СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

УЧЕБНОЕ ПОСОБИЕ

М. С. Эпштейн

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ С

ИНФОРМАТИКА
И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ACADEMIA

СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

М.С. ЭПШТЕЙН

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ С

Допущено
Министерством образования Российской Федерации
в качестве учебного пособия для студентов образовательных учреждений
среднего профессионального образования

Москва
Издательский центр «Академия»
2007

УДК 681.3(075.32)
ББК 32.81я723
Э736

Рецензенты:
доц. Санкт-Петербургского государственного политехнического университета, канд. физ-мат. наук *Ю. П. Ильин;*
преподаватель Московского Государственного колледжа информационных технологий *Н. В. Валиулина*

Эпштейн М. С.

Э736 Практикум по программированию на языке С : учеб. пособие для студ. сред. проф. образования / М.С.Эпштейн. — М. : Издательский центр «Академия», 2007. — 128 с. ISBN 978-5-7695-3626-7

Приведены девять лабораторных работ. Содержатся необходимые для их выполнения сведения по соответствующим разделам программирования, набор индивидуальных вариантов заданий, пример программы решения аналогичной задачи и требования к знаниям, умениям и навыкам, которыми студенты должны овладеть в результате выполнения работ. Включены разделы, содержащие информацию о типичных ошибках.

Для студентов средних профессиональных учебных заведений.

УДК 681.3(075.32)
ББК32.81я723

Оригинал-макет данного издания является собственностью Издательского центра «Академия», и его воспроизведение любым способом без согласия правообладателя запрещается

© Эпштейн М.С., 2007
© Образовательно-издательский центр «Академия», 2007
© Оформление. Издательский центр «Академия», 2007

ISBN 978-5-7695-3626-7

ПРЕДИСЛОВИЕ

В настоящее время язык С является одним из наиболее распространенных и востребованных языков программирования. Этот язык представляет собой фундамент, на котором строится современное программирование, и поэтому является обязательным элементом подготовки современных программистов.

Руководство ориентировано на курс практических (лабораторных) работ продолжительностью примерно 28...30 академических часов.

Работы содержат задачи, посвященные различным аспектам изучаемой темы, что позволяет за счет исключения одной из предлагаемых задач согласовывать продолжительность курса с возможностями группы.

Содержание работ охватывает основные аспекты программирования на языке С.

Практическая работа 1 ориентирована на обучение студентов методике составления и отладки линейных программ, знакомство с интегрированной средой программирования BC и справочной системой среды.

Практическая работа 2 предполагает изучение методов отладки программ с помощью встроенного отладчика среды BC и составление линейной программы средней сложности с использованием библиотеки математических функций языка С.

Практические работы 3 и 4 посвящены соответственно программированию задач с разветвлениями вычислительного процесса и программированию циклических вычислительных процессов, включая использование вложенных условных операторов и вложенных циклов. При выполнении указанных работ внимание студентов специально акцентируется на методологии отладки программ и поиска ошибок периода выполнения с помощью «горячих» клавиш.

Практическая работа 5 помогает студентам научиться создавать и использовать в своих программах собственные функции.

Практическая работа 6 помогает научиться строить многомодульные программные проекты, включающие в себя большое количество пользовательских функций.

Практические работы 7 и 8 посвящены использованию одномерных и двухмерных массивов и технике передачи массивов в функции.

В практической работе 9 студенты изучают символьные строки и встроенную библиотеку функций для работы со строками.

Описание каждой работы содержит необходимые для ее выполнения сведения по соответствующим разделам программирования, набор индивидуальных вариантов заданий, пример программы решения аналогичной задачи и требования к знаниям, умениям и навыкам, которыми студенты должны овладеть в результате выполнения работы. Кроме того, в описание каждой работы включены разделы, содержащие информацию о типичных ошибках, которые, как показывает практика, наиболее часто допускаются студентами, а также контрольные вопросы для закрепления материала.

При выборе материала для заданий автор учитывал преемственность обучения. В частности, для выполнения Практических работ *2,* 3 и 6 от студентов требуется вспомнить разделы из курса элементарной математики (геометрии и тригонометрии), а для выполнения Практической работы 5 — из курса вычислительных методов. В разделах «Справочные сведения», которые включены в описание каждой из работ, приведены краткие справки, содержащие всю необходимую для выполнения работы информацию из сопутствующих курсов.

Практические работы могут выполняться студентами с использованием любого из доступных им компиляторов. По мнению автора наиболее удобным для первоначального изучения языка С является интегрированная среда программирования Borland С версий 3.0/3.1. Хотя эти версии не позволяют создавать современные 32-разрядные приложения, они обладают другими более важными для студентов качествами:

интерфейс сред ВС3.0/3.1 унифицирован со средой Turbo Pascal 7.0, которая, как правило, уже знакома студентам к моменту начала изучения языка С, поэтому они могут не отвлекаться на изучение новой среды;

среды ВС3.0/3.1 предъявляют минимальные требования к объему памяти на жестком диске (примерно 10/40 Мбайт соответственно), что позволяет установить их на компьютерах с самыми скромными возможностями;

среды ВС3.0/3.1 предоставляют пользователям удобную справочную систему, а также эффективные средства поиска ошибок и отладки программ;

в отличие от многих других версий указанные версии устойчиво поддерживают русский язык, что позволяет легко комментировать разрабатываемые программы (это особенно важно на начальном этапе изучения языка).

Данное учебное пособие ориентировано именно на использование указанных сред разработки программ С. Как показывает практика, овладев одной из этих сред, студенты легко переходят к более современным системам, включая среду визуального программирования C++ Builder.

Автор благодарит доцента канд. техн. наук Ю. П. Ильина и преподавателя Н. В. Валиулину за полезные советы, заместителя директора СПИШЭ по методической работе Н. А. Васильеву за внимание и помощь в решении организационных вопросов, возникавших в процессе подготовки рукописи к печати, и Н. В. Семенютину, взявшую на себя нелегкий труд по компьютерному набору текста.

Практическая работа 1

ИЗУЧЕНИЕ ИНТЕГРИРОВАННОЙ СРЕДЫ BC И АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ ЯЗЫКА С. ПРОГРАММИРОВАНИЕ И ОТЛАДКА ПРОСТЕЙШЕЙ ЗАДАЧИ

1.1. Цель работы

1. Изучить структуру интегрированной среды Borland C (BC) — версия 3.1 и освоить ее основные возможности.
2. Изучить справочную систему среды BC и научиться использовать ее на практике.
3. Изучить функции ввода-вывода данных и управления выводом на экран.
4. Научиться программировать задачи с простым линейным (без разветвлений и циклов) алгоритмом.
5. Практически освоить правила записи арифметических выражений и правила использования скобок при записи сложных «многоэтажных» выражений
6. Изучить правила преобразования данных при выполнении бинарных арифметических операций и операции присваивания.

1.2. Справочные сведения

Интегрированная среда Borland C (BC) версия 3.1 предназначена для разработки, отладки и выполнения программ, написанных на языке C/C-н-. Структура и возможности этой системы за редким исключением аналогичны структуре и возможностям системы Turbo Pascal версии 7.0, которая должна быть хорошо знакома студентам из соответствующего курса (совпадают основные пункты главного и вложенных меню, выполняемые ими функции, а также «горячие» клавиши). Поэтому для работы в среде BC студентам достаточно восстановить имеющиеся у них знания и навыки.

Главное меню системы содержит следующие пункты:

1. *E* — системное меню;
2. *File* — работа с файлами;
3. *Edit* — редактирование текста программы;
4. *Search* — поиск и замена фрагментов текста;
5. *Run* — выполнение программы;
6. *Compile* — компиляция программы;
7. *Debug* — отладочные средства среды BC 3;
8. *Project* — организация программного проекта (многофай-

ловой программы);

9. *Options* — настройка интегрированной системы (установка режимов компиляции программы и работы среды);
10. *Windows* — управление окнами среды;
11. *Help* — справочная система.

Каждый из пунктов главного меню имеет ряд подпунктов, организованных в форме вложенных (выпадающих) меню, возможности которых изучаются студентами в процессе проведения лабораторных работ.

Далее приводятся основные сведения по элементам языка С, необходимые для программирования операций ввода-вывода при работе с консолью и файлами, а также для записи на языке С простых операторов, содержащих арифметические операции.

1.2.1. Структура программы на языке С

Любая программа на языке С начинается с инструкций подключения к программе заголовочных файлов, обеспечивающих подключение библиотечных функций.

Эти инструкции имеют вид

Mnclude < имя заголовочного файла>

Все заголовочные файлы имеют расширение *h*. Как правило, в программе присутствуют по крайней мере две инструкции:

Mnclude <stdio.h>
Mnclude <conio.h>

первая из которых обеспечивает подключение функций ввода-вывода при работе с экраном и файлами, а вторая — функции управления работой монитора (очистка экрана, перемещение курсора, определение цветов фона и символов и т.п.). Кроме того, часто используются инструкции

Mnclude <math.h>
Mnclude <string.h>
Mnclude<dir.h>
Mnclude <dos.h>

которые обеспечивают подключение к программе математических функций, функций работы со строками, функций для работы с дисками и каталогами и DOS-функциями.

Каждая инструкция *include* записывается с новой строки. В конце инструкции никакие разделительные знаки не используются.

После всех инструкций подключения заголовочных файлов обычно помещают описание главной функции программы. Оно начинается с заголовка функции

void main ()

за которым никакие знаки препинания не ставятся.

За заголовком следует тело этой функции — последовательность выполняемых операторов, заключенная в фигурные скобки, которые имеют смысл аналогичный *begin* и *end* в языке *Pascal.* Каждый оператор обязательно должен завершаться символом «;» (точка с запятой). Обычно каждый из операторов записывается с новой строки, что обеспечивает хороший обзор программы.

Если программа содержит несколько функций, то они располагаются последовательно одна за другой, причем функция *main* может занимать в этой последовательности произвольное место. Выполнение программы в любом случае начинается с выполнения функции *main.*

Внимание. В конце программы после закрывающейся фигурной скобки, завершающей последнюю функцию, в отличие от программы на языке *Pascal* никаких знаков препинания ставить не следует.

1.2.2. Работа с консолью

Ввод (чтение) данных с консоли (клавиатуры) выполняется с помощью функции

scanf (строка форматов вводимых данных,
список адресов вводимых данных); (1.1)

а вывод данных на консоль (экран монитора) осуществляется с помощью функции

printf (строка форматов выводимых данных,
список выводимых значений); (1.2)

В списке адресов вводимых данных в функции (1.1) необходимо перед именем каждого данного поставить знак адреса — *&* (амперсенд).

В качестве списка выводимых данных в функции (1.2) необходимо перечислить имена всех выводимых данных, разделив их запятыми.

В строке форматов каждому вводимому или выводимому данному должен соответствовать свой формат, который определяет, в какой конкретно форме должно выводится или вводится соответствующее данное. Каждый формат начинается с символа «%». В простейшем случае непосредственно за символом «%» следует спецификация формата.

Основные спецификации формата:

i — целое десятичное число со знаком;
d — аналогично спецификации *i;*
li — длинное целое десятичное число со знаком;
ld — аналогично спецификации li;
u — целое десятичное число без знака;
lu — длинное целое десятичное число без знака;
o — целое число без знака в восьмеричной форме;
x — целое число без знака в шестнадцатеричной форме (используются строчные буквы *abcdef);*
X — целое число без знака в шестнадцатеричной форме (используются прописные буквы *ABCDEF);*
f — десятичное число с плавающей точкой вида [-]xx.xxxx;
lf — длинное десятичное число с плавающей точкой вида [-]xx.xxxx;
e — десятичное число с плавающей точкой в экспоненциальной форме вида [-Jxx.xxx *e* [-] xx;
E — десятичное число с плавающей точкой в экспоненциальной форме вида [-jxx.xxx *E* [-]xx;
le — длинное десятичное число с плавающей точкой в экспоненциальной форме вида [-Jxx.xxx *e* [-]xx:
lE — длинное десятичное число с плавающей точкой в экспоненциальной форме вида [-]xx.xxx *E* [-]xx:
g — наиболее короткий из форматов f или *e;*
G — наиболее короткий из форматов f или *E;*
lg — наиболее короткий из форматов *lf* или *l*e;
lG — наиболее короткий из форматов *lf* или *lE;*
c — одиночный символ;
s — строка (последовательность символов, заключенная в кавычки).

При выводе на экран значений переменных целого типа между символом «%» и кодом формата можно вставить число, определяющее ширину поля вывода — количество позиций, отводимых для изображения значения. При выводе данных с плавающей точкой в качестве ширины поля вставляется конструкция вида *a.Ь,* где я представляет собой полное количество позиций, отводимых для вывода значения переменной с учетом знака и десятичной точки, а *b* — количество выводимых знаков после запятой (по умолчанию после запятой выводится шесть знаков).

Если заданная ширина поля вывода недостаточна для изображения числа (с учетом знака и десятичной точки), необходимое количество позиций добавляется автоматически, т. е. данное всегда выводится полностью без усечения. Если ширина поля избыточна, то лишние позиции слева автоматически заполняются пробелами. При необходимости заполнить пробелами лишние позиции справа (выровнять число по левому краю поля) в формате

перед заданной шириной поля достаточно вставить знак «минус».

Перед любым из форматов в строке форматов можно задать одиночный символ или последовательность символов. В случае вывода указанная последовательность символов будет выводиться на экран непосредственно перед значением соответствующей величины. В случае ввода пользователь программы должен набрать на клавиатуре перед задаваемым значением указанную перед форматом последовательность символов.

При выводе на экран в качестве указанной последовательности символов можно, в частности, задать символ «\n» для вывода с новой строки экрана, символ «пробел» для разделения данных между собой или имя переменной для вывода значений данных вместе с их именами.

Например, если задано

```
int a = −22, b = 10;
float d1 = 7.234, d2 = −2.3769:
char c = '#';
```

то функция

```
printf ("%c_a= %4d_b= %i_b8= %4o\n_d1= %6.3f_d2= %.2f_%c",
c, a, d1, d2, c),
```

где символ «␣» обозначает пробел, а имя b8 присвоено переменной b, выраженной в восьмеричной форме, выведет на экран следующую информацию:

```
#_ a = _ −22_ b = 10 _b8 = _ _12
_ d1 = _7.234 _d2 = −2.38 _#
```

Функция чтения с клавиатуры

```
scanf (%c %id1 = %f d2 = %f",&c,&a,&d1,&d2);
```

для ввода указанных значений требует набора на клавиатуре следующей информации:

```
#_ −22_d1 = 7.234_d2 = −2.3769
```

(в качестве разделителей здесь вместо пробела можно нажимать клавишу [Enter], т.е. вводить каждое из данных с новой строки).

1.2.3. Управление выводом на экран

Имеется большой набор библиотечных функций для управления выводом на экран, которые определены в файле *conio.h.* Приведем основные из этих функций:

clrscr() — очищает экран;

getch() — ожидает нажатия любой клавиши;

textcolor (цвет) — устанавливает цвета символов;

textbackground (цвет) — устанавливает цвета фона: конкретный цвет задается номером или английским словом, набранным прописными буквами (RED, BLUE и т.п.);

highvideo()— устанавливает вывод повышенной яркостью;

lovvideo() — устанавливает вывод пониженной яркостью;

normvideo() — устанавливает вывод нормальной яркостью;

gotoxy(номер позиции в строке, номер строки на экране) — устанавливает курсор в заданную позицию на экране;

wherex () — возвращает текущий номер позиции курсора на строке экрана;

wherey () —- возвращает текущий номер строки, на которой находится курсор.

Например, оператор

gotoxy(wherex(), *wherey()*+1); переводит курсор на следующую строку в ту же позицию, в которой он находился на предыдущей строке до выполнения этого оператора.

Для окраски всей поверхности экрана желаемым цветом достаточно задать цвет фона и затем очистить экран, т.е. выполнить последовательность функций

```
textbackground (цвет);
clrscr();
```

Для вывода на экран информации в соответствии со сделанными установками цвета и яркости вместо функций ввода и вывода (1.1) и (1.2) следует использовать функции

cscanf (строка форматов вводимых данных,
список адресов вводимых данных); (1.3)

cprintf (строка форматов выводимых данных,
список выводимых данных); (1.4)

Параметры этих функций полностью совпадают с(1.1)и(1.2), различие заключается только в добавлении префикса «с» в именах функций. Кроме того, для перехода на новую строку вместо символа «\n» в функции *cprintf* в строке форматов следует использовать последовательность управляющих символов *«\n\r»*.

1.2.4. Работа с файлами

Для чтения информации из файла или записи ее в файл требуется выполнить следующие действия.

1. Описать логическое имя файла с помощью оператора

FILE * логическое имя;

например *FILE * myfile:*

2. Открыть файл (связать логическое имя файла с физическим именем) с помощью оператора

логическое имя файла =
fopen (физическое имя файла, режим доступа к файлу);

Основные режимы доступа: «w» — открыть файл для записи информации; «а» — открыть файл для добавления информации; «r» — открыть файл для чтения информации.

При использовании режима «w», если файл с заданным физическим именем отсутствует, файл создается, в противном случае он обновляется. В режиме "а" при отсутствии файла таковой создается, а в режиме *«r»* генерируется сигнал ошибки.

После открытия файла ввод (чтение) информации из файла или запись (вывод) ее в файл осуществляется соответственно с помощью функций:

fscanf (логическое имя файла,
строка форматов вводимых данных,
список адресов вводимых данных); (1.5)

fprintf (логическое имя файла,
строка форматов выводимых данных,
список выводимых данных); (1.6)

Строка форматов и списки данных или их адресов в (1.5), (1.6) формируются аналогично функциям (1.1), (1.2).

После завершения работы с файлом его следует закрыть, для чего используется функция

fclose (логическое имя файла); (1.7)

1.2.5. Арифметические операции и операция присваивания

В языке С имеются две унарных (работающих с одним операндом) арифметических операции: операция минус унарный, которая меняет знак своего единственного операнда, и операция явного преобразования типа, которая преобразует тип операнда к заданному типу.

Например, если определено

float d= 1.23;

то результатом операции *-d* является отрицательное действительное число -1.23, а операция *(int)d* преобразует *d* к целому типу, т.е. дает целое число 1.

Существует пять бинарных (имеющих два операнда) арифметических операций в языке С: + сложение; - вычитание; * умножение; / деление; % остаток от деления (деление по модулю 2).

Операция остаток от деления является корректной только тогда, когда оба ее операнда являются целыми. Результат этой операции всегда есть целое число, знак которого совпадает со знаком первого из операндов. Остальные четыре операции могут применяться при операндах разных типов, при этом тип результата определяется в соответствии с таблицей преобразования типов, приведенной в Приложении 1.

Например, фрагмент программы

```
int i = 5, j = 3, k, k1, k2, k3;
float f1 = 2.5; f2 = 0.5, g, g1, g2;
k = i + j; k1 = i/j; k2 = –i%j; k3 = (int)f1/i;
g = f1/f2; g1 = f1/i; g2 = (int)f1/f2;
```

дает следующие результаты:

```
k = 8; k1 = 1; k2 = –2; k3 = 0;
g = 5.0; g1 = 0.5; g2 = 4.0;
```

Следует обратить внимание на то, что результат деления целого числа на целое является целым числом. При этом результат формируется путем отбрасывания дробной части частного, т.е. никакого округления не выполняется.

