанию по строкам и столбцам, т.е. A[I, 1] < A[I, 2] < ... < A[I, M] (при всех I), A[1, J] < A[2, J] < ... < A[N, J] (при всех J). Найти элемент массива, равный заданному числу X и отпечатать его индексы (I,J). Напечатать слово «НЕТ», если такого элемента не окажется. X можно сравнить не более, чем с M+N элементами массива.
11. Написать программу, которая в двухмерном массиве A(N, M) целых чисел, таком, что для всех I от 1 до N, J от 1 до M–1 выполняется A(I, J) > A(I, J+1) и для всех I от 1 до N–1 выполняется A(I, M) > A(I+1, M), находит все элементы A(I, J), равные J+I, или устанавливает, что таких элементов нет.
12. Написать программу, которая в двухмерном массиве A(N, M) целых чисел, таком, что для всех I от 1 до N, J от 1 до M–1 выполняется A(I, J) < A(I, J+1) и для всех I от 1 до N–1 выполняется A(I, M) < A(I+1, M), находит все элементы A(I, J), равные J+I, или устанавливает, что таких элементов нет.
13. Задана прямоугольная таблица A[1:N, 1:N], элементы которой равны 0 или 1, причем A[i, i] = 0 для любого i. Необходимо найти, если они есть, такие строку i_0 и столбец j_0, чтобы в столбце j_0 были все 0, а в строке i_0 – все 1 (кроме элемента A[i_0, i_0], равного 0).
14. На плоскости задается выпуклый N-угольник целочисленными координатами своих вершин в порядке обхода по контуру. Вводятся координаты точки (X, Y). Определить: а) является ли точка вершиной N-угольника; б) принадлежит ли она N-угольнику.
15. На прямой своими концами заданы N отрезков и точка X. Определить, принадлежит ли точка межотрезочному интервалу. Если да, то указать концевые точки этого интервала. Если нет, то найти: а) какому количеству отрезков принадлежит точка; б) каким именно отрезкам принадлежит точка.

16. На прямой своими концами заданы N отрезков. Найти точку, принадлежащую максимальному числу отрезков.
17. Вводится последовательность из n натуральных чисел. Необходимо определить наименьшее натуральное число, отсутствующее в последовательности.
18. Заданы массивы A[1..n] и B[1..n]. Необходимо найти такие индексы i_0 и j_0, что $a_{i0} + b_{j0} = \max (a_i + b_j)$, где $1 \le i \le n$, $1 \le j \le m$, $i < j$.
19. Мажорирующим элементом в массиве A[1..N] будем называть элемент, встречающийся в массиве более N/2 раз. Легко заметить, что в массиве может быть не более одного мажорирующего элемента. Например, массив 3, 3, 4, 2, 4, 4, 2, 4, 4 имеет мажорирующий элемент 4, тогда как в массиве 3, 3, 4, 2, 4, 4, 2, 4 мажорирующего элемента нет. Необходимо определить, есть ли в массиве мажорирующий элемент, и если есть, то какой.
20. Имеется N камней весом A_1, A_2, ..., A_N. Необходимо разбить их на две кучи таким образом, чтобы веса куч различались не более чем в 2 раза. Если этого сделать нельзя, то указать это.
21. Ввести две целочисленные таблицы A [1:10] и B[1:15]. Разработать алгоритм и написать программу, которая проверяет, являются ли эти таблицы похожими. Две таблицы называются похожими, если совпадают множества чисел, встречающихся в этих таблицах.
22. Имеются числа $A_1,A_2,...,A_N$ и $B_1,B_2,...,B_N$. Составить из них N пар (A_i, B_j) таким образом, чтобы сумма произведений пар была максимальна. Каждое A_i и B_j в парах встречаются ровно по одному разу.
23. Имеются числа A_1, A_2, ..., A_N и B_1, B_2, ..., B_N. Составить из них N пар (A_i, B_j) таким образом, чтобы сумма произведений пар была минимальна. Каждое A_i и B_j в парах встречаются ровно по одному разу.
24. В музее регистрируется в течение дня время прихода и ухода каждого посетителя. Таким образом, за день получены N пар значений, где пер-

вое значение в паре показывает время прихода посетителя и второе значение – время его ухода. Найти промежуток времени, в течение которого в музее одновременно находилось максимальное число посетителей.