В одном выражении может содержаться произвольная последовательность арифметических операций. Порядок их выполнения определяется в соответствии с таблицей приоритетов операций (см. Приложение 1). Этот порядок может быть при необходимости изменен с помощью круглых скобок, расставленных в соответствующих местах. Как следует из таблицы приоритетов, скобки имеют наивысший приоритет.

Операция присваивания имеет вид

A = 6,

где A — имя простой переменной, которой следует присвоить значение b; b — число либо простая переменная или выражение. Операция присваивания выполняется следующим образом. Сначала вычисляется выражение b. Далее, если тип переменной A совпадает с типом полученного результата, этот результат присваивается переменной A, в противном случае сначала результат автоматически преобразуется к типу переменной A и только после этого выполняется присваивание. Вычисление выражения *b* всегда предшествует присвоению, поэтому, например, действитель-

ная величина f в результате выполнения операции присваивания f= 6/5 получит значение 1.0, а не 1.2, как это может показаться на первый взгляд.

1.3. Содержание работы

Для освоения практических приемов программирования на языке С и работы в среде ВС студентам предлагается задача, предусматривающая ввод данных, выполнение вычислений, предусмотренных индивидуальным заданием, и вывод полученных результатов на консоль и в файл. Все данные рекомендуется выводить с их именами. Файл с заданным в индивидуальном задании именем можно расположить на любом диске или директории, открытом для записи (в частности, можно использовать флоппи-диск А).

Программа, созданная в ходе выполнения работы, должна включать в себя:

комментарий, содержащий сведения о назначении программы, ее авторе и дате написания;

подключение необходимых библиотек стандартных функций (с помощью директивы # include);

заголовок главной функции main ();

процедурный блок, содержащий описание переменных и констант, последовательность выполняемых операторов, а также комментарии, поясняющие работу отдельных частей программы.

Отчет по работе должен содержать:

перечисление всех подпунктов из пунктов 2...6 и 10, 11 основного меню среды ВС с подробным описанием функций, которые они выполняют;

формулировку задачи, предложенной в индивидуальном задании;

программу решения задачи с комментариями к основным операторам;

пример работы программы (введенные данные и полученные результаты).

Литература: [1] с. 7-33, 63-68, 142-146, 277-280. 1.4.

Индивидуальные задания

Вариант 1.1.

Ввести с клавиатуры целые i1, i2, i3 и действительные *d1, d2, d3* величины. Вычислить

$$X = \frac{10^3}{3} * \frac{i1}{i2+i3} * (d1+d2+d3);$$

$$Y = \frac{d1}{d2*d3} + \frac{i3}{i1+i2};$$

$$Z = d1 * \frac{i1+i2}{d2} + \frac{i3}{i1*i2}.$$

Введенные величины и результаты расчетов вывести в файл *f1.txt* и на экран синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана.

Вариант 1.2.

Ввести с клавиатуры целые *i, j, к* и действительные *x, y* величины. Вычислить

$$A = \frac{i+j}{k} * \frac{x}{i+j+k} + \frac{1}{2} * \frac{i}{j} + y;$$

$$B = x + y + \frac{i+j}{k} + \frac{k}{i*j};$$

$$C = 10^{-9} * \left(\frac{i}{k} + \frac{k}{j} + \frac{j}{i}\right).$$

Введенные величины и результаты расчетов добавить в конец файла f2.*txt* и вывести на экран желтым цветом повышенной яркости на синем фоне. Весь экран предварительно окрасить синим цветом.

Вариант 1.3.

Ввести с клавиатуры целые *a1, a2, a3* и действительные *m, n* величины. Вычислить

$$P = 10^{-3}\left(\frac{4}{5} + \frac{a1*a2}{a3} + \frac{a1}{a2*a3}\right);$$

$$R = m*n + \frac{m}{a1+a2} + \frac{n}{a2+a3};$$

$$S = \left(\frac{2}{a1} + \frac{2}{a2} + \frac{2}{a3}\right) * \frac{a1*m}{a2*n}.$$

Введенные величины и результаты расчетов вывести в файл *f3.txt* и на экран. Целые числа выводить синим цветом повышенной яркости, а действительные — зеленым цветом пониженной

яркости. Вывод выполнить на желтом фоне с 10-й позиции пятой строки экрана.

Вариант 1.4.

Ввести с клавиатуры целые i, *j, к* и действительные *a, b, c* величины. Вычислить

$$M = 2 * 10^5 * \left(\frac{i}{k} + \frac{j}{i} + \frac{k}{j}\right);$$

$$N = \frac{a * b}{c} + \frac{b}{a * c};$$

$$L = \frac{4}{5} - \frac{i * k}{j} - a - b - c.$$

Введенные величины и результаты расчетов вывести в файл *f4.txt* и на экран синим цветом повышенной яркости на желтом фоне. Вывод организовать в столбик начиная с 10-й позиции третьей строки.

Вариант 1.5.

Ввести с клавиатуры целые α, β, μ и действительные *d1, d2* величины. Вычислить

$$L = \frac{1}{2} * \left(\frac{\alpha * \beta}{\mu} + \frac{\beta}{\alpha * \mu}\right);$$

$$K = d1 + \left(\frac{1}{\alpha} + \frac{1}{\beta} + \frac{1}{\mu}\right) * d2 * 10^{-2};$$

$$S = \frac{3}{5} + \frac{d1}{d2} * (\alpha + \beta + \mu).$$

Введенные величины и результаты расчетов добавить в конец файла *f5.txt* и вывести на экран на желтом фоне синим цветом повышенной яркости для действительных чисел и красным цветом повышенной яркости для целых чисел. Предварительно весь экран закрасить белым цветом.

Вариант 1.6.

Ввести с клавиатуры целые *a, b, c* и действительные *d, e* величины. Вычислить

$$A = \frac{8}{5} + \frac{a * b}{c} + \frac{a}{b + c};$$

$$B = \frac{d}{a+b} + \frac{e}{a*b} + \frac{a+b+c}{d+e};$$

$$Q = -d*10^6 + \frac{a}{b}.$$

Введенные величины и результаты расчетов вывести в файл *f6.txt* и на экран синим цветом повышенной яркости на желтом фоне. Целые величины вывести с 10-й позиции пятой строки экрана, а действительные — с 10-й позиции седьмой строки. Экран предварительно окрасить зеленым цветом.

Вариант 1.7.

Ввести с клавиатуры целые *p, r, q* и действительные *a, b, c* величины. Вычислить

$$M = \frac{0,8}{p} + \frac{1}{r} + \frac{0,5}{q};$$

$$N = 10^2\left(\frac{p*q}{r} - \frac{q}{p*r} + a\right);$$

$$L = (p*r + r*q + q*p).$$

Введенные величины и результаты расчетов добавить в файл *fl.txt* и вывести на экран синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана.

Вариант 1.8.

Ввести с клавиатуры целые *g, f, h* и действительные *d1, d2* величины. Вычислить

$$X = \frac{3}{4} + \frac{g*f}{h} + \frac{f}{g*h};$$

$$Y = \frac{d1+d2}{d1} - \frac{g+h/f}{d2};$$

$$Z = 10^5 + d1 + d2.$$

Введенные величины и результаты расчетов вывести в файл *f8.txt* и на экран черным цветом пониженной яркости на белом фоне с седьмой позиции 14-й строки экрана. Экран предварительно окрасить голубым цветом.

Вариант 1.9.

Ввести с клавиатуры целые $i1$, $i2$ $i3$и действительные *a, b* величины. Вычислить

$$\gamma = \frac{1}{2} * \frac{a}{b + a/b};$$

$$\theta = \frac{1}{i1} + \frac{1}{i2} + \frac{1}{i3};$$

$$\varphi = \frac{i1}{i2 * i3} + \frac{i1 * i3}{i2}.$$

Введенные величины и результаты расчетов добавить в конец файла *f9.txt* и вывести на экран желтым цветом повышенной яркости на синем фоне в столбик с 12-й позиции четвертой строки экрана.

Вариант 1.10.

Ввести с клавиатуры целые *a, b, c* и действительные gl, *g2* величины. Вычислить

$$P = \frac{10^3}{g1 * g_2} + \frac{1}{a + b};$$

$$R = \frac{a * b}{c} + \frac{b}{a * c};$$

$$T = \frac{a + b}{c + b} + \frac{a + b/c}{c + a/b}.$$

Введенные величины и результаты расчетов вывести в файл f10.txt и на экран белым цветом повышенной яркости на синем фоне. Данные целого типа вывести с 12-й позиции четвертой строки экрана, а данные действительного типа — с 12-й позиции седьмой строки.

Вариант 1.11.

Ввести с клавиатуры целые i, *j, к* и действительные *m, n* величины. Вычислить

$$V = \frac{10^3 + 10^4}{i} + \frac{i}{j + k} + \frac{j * k}{i};$$

$$W = \frac{i + m/n}{j + i/k} + \frac{1}{5};$$

$$Z = \frac{1}{i} + \frac{1}{k} + \frac{1}{j} + \frac{1}{m} + \frac{1}{n}.$$

Введенные величины и результаты расчетов вывести в файл *fll.txt* и на экран, который предварительно окрасить синим цветом. Действительные числа вывести с 12-й позиции четвертой строки экрана желтым цветом, целые — с 15-й позиции восьмой строки белым цветом.

Вариант 1.12.

Ввести с клавиатуры целые c1, c2, c3 и действительные *si, s2* величины. Вычислить

$$A = \frac{4}{10} + \frac{c1}{c2 * c3} + \frac{c1 * c3}{c2};$$

$$B = \frac{s1 + c2/c1}{s2 + c2/c3} - \frac{1}{c1} - \frac{1}{s1};$$

$$C = 10^3 * \left(\frac{s1 + c1}{s2 + c2} + c3 \right).$$

Введенные величины и результаты расчетов вывести в файл *f12.txt* и на экран белым цветом повышенной яркости на желтом фоне с пятой позиции четвертой строки экрана. Экран предварительно окрасить синим цветом.

Вариант 1.13.

Ввести с клавиатуры целые *x, y, z* и действительные *v, w* величины. Вычислить

$$F = \frac{5x}{3y} + \frac{2y}{3z} + \frac{4v}{3w} + \frac{1}{2} xyz;$$

$$G = \frac{xy}{z} + \frac{y}{xz} + \frac{2 + y/z}{3 + w/v};$$

$$H = 10^5 * xyz + 10^3 * vw.$$

Введенные величины и результаты расчетов добавить в конец файла f13.txt и на экран синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана. Экран предварительно окрасить зеленым цветом.

Вариант 1.14.

Ввести с клавиатуры целые t, u, z и действительные $z1$, $z2$ величины. Вычислить

$$\alpha = 10^4 * z1 + \frac{10^3 t}{uz};$$

$$\lambda = \frac{z1}{z2} + \frac{u}{t} + \frac{z2}{z};$$

$$\beta = \frac{tu}{z} + \frac{u}{t} + \frac{z1}{u} + z2.$$

Введенные величины и результаты расчетов вывести в файл f14.txt и на экран синим цветом повышенной яркости в столбик с 12-й позиции четвертой строки экрана. Экран предварительно окрасить белым цветом.

Вариант 1.15.

Ввести с клавиатуры целые *a, b, c* и действительные *e,f* величины. Вычислить

$$K = \frac{a + b/c}{c/a + b} + e * f;$$

$$L = 10^3 \left(\frac{a}{bc} + \frac{ef}{a} \right) + 10^4 \frac{bc}{a};$$

$$M = \frac{1}{a} + \frac{2}{b} + \frac{3}{c} + \frac{4}{e} + \frac{5}{f}.$$

Введенные величины и результаты расчетов вывести в файл f15.*txt* и на экран желтым цветом повышенной яркости на синем фоне с шестой позиции седьмой строки экрана.

Вариант 1.16.

Ввести с клавиатуры целые *a1, a2, a3* и действительные b1, *b2* величины. Вычислить

$$A = \frac{1 + a1/a2}{1 + a3/a2} + \frac{b1}{b2} + \frac{a1 * b1}{a2 + b2};$$

$$B = 10^3 b1 + 10^4 b2 + \frac{a1}{a2 + a3};$$

$$C = \frac{1}{2} + \frac{2}{a1} + \frac{3}{b1} - \frac{4}{a2} - \frac{5}{b2}.$$

Введенные величины и результаты расчетов вывести в файл *f16.txt* и на экран. Целые значения выводить синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана, действительные — зеленым цветом повышенной яркости на красном фоне с 12-й позиции седьмой строки. Экран предварительно окрасить белым цветом.

Вариант 1.17.

Ввести с клавиатуры целыеρ, π, μ и действительные *x, y* величины. Вычислить

$$Q = \frac{1}{2} + 10^3\rho + 10^4\pi + 10^5\mu;$$

$$T = \frac{\rho}{\pi\mu} + \frac{\pi}{\mu} + \frac{x+y}{\rho} + \frac{\pi+\mu}{y};$$

$$S = \frac{x+\rho/\pi}{y+\rho/\mu} + \frac{\rho+\mu+\pi}{x+y}.$$

Введенные величины и результаты расчетов вывести в файл /17.*txt* и на экран зеленым цветом повышенной яркости на голубом фоне с третьей позиции пятой строки экрана.

Вариант 1.18.

Ввести с клавиатуры целые *m, n, p* и действительные *d1, d2* величины. Вычислить

$$C = \frac{m+n}{m+p} + 10^{-3}\frac{d1}{d2};$$

$$S = \frac{m}{np} + \frac{nm}{p} + \frac{d1}{n} + \frac{p}{d2};$$

$$Z = \frac{p}{m} * \frac{d1+m/n}{d2+m/p}.$$

Введенные величины и результаты расчетов вывести в файл f18.txt и на экран синим цветом повышенной яркости на желтом фоне в столбик с 12-й позиции четвертой строки экрана.

Вариант 1.19.

Ввести с клавиатуры целые*k, kk, kkk* и действительные *m, n* величины. Вычислить

$$W = \frac{3}{2} + \frac{kkk}{k+kk} + \frac{k+kkk}{kk};$$

$$U = \frac{10^2+1/k}{10^2} + \frac{m+kk/k}{n+kkk/k};$$

$$V = \left(\frac{3+kkk}{3+kk} - \frac{3,3+k}{3,3+kkk}\right)10^{-3}.$$

Введенные величины и результаты расчетов вывести в конец файла f19.*txt* и на экран белым цветом повышенной яркости на коричневом фоне с третьей позиции шестой строки экрана. Экран предварительно окрасить голубым цветом.

Вариант 1.20.

Ввести с клавиатуры целые *a, b, c* и действительные *alpha, beta* величины. Вычислить

$$J = 10^{-2}ab + \frac{2a}{bc} + \frac{3bc}{a};$$

$$K = \frac{10^{-2}}{a+b} * \left(\frac{2}{a} + \frac{alpha}{b} - \frac{c}{b * beta}\right);$$

$$L = \frac{abc}{1+alpha} + \frac{abc}{1+beta} + \frac{1}{5}.$$

Введенные величины и результаты расчетов вывести в файл *f20.txt* и на экран синим цветом повышенной яркости на желтом фоне. Целые числа вывести желтым цветом с четвертой позиции восьмой строки, а действительные — синим цветом со второй позиции пятой строки. Экран предварительно окрасить белым цветом.

Вариант 1.21.

Ввести с клавиатуры целые *t1, t2, t3* и действительные *d, e* величины. Вычислить

$$Q1 = \frac{t1 + t2/t3}{d} * \frac{e}{t3 + t2/t1};$$

$$Q2 = \left(\frac{1}{t1} - \frac{2}{t2} + \frac{3}{t3}\right)10^{-3};$$

$$Q3 = \frac{1}{3} + \frac{t1}{t2 * t3} + \frac{t1 * t3}{t2} + \frac{d}{e + 0,5}.$$

Введенные величины и результаты расчетов вывести в файл *f11.txt* и на экран синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана. Экран предварительно окрасить коричневым цветом.

Вариант 1.22.

Ввести с клавиатуры целые *ml, mn2, m3* и действительные *a, b* величины. Вычислить

$$K1 = \frac{m1m2}{m3} + \frac{m1}{m2m3} + \frac{a}{b};$$

$$K2 = m1m2 + m2m3 + m3m1 + 21;$$

$$K3 = \frac{49}{50} + \frac{3}{50} * m1 + \frac{11}{50} * m2 + \frac{21}{50} m3.$$

Введенные величины и результаты расчетов вывести в файл f22. txt и на экран. Целые значения выводить синим цветом повышенной яркости на желтом фоне с 12-й позиции четвертой строки экрана, действительные — синим цветом пониженной яркости на красном фоне с 12-й позиции седьмой строки. Экран предварительно окрасить белым цветом.

Вариант 1.23.

Ввести с клавиатуры целые i, j, k, n и действительные v, w величины. Вычислить

$$A = \frac{7}{100} + \frac{i * j}{k * n} + \frac{i}{j} * \frac{k}{n};$$

$$B = \frac{1}{2} + \frac{i}{jk} - \frac{nk}{j} + j;$$

$$C = 100 - i - j - k - n.$$

Введенные величины и результаты расчетов вывести в конец файла f23.txt и на экран желтым цветом пониженной яркости на синем фоне с третьей позиции седьмой строки экрана. Экран предварительно окрасить голубым цветом.

Вариант 1.24.

Ввести с клавиатуры целые *a, b, c* и действительные *d, e* величины. Вычислить

$$W1 = 3 + \frac{a}{bc} + \frac{ab}{c} + \frac{d}{ef} + \frac{f}{de};$$

$$W2 = a * c - b - c + b * c;$$

$$W3 = d * \frac{a}{b} - \frac{c}{b} * e + 2,5.$$

Введенные величины и результаты расчетов вывести в файл *f24.txt* и на экран желтым цветом повышенной яркости в столбик с 12-й позиции четвертой строки экрана. Экран предварительно окрасить белым цветом.

1.5. Пример программы

Условие задачи. Ввести с клавиатуры целые *i1, i2, i3* и действительное *a* величины. Вычислить

$$\alpha = \frac{3}{4} * \frac{i1 + i2}{i3} + \frac{2}{3} * \frac{i1 + i3}{i2};$$

$$\beta = \frac{a + \dfrac{i1}{i2 * i3}}{a + \dfrac{i1 * i3}{i2}};$$

$$\gamma = i1 * i2 + i2 * i3 + i3 * i1.$$

Введенные величины и результаты расчетов вывести в файл ff.txt и экран. Целые значения выводить желтым цветом, действительные — белым цветом. Экран предварительно окрасить синим цветом.

Программа

```
/*********************************************
Лабораторная работа № 1.
Программирование линейной задачи.
Автор Скумбриевич Егор, гр. 441 к
31.12.2006
*********************************************/
#include <stdio.h>
#include <conio.h> // подключение библиотек
   void main()      // заголовок главной функции
    { int i1, i2, i2, gamma;
      double a, alpha,beta; // описание переменных
      textbackground (BLUE);
      clrscr();        // окраска экрана
                  /* Ввод исходных данных */
      printf ("Введите целые i1, i2, i3 и действительное a > ");
      scanf (" %i %i %i %lf", &i1, &i2, &i3, &a);
                        /* вычисления */
      alpha = 3.0/4*(i1 + i2)/i2 + 2.0/3*(i2 + i3)/i1;
      beta = (a + (double)i1/(i2*i3))/(a + (double)i1*i3/i2);
      gamma = i1*i2 + i2*i3 + i3*i1;
                  /* вывод результатов в файл */
      FILE * file;
      File = fopen("A:\\ff.txt", "w"); // открытие файла
```

```
  fprintf(file, "\n i1 = %i i2 = %i i3 = %i a = %lf"
                "\n alpha = %lf beta = %lf gamma = %i",
                i1, i2, i3, a, alpha, beta, gamma); //вывод в файл
  fclose(file); // закрытие файла

                /* вывод результатов на экран */
  textcolor(YELLOW); // установка цвета символов для вывода
                     // данных целого типа
  cprintf («\n\n Целые: i1 = %d i2 = %d i3 = %d gamma = %d»,
                                          i1, i2, i3, gamma);
  textcolor (WHITE);   // установка цвета символов для
                       // вывода данных действительного типа
  cprintf( "\n Целые: i1 = %i i2 = %i i3 = %i"
  "\nДействительные: a = %.3lf alpha = %.3lf beta = %.3lf",
  i1,i2,i3,a,alpha,beta);
  getch(); // ожидание нажатия любой клавиши
}
```

```
Результаты работы программы.
Введите целые i1, i2, i3 и действительное a > 1 2 3 1.5
Целые: i1 = 1 i2 = 2 i3 = 3 gamma = 11
Действительные: a = 1,5 alpha = 2,417 beta = 0,556
```

1.6. Типичные ошибки при выполнении работы

При делении целого числа на целое результат по определению является целым числом, т. е. происходит округление частного до целого. Таким образом, дробная часть результата теряется. Для устранения ошибки необходимо хотя бы один из операндов операции деления явным образом преобразовать к действительному типу, что обеспечит результат действительного типа, исключающий округление.

При программировании выражения типа $\frac{a}{bc}$ часто используется некорректная форма записи ***a/b*c***, которая интерпретируется компилятором в виде $a * / b * c$. Правильные варианты: *a/b/c* или *a/(b*c)*.

1.7. Требования к студентам

В результате выполнения работы студенты должны: знать основные пункты и подпункты меню интегрированной системы BC и уметь пользоваться ими на практике;

освоить справочную систему среды BC и уметь ее использовать на практике;

знать структуру программы на языке С. Уметь корректировать текст программы, в частности уметь выделить, удалить, скопировать, перенести в заданное место, отыскать по образцу и заменить фрагмент программы, используя для этого встроенные возможности среды BC;

уметь транслировать и выполнять программы в среде BC;

знать операции языка С;

знать правила записи и уметь корректно записывать сложные арифметические выражения на языке С. Знать приоритеты арифметических операций и правила преобразования типов данных при выполнении этих операций;

уметь программировать операции ввода-вывода при работе с консолью и файлами;

знать основные функции для управления выводом информации на экран монитора.

Контрольные вопросы

1. Перечислите обязательные компоненты программы на языке С.
2. Какая функция используется для вывода данных на экран?
3. Какая функция используется для ввода данных с клавиатуры?
4. Как описать логическое имя файла (файловую переменную) и связать его с физическим файлом?
5. Как записать информацию в файл и считать ее из файла?
6. Каким образом можно очистить экран и окрасить его заданным цветом?
7. Как установить указатель на заданную позицию экрана?
8. Какие операции называются унарными и какие бинарными?
9. Перечислите арифметические операции языка С.
10. Какова специфика выполнения операции деления, когда оба операнда являются целыми числами?
11. Что делает операция «%»?
12. Как работает операция присвоения?
13. Как войти в справочную систему среды BC?
14. Как откомпилировать программу?
15. Как запустить программу на выполнение?
16. Как сохранить текст созданной программы на жестком диске?

Практическая работа 2

РАЗРАБОТКА И ОТЛАДКА ЛИНЕЙНОЙ ПРОГРАММЫ, ИСПОЛЬЗУЮЩЕЙ СТАНДАРТНЫЕ БИБЛИОТЕЧНЫЕ ФУНКЦИИ

2.1. Цель работы

1. Углубить и закрепить полученные в Практической работе 1 знания об интегрированной среде программирования BC, структуре линейной программы на языке С и методике отладки программы.

2. Освоить методику отладки программ с помощью быстрых («горячих») клавиш. Изучить и опробовать на практике работу основных «горячих» клавиш.

3. Изучить основные функции библиотеки математических функций языка С и использовать их при составлении программы расчета элементов геометрических фигур.

2.2. Справочные сведения

Для освоения практических приемов программирования на языке С и работы в среде BC студентам предлагается задача, предусматривающая вычисление и вывод на консоль компьютера параметров геометрических фигур, указанных в индивидуальном задании.

Фигуры более сложной конфигурации рекомендуется расчленить на треугольники и выполнять вычисления для каждого из них. Предполагается, что условия существования заданной фигуры всегда выполняются, поэтому никакой проверки существования в программе не требуется.

В формулах и заданиях для треугольников (рис. 2,1, о) используются следующие обозначения: *a, b, c* — стороны треугольника; *A, B, C* — углы треугольника, противолежащие соответствующим сторонам; h_a, h_b, h_c — высоты треугольника, опущенные соответственно на стороны *a, b* и с; *p, S* — соответсвено половина периметра и площадь треугольника; r, *R* — радиус соответственно вписанной и описанной окружностей.

Приведем основные теоремы и формулы, необходимые для решения треугольников:

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$ — теорема синусов;

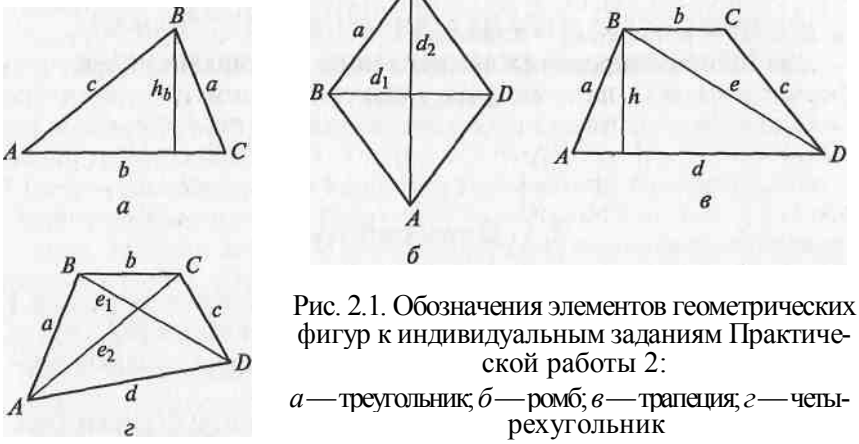

Рис. 2.1. Обозначения элементов геометрических фигур к индивидуальным заданиям Практической работы 2:

а — треугольник; *б* — ромб; *в* — трапеция; *г* — четырехугольник

— теорема косинусов;

$$a^2 = b^2 + c^2 - 2bc\cos A$$

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

— формула Герона;

$$R = \frac{a}{2\sin A} = \frac{b}{2\sin B} = \frac{c}{2\sin C};$$

$$r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}};$$

$$h_b = a\sin C = c\sin A.$$

В формулах и заданиях для ромба (рис. 2.1, *б)* используются следующие обозначения: *a* — сторона ромба; *A, B, C, D* — углы ромба; d_1, d_2 — диагонали ромба; *p, S*— соответственно периметр и площадь ромба.

В формулах и заданиях для трапеции (рис. 2.1, *в)* используются следующие обозначения: *a, c* — боковые стороны; *b, d*— соответственно верхнее и нижнее основания; *A, B, C, D* — углы; *h, e* — соответственно высота и диагональ; *p, S* — соответственно периметр и площадь.

При расчете элементов трапеции кроме приведенных для треугольников соотношений используются формулы:

$$A + B = C + D = 180°;$$

$$S = \frac{1}{2}(b+d)h.$$

В формулах и заданиях для четырехугольника общего вида (рис. 2.1, *г)* используются следующие обозначения: *a, b, c, d*— стороны; *A, B, C, D* — углы; e_1, e_2 — диагонали.

Для программирования вычислений с помощью приведенных формул в языке С предусмотрена библиотека математических функций, которая содержит, в частности, стандартные функции для вычисления тригонометрических функций sin(x), cos(x) и tan(x), а также обратных тригонометрических функций asin(x), acos(x) и atan(x). Указанная библиотека подключается к программе с помощью директивы

 # include <math.h>

или

#include «math.h»

Внимание. Аргументы тригонометрических функций обязательно следует выражать в радианах. Результаты вычисления обратных тригонометрических функций также представляются в радианах. Для пересчета углов из градусов в радианы служит соотношение

$$\varphi_{рад} = \frac{\pi}{180} * \varphi_{град}.$$

Примечание. В библиотеке математических функций отсутствует стандартная функция для вычисления котангенса, который может быть вычислен по формуле

$$c\tan(x) = 1/\tan(x).$$

Для вычисления арккотангенса следует использовать соотношение

$$ac\tan(x) = a\cos\left(\frac{x}{\sqrt{1+x^2}}\right).$$

2.3. Содержание работы

Требуется составить алгоритм, написать, отладить и выполнить в среде ВС программу, которая для фигуры, указанной в индивидуальном задании:

вводит с клавиатуры значения всех исходных данных, перечисленных в индивидуальном задании (значения углов задаются в градусах);

вычисляет все параметры заданной фигуры, которые перечислены в подразд. 2.2 и не входят в число исходных данных;

Таблица 2.1

Клавиша	Fl	F2	F3	F4	F5	F6	F7	F8	F9	F10
—	2	2	2	3	2	2	2	2	2	2
Ctrl	2	3	2	3	2 '	—	4	4	2	—
Alt	2	—	2	3	2	—	2	2	2	—

выводит на экран монитора и в файл *myfile.txt,* расположенный на любом из доступных для записи дисков, все исходные данные и результаты расчета, при этом значения всех углов должны быть выражены в градусах.

В процессе отладки и выполнения программы необходимо с помощью справочной системы или литературы (например, [1]) познакомиться с основными «горячими» клавишами интегрированной среды BC и опробовать их на практике.

Отчет по работе должен содержать:

1. Таблицу «горячих» клавиш системы BC (табл. 2.1).

В графах таблицы следует привести (на русском языке) сведения о назначении соответствующей клавиши или комбинации клавиш. В ходе настоящей работы заполняются только графы, помеченные цифрой 2 (остальные графы будут заполнены при выполнении следующих работ).

2. Формулировку задачи, расчетные формулы, текст програм мы, исходные данные и результаты расчетов.

Литература: [1] с. 7-36, 277 —280.

2.4. Индивидуальные задания

Вариант 2.1.

В треугольнике (см. рис. 2.1, а) заданы две стороны a и b и угол между ними *C.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.2.

В треугольнике (см. рис. 2.1, о) заданы сторона *a* и прилегающие к ней углы 2? и *C.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.3.

В треугольнике (см. рис. 2.1, *а)* заданы три стороны *a, Ь* и *с.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.4.

В треугольнике (см. рис. 2.1, *а)* заданы две стороны *a, b* и площадь *S.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.5.

В треугольнике (см. рис. *2.1, а)* заданы сторона *a,* угол Си площадь *S.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.6.

В треугольнике (см. рис. 2.1, *а)* заданы сторона *Ь,* угол *А* и радиус описанной окружности *R.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.7.

В треугольнике (см. рис. 2,1, *а)* заданы углы *А, В и* радиус описанной окружности *R.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.8.

В треугольнике (см. рис. 2.1, *а)* заданы стороны *а, Ъ* и радиус описанной окружности *R.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.9.

В треугольнике (см. рис. 2.1, я) заданы стороны *а, Ъ* и половина периметра *р.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.10.

В треугольнике (см. рис. 2.1, *а)* заданы углы *А, С и* высота h_b. Вычислить и вывести на экран и в файл остальные элементы.

Вариант 2.11.

В треугольнике (см. рис. 2.1, А) заданы стороны а, с и высота h_b. Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.12.

В треугольнике (см. рис. 2.1, *а)* заданы угол *А,* сторона с и высота h_b. Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.13.
В треугольнике (см. рис. *2.1, а)* заданы координаты вершин x_a, $y_a x_b$, y_b и x_c, y_c. Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.14.
В ромбе (см. рис. 2.1, *б)* заданы сторона *а* и угол *А.* Вычислить и вывести на экран и в файл остальные элементы треугольника.

Вариант 2.15.
В ромбе (см. рис. 2.1, *б)* заданы диагонали d_1 и d_2. Вычислить и вывести на экран и в файл остальные элементы ромба.

Вариант 2.16.
В ромбе (см. рис. 2.1, *б)* заданы диагональ *d*\ и площадь *S.* Вычислить и вывести на экран и в файл остальные элементы ромба.

Вариант 2.17.
В ромбе (см. рис. *2.1,6)* заданы сторона *а* и диагональ d_x. Вычислить и вывести на экран и в файл остальные элементы ромба.

Вариант 2.18.
В ромбе (см. рис. 2.1, *б)* заданы угол *А* и диагональ d_x. Вычислить и вывести на экран и в файл остальные элементы ромба.

Вариант 2.19.
В трапеции (см. рис. 2.1, *в)* заданы стороны *a, b, c, d.* Вычислить и вывести на экран и в файл остальные элементы трапеции.

Вариант 2.20.
В трапеции (см. рис. 2.1, *в)* заданы стороны *a, b, dn* высота *h.* Вычислить и вывести на экран и в файл остальные элементы трапеции.

Вариант 2.21.
В трапеции (см. рис. 2.1, *в)* заданы стороны *a, b, d* и угол *А.* Вычислить и вывести на экран и в файл остальные элементы трапеции.

Вариант 2.22.
В трапеции (см. рис. 2.1, *в)* заданы стороны *a, b,* высота *h* и диагональ *е.* Вычислить и вывести на экран и в файл остальные элементы трапеции.

Вариант 2.23.
В четырехугольнике (см. рис. 2.1, г) заданы четыре стороны *a, b, c, d и* диагональ e_1. Вычислить и вывести на экран и в файл остальные элементы четырехугольника.

Вариант 2.24.

В четырехугольнике (см. рис. 2.1, *г)* заданы стороны *a, b, d* и углы *A* и *B*. Вычислить и вывести на экран и в файл остальные элементы четырехугольника.

2.5. Дополнительное задание для «мастеров»

При выводе результатов на экран имена переменных выводить желтым цветом повышенной яркости, а значения переменных — белым. Весь экран предварительно окрасить синим цветом.

2.6. Пример программы

Условие задачи. В треугольнике заданы две стороны *a, b и* угол напротив одной из них *A*. Определить сторону c, углы B и C и площадь треугольника *S* (линейные размеры вводить и выводить в см, площадь — в см^2, углы — в градусах).

Расчетные формулы. При решении задач воспользуемся следующими формулами:

из теоремы синусов —

$$B = \arcsin\left(\frac{b}{a}\sin A\right);$$

сумма углов треугольника 180° —

$$C = 180 - (A + B);$$

из теоремы синусов —

$$c = a\frac{\sin C}{\sin A};$$

$$S = \frac{1}{2}ab\sin C.$$

Программа

```
Лабораторная работа № 2.
Решение треугольника по двум сторонам и углу
напротив одной из них.
Автор Корейко Александр Иванович, гр. 440
20.09.2006
```

```
#include "stdio.h"
#include "conio.h"
#nclude "math.h "          // подключение заголовочных файлов,
                           // библиотек стандартных функций
      void main ()         // заголовок главной функции
        {float a, b, c, A, B, C, S; float Ap, Bp,
      Cp; /*углы в радианах*/ float const pi =
      3.141593;

      /*очистка экрана и ввод данных*/
      clrscr ();
      print/("Введите стороны a и Ъ (см) >");
      scan/(" %/%/",& a, &Ь);
      print/("\n- Введите угол A (град)> ");
      scan/("%/",&A);

      /*вычисление углов B и C*/
      Ap = A*pi/180;           // перевод угла A в радианы
      Bp = asin(b/a*sin(Ap));
      B = Bp* 180/pi;          // перевод угла B в градусы
      C = 180- (Ал-B);
      Cp = C*pi/180;           // перевод угла C в радианы

      /*вычисление стороны с и площади треугольника*/ c =
      a*sin (Cp)/sin(Ap); S= 1.0/2* a*b*sin(Cp);

      /* печать результатов*/ print/
      ("\n. \n Результаты:"
           "\n Сторона c = % 7.2/см"
           "\n Углы B = % 6.2/град, C= % 6.2/град"
           "\n Площадь S= % 7.2/кв. см",
          c,B,C,S);

      /* Вывод результатов в файл */
      FILE */;
      F=/open("myfile.txt", "w"); // открытие файла
      /print/    (f, "\n. \n Результаты: " \n Сторона c =
      %12/.см"
           "\n Углы B = % 6.0 /град, C= % 6.0/град. "
           "\n Площадь S= % 7.2/кв.см",
       c,B,C,S); /close(f); Ц закрытие
    файла getch (); }
```

Р е з у л ь т а т ы работы программы.
Введите стороны *a и b* (см) > 1 1
Введите угол *A* (°) > 45

Результаты: сторона *c* =1,42 см углы *B* = 45°, *C* =*90°* площадь S=0,5 см^2

2.7. Типичные ошибки при выполнении работы

Аргументы библиотечных тригонометрических функций ошибочно задаются в градусах. Компилятор такую ошибку не перехватывает, поэтому ее обнаружение затруднительно.

Функция вычисления тангенса ошибочно записывается в виде tg(x), а обратные тригонометрические функции в виде arcsin(x), arcos(x) и arctg(x). В языке С принято написание: tan(x), asin(x), acos(x) и atan(x).

2.8. Требования к студентам

В результате выполнения работы студенты должны:

знать основные функции «горячих» клавиш системы ВС и уметь использовать их на практике при составлении и отладке программ;

знать основные функции математической библиотеки языка С и уметь использовать их на практике.

Контрольные вопросы

1. Как обеспечить в программе доступность библиотеки математических функций?
2. В какой форме следует задавать аргументы тригонометрических функций?
3. Как на языке С записываются функции вычисления десятичного и натурального логарифмов?
4. Как быстро получить справку по интересующей пользователя функции?
5. Как посмотреть и запустить на выполнение содержащийся в справочной системе среды ВС пример, иллюстрирующий работу выбранной функции?
6. Как выполняется пошаговая трассировка программы?
7. В чем заключается разница в работе «горячих» клавиш [F7] и [F8]?
8. Как обеспечить останов работающей программы перед выполнением определенного оператора?
9. Как определить значение какой-либо переменной или выражения в точке останова?
10. Как в точке останова изменить существующее значение переменной?
11. Каким образом можно продолжить работу программы после останова?
12. Как после останова работы программы и внесения в нее корректив пустить программу сначала?

Практическая работа 3

ПРОГРАММИРОВАНИЕ ЗАДАЧ С РАЗВЕТВЛЕНИЯМИ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

3.1. Цель работы

1. Научиться составлять и анализировать схемы программ с разветвлениями вычислительного процесса и овладеть средствами языка С для реализации разветвлений.

2. Освоить на практике средства отладки ветвящихся процессов, предоставляемые средой ВС.

3.2. Справочные сведения

Для реализации разветвлений в программах, написанных на языке С, имеются следующие средства.

1.Условный оператор

if(условие) *Op1 [else Op2]*

Здесь условие представляет собой любое выражение, имеющее значение 0 (ложь) либо не 0 (истина), а *Op1* и *Op2* — любые простые или составные операторы либо блоки. В частности, в качестве *Opl* или *Op2* может использоваться другой условный оператор.

Функционирование условного оператора иллюстрирует блок-схема, изображенная на рис. 3.1, *а.* Фрагмент условного оператора, заключенный в квадратные скобки, может быть опущен (не-

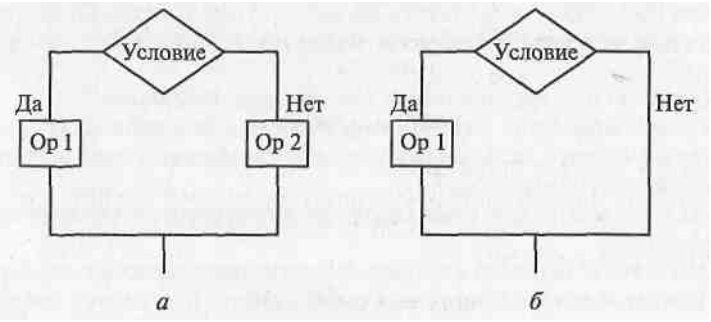

Рис. 3.1. Схемы выполнения условного оператора:
а — полная форма оператора; *б* — неполная форма оператора

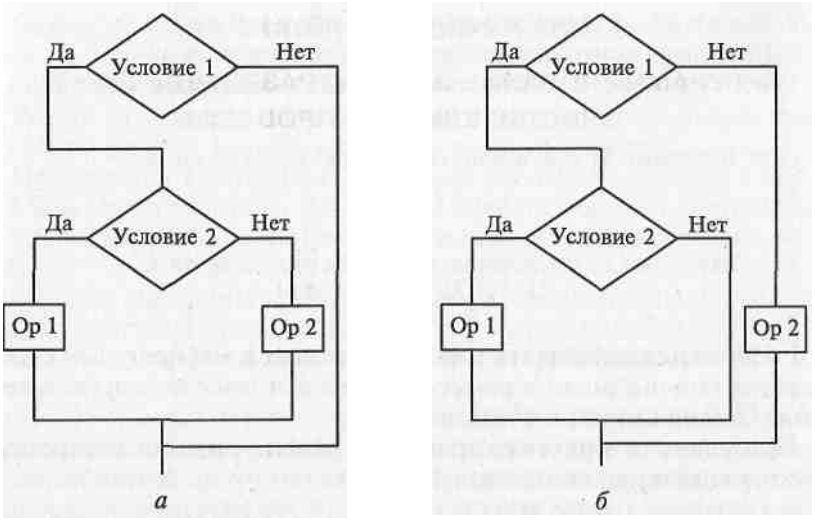

Рис. 3.2. Схемы вложенных условных операторов: *а* — оператор (3.1); *б* — оператор (3.2)

полная форма условного оператора). Блок-схема для подобной ситуации изображена на рис. 3.1, *б.*

В случае вложенных операторов *if* действует правило, согласно которому каждый из членов *else* относится к ближайшему к нему *if.* Изменить это правило можно используя фигурные скобки.

Например, в операторе

(3.1)

if(условие 1) *if* (условие 2) *Op* 1 *else op* 2

else относится к условию 2, что соответствует схеме, изображенной на рис. 3.2, *а,* а для оператора

if(условие 1) {*if* (условие 2) *Op* 1} *else Op* 2 **(3.2)**

относится к условию 1 (см. схему на рис. 3.2, *б).*

2. Условная операция

(условие)? *в*1 : *в*2,

где вl, в*2* — любые выражения.

Результатом выполнения условной операции является *в*\, если условие истинно, либо *el,* если условие ложно.

Например, оператор

$$max == (x>y)?\ x : y;$$

присваивает переменной *max* наибольшее из значений переменных x и *y.*

3. Оператор множественного выбора *switch*
(выражение целого либо символьного типа) { *case*
значение 1: последовательность операторов 1
break;
case значение 2: последовательность операторов 2
break;

case значение *к:* последовательность операторов *к*
break;

[default: последовательность операторов *к*+ 1]
} где *default* не является
обязательным.

При выполнении оператора *switch* реализуется последовательность операторов, соответствующая текущему значению выражения, указанного в круглых скобках. Если это значение не совпадает ни с одним из перечисленных значений и член *default* присутствует, выполняется следующая за ним последовательность операторов.

Если в указанной ситуации член *default* отсутствует, то действие оператора *switch* аналогично пустому оператору.

При формировании условий, определяющих выполнение условного оператора и условной операции, следует руководствоваться следующими положениями.

В языке С любое отличное от нуля значение (положительное или отрицательное) интерпретируется как истинное и только нулевое значение — как ложное.

Формирование условий выполняется с помощью операций сравнения:

= = — равно;
! = — не равно, и
операций отношения:
> — больше;
>= — больше или равно;
< — меньше;
<= — меньше или равно,
результатом которых является значение 1 (истина) или 0 (ложь).

Простые условные выражения могут при необходимости объединяться в более сложные с помощью логических операций:

&& - и;
|| — или;
! — не.

Результат бинарной операции && равен 1 (истина) только тогда, когда оба ее операнда являются истинными, и равен нулю (ложь), если хотя бы один из операндов ложный.

Результат бинарной операции || равен 1 (истина), если хотя бы один из операндов истинный, и равен нулю только тогда, когда оба операнда ложные.

Результат унарной операции ! равен отрицанию значения своего операнда, т. е. меняет истину на ложь, и наоборот.

Приоритеты выполнения операций указаны в таблице операций (см. Приложение 1). При необходимости порядок выполнения операций может быть изменен путем использования круглых скобок.

Отладку программ с ветвлениями удобно выполнять с помощью отладочных средств системы BC, которые позволяют остановить выполнение программы в любой (критической) точке и с помощью пункта меню *Debug -> Evaluate* или комбинации клавиш [Ctrl] + [F4] просмотреть значение любой известной в этой точке переменной либо выражения, определяющих направление разветвления. Установить точку наблюдения можно с помощью курсора или пункта меню *Debug —> Toggle break point* (комбинация клавиш [Ctrl] + [F8]). Достижение точки наблюдения в первом случае осуществляется с помощью пункта меню *Run -» go to cursor* (клавиша [F4]), а во втором случае с помощью пункта *Run -> Run* ([Ctrl] + [F9]). Достигнуть точки наблюдения можно также с помощью пошагового выполнения программы (клавиши [F7] или [F8], первая из которых при трассировке программы сканирует все вызываемые программой функции, а вторая не делает этого, выполняя всю вызываемую функцию за один шаг).

Литература: [1] с. 74—78.

3.3. Содержание работы

Требуется составить схему программы, разработать, отладить и выполнить программу для решения задачи.

Плоскость *xOy* разделена на четыре области, обозначенные буквами *A, B, Cu D* (рис. 3.3). Способ деления определяется вариантом индивидуального задания. Программа должна обеспечить ввод с клавиатуры координат двух точек x_a, y_a и x_b, y_b, определить и вывести на печать:

коды областей, в которых лежат указанные точки;

направление отрезка *ab* — вертикальное, горизонтальное, наклонное вниз или вверх по отношению к положительному направлению оси *x*.

Кроме того, в программе следует предусмотреть меню, определяющее, каким цветом должны выводиться результаты.

В ходе отладки программы следует опробовать отладочные средства среды BC, помогающие следить за ходом вычислительного

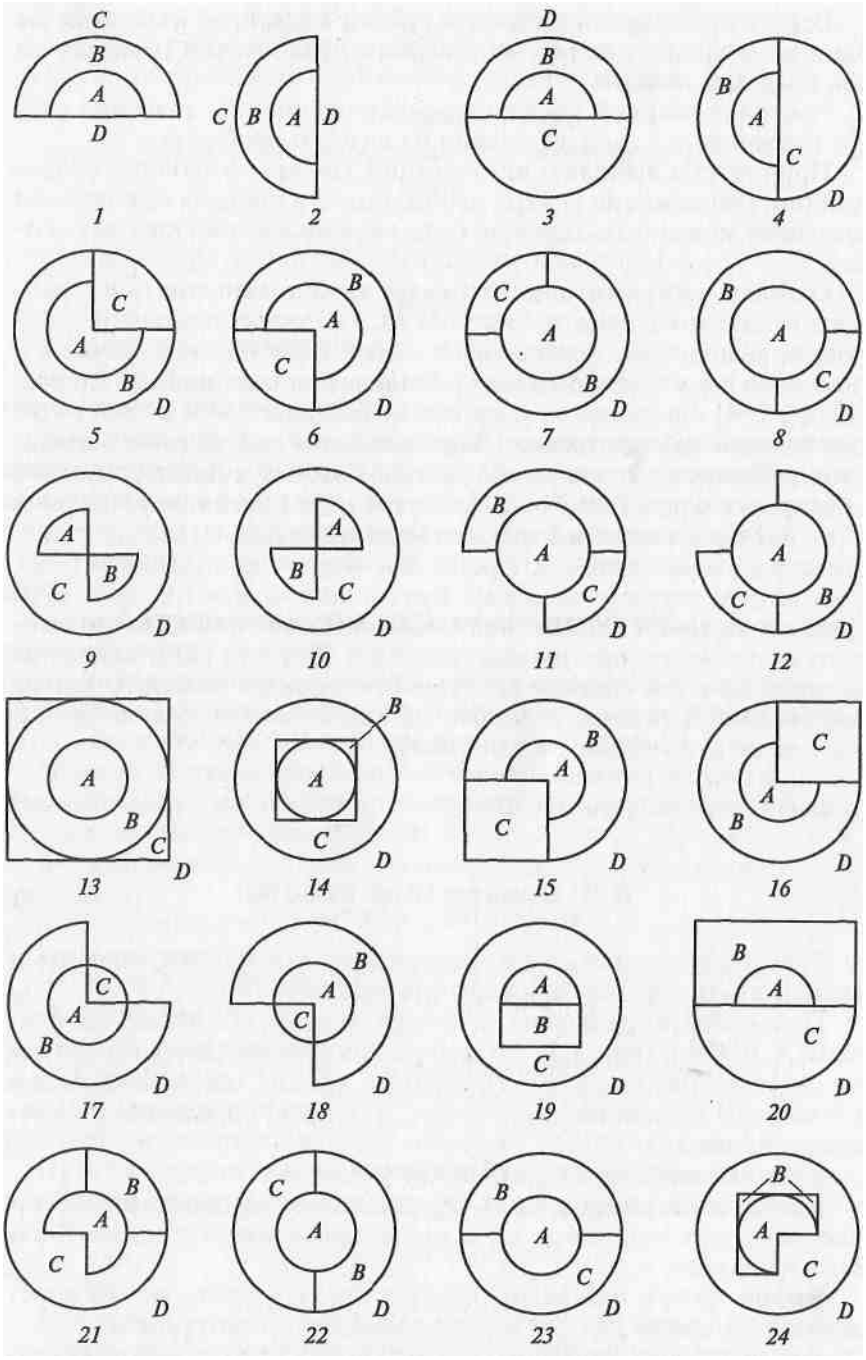

Рис. 3.3. Варианты индивидуальных заданий к Практической работе 3

процесса при наличии разветвлений, и заполнить в таблице «горячих клавиш» (см. Практическую работу 2, табл. 2.1) графы, помеченные цифрой 3.

Отчет по работе должен содержать:

формулировку задачи;

схему программы;

текст программы с комментариями;

пример результатов работы программы;

таблицу «горячих клавиш» из Практической работы 2 (см. табл. 2.1), дополненную новой информацией.

3.4. Индивидуальные задания

Схема деления плоскости *xOy* на области *A, B,* C и *D* для вариантов заданий показаны на рис. 3.3. Центры окружностей совпадают с началом координат. Радиусы внутренней и наружной окружностей принять равными 1 и 3 соответственно.

3.5. Дополнительное задание для «мастеров»

1. Дополнить программу таким образом, чтобы она отслеживала ситуации, когда точка попадает на границу между областями, и выводила на экран имена приграничных областей.

2. Дополнить программу, обеспечив ввод с клавиатуры координат еще одной точки *c.* Программа должна определить, лежит ли точка *c* на отрезке *ab,* и выдать на экран результат.

3.6. Пример программы

Условие задачи. Ввести с клавиатуры координаты точки на плоскости *X* и *Y* и определить, в каком квадранте плоскости *xOy* лежит точка.

Приведем три различных варианта решения задачи, блок- схемы которых показаны соответственно на рис. 3.4, 3.5 и 3.6.

Программа

Лабораторная работа № 3.
Определение положения точки.
Вариант А (см. рис. 3.4).
Автор Балаганов Александр, гр. 441
15.10.2006

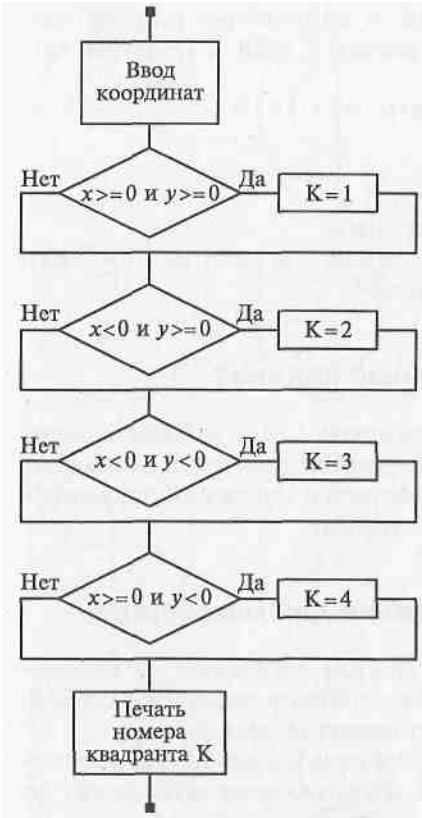

Рис. 3.4. Схема программы (вариант А) к Практической работе 3

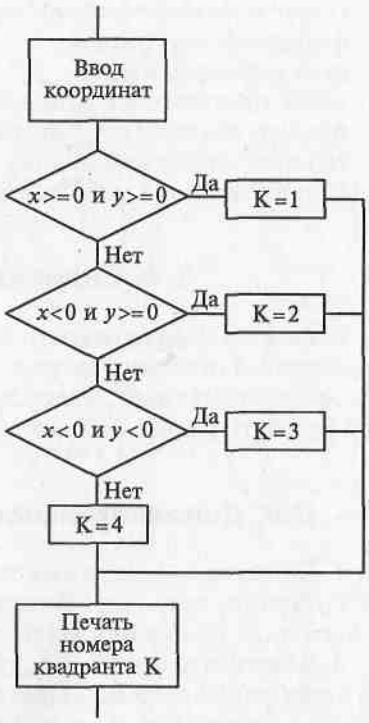

Рис. 3.5. Схема программы (вариант В) к Практической работе 3

```
# include <stdio.h>
# include <conio.h>
void main ()
{float x,y; int K; /*K — номер квадранта */.
                    / *ввод исходных данных */
 clrscr();
 print/("Введите координаты точки x и y > ");
 scanf ("%f %f",&x, &y):
              /* определение номера квадранта*/
if (x> = 0 && Y> = 0) k = 1;
if (x<0 && Y> = 0) k = 2;
if (x<0 && Y<0) k = 3;
if (x> = 0 && Y<0) k = 4;
```

```
 /*печать результата*/
 print/("\n\n Точка лежит в квадранте % i",k);
 getchQ;
}
```

Лабораторная работа № 3.
Определение положения точки.
Вариант В (см. рис. 3.5).
Автор Козлевич Адам Казимирович, гр. 441
04.09.2006

```
#include "stdio.h" ^include
"conio.h" void mainQ {double x,
y; int k; И номер квадранта
clrscrQ;
      print/("\n Введите координаты Xu Y>");
      scan/ ("%l/ %lf", &x, &y);
```

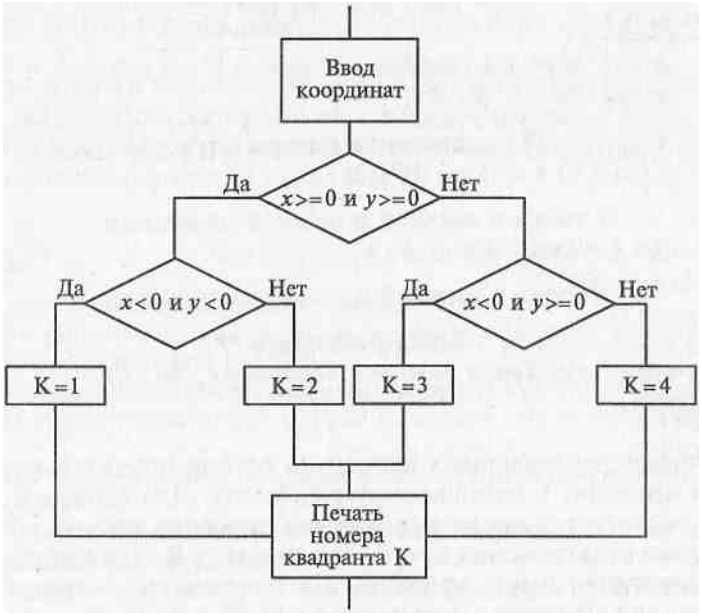

Рис. 3.6. Схема программы (вариант С) к Практической

```
/* определение номера квадранта*/
if(x> = 0 &&y> = 0) к= 1; else i/(x<0
&& y> = 0) к = 2; elsei/(x<0 && y< 0)
& = 3 ;  else k= 4;
  /*печать результата*/ print/("\n\n Точка лежит в
квадранте %i",k); getch();
}
```

Лабораторная работа № 3.
Определение положения точки.
Вариант С (см. рис. 3.6).
Автор Паниковский Михаил Самуэлевич, гр. 443
17.02.2006

```
#include <stdio.h>
#include <conio.h>
void main() {float
x,y; int k;
                    I* ввод координат */
   clrscrQ;
   printf( "Введите координаты точки X и Y>")\
   scanf("%f%f",&x,&y);
                /* определение квадранта */
   if(y>=0) k = (x>=0)?1:2;
      Ц точка в верхней половине плоскости else k
   = (x<0)?3:4:
      // точка в нижней половине плоскости
                    /* вывод результата */ print/("\n\n
   Точка лежит в квадранте %d",k); getchQ;
   }
```

Сравнение приведенных вариантов весьма поучительно, хотя все они приводят к одинаковому результату. Для варианта А результат достигается после выполнения проверки восьми условий независимо от положения точки. Для варианта В в наихудшем случае проверяются шесть условий, а в наилучшем — только два. Наконец, для варианта С результат в любой ситуации достигается после проверки всего двух условий. Таким образом, очевидно, что

вариант В предпочтительнее варианта А, а вариант С предпочтительнее вариантов А и В, так как обеспечивает наиболее экономное решение.

Этот пример показывает, как важно правильно организовать процесс ветвления в сложной ситуации.

Примечание. Обратите внимание на взаимное расположение операторов г/и *else* при записи программы в варианте В: каждый оператор *else* помещен под своим *if*.

Настоятельно рекомендуется строго придерживаться этого правила, так как оно существенно улучшает чтение (и отладку) программ, содержащих сложную иерархию условных операторов.

3.7. Типичные ошибки при выполнении работы

Операция сравнения ошибочно записывается в виде =. Следует записывать = =.

Составные операторы необходимо заключать в фигурные скобки.

3.8. Требования к студентам

В результате выполнения работы студенты должны:

знать синтаксис операторов, организующих ветвящиеся процессы;

знать и уметь использовать на практике отладочные средства среды ВС при отладке программ с разветвлением;

уметь анализировать иерархию условных операторов с точки зрения оптимальности ее организации.

Контрольные вопросы

1. Какие функции выполняет условный оператор?
2. Что такое полный и неполный условный оператор? В чем разница между ними? Нарисуйте их схемы.
3. Что такое вложенный условный оператор? Как определяется соответствие между операторами *if* и *else* во вложенных условных операторах?
4. Что такое условная операция и чем она отличается от условного оператора?
5. С помощью каких операций формируется условие в условном операторе и условной операции?
6. Что такое составной оператор и когда его следует применять внутри условного оператора?
7. Из каких компонент состоит оператор множественного ветвления?

8. Какова роль оператора *break* в операторе множественного ветвления?

9. Дайте характеристику служебного слова *default?* Как работает оператор *switch* при отсутствии оператора *default?*

10. Как выполняется отладка программ, содержащих ветвления вычислительного процесса?

11. Постройте меню, выбирающее цвет вывода текста.

12. Как отлаживать программы с ветвлениями вычислительного процесса?

Практическая работа 4

ПРОГРАММИРОВАНИЕ ЗАДАЧ С ЦИКЛАМИ

4.1. Цель работы

1. Овладеть синтаксисом различных вариантов реализации циклических процессов на языке С, научиться составлять, отлаживать и выполнять циклические программы.

2. Практически освоить средства отладки циклических вычислительных процессов, предоставляемые средой ВС.

4.2. Справочные сведения

Язык С поддерживает три варианта циклов.

1. Цикл с предусловием:

while (условие) тело цикла.

Здесь условие представляет собой любое выражение, принимающее значение 0 (ложь) или не 0 (истина), тело цикла — простой (или составной) оператор либо блок.

Вначале вычисляется условие. Если оно оказывается истинным, то выполняется тело цикла; в противном случае управление передается оператору, следующему за телом цикла. Если при входе в цикл условие ложно, то тело цикла не будет выполнено ни разу.

Блок-схема цикла с предусловием представлена на рис. 4.1, *а.*

2. Цикл с постусловием:

do тело цикла *while (условие)* отличается от цикла с предусловием тем, что здесь вначале выполняется тело цикла, затем вычисляется заданное условие. Повторное выполнение тела цикла происходит до тех пор, пока указанное условие не окажется ложным. При использовании цикла с постусловием тело цикла всегда будет выполнено по крайней мере один раз.

Блок-схема цикла с постусловием представлена на рис. 4.1, *б.*

3. Цикл *for.* Этот цикл, представляющий собой очень широкое обобщение цикла с предусловием, имеет вид

```
for (список начальных присвоений;
        условие;
        список модификаций переменных)
тело цикла
```

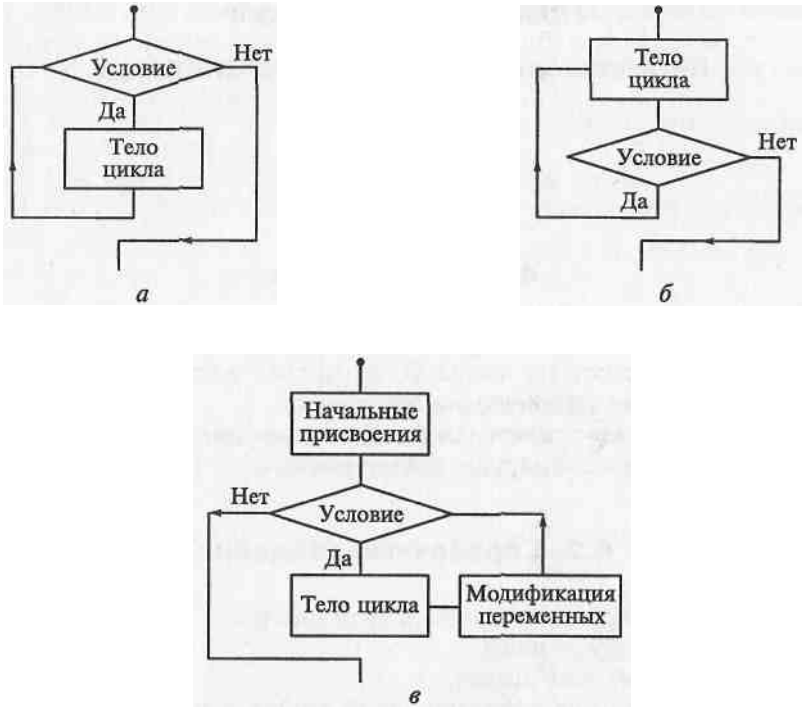

Рис. 4.1. Схемы выполнения операторов циклов: *а* — оператор *while; б* — оператор *do ... while; в* — оператор *for*

Здесь список начальных присвоений представляет собой последовательность выражений, разделенных, между собой запятыми, которые определяют значения любых переменных перед началом цикла. Список модификаций — последовательность выражений, разделенных между собой запятыми, которые определяют, как модифицируются любые переменные перед каждым повторением цикла. Значение заданного условия, которое вычисляется перед каждым выполнением тела цикла, определяет, будет ли продолжен или завершен цикл. Блок-схема цикла *for* показана на рис. 4.1, *в*.

Для любого из циклов тело цикла может, в свою очередь, представлять собой оператор цикла (того или иного вида) либо содержать такой оператор. Подобные структуры называются вложенными циклами.

Среда BC предлагает пользователям эффективные средства отладки циклических вычислительных процессов. Эти средства сосредоточены в выпадающем меню пункта *Debug -> Watches* главного меню. Оно позволяет задать, редактировать или исключить

любые переменные или выражения, значения которых будут выводиться в окно наблюдения, если останавливать выполнение программы (любыми изученными в предыдущих работах средствами) внутри тела цикла при каждом очередном его выполнении.

Аналогичных результатов можно достигнуть, используя комбинации «горячих» клавиш [Ctrl] + [F7] и [Ctrl] + [F8].

Литература: [1] с.78 —82.

4.3. Содержание работы

Работа состоит из решения двух задач.

Первая задача заключается в составлении циклической программы, которая вычисляет сумму ряда

$$S_K = \sum_{k=k_0}^{K} a_k = a_{k_0} + a_{k_0+1} + a_{k_0+2} + \ldots + a_K$$

или произведение

$$P_K = \prod_{k=k_0}^{K} a_k = a_{k_0} * a_{k_0+1} * a_{k_0+2} + \ldots + a_K.$$

Количество членов, которое необходимо учесть при вычислении суммы или произведения, в индивидуальных вариантах задания определяется одним из следующих способов:

K вводится с клавиатуры;

K определяется из условия $|a_K| < \varepsilon$;

K определяется из условия $\left|\frac{a_k}{S_k}\right| < \varepsilon$ или $\left|\frac{a_k}{P_k}\right| < \varepsilon$,

где ε — некоторое заданное малое положительное число.

Примечание. При вычислении сомножителей вида $(-1)^k$, x^k, $k!$ следует использовать «мерцающий счетчик» и метод накопления (см. пример программ Практической работы 4). Применение библиотечной функции pow *(x,y)* не рекомендуется.

Вторая задача заключается в составлении программы, содержащей вложенные циклы, которая рассчитывает и выводит на экран таблицу функции двух переменных f(*x,y)* при $x = x_0\,(\Delta x)\,x_1$ и $y = y_0\,(\Delta y)\,y_1$.

Примечание. Запись вида $z = z_0\,(\Delta z)\,z_1$ означает, что z изменяется от z_0 до z_1 с шагом Δz.

Таблица с двумя входами должна быть оформлена, как табл. 2.1. Таблицу следует представить с шапкой и делением на графы. Если некоторые из значений *f(x,y)* оказываются неопределенными (например, возникает деление на нуль или аргументы элементарных функций, входящих *в/(x,y),* выходят за допустимые пределы), то в соответствующих графах таблицы следует поместить символ «-» или «*».

В процессе отладки программ студенты должны освоить средства среды BC для отладки циклических программ и заполнить графы таблицы, помеченные цифрой 4 (см. табл. 2.1).

Отчет по работе для каждой из задач должен содержать:

формулировку задачи;

схему программы;

текст программы;

результаты работы программы;

таблицу «горячих клавиш» с внесенными в нее дополнениями.

4.4. Индивидуальные задания

Вариант 4.1.

1. Вычислить и вывести на печать сумму

$$S = \sum_{k=1}^{K} (-1)^{k-1} \frac{k^2 + k + 1}{x^k};$$

x и K ввести с клавиатуры.

2. Вывести на экран таблицу функции

$$f(x, y) = \frac{\arcsin(x + y)}{x + y}$$

при $x = 0(0,1)0,5$; $y = 0,1(0,05)0,75$.

Вариант 4.2.

1. Вычислить и вывести на экран

$$S = \sum_{k=0}^{K} \frac{(-2)^k}{k^2 + 1};$$

K ввести с клавиатуры.

2. Вывести на экран таблицу функции

$$f(x, y) = \frac{\sin(x) + \sin(y)}{x^2 + y^2}$$

при $x = 0(0,1)0,5$; $y = -1(0,2) + 1$.

Вариант 4.3.

1. Вычислить и вывести на экран

$$P = \prod_{k=1}^{K} \frac{(-1)^k * \left[k^2 + k + 1\right]}{1 * 2 * 3 * \ldots * k};$$

целое K ввести с клавиатуры.

2. Вывести на экран таблицу функции

$$f(x, y) = \frac{e^x}{x^2 + y^2 - 1}$$

при x = 0(0,25)2; y =0(0,1)1,

Вариант 4.4.

1. Вычислить и вывести на экран

$$S = \sum_{k=0}^{K} \frac{(-x)^k}{k^2 + 2} \text{ при } |x| < 1;$$

x ввести с клавиатуры, K определить из условия

$$\left|\frac{a_k}{S}\right| \le 0,001.$$

2. Вывести на экран таблицу функции

$$f(x, y) = \frac{\sin 2x + \cos 2x}{\sin x}\left(y^2 + x^2\right)$$

при x = -1(0,25)
+ 1; y = 0(0,1)2.

Вариант 4.5.

1. Вычислить и вывести на экран

$$P = \prod_{k=1}^{K} \frac{\sin(kx)}{1 * 3 * 5 * \ldots * (2k - 1)};$$

x ввести с клавиатуры, K определить из условия

$$|a_k| < 0,01.$$

2. Вывести таблицу функции

$$f(x, y) = \frac{e^x + e^{-x}}{e^y - e^{-y}} \arcsin\left(\frac{x + y}{2}\right)$$

при x = 0(0,25)2; y = -0,5(0,1) + 0,5.

Вариант 4.6.

1. Вычислить и вывести на экран

$$S = \sum_{k=0}^{K} (-1)^k \frac{k^2 - k + 1}{k^4 + k + 10};$$

K ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x, y) = \frac{\ln(x - y)}{x^2 + y^2 + x + y + 1}$$

при x= 1(1)5; *y* = 0(0,25)2.

Вариант 4.7.

1. Вычислить и вывести на экран

$$S = \sum_{k=1}^{K} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \text{ при } |x| < 1;$$

x ввести с клавиатуры, *K* определить из условия

$$\left|\frac{a_k}{S}\right| < 0,001.$$

2. Вывести таблицу функции

$$f(x, y) = \frac{x + y - 2}{x^2 + y^2 - 4} \arccos(x + y)$$

при x = -1(0,25)1; *y* = **-1(0,1)1.**

Вариант 4.8.

$$P = \prod_{k=0}^{K} (-1)^k \frac{k+1}{k^3 + k + 1} x^k \text{ при } |x| < 0,1;$$

1. Вычислить и вывести на экран

x ввести с клавиатуры, *K* определить из условия

$$|a_k| < 0,01.$$

2. Вывести $f(x, y) = \frac{x + y + 1}{\sin x \sin y} \lg\left(\frac{x + y}{2}\right)$

таблицу

функции

при *x* = -1(0,25)1; *y* = 0(0,2)2.

Вариант 4.9.

1. Вычислить и вывести на экран

$$S = \sum_{k=1}^{K} (-1)^{k+1} \frac{x^{2k-1}}{(2k-1)!} \text{ при } |x| < 1;$$

x ввести с клавиатуры, *K* определить из условия

$$|a_k| < 0,0001.$$

2. Вывести таблицу функции

$$f(x, y) = \frac{x^2 + y^2 + 2}{x^2 - y^2}$$

при *x* = -1(0,2)1; *y* = 0(1)10.

Вариант 4.10.

1. Вычислить и вывести на экран

$$P = \prod_{k=1}^{K} \frac{x^k (kx + 1)}{2(kx)^2 + 3kx + 1} \text{ при } |x| < 1;$$

Ки x ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x, y) = \frac{2x + 3y - 1}{x^2 + y^2 - xy}$$

при x = -1(0,2) + 1; *y* = -1(0,1)1.

Вариант 4.11.

1. Вычислить и вывести на экран

$$S = \sum_{k=0}^{K} (-2)^k \frac{\sin kx}{k^2 + k + 1};$$

Ки x ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x, y) = \frac{\sin x + \cos y}{\sin x - \cos y}$$

при *x* = 0(π/10)π; *y* = 0(π/10)π.

Вариант 4.12.

1. Вычислить и вывести на экран

$$S = \sum_{k=1}^{K} \frac{e^{-kx}}{k!} \text{ при } |x| < 1;$$

x ввести с клавиатуры, *K* определить из условия

$$|a_k| < 0,0001.$$

2. Вывести таблицу функции

$$f(x,y) = \frac{\operatorname{tg} x}{\sin x + \cos x}$$

при x-$0(\pi/5)2\pi$; $y = -\pi(\pi/10)\pi$.

Вариант 4.13.

1. Вычислить и вывести на экран

$$P = \prod_{k=1}^{K} \frac{x^k \lg kx}{k^2 + k + 1};$$

K и x ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x,y) = \frac{x^2 + \frac{y}{3} + 7}{x - y}$$

при $x = 0(0,5)5$; $y = -1(1)3$.

Вариант 4.14.

1. Вычислить и вывести на экран

$$S = \sum_{k=1}^{K} (-1)^k \frac{x^k}{k} \sin kx \quad \text{при } |x| < 1;$$

x ввести с клавиатуры, *K* определить из условия

$$|a_k| < 0,0001.$$

2. Вывести таблицу функции

$$f(x,y) = \frac{x + y}{\sin x + \sin y}$$

при $x = 0(\pi/10)2\pi$; $y = 0(\pi/3)2\pi$.

Вариант 4.15.

1. Вычислить и вывести на экран

$$S = \sum_{k=0}^{K} \frac{1}{10^k} \frac{x+2}{x^{k+1}} \quad \text{при } |x| < 1;$$

x ввести с клавиатуры, *K* определить из условия

$$Щ- < 0{,}0001.$$

2. Вывести таблицу функции

$$f(x,y) = \frac{x^2 + y^2}{x^3 - y^3}$$

при *x* = -1(0,2)1; *y* = -2(0,5)2.

Вариант 4.16.

1. Вычислить и вывести на экран

$$P = \prod_{k=0}^{K} \frac{e^{-kx} \sin kx}{k!};$$

K и *x* ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x,y) = \frac{x\cos x + y\cos y}{x - y}$$

при $x = 0(\pi/8)\pi;\ y = 0(\pi/16)\pi.$

Вариант 4.17.

1. Вычислить и вывести на экран

$$S = \frac{3}{4} + \sum_{k=0}^{K} \frac{(-2)^k k + \sin kx}{(k+1)!};$$

x и K ввести с клавиатуры. 2.

Вывести таблицу функции

$$f(x,y) = \frac{x\cos y + y\sin x}{x^2 - y^2}$$

при $x = -\pi(\pi/4)\pi;\ y = -2\pi(\pi/4)2\pi.$

Вариант 4.18.

1. Вычислить и вывести на экран

$$S = \sum_{k=1}^{K} \frac{1}{100^k} \frac{\sin kx + \cos kx}{(kx)^2};$$

x ввести с клавиатуры, *K* определить из условия

$$|a_k| < 0{,}0001.$$

2. Вывести таблицу функции

$$f(x,y)=\frac{x\ln y+y\lg x}{x^3-y^3}$$

при x= 1(0,1)2; *y* = -2(0,5)2.

Вариант 4.19.

1. Вычислить и вывести на экран

$$S=\sum_{k=0}^{K}\frac{1}{2^k}\frac{\sin x}{x^{k+1}}\ \text{при}\ |x|<1;$$

x ввести с клавиатуры, *K* определить из условия

$$\left|\frac{a_k}{S}\right|<0,0001.$$

2. Вывести таблицу функции

$$f(x,y)=\frac{2x^3+3y^3}{3x^2-2y^2}$$

прих= 1(0,1)2; *y* = -2(0,5)2.

Вариант 4.20.

1. Вычислить и вывести на экран

$$S=\sum_{k=0}^{K}\left(\frac{1}{2^k}+\frac{1+kx}{1+kx^2}\right)+\frac{k}{2};$$

x и *K* ввести с клавиатуры. 2.
Вывести таблицу функции

$$f(x,y)=\frac{x\sin y+y\cos x}{x^3-y^3}$$

при *x* =–π(π/4)π; *y* = –2π(π/4)2π.

Вариант 4.21.

1. Вычислить и вывести на экран

$$S=\sum_{k=1}^{K}\frac{e^{-kx/2}}{k!}\ \text{при}\ |x|<1;$$

X ввести с

клавиатуры, *K* определить из условия

$$|a_k|<0,001.$$

2. Вывести таблицу функции

$$f(x,y)=\sqrt{\frac{x+1}{y+3/2}}$$

при x = 0(0,1)1; y = -3(0,5)3.

Вариант 4.22.

1. Вычислить и вывести на экран

x ввести с

$$S=\sum_{k=1}^{K}\frac{(x-1)^k}{kx^k} \text{ при } x>\frac{1}{2};$$

клавиатуры, K определить из условия

$$|a_k|<0,001.$$

2. Вывести таблицу функции

$$f(x,y)=\frac{x^2+x+1}{y^2+y+1}$$

при x = 0(0,2)2; y = -5(1)5.

Вариант 4.23.

1. Вычислить и вывести на экран

$$S=\frac{3}{5}+\sum_{k=1}^{K}\frac{k^2x^2-kx+2}{kx} \text{ при } |x|<1;$$

хя К ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x,y)=\sqrt[3]{\frac{x+y}{xy}}$$

при x= 0(0,1)1; ^= 1(0,1)2,5.

Вариант 4.24.

1. Вычислить и вывести на экран

$$P=\prod_{k=1}^{K}\left(\frac{1}{k}+\frac{k+1}{k+2}\right);$$

K ввести с клавиатуры.

2. Вывести таблицу функции

$$f(x,y)=\sqrt[4]{\frac{\sin(xy)}{x^2+y^2}}$$

при x = 0(0,1)1; *y* = -3(0,5)3.

4.5. Дополнительное задание для «мастеров»

Если при вычислении суммы ряда число членов вводится с клавиатуры, дать оценку точности вычисления суммы, в остальных случаях вывести на экран таблицу зависимости результата от заданной точности.

4.6. Пример программы

Условие задачи. Вычислить и вывести на экран

$$S=\sum_{k=0}^{K}\frac{(-1)^k}{(k+1)!};$$

K определить из условия $|a_k|<0,0001.$

Программа

```
                Лабораторная работа № 4.
                Вычисление суммы ряда.
                Автор Берлага Фома, гр. 442
                01.01.2006
        = = = = = = = = = = = = = = = = = = = = = = = = = */

# include <stdio.h>
# include <conio.h>
# include <math.h>
main ()
{int k, mult; float a, S,f,eps =
  0,0001;
                        /* к — счетчик членов ряда;
                          mult — так называемый мерцающий счетчик,
                        попеременно принимающий
                        значения +1 и -1;
                        /— значение 1/факториал;
                        а — k-й член ряда;
                        eps — заданная погрешность
                        */
```

```
for(S= 0, mult = ~l,f= 1, a= 1, k= 0;fabs(a)>= eps; k++) {
  mult * = -1; // модификация мерцающего счетчика //=
  ((double)k+1); // накопление значения
                 // 1 /факториал;
   a = mulf-f; // к-й член ряда;
   S+ = a;          И накопление суммы;
  }

 /* вывод результата */
clrscr (); gotoxy (10,3);
print/(" Сумма ряда равна % 8.3/', S); getch
(); }
```

4.7. Типичные ошибки при выполнении работы

Результат деления целого числа на целое есть целое число. Для получения правильного результата необходимо явным образом с помощью операции (тип) преобразовать один или оба операнда операции деления к типу *float* или *double.*

Внимание. Компилятор такую ошибку не перехватывает!

Библиотечная функция *abs* имеет аргумент типа *int.* Использование ее с аргументом вещественного типа приводит к потере дробной части результата.

Для получения правильного результата необходимо использовать функцию *fobs,* аргумент и возвращаемое значение которой имеют тип *double.*

Счетчик цикла должен быть целым числом. Вещественные числа представляются в памяти компьютера с погрешностью, поэтому при использовании в качестве счетчика вещественной переменной число повторений цикла может на единицу отличаться от планируемого.

При вычислении *к-го* члена ряда величину *к* целесообразно преобразовать к действительному типу, что обеспечит возможность суммировать ряды с числом членов, превышающим предельное для типа *int* значение 32767.

4.8. Требования к студентам

В результате выполнения работы студенты должны:

знать синтаксис и характер функционирования всех вариантов циклов в языке С. Уметь составлять и отлаживать циклические программы в среде ВС;

владеть средствами отладки циклических программ в среде ВС; владеть средствами форматного вывода информации и оформления таблиц.

Контрольные вопросы

1. Чем отличаются циклы с предусловием от циклов с постусловием?
2. Опишите последовательность функционирования циклов *while* и *do while*. Какой из этих циклов является циклом с предусловием, а какой — с постусловием?
2. Опишите схему работы оператора цикла *for*. Является ли цикл *for* циклом с предусловием или циклом с постусловием?
3. Что такое неполный цикл *for?*
4. Как более прозрачно может быть записан цикл *for* (; *i*< 10;)?
5. Сколько раз символ «;» (точка с запятой) должен обязательно содержаться внутри заголовка неполного оператора *for?* полного?
6. Как будет работать цикл с заголовком *for* (/= *1;* ; /++)?
7. Для какого из двух циклов

while(l) /++;

while(O) йн-тело цикла не будет выполнено ни одного раза? Как будет работать второй из этих циклов, если /' имеет тип *float?* тип *int?*

8. Цикл

for (i= Г, i<W. /++); не содержит тела. Будет ли работать такой цикл? Что получится в результате?

9. Какие циклы называются вложенными? Какую глубину вложения циклов допускает язык С?
10. Что такое мерцающий счетчик? Для чего он применяется?

Практическая работа 5

ФУНКЦИИ В ЯЗЫКЕ С

5.1. Цель работы

1. Овладеть синтаксисом написания функций и их прототипов, методикой составления и отладки программ, содержащих функции, спецификой передачи параметров в функцию и возврата полученных результатов.

2. Научиться программировать на языке С типовые задачи вычислительной математики (численное интегрирование, решение алгебраических и дифференциальных уравнений).

5.2. Справочные сведения

Функции в языке С записываются в форме: Заголовок функции Тело функции Заголовок функции представляет собой конструкцию следующего вида:

[тип возвращаемого значения]

имя функции (список формальных параметров), где имя функции — уникальный идентификатор, определяемый пользователем; список формальных параметров — перечень параметров, передаваемых в функцию, с указанием их типов.

Тип возвращаемого значения — необязательный компонент заголовка, по умолчанию принимаемый как *int.* Если функция ничего не возвращает в вызывающую программу, то в качестве типа возвращаемого значения задается *void.*

Тело функции представляет собой блок, содержащий описание локальных переменных и последовательность выполняемых операторов. Результат работы функции возвращается с помощью оператора возврата:

return возвращаемое значение; Допускается несколько операторов *return* в одной функции. Выход из функции осуществляется при достижении любого из операторов *return* или после достижения конца тела функции. Для функции типа *void* оператор возврата имеет вид

return;

Такая функция может вообще не содержать операторов возврата. В этом случае работа функции завершается после выполнения последнего оператора.

Параметры могут передаваться в функцию по имени (значению) или по адресу (указателю). В первом случае значения переменных, переданных в функцию, после выхода из нее сохраняют свои исходные значения несмотря на то, что внутри функции они были изменены. Во втором случае значения переменных, указатели на которые переданы в функцию, могут быть изменены внутри функции и сохраняют новые значения после выхода из функции.

Все функции, входящие в программу, записываются последовательно друг за другом, включая главную функцию, имеющую имя *main*. Эта функция может занимать любую позицию в последовательности функций, однако она всегда выполняется первой.

Любая из функций программы доступна для всех функций. Вызов осуществляется в виде

имя функции {список фактических параметров), где в качестве фактических параметров могут задаваться любые переменные или выражения, определенные в вызывающей функции.

Рекомендуется в вызывающей функции описать вызываемую функцию, задав в разделе описаний прототип последней. Прототип совпадает по форме с заголовком функции, за которым вместо тела следует точка с запятой.

В целях практического освоения программирования функций студентам предлагается составить программы, реализующие типовые алгоритмы вычислительной математики. Далее приводится краткое описание этих алгоритмов.

5.2.1. Решение трансцендентного уравнения методом последовательных приближений

Уравнение приводится к виду

$$x = f(x). \quad (5.1)$$

Задается некоторое начальное приближение для корня x_0 и строится итерационный процесс уточнения значения корня:

$$x_{i+1} = f(x_i),\ i = 0,1,2\ldots \quad (5.2)$$

Процесс завершается при выполнении условия

$$|x_{i+1} - x_i| < \varepsilon, \quad (5.3)$$

где е — заданная допустимая погрешность.

Доказывается, что итерационный процесс (5.2) сходится к решению уравнения (5.1) при выполнении в окрестности корня условия

$$\left|\frac{df(x)}{dx}\right| < 1, \qquad \textbf{(5.4)}$$

которое необходимо учитывать при привидении уравнения к виду (5.1). Например, для уравнения

$$x\mathrm{tg}x = a$$

итерации расходятся, если уравнение представить в виде

$$x = \mathrm{actg}\, x,$$

и сходятся в случае

$$x = \mathrm{arctg}\frac{a}{x}.$$

5.2.2. Численное интегрирование методом Симпсона

Для вычисления интеграла

$$I = \int_a^b f(x)dx \qquad \textbf{(5.5)}$$

отрезок интегрирования *ab* делится на *n* равных частей (и — четное) и используется квадратурная формула

$$I = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \ldots + y_n), \qquad (5.6)$$

где $h = \frac{b-a}{n}$, $y_i = f(x_i) = f(a+ih)$.

5.2.3. Численное интегрирование дифференциального уравнения методом Рунге—Кутта

Для численного интегрирования дифференциального уравнения

$$\frac{dy}{dx} = f(x,y), \quad x_0 \le x \le X \qquad (5.7)$$

с начальным условием

$y(x = x_0) = y_0$ (5.8) выбирается некоторый достаточно малый шаг интегрирования

$$h = \frac{X - x_0}{n}, \tag{5.9}$$

после чего приближенное значение искомой функции y в точках

$$x_i = x_0 + h * i, \ (i = 1, 2, \ldots n)$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{5.10}$$

последователь

но вычисляется по формуле

где

$$k_1 = hf(x_i, y_i);$$
$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right);$$
$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right);$$
$$k_4 = hf(x_i + h, y_i + k_3).$$

Литература: [1] с. 97—108.

Отчет по работе должен содержать: формулировку задачи; описание метода; программу решения; результаты работы программы.

5.3. Содержание работы для студентов специальности 2203

Требуется составить, отладить и выполнить в среде ВС программы решения задач, сформулированных в индивидуальном задании.

При решении трансцендентного уравнения следует привести уравнение к виду (5.1) и написать две отдельные функции, первая из которых вычисляет правую часть уравнения (5.1), а вторая

реализует итерационный процесс (5.2). В главной программе следует ввести с клавиатуры погрешность вычисления корня e, вычислить корень и проверить результат путем подстановки его в исходное уравнение. Вывести на экран полученное значение корня и погрешность.

При численном интегрировании следует написать две функции, первая из которых вычисляет подынтегральное выражение, а вторая реализует вычисление интеграла по формуле (5.6) в зависимости от заданного числа *n*.

В главной программе обеспечить вычисление и вывод на экран таблицы результатов интегрирования в зависимости от *n* при *n* = 100 (50) 300.

При решении дифференциального уравнения методом Рунге—Кутта написать две функции, первая из которых вычисляет правую часть уравнения (5.7), а вторая реализует интегрирование по формуле (5.10) и печать результатов после каждого шага в виде таблицы *y -f(x)*. В главной программе обеспечить печать шапки таблицы и вызов функции, реализующей процесс интегрирования.

5.4. Индивидуальные задания для студентов специальности 2203

Вариант 5.1.

1. Вычислить интеграл вероятности

$$\Phi = \sqrt{\frac{2}{\pi}} \int_0^1 \mathrm{e}^{-0,5x^2}\,\mathrm{d}x.$$

2. Решить трансцендентное уравнение

$$x\mathrm{e}^{x^2} = 1.$$

Вариант 5.2.

1. Проинтегрировать дифференциальное уравнение

$$y' = \sin x + \cos y$$

на отрезке $0 \le x \le \pi$ при *y(0)* = 0.

2. Вычислить интеграл

$$I = \int_5^{10} \frac{\mathrm{d}x}{x \ln x}.$$

Вариант 5.3.

1. Решить алгебраическое уравнение

$$x^3 - x - 1 = 0.$$

2. Проинтегрировать

дифференциальное уравнение

$$y' = ye^{-x}$$

на отрезке $0 \le x \le 2$ при $y(0)$ - 1.

Вариант 5.4.

1. Вычислить интеграл

$$I = \int_0^1 \frac{\arcsin x}{(x+1)^2}\,dx.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \sin x \cos y + xy$$

на отрезке $0 \le x \le \pi$ при $y(0) = 0$.

Вариант 5.5.

1. Проинтегрировать дифференциальное уравнение

$$y' = x^2 + y^2$$

на отрезке $-1 \le x \le 1$ при y(-1) = 0.

2. Решить уравнение

$$x(x^2 + 4) = 8.$$

Вариант 5.6.

1. Вычислить интеграл

$$I = \int_0^1 \frac{xe^x}{(1+x^2)}\,dx.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \sin x \cos y$$

на отрезке $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$ при $y\left(-\frac{\pi}{2}\right) = -1.$

Вариант 5.7.

1. Решить алгебраическое уравнение

$$x(1 + x^2) = 1.$$

2. Вычислить интеграл

$$I = \int_1^2 \frac{dx}{(1+x+x^2)\sqrt{1+x+x^2}}.$$

Вариант 5.8.

1. Вычислить интеграл

$$I = \int_1^2 \frac{\sin x}{x} dx.$$

2. Проинтегрировать дифференциальное уравнение

$$-\frac{\pi}{4} \le x \le \frac{\pi}{4} \text{ при } y\left(-\frac{\pi}{4}\right) = 0.$$

на отрезке

Вариант 5.9.

1. Вычислить интеграл

$$I = \int_0^1 \frac{\sin x \cos x}{x^2+x+1} dx.$$

2. Решить алгебраическое уравнение

$$x^2(x+1) = 1.$$

Вариант 5.10.

1. Решить трансцендентное уравнение

$$\sin x + \cos x - x = 0.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \frac{x^2+1}{\sqrt{x^3+x+1}}$$

на отрезке $0 \le x \le 1$ при $y(0) = 1$.

Вариант 5.1,1.

1. Проинтегрировать дифференциальное уравнение

$$y' = y + \sin x$$

на отрезке $0 \le x \le \frac{\pi}{2}$ при $y(0) = 0$.

2. Вычислить интеграл

$$I = \int_0^2 \frac{e^x + e^{-x}}{x^2 + 1} dx.$$

Вариант 5.12.

1. Решить трансцендентное уравнение

$$\frac{x-1}{\sin x} = 1.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \sin y$$

на отрезке $0 < x < 1$ при $y(0) = 0{,}5$.

Вариант 5.13.

1. Вычислить интеграл

$$I = \int_1^5 \frac{\lg x}{\sqrt{x+1}} dx.$$

2. Решить трансцендентное уравнение

$$\frac{x^2 - 1}{\sqrt{x^2 + x + 1}} = 1.$$

Вариант 5.14.

1. Вычислить интеграл

$$I = \int_0^{\frac{\pi}{2}} \frac{\sin x \cos x}{2 + \sin^2 x - \cos^2 x} dx.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \frac{x}{\sqrt[3]{x+y}}$$

на отрезке $0 \le x \le 1$ при $y(0) = 1$.

Вариант 5.15.

1. Решить трансцендентное уравнение

$$\frac{x-1}{\cos x} = 1.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \cos(x+1)\operatorname{tg}x + y$$

на отрезке $0 \le x \le 1$ при y(0) = 0.

Вариант 5.16.

1. Вычислить интеграл

$$I = \int_0^{\frac{\pi}{4}} \frac{\operatorname{tg} x}{\sqrt{x+1}}\,dx.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \sqrt{y^2 + y - 1}$$

на отрезке $0 \le x \le 2$ при $y(0) = 1$.

Вариант 5.17.

1. Решить трансцендентное уравнение

$$x^2 - \sqrt[3]{x+4} = 0.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = x + \sqrt{y}$$

на отрезке $-1 \le x \le 1$ при $y(-1) = 1$.

Вариант 5.18.

1. Вычислить интеграл

$$I = \int_0^1 \frac{e^x \sin x}{x+1}\,dx.$$

2. Решить трансцендентное уравнение

$$x^3 - \cos x - 1 = 0.$$

Вариант 5.19.

1. Решить трансцендентное уравнение

$$\sqrt[3]{x^2 + 3x + 1} - x + 1 = 0.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = \frac{x^3 + 2}{\sqrt{2x^2 + x + 1}}$$

на отрезке $0 \le x \le 1$ при $y(0) = 1$.

Вариант 5.20.

1. Проинтегрировать дифференциальное уравнение

$$y' = y + 2\cos x^2$$

на отрезкепри $y(0) = 0$.

$$0 \le x \le \frac{\pi}{2}$$

2. Вычислить интеграл

$$I = \int_0^2 \frac{\sqrt[3]{\sin(x-1)}}{2x^3+5}\,dx.$$

Вариант 5.21.

1. Вычислить интеграл

$$I = \int_0^1 \frac{\sin x \operatorname{tg} x}{\sqrt{x+1}}\,dx.$$

2. Решить уравнение

$$3\sin^2(x) = x.$$

Вариант 5.22.

1. Вычислить интеграл

$$I = \int_0^1 \frac{x+\sin x}{x+\cos x}\,dx.$$

2. Решить трансцендентное уравнение

$$x^3 - \sin x - 1 = 0.$$

Вариант 5.23.

1. Проинтегрировать дифференциальное уравнение

$$y' = \sqrt{y} + 2\sin x^2$$

на отрезке при $y(0) = 0$.

$$0 \le x \le \frac{\pi}{2}$$

2. Вычислить интеграл

$$I = \int_0^2 \frac{\sqrt[3]{\cos(x+1)}}{3x^3 + \frac{x}{4}}\,dx.$$

Вариант 5.24.

1. Решить трансцендентное уравнение

$$x\cos x = x + \frac{1}{3}.$$

2. Проинтегрировать дифференциальное уравнение

$$y' = xy + 2$$

на отрезке $0 \le x \le 1$ при $y(0) = 0{,}5$.

5.5. Пример программы для студентов специальности 2203

Условие задачи. Методом трапеций вычислить интеграл

$$I = \int_0^1 x\cos x dx.$$

Квадратурная формула трапеций имеет вид

$$I = h\left(\frac{y_0}{2} + y_1 + y_2 + \ldots + y_{n-1} + \frac{y_n}{2}\right).$$

Программа

Лабораторная работа № 5.
Численное интегрирование методом трапеций.
Вариант А.
Автор Синицкая Зося Викторовна, гр. 445
20.01.2006

```
# include <stdio.h>
# include <conio.h>
# include <math.h>

        1*главная процедура*I
\oid mainQ
{float a, b,I;                          // описание переменных
int n = 10;
float integral (float a, float b, int n);       // прототип функции
                                          // вычисления интеграла
```

```
clrscr ();
/= integral (0, 1, n); printf("\n Значение интеграла равно %8.4/", I);
getch ();
}

// ------------------------------------------------------
/* функция, вычисляющая интеграл */
float integral (float a, float b, int ri)
 {float S, h;
 int i;
float f (float x);          //прототип функции
 // вычисления подинтегрального // выражения h = (b -
 a)/n;
 S = (f(a) +f(b))/2;                    // сумма крайних членов ряда
for (i= 1; / < n; г ++)
     S + =f(a + h*i);                   Ц вычисление ряда
 return h*S; }
 // ---------------------------------------------------------------------------
 /* функция, вычисляющая подынтегральное
            выражение */ float
f (float x)
     { return x*cos (x);
     }
```

5.6. Содержание работы для студентов специальностей 2201, 2202

Требуется разработать функцию, которая выводит на экран строку из N звездочек заданного цвета в позицию экрана с координатами x_0, y_0

С помощью указанной функции разработать функцию, которая выводит на экран фигуру, указанную в индивидуальном задании (рис. 5.1).

Цвет звездочек, формирующих фигуру, и координаты ее положения на экране ввести с клавиатуры.

5.7. Дополнительное задание для «мастеров» специальностей 2201, 2202

Написать функции для вывода на экран нескольких фигур (букв), с помощью которых вывести на экран целое слово.

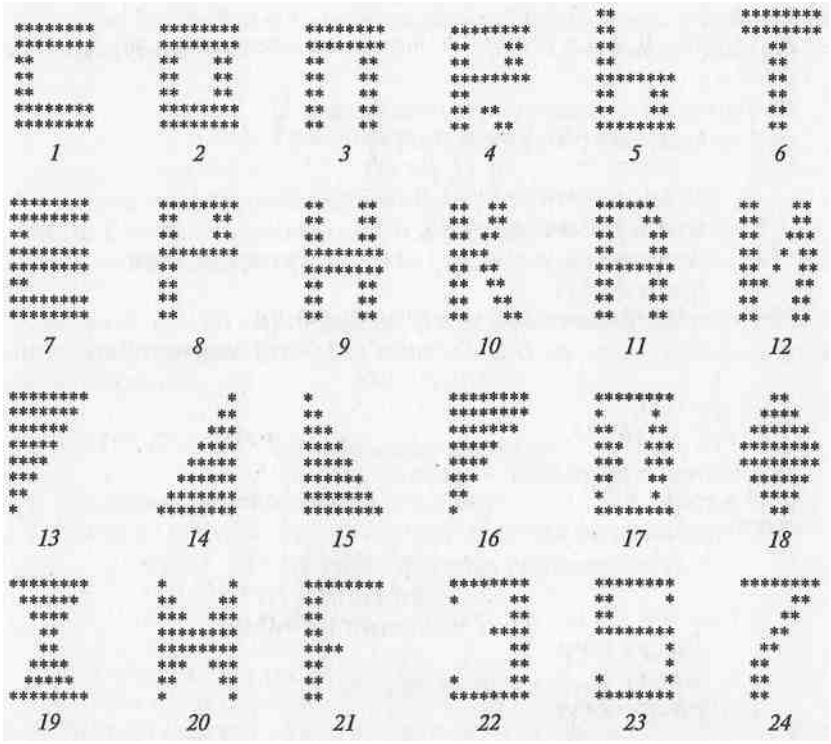

Рис. 5.1. Варианты индивидуальных заданий к Практической работе 5

5.8. Пример программы для студентов специальностей 2201, 2202

Условие задачи. Написать функцию, которая выводит на экран прямоугольник заданного цвета с координатами верхнего левого угла x_0 y_0 и координатами нижнего правого угла $x_ь$ y_1 В главной программе с помощью созданной функции вывести на экран фигуру в форме буквы Г. Цвет и положение верхнего левого угла буквы ввести с клавиатуры.

Программа

```
                Лабораторная работа № 5. Вывод на
экран фигуры в форме буквы Г.
                Вариант В.
Автор Полесов Виктор Михайлович, гр. 321
                5.11.2006
                                                /
```

```
Шпсшае <stdio.h>
#include <conio.h>

/* Функция — рисует прямоугольник */
void rectangleiint color, int xO,int yO, intxl,
intyl) {int y; textbackground(color);
        for (y = yO; y < — yl; y++) // счетчик строк
        (gotoxy(xO,y); while (wherexQ < = xl) cprintfi" ");
                                                    //вывод одной строки
      }}

/* Главная программа — рисует фигуру */ void
main()
      {int color, xO,yO; void rectangleiint color,
        int xOJnt yO, intxl, intyl); И прототип
        функции clrscii);
        printfi" Введите цвет фигуры > ");
        scanf{"%i",&color); printf( "\n Введите
        координаты верхнего "
              " левого угла фигуры > ");
        scanfi'Voi %i", &x0, &yO); clrscti);
        rectangleicolor, xO, y0, xO + 20, y0+ 5);
    Ц размеры длины и ширины //перекладины
    приняты // соответственно 20 и 5
    rectangleicolor,xO,yO + 6, x0+ 5, y0+ 10); II
    размеры ножки — 5 и 10 getchi); }
```

5.9. Типичные ошибки при выполнении работы

Если в последовательности функций вызываемая функция расположена после вызывающей, то последняя обязательно должна содержать прототип вызываемой функции. Ошибка, связанная с нарушением этого правила перехватывается компилятором и, следовательно, не является фатальной.

Задание прототипов для всех вызовов функций — хороший стиль программирования.

5.10. Требования к студентам

В результате выполнения работы студенты должны:

знать синтаксис написания и вызов функций на языке С;

уметь составлять и отлаживать циклические программы в среде ВС;

уметь отлаживать программы, содержащие несколько функций;

знать алгоритмы решения перечисленных задач вычислительной математики.

Контрольные вопросы

1. Как формируется заголовок функции?
2. Какие из приведенных заголовков функций содержат ошибки:

*voidfl {int i, int *j, double s)*
voidf2{int i,j, double a,b, char c)
void {int i, int k,) intf4(int i,
char c)

3. Что представляет собой тело функции?
4. В каких случаях оператор *return* задается без параметров?
5. Сколько величин можно возвратить из функции с помощью оператора *return!*
6. Может ли функция содержать оператор *return* более одного раза?
5. Зачем нужен прототип функции? В каких случаях прототип не является обязательным?
6. Чем отличается синтаксис написания прототипа функции от синтаксиса ее заголовка?

Практическая работа 6

МНОГОМОДУЛЬНЫЙ ПРОГРАММНЫЙ ПРОЕКТ

6.1. Цель работы

1. Изучить возможные способы организации больших программ (программных проектов), исходный текст которых включает в себя ряд функций, расположенных в отдельных файлах.

2. Практически освоить технику сборки программных проектов на уровне исходных и объектных модулей и технику отладки программных проектов.

6.2. Справочные сведения

Программирование больших задач целесообразно разбить на серию отдельных более мелких и простых подзадач, каждая из которых реализуется в виде отдельной функции, получающей в качестве входной информации только те данные, которые необходимы для решения конкретной подзадачи, и возвращающей в вызывающую функцию результаты своей работы.

Программу, в которой реализован указанный подход, называют *программным проектом.* Обычно программный проект представляет собой иерархию функций, каждая из которых может вызывать любую другую функцию проекта, необходимую для ее работы.

Программные проекты весьма привлекательны с точки зрения экономии средств и времени, затрачиваемых на их разработку.

Действительно, при хорошо спланированном проекте каждая функция или группа связанных между собой функций могут разрабатываться, отлаживаться и тестироваться независимо от других, что, в частности, дает возможность параллельно работать над проектом целой бригаде программистов, каждый из которых может целиком сосредоточиться на решении своей проблемы. Весь проект в целом собирается, отлаживается и тестируется только после завершения разработки и отладки всех подзадач.

При параллельной разработке каждую функцию или группу функций размещают в отдельном файле. Это удобно с точки зрения возможности использовать один раз отлаженную и протестированную функцию в нескольких различных проектах.

Существуют различные технологии сборки файлов (модулей) в проект.

Сборка на уровне исходных модулей. Это наиболее простой способ. Он заключается в подключении каждого модуля проекта к главной функции программы с помощью инструкции препроцессора

#include < имя файла>

В данном случае в оперативной памяти компьютера перед началом компиляции каждая из указанных инструкций автоматически замещается текстом соответствующего файла, т.е. формируется исходный текст программы, где все составляющие его функции следуют друг за другом в порядке их сборки. Компилятор компилирует этот текст, образуя единственный объектный модуль (с расширением *.obj),* который обрабатывается редактором связей, в результате чего создается готовый к работе выполняемый модуль (с расширением *.exё).*

Необходимым условием успешного преобразования объектного модуля в выполняемый является наличие в каждой функции, вызывающей другую функцию, прототипа последней, если в исходном модуле вызываемая программа расположена после вызывающей. Главный недостаток сборки проекта на уровне исходных модулей заключается в необходимости полной перекомпиляции всех функций проекта после исправления ошибки только в одной или нескольких функциях, что в случае большого проекта может потребовать значительного времени.

Сборка проекта на уровне объектных модулей. Это более изощренная технология сборки, которая заключается в использовании так называемого файла проекта (с расширением *.prj),* в который следует включить список всех файлов проекта. Файл проекта рекомендуется формировать с помощью пункта меню интегрированной системы *Project.* Подпункт этого меню *Open project* позволяет создать новый или открыть существующий файл проекта. Подпункты *Add item...* и *Delete item...* обеспечивают соответственно внесение в проект и удаление из него заданного файла.

После внесения в проект всех файлов, содержащих тексты функций, файл проекта может быть откомпилирован («горячая» клавиша [F9]) либо откомпилирован и выполнен (комбинация «горячих» клавиш [Ctrl] + [F9]); при этом в результате компиляции для каждого исходного файла формируется свой собственный объектный файл, количество которых совпадает с количеством исходных файлов. Редактор связей собирает все объектные модули в общий выполняемый файл, который является единственным.

Достоинством сборки проекта на уровне объектных модулей является то, что при корректировании одного или нескольких исходных файлов только эти файлы подвергаются перекомпилированию, что существенно сокращает полное время компиляции проекта. Необходимым условием успешного завершения процесса компиляции является наличие прототипов вызываемых функций в каждой функции, вызывающей другие функции.

6.3. Содержание работы

Требуется составить отладить и выполнить программу, которая вычисляет налог на предлагаемый в индивидуальном задании участок, содержащий земельные угодья и пруды. Программу следует разработать в форме проекта.

Для выполнения работы необходимо:

разделить предлагаемый участок на фрагменты простой формы (прямоугольник, треугольник, трапеция);

составить функции, вычисляющие площади необходимых фигур, поместив каждую из этих функций в отдельный файл;

отладить и протестировать каждую из функций, создав для этого специальные тестирующие программы, в которые тестируемые функции включаются с помощью инструкции *include;*

определить набор исходных данных, необходимых для вычисления всех площадей;

написать главную программу, которая вводит с клавиатуры все необходимые размеры участка и ставки налога на землю и пруды, р./м^2, с помощью разработанных и отлаженных функций вычисляет суммарную площадь земли и воды на участке и выводит на экран сумму причитающегося за пользование участка налога;

с помощью интегрированной системы создать файл проекта, откомпилировать его и решить задачу. Проверить результат, рассчитав налог вручную, используя для этого встроенный в систему калькулятор (комбинация клавиш [Ctrl] + [F4]).

Литература: [1], с. 27 —28.

Отчет по работе должен содержать:
формулировку задачи, схему участка и принятые обозначения; тексты всех функций, входящих в проект; тексты тестовых программ, проверяющих работу каждой из функций, и результаты тестирования; текст главной программы; состав файла проекта; результаты работы программы.

6.4. Индивидуальные задания

Варианты схем участков представлены на рис. 6.1. Части участков, занятые прудами, на схемах залиты черным цветом. Ставки налогов на землю и воду соответственно равны Hland и Hwater.

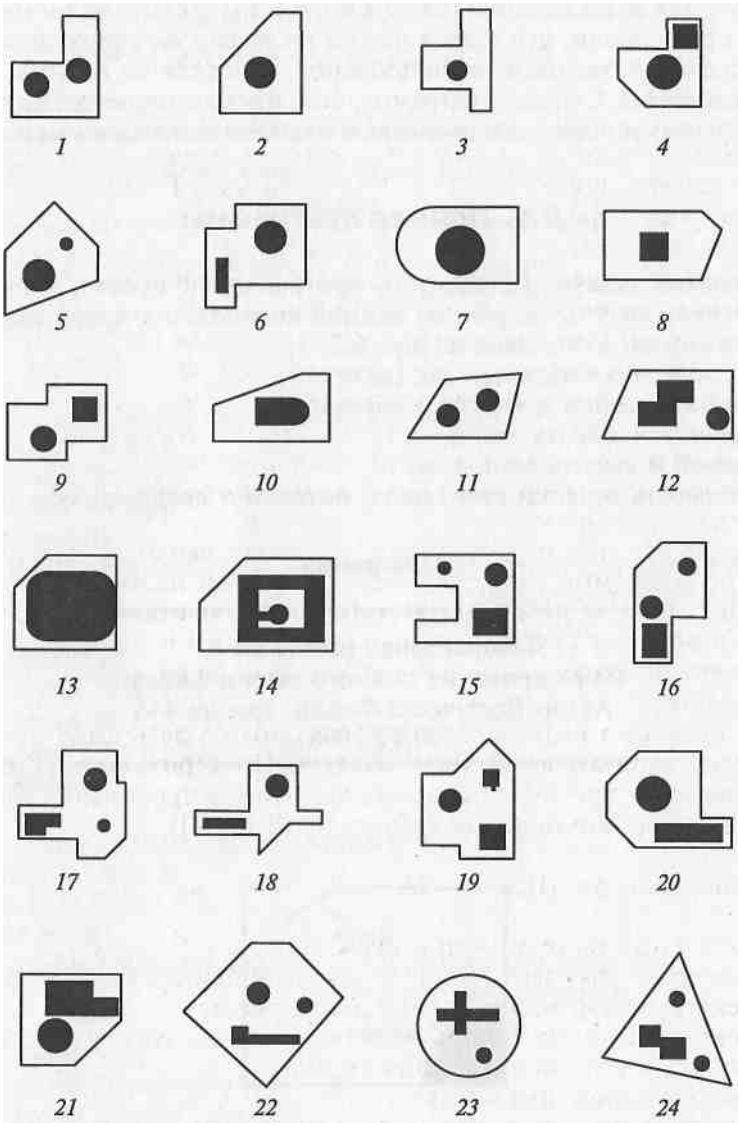

Рис. 6.1. Варианты индивидуальных заданий к Практической работе 6

6.5. Дополнительное задание для «мастеров»

Задание 6.1. Дополнить программу функцией, которая получает в качестве параметров ставки налогов и выводит таблицу зависимости суммы налога от налоговых ставок при уменьшении и увеличении каждой из ставок налога на 20, 15, 10 и 5%.

Задание 6.2. Разделить землю на два сорта: пахотную и не пригодную для возделывания (овраги и т.п.). Вычислить налог на участок при условии, что ставка налога на землю, не пригодную для сельскохозяйственного использования, понижается на 40%.

Задание 6.3. Собрать программу, пользуясь двумя методами сборки. Сформулировать достоинства и недостатки каждого метода.

6.6. Пример программы

Условие задачи. Разработать программный проект, который вычисляет затраты на ремонт ванной комнаты с угловой ванной. Схема комнаты показана на рис. 6.2.

Данные, необходимые для расчета:
длина, ширина и высота комнаты, м;
ширина и высота двери, м;
радиус и высота ванны, м;
стоимость отделки стен, пола, потолка и двери, р./м^2.

Программа

Лабораторная работа № 6.
Формирование полного имени файла.
Автор Востриков Федор, группа 445
31.12.2006

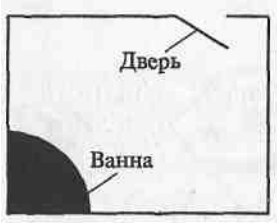

Рис. 6.2. Схема ванной комнаты для примера программы к Практической работе б

Модули проекта

```
//..... файл FO.cpp .....

/*        Функция расчета площади круга        */
/*                по его диаметру                */

double F0(double D) // D — Диаметр круга
        {const double pi = 3.141593; return
        pi*D*D/A;
        }
//..... файл F4.cpp .....

/* Функция расчета площади прямоугольника */
/*                по его сторонам                */

double F4(double a, double b) // a,b — стороны
                              // прямоугольника
  {return a*b; }

//..... программа тестирования функции F0 .....
^include "stdio.h"
^include "conio.h"
#include "FO.cpp"
void mainQ
{ double FO(double);                    // прототип функции FO
    // в данной ситуации не // обязателен double D;
    printfi "Введите D> ");
    scanA"%lf'\&D);
    print/("\n площадь = %//", F0\D));
    getchQ;
  }
//..... программа тестирования функции F4 .....
Mnclude "stdio.h" ^include "conio.h" void mainQ { double
F4(double,doudle);   //прототип функции F4
    // в данной ситуации // обязателен double a,b;
    printfC'Beedume a и b > ");
    scanf("%lf%lf",&a,&b); print/("\n
    площадь = %lf", FA(a,b));
```

```
    getchQ;
  } Mnclude
"F4.cpp"

    II .....файл main.cpp .....

     /*     главная функция расчета стоимости   */
     /*              отделки ванной комнаты            */


#include "stdio.h"
^include "conio.h"
void mainQ
    {double FQ(double);
     doubleF4(double, double);
                              //прототипы обязательны
     double Aroom, Broom, Hroom,
     II ширина, длина и
     // высота комнаты, м
     Adoor,Hdoor,                    // ширина и высота
                                     // двери, м
     Rbath, Hbath,                   // радиус и высота
                                     // стенки чаши, м
     Zwall, Zdoor, Zceil, Zfloor,
     II стоимость отделки // стен, двери, // и пола, р./м2
     clrscrQ; printj{ "Введите размеры комнаты — "
                                          "длина,ширина,высота, м>");
     scanfi "%lf%lf%lf",&Aroom, &Broom, &Hroom); printf(
     "\nВведите размеры двери — высота, "
                                                "ширина, м>");
     scanf{"%lf%lf", &Adoor, &Hdoor); printf{ "\nВведите
     размеры ванны —
                                              радиус,высота, м > «);
     scanK"%lf%lf", &Rbath, &Hbath); printf{ "\nВведите
     стоимости отделки стен,"
                                   "двери, потолка и пола, р./м2, \n> ");
     scan/{"%lf%lf%lf%lf",
              &Zwall,&Zdoor, &Zceil, &Zfloor);
     double Sceil = F4(Aroom,Broom);
                                  //площадь потолка
     double Sbath = F0(2*Rbath)/4;
                                     //площадь, занятая ванной,
     double Sfloor = Sceil-Sbath;     //площадь пола
```

double Sdoor — F4(Adoor,Hdoor); //площадь двери
double Swall — 2(F4(Aroom,Hroom)+*
*F4(Broom, Hroom)) -Sdoor-2*F4(Rbath,Hbath);* //площадь стен за вычетом площадей, //занятых дверью и ванной *double Price = Zceil*Sceil + Zfloor*Sfloor +Zdoor*Sdoor + Zwalt* Swall;*
II полная стоимость отделки *printf["\n\n ***** Полная стоимость отделки "* "ванной комнаты составляет "
%7.21/> *",Price); getchQ; }*

//.....состав файла проекта bathroom.prj

main.cpp
FO.cpp
F4.cpp

Результаты работы программы.
Введите размер комнаты — длина, ширина, высота, м, > 3 2 2.5.
Введите размер двери — высота, ширина, м > 2 1.
Введите размеры ванны — радиус, высота, м > 1 0.5.
Введите стоимость отделки стен, двери, потолка, пола, р./м^2
> 200 100 150 250.
***** Полная стоимость отделки ванной комнаты составляет 6803.65 р.

6.7. Типичные ошибки при выполнении работы

Нарушается правило, согласно которому прототип вызываемой функции обязательно должен присутствовать в вызывающей, если последняя предшествует первой в принятой последовательности функций. Для того чтобы избежать ошибки, рекомендуется задавать прототипы во всех функциях, вызывающих другие функции.

6.8. Требования к студентам

В результате выполнения работы студенты должны: знать Два способа сборки модулей многомодульного программного проекта, особенности, достоинства и недостатки каждого из них;

иметь практические навыки формирования проектов с помощью пункта интегрированной среды *Project*.

Контрольные вопросы

1. Какие способы сборки многомодульной программы вы знаете?
2. Объясните смысл понятий «исходный модуль», «объектный модуль» и «исполняемый модуль».
3. Как выполняется сборка программы на уровне исходных модулей?
4. Какие средства используются при сборке программы на уровне объектных файлов?
5. Что такое файл проекта? Какое расширение имеет такой файл? Из чего он состоит?
6. Что такое объектный файл? Какое расширение он имеет?
7. Сколько объектных файлов содержит проект в каждом варианте его сборки?
8. В чем заключаются основные достоинства и недостатки каждого из способов сборки программы?

Практическая работа 7

ОДНОМЕРНЫЕ МАССИВЫ

7.1. Цель работы

1. Освоить технику использования массивов при программировании на языке С. Изучить применение адресной арифметики при работе с массивами.
2. Научиться писать функции с параметрами-массивами.

7.2. Справочные сведения

Одномерные массивы представляют собой упорядоченную последовательность переменных одинакового типа (тип массива). Элементы массива идентифицируются общим групповым именем (имя массива) и различаются между собой только порядковым номером (индекс элемента).

7.2.1. Описание массивов и работа с массивами с помощью индексов и указателей

Одномерные массивы описываются в языке С по следующей формуле:

тип элементов массива имя массива [количество элементов] = =
{список начальных значений};

Например,
double Ml[3] = {25, 4, -2}; *char* S[5] = {'a','b','c','d','e'}. Нумерация элементов в массиве всегда начинается с нуля. Поэтому начальный элемент указанного массива *M* определяется в виде M1[0], а конечный — Ml [2].