25. Даны целые M и N и вектор действительных чисел X[1..N]. Найти целое число i ($1 \le i \le N–M$), для которого сумма x[i] + ... + x[i + M] ближе всего к нулю.
26. Есть два отсортированных в порядке неубывания массива A[1, N] и B[1, M]. Получить отсортированный по неубыванию массив C[1, N+M], состоящий из элементов массивов A и B («слить» вместе массивы A и B).
27. Дан массив X[1..N]. Необходимо циклически сдвинуть его на k элементов вправо (т.е. элемент X[i] после сдвига должен стоять на месте X[i + k]; тут мы считаем, что за X[N] следует X[1]). Разрешается использовать только несколько дополнительных слов памяти (Дополнительного массива создавать нельзя!).
28. Сколько палиндромов длиной n можно образовать, используя 26 букв? (Палиндром – выражение, которое читается одинаково как слева направо, так и наоборот).
29. Получить все счастливые номера от 0 до 999999. Номер является счастливым, если сумма первых трех цифр равна сумме трех его последних цифр. Если в числе меньше шести цифр, то недостающие начальные цифры считаются нулями.
30. Составить программу, которая печатает значение элементов матрицы, образованной из исходной матрицы, если вычеркнуть все строки и столбцы, содержащие хотя бы один нулевой элемент.

Задачи со строками:

1. Задать словарь. Найти в нем все анаграммы (слова, составленные из одних и тех же букв). Стандартных функций работы со строками не использовать.

2. Ввести массив строк. Каждую строку обработать так, чтобы все слова, состоящие только из цифр, были удалены, их сумма стояла последним словом. Дополнительных строк или массивов не создавать, стандартные функции работы со строками не использовать.
3. Ввести матрицу. Элементы матрицы – строки. Подсчитать сумму кодов символов каждого слова и, если сумма оказалась четной, развернуть зеркально это слово в строке. Полученные матрицы вывести на экран. Стандартных функций работы со строками не использовать.
4. Обработать массив строк (длина каждой строки не более 100 символов). Если в строке встречена последовательность одинаковых символов, заменить их кодом 255, за которым следует код этого символа и количество одинаковых символов. Стандартных функций работы со строками не использовать.
5. Написать программу, которая, анализируя массив, состоящий из строк, выводит на экран количество слов женского, мужского и среднего рода. Стандартных функций работы со строками не использовать.
6. Имеется массив строк. На том же месте, не заводя других массивов, записать слова в обратном порядке, рассматривая все строки, т.е. первое слово первой строки станет последним словом последней строки (если объединить строки, будет гораздо проще. Нужно учитывать, что в строке может не хватить места для очередного слова). Стандартных функций работы со строками не использовать.
7. В массиве строк найти среднее значение длины строки. Строки, длина которых больше среднего, – обрезать, меньше – добавить пробелы. Полученный массив вывести на экран. Стандартных функций работы со строками не использовать.
8. Перемещая указатель, рассортировать введенный с клавиатуры массив строк в алфавитном порядке. Ввод, сортировку и вывод массива выпол-

нить в отдельных функциях. Стандартные функции работы со строками не использовать.

9. Реализовать функции работы со строками (объединение на старом, на новом месте; сравнения; запись строк в обратном порядке – реверс строк). Стандартных функций работы со строками не использовать.
10. Ввести строки и за один просмотр в функции определять частоту встречаемости в строке цифры и буквы латинского алфавита. Стандартных функций работы со строками не использовать.
11. Рассортировать массив строк в алфавитном порядке, перемещая строки. Стандартных функций работы со строками не использовать.
12. Написать функции (копирование, сравнение, объединение) для работы со строками, используя указатели. Стандартных функций работы со строками не использовать.
13. Используя указатели, написать функции перевода целых чисел в строку и строки в число; функции перевода вещественных чисел в строку, перевода строки в вещественное число. Стандартных функций работы со строками не использовать.
14. Ввести массив строк. В функции для каждой строки проверить, является она симметричной или нет. (Симметричной считается строка, которая одинаково читается слева направо и справа налево). Вывести на экран саму строку и результат её обработки. Стандартных функций работы со строками не использовать.
15. Ввести массив строк. В функции для каждой строки проверить, является она палиндромом (симметричной с точностью до пробелов) или нет. Например, палиндромами являются цепочки: «АРГЕНТИНА МАНИТ НЕГРА» и «А РОЗА УПАЛА НА ЛАПУ АЗОРА». Вывести на экран саму строку и результат её обработки. Стандартных функций работы со строками не использовать.

16. Ввести строку и вычислить произведение входящих в эту строку целых чисел (без учета их знаков). Например, для строки «kjjjkkj2.5jkjn,,,hfd45jgfvjlkfdii10,2hfhg» произведение 2·5·45·10·2=9000. Результат вывести на экран. Стандартных функций работы со строками не использовать.
17. Ввести строку и вычислить сумму входящих в неё цифр, причем знак очередного слагаемого должен быть противоположным знаку предыдущего слагаемого. Например: для строки «asdd1vnb24vnf63vbn,-5h-2kk» сумма S=1–2+4–6+3–5+2= –3. Результат вывести на экран. Стандартных функций работы со строками не использовать.
18. Ввести строку и определить наибольшее записанное в этой строке целое число (без учета знака числа). Например, для строки «sdfvgsd1.9fdmjgvb-15.25dnj05» наибольшее целое число 15. Результат вывести на экран. Стандартных функций работы со строками не использовать.
19. Ввести строку и определить все входящие в неё символы. Например: строка «abccbbbabba» состоит из символов «a», «b» и «c». Результат вывести на экран. Стандартных функций работы со строками не использовать.
20. Ввести строки символов. Среди литер этого текста особую роль играет знак #, появление которого означает отмену предыдущей литеры текста; k знаков # отменяют k предыдущих литер (если такие есть). Преобразовать строку с учетом роли знака #. Например, строка «VR#Y##HELO#LO» должна быть напечатана в виде: «HELLO». Результирующую строку вывести на экран. Стандартных функций работы со строками не использовать.
21. Ввести строки символов. Определить, какой символ встречается в этой строке подряд наибольшее число раз. В ответе указать символ, образующий самую длинную последовательность, длину последовательности и номер символа, с которого она начинается. Например, в строке «asadddbbbbababaaaaaahhgg» символ a образует последовательность дли-

ной в 6 символов, начиная с символа с номером 15. Стандартных функций работы со строками не использовать.

22. Ввести строки, состоящие из строчных букв и пробелов, в функции определять слово наибольшей длины, которое начинается и заканчивается на одну и ту же букву. Например: строка – «револьвер системы наган», слово – «револьвер». Стандартных функций работы со строками не использовать.
23. Ввести строки, в которых имеются круглые, фигурные и квадратные скобки. Проверить, правильно ли они расставлены. Результат вывести на экран. Стандартных функций работы со строками не использовать.
24. Ввести две строки. В функции определить, являются ли они анаграммами (т.е. одна строка получена из другой перестановкой букв). Например: строки БУК и КУБ или СОЛЬ и ЛОСЬ являются анаграммами. Стандартных функций работы со строками не использовать.
25. Ввести строку символов, содержащую два или более слов, разделенных пробелами. Написать функцию, меняющую местами все четные и нечетные слова в строке, предполагая, что за один раз можно менять местами не более двух символов. Результат вывести на экран. Стандартных функций работы со строками не использовать.
26. Слово-донор – это слово, из которого составляются другие слова. Например, из слова МОНИТОР можно получить МОТОР, РОТ, ТИР и др. Вхождение каждой буквы в новое слово допускается не более того числа раз, с каким она входит в слово-донор. Ввести строку символов, содержащую несколько слов, разделенных пробелами. Известно, что первое слово в этой строке является донором. Выбрать из оставшихся слов в строке те, которые можно получить из заданного слова-донора. Стандартных функций работы со строками не использовать.