Имя массива, употребленное без индексов, определяет адрес начала массива, т. е. в рассматриваемом примере

M1 = &M1[].

Адрес k-го элемента массива *M* определяется по формуле *M+ K,* т.е.

M+ K=&M[к].

Соответственно, доступ к нулевому и *к-му* элементу массива *M* может быть осуществлен в виде **M* и **(M+ к)*.

Последовательный перебор элементов в массиве может быть осуществлен с помощью циклов трех видов, которые мы проиллюстрируем на примере обнуления массива *M:*

```
int M[5\, i, * p, * plast;
//Вариант 1 — использование индексов:
for (/= 0; /<5; /++);
    M[i] = 0. //Вариант 2 — использование индексов и
указателей: for (i= 0; /<5; /'++);
    *(M+i) = 0. //Вариант 3 — использование
указателей: for (p-M, plast = M + 4; p <= plast; p
++)
    *p = 0; // M+ 4 — адрес последнего элемента массива M.
```

Для передачи массива в функцию рекомендуется передать в последнюю адрес начала массива и количество элементов в массиве, а внутри тела функции работать с элементами массива, используя указатели. Подобный подход обеспечивает возможность строить функции, которые могут работать с массивами произвольного размера.

7.2.2. Динамические массивы

Под динамическими массивами понимаются массивы, размерность которых заранее неизвестна. Она определяется в ходе выполнения программы, например вводится с клавиатуры в качестве исходных данных или вычисляется в процессе расчета, поэтому размещение таких массивов в памяти на этапе компиляции программы так, как это имеет место для обычных (статических) массивов, не представляется возможным.

Для размещения динамического массива определенного типа можно описать указатель на этот тип и передать в этот указатель результат работы функции *malloc(size),* где *size* — число байтов памяти, необходимых для размещения массива. Например, чтобы ввести с клавиатуры целое число *N* и разместить в памяти динамических массив из 7V вещественных чисел, достаточно написать фрагмент программы

```
float N, *mass;
 scanf("%i",&N);
 mass= malloc(N*sizeof(float));
```

Далее с указателем *mass* можно работать как с указателем на начало массива обычным образом (см. подразд. 7.2.1). Память, занятую массивом, можно в любой момент освободить, использовав для этого функции *free(pt),* где/tf — имя соответствующего указателя, т.е. в рассмотренном примере достаточно задать *free(mass);*

Для работы указанных функций необходимо подключить библиотеку динамического управления памятью с помощью инструкции

#nclude <alloc.h>

Литература: [1], с. 83 — 84, 89-95.

7.3. Содержание работы

Каждая работа предполагает решение задачи в соответствии с вариантом индивидуального задания. Отчет по работе должен содержать: формулировку задания; текст программы с комментариями; пример результатов.

7.4. Индивидуальные задания

Для каждого из предложенных вариантов необходимо составить и отладить указанную функцию. В главной программе, если в задании не предложено другое, следует описать два массива указанного в задании типа *S* [8] и *S2* [4]. Элементы первого массива задать при описании, второго — ввести с клавиатуры. Обработать каждый из массивов с помощью составленной функции и результаты вывести на экран.

Вариант 7.1.

Написать функцию, которая для заданного в качестве параметра массива типа *char* вычисляет и возвращает число элементов массива, являющихся цифрами, и заменяет эти элементы символом «#».

В главной программе вычислить суммарное число цифр в обоих массивах.

Вариант 7.2.

Написать функцию, которая для заданного в качестве параметра массива типа *int* меняет местами соседние элементы с четными и нечетными номерами.

Вариант 7.3.

Написать функцию, которая возвращает сумму всех отрицательных элементов заданного в качестве параметра массива плавающих чисел и заменяет эти элементы значениями их абсолютных данных.

Вариант 7.4.

Написать функцию, которая для заданного в качестве параметра массива типа *char* заменяет все малые латинские буквы символом «$» и возвращает количество выполненных замен.

Вариант 7.5.

Написать функцию, которая транспонирует заданный в качестве параметра массив типа *int* (т. е. меняет местами равноудаленные от концов массива элементы).

Вариант 7.6.

Написать функцию, которая возвращает среднее арифметическое значение элементов заданного в качестве параметра массива целого типа, а также заменяет нулем элементы, значения абсолютных данных которых лежат в диапазоне 1... 5.

Вариант 7.7.

Написать функцию, которая возвращает количество положительных элементов заданного в качестве параметра массива типа *double.*

В главной программе определить суммарное число таких элементов в обоих заданных массивах.

Вариант 7.8.

Написать функцию, которая почленно суммирует первые четыре элемента двух заданных в качестве параметров массивов типа *double,* помещая результаты в третий массив.

В главной программе описать три массива действительного типа: *M,1[8], M2[10]* и *M3[4].* Первый массив ввести с клавиатуры, второй задать при описании, а третий вычислить с помощью указанной функции.

Массивы вывести на монитор.

Вариант 7.9.

Написать функцию, которая для заданного в качестве параметра массива типа *char* возвращает сумму кодов его элементов и заменяет все вхождения буквы а на знак «?».

Вариант 7.10.

Написать функцию, которая для заданного в качестве параметра массива действительного типа возвращает число элементов, значения абсолютных данных которых лежат в диапазоне 0... 1, и меняет значения этих элементов на их порядковый номер.

Вариант 7.11.

Написать функцию, которая для заданного в качестве параметра массива типа *double* формирует новый массив, элементы которого содержат синусы значений элементов исходного массива, и возвращает в вызывающую программу сумму квадратов синусов элементов.

Вариант 7.12.

Написать функцию, которая для заданного в качестве параметра массива типа *double* заменяет его элементы значениями косинусов от исходных значений и возвращает количество отрицательных элементов.

В главной программе обеспечить вычисление суммы отрицательных элементов обоих заданных массивов.

Вариант 7.13.

Написать функцию, которая для двух заданных в качестве параметров массивов типа *int* формирует третий массив из четырех элементов, значения которых равны единице, если соответствующие элементы исходных массивов имеют одинаковые знаки, и 0 — в противном случае.

В главной программе описать три массива действительного типа: *M1[8],* M2[10] и M3[4]. Первый массив ввести с клавиатуры, второй задать при описании, а третий вычислить с помощью указанной функции.

Результаты вывести на монитор.

Вариант 7.14.

Написать функцию, которая для заданного в качестве параметра массива типа *char* возвращает сумму кодов всех его элементов и заменяет все вхождения символа «#» на букву А. В главной программе определить суммарное значение кодов символов в обоих массивах.

Вариант 7.15.

Написать функцию, которая для двух заданных в качестве параметров массивов типа *double* формирует третий массив, элементы которого равны сумме соответствующих элементов исходных массивов, и возвращает в вызывающую программу сумму квадратов элементов полученного массива.

В главной программе описать три массива действительного типа: A1[10], A2[10] и A3[10]. Первый массив ввести с клавиатуры, второй задать при описании, а третий вычислить с помощью указанной функции.

Результаты вывести на монитор.

Вариант 7.16.

Написать функцию, которая упорядочивает по возрастанию элементы заданного в качестве параметра массива типа *int* и возвращает среднее арифметическое значение его элементов.

Вариант 7.17.

Написать функцию, которая возвращает порядковый номер максимального по абсолютной величине элемента заданного в качестве параметра массива типа *double.*

Вариант 7.18.

Написать функцию, которая обнуляет первый из заданных в качестве параметров массив вещественного типа, если среднее арифметическое значение элементов второго массива меньше единицы.

Вариант 7.19.

Написать функцию, которая возвращает количество элементов заданного в качестве параметра массива целого типа, превышающих среднее арифметическое значение его элементов.

Вариант 7.20.

Написать функцию, которая для заданного в качестве параметра массива целого типа вычисляет квадратный корень из суммы квадратов значений элементов с нечетными номерами и среднее арифметическое значение элементов с четными номерами и возвращает наименьшее из этих двух данных.

Вариант 7.21.

Написать функцию, которая получает в качестве параметров два массива литерного типа и формирует массив целого типа. Элементы этого массива равны единице, если соответствующие элементы исходных массивов совпадают, и равны нулю в противном случае.

В главной программе описать два массива литерного типа и один целого.

Размерность всех массивов 10.

Первый массив ввести с клавиатуры, второй задать при описании, а третий вычислить с помощью указанной функции.

Результаты вывести на монитор.

Вариант 7.22.
Написать функцию, которая для заданного в качестве параметра массива действительного типа возвращает квадратный корень из суммы квадратов его элементов с нечетными номерами.

Вариант 7.23.
Написать функцию, которая упорядочивает по алфавиту элементы заданного в качестве параметра массива символьного типа.

Вариант 7.24.
Написать функцию, которая для заданного в качестве параметра массива действительного типа возвращает номер элемента, имеющего наибольшее значение синуса.

7.5. Дополнительное задание для «мастеров»

В создаваемой программе один из массивов разместить динамически. Его размер ввести с клавиатуры.

7.6. Пример программы

Условие задачи. Написать функцию, которая вычисляет и возвращает в вызывающую программу сумму элементов массива целых чисел. В главной программе определить массив из пяти целых чисел, вычислить и вывести на экран сумму элементов этого массива.

Программа

Лабораторная работа № 7.
Вычисление суммы элементов массива.
Автор Хворобьев Федор Никитич, гр. 444ф
30.06.2006

```
^include "stdio.h"
#include "conio.h"

void mainQ
   { int Mass[5] = {2, 4, —3, 55, 0}; int summ
     {int *, int); // прототип функции clrscii);
```

```
printf{ "Сумма элементов массива Mass равна %d",
sum(Mass,50)); getchQ; }
// функция вычисления суммы элементов массива
intsumm(int*A, int Net) { int Sum, i: for (Sum = 0,1=0; I <
Nel; /++)
        Sum+ = *(A + i);
    return Sum; }
```

7.7. Типичные ошибки при выполнении работы

Нумерация элементов массива начинается с нуля, поэтому последний элемент массива, содержащего *N* элементов, имеет номер *N* - 1, а не Ж Компилятор ошибку не перехватывает. Как правило, ошибка носит катастрофический характер.

Для того чтобы функция могла обрабатывать массивы любых данных кроме адреса начала массива, в нее следует передать длину массива.

7.8. Требования к студентам

В результате выполнения работы студенты должны:
знать синтаксис описания массивов на языке С;
уметь обращаться к элементам массивов по индексу и адресу;
уметь передавать массивы в функции в качестве аргументов.

Контрольные вопросы

1. Как описать одномерные массивы заданной размерности?
2. Как инициализировать одномерный массив?
3. Какое значение имеет индекс первого элемента массива? последнего?
4. Что означает имя массива без индекса?
5. Как обратиться к г'-му элементу массива с помощью индекса и с помощью адреса (указателя)?
6. Когда необходимо использовать динамические массивы?
7. Как распределить память под динамический массив?
8. Как освободить память, занятую динамическим массивом?
9. Как обратиться к /'-му элементу динамического массива?

Практическая работа 8

ДВУХМЕРНЫЕ МАССИВЫ

8.1. Цель работы

1. Освоить технику использования двухмерных массивов при программировании на языке С. Изучить применение адресной арифметики при работе с двухмерными массивами.
2. Научиться писать функции с параметрами-массивами.

8.2. Справочные сведения

Двухмерные массивы описываются и (если нужно) инициализируются в форме: тип элементов

имя массива [количество строк] [количество столбцов] = =
{{список начальных значений элементов
первой строки},
{список начальных значений элементов
второй строки},

{список начальных значений элементов
последней строки}};

Например,

double 5[2][3] = {{2,0, 3,1, 7,0}, {-ПО, -2,0, -310,5}};

Нумерация строк и столбцов в массиве всегда начинается с нуля.

Имя элемента с одним индексом определяет адрес начала соответствующей строки, т.е.

S[K] = & S[K][0],

следовательно, *S[0]* определяет адрес начала массива.

Доступ к элементу массива, расположенному в i-Й строке и j-м столбце, осуществляется в виде S[i][j] либо с использованием адреса в виде

** (S[0] + i*Nstb+j),*

где *Nstb* — число столбцов массива.

Выражение *S[0] + FNstb + j* соответственно определяет адрес элемента *S[i]* [/'].

При использовании двухмерных массивов в качестве параметров функции рекомендуется передать в последнюю адрес начала массива, а также число строк и столбцов. При этом в теле функции доступ к элементам массива должен осуществляться исключительно с помощью адресной арифметики.

Литература: [1] с. 86 —87.

8.3. Содержание работы

Каждая работа предполагает и решение задачи, предусмотренной вариантом индивидуального задания.

Отчет по работе для каждой задачи должен содержать:

формулировку задания;

текст программы с комментариями;

пример результатов.

8.4. Индивидуальные задания

Вариант 8.1.

Написать функцию, которая меняет местами первую строку и последний столбец квадратной матрицы.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 5*5 и *B* размерностью 3*3, инициализировав последний массив значениями 5, 3, 7, -1, -3, -5, 4, 7, 9, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на кран.

Вариант 8.2.

Написать функцию, которая складывает первую и последнюю строки квадратного массива и записывает результат в первый столбец.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 7*7 и *B* размерностью 3*3, инициализировав последний массив значениями 5, 3, 7, 0, 0, 0, 5, 7, 3, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.3.

Написать функцию, которая меняет значения диагональных элементов квадратной матрицы на значения соответствующих элементов заданного одномерного массива.

Написать программу, которая описывает двухмерный массив *A* размерностью 3*3 и одномерный *B* размерностью 3, инициализировав последний массив значениями 5, 3, 7, а первый ввести с клавиатуры.

Применить к массиву *A* указанную функцию и вывести массивы *A* и *B* на экран.

Вариант 8.4.

Написать функцию, которая складывает соответствующие элементы двух заданных массивов и заносит результаты в третий массив. Все три массива имеют одинаковые размерности.

Написать программу, которая описывает три двухмерных массива: *A, B и C* размерностью 4*2, инициализировав массив *A* значениями 5, 3, 7, 1, 3, -5, 4, 7, а значения элементов массива *B* ввести с клавиатуры.

Результаты сложения массивов *A* и *B* занести в массив C. Вывести все три массива на экран.

Вариант 8.5.

Написать функцию, которая меняет местами первую строку и последний столбец квадратной матрицы.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 5*5 и *B* размерностью 3*3, инициализировав последний массив значениями 5, 3, 7, -1, -3, -5, 4, 7, 9, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.6.

Написать функцию, которая меняет местами диагонали квадратной матрицы.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 5*5 и *B* размерностью 3*3, инициализировав последний массив значениями 5, 3, 7, -1, -3, -5, 4, 7, 9, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.7.

Написать функцию, которая суммирует элементы строк двухмерного массива и заносит результаты в одномерный массив, размерность которого равна числу строк двухмерного массива.

Написать программу, которая описывает двухмерный массив *A* размерностью 4*2, вводит этот массив с клавиатуры, помещает суммы элементов строк этого массива в одномерный массив *B* и выводит массивы *A* и *B* на экран.

Вариант 8.8.

Написать функцию, которая находит и возвращает в вызывающую программу максимальный по модулю элемент заданного двухмерного массива.

Написать программу, которая описывает двухмерный массив *A* размерностью 4*2. Значения элементов массива *A* ввести с клавиатуры.

Вывести на экран массив *A* по столбцам, а также результаты применения к этому массиву созданной функции.

Вариант 8.9.

Написать функцию, которая меняет местами последнюю строку и первый столбец квадратной матрицы.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 4*4 и *B* размерностью 3*3, инициализировав последний массив значениями 4, *2,* 7, -1, -5, -7, 9, 4, 1, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.10.

Написать функцию, которая складывает первый и последний столбцы квадратного массива и записывает результат на место первой строки.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 7*7 и *B* размерностью 3*3, инициализировав последний массив значениями 7, 4, 7, 0, 0, 0, 1, 7, 3, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.11.

Написать функцию, которая меняет значения элементов заданного столбца квадратной матрицы на значения соответствующих элементов одномерного массива.

Написать программу, которая описывает двухмерный массив *A* размерностью 3*3 и одномерный *B* размерностью 3, инициализировав последний массив значениями 5, 3, 7, а первый ввести с клавиатуры.

Применить к массиву *A* указанную функцию и вывести массивы *A* и *B* на экран.

Вариант 8.12.

Написать функцию, которая перемножает соответствующие элементы двух заданных массивов и заносит результаты в третий массив.

Размерности всех трех массивов одинаковы.

Написать программу, которая описывает три двухмерных массива: *A, B и C* размерностью 4*2, инициализировав массив *A* значениями 1, 3, 7, 1, 3, 5, 4, 2, а значения элементов массива *B* ввести с клавиатуры. Результаты перемножения массивов *A и B* занести в массив C.

Вывести все три массива на экран.

Вариант 8.13.

Написать функцию, которая меняет местами последнюю строку и первый столбец квадратной матрицы.

Написать программу, которая описывает два двухмерных массива: *A* размерностью 3*3 и *B* размерностью 4*4, инициализировав последний массив значениями 5, 3, 7, -1, -3, -5, 4, 7, 9, 0, 1, 2, 1, 5, 3, 2, а первый ввести с клавиатуры.

Применить к обоим массивам указанную функцию и вывести результат на экран.

Вариант 8.14.

Написать функцию, которая суммирует элементы столбцов двухмерного массива и заносит результаты в одномерный массив, размерность которого равна числу столбцов двухмерного массива.

Написать программу, которая описывает двухмерный массив *A* размерностью 4*2, вводит этот массив с клавиатуры, помещает суммы столбцов этого массива в одномерный массив *B* и выводит массивы *A* и *B* на экран.

Вариант 8.15.

Написать функцию, которая находит и возвращает в вызывающую программу номер строки заданного двухмерного массива, имеющего максимальную по модулю сумму элементов.

Написать программу, которая описывает двухмерный массив *A* размерностью 4*2. Значения элементов массива *A* ввести с клавиатуры.

Вывести на экран массив *A* по столбцам, номер строки этого массива, найденный с помощью разработанной функции, а также саму строку.

Вариант 8.16.

Написать функцию, которая для заданных двухмерных массивов *A, B* и *C* типа *double* одинаковой размерности вычисляет массив *D* по формуле

$$Di,j = Ai,j+Bi,j*Ci,j$$

Написать программу, которая вводит с клавиатуры три массива типа *double Ml, Ml* и *МЪ* размерностью 2*3, вычисляет по указанной формуле массив *Xu* выводит его по строкам на экран.

Вариант 8.17.

Написать функцию, которая для заданного двухмерного массива типа *int* вычисляет и возвращает значение среднего арифметического элементов *к-то* столбца и л-й строки.

Написать программу, которая описывает и задает значения элементов массива *A* размерностью 3*4, описывает и вводит с клавиатуры массив *B* размерностью 2*3, вычисляет и выводит на экран сумму средних арифметических первого столбца и второй строки обоих массивов.

Вариант 8.18.

Написать функцию, которая располагает столбцы заданного в качестве параметра двухмерного массива целого типа в порядке убывания сумм их элементов.

В главной программе описать два массива целого типа размерностью 2*4 и 3*5. Первый массив ввести с клавиатуры, второй инициализировать при описании.

С помощью разработанной функции упорядочить столбцы обоих массивов и вывести массивы на экран монитора.

Вариант 8.19.

Написать функцию, которая