27. Даны две строки: S1 и S2. В функции определить, сколько символов этих строк, стоящих на местах с одинаковым номером, совпадает? Индексов не использовать. Результат вывести на экран.
28. Ввести строку символов и зашифровать её по следующему принципу: заменить каждый символ на следующий по порядку символ таблицы ASCII. Индексов не использовать. Результат вывести на экран.
29. Ввести строку символов и в функции преобразовать её, оставив в ней только арабские цифры с сохранением порядка их следования. Индексов и стандартных функций работы со строками не использовать. Результат вывести на экран.
30. Ввести строку символов и в функции преобразовать её, вычеркнув в данной строке каждую *k*-ю букву. Индексов и стандартных функций работы со строками не использовать. Дополнительных массивов не создавать. Результат вывести на экран.

Литература

1. Основная

1.1. Касаткин А.И., Вальвачев А.Н. Профессиональное программирование на языке Си: От Turbo C к Borland C++: Справ. пособие / Под общ. ред. А.И. Касаткина. – Мн.: Высш. шк., 1992.

1.2. Касаткин А.И. Профессиональное программирование на языке Си. Управление ресурсами: Справ. пособие. – Мн.: Высш. шк., 1992.

1.3. Керниган Б., Ритчи Д. Язык программирования Си. – М.: Финансы и статистика, 1992.

1.4. Подбельский В.В., Фомин С.С. Программирование на языке Си. Учеб. пособие. – М.: Финансы и статистика, 2000.

1.5. Скляров В.А. Программирование на языках Си и Си++. – М.: Высш. шк., 1996.

2. Дополнительная

2.1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979.

2.2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985.

2.3. Демидович Е.М. Учеб. пособие по курсу «Основы алгоритмизации и программирования» для студ. спец. «Вычислительные машины, системы и сети». – Мн.: БГУИР, 1999.

2.4. Демидович Е.М. и др. Рабочая программа, методические указания и контрольные задания по курсу «Основы алгоритмизации и программирования» для студентов специальности «Вычислительные машины, системы и сети» заочной формы обучения. – Мн.: БГУИР, 2000.

2.5. Демидович Е.М. Основы алгоритмизации и программирования. Язык СИ: Пособие для студ. БГУИР. – Мн.: Бестпринт, 2001.

2.6. Касаткин А.И. Профессиональное программирование на языке Си: Системное программирование. – Мн.: Высш. шк., 1993.s

Св. план 2003, поз. 42

Учебное издание

Бушкевич Алексей Владимирович
Лукьянова Ирина Викторовна

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методическое пособие

для студентов специальности 40 02 01
«Вычислительные машины, системы и сети»
заочной формы обучения

В 2-х частях

Часть 1

Редактор Т. Н. Крюкова
Корректор Е. Н. Батурчик

Подписано в печать 18.04.2003. Формат 60×84 1/16. Бумага офсетная.
Печать ризографическая. Гарнитура «Таймс». Усл. печ. л. 1,98.
Уч.-изд. л. 1,5. Тираж 75 экз. Заказ 26.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП №156 от 30.12.2002
Лицензия ЛВ №509 от 03.08.2001.
220013, Минск, П. Бровки, 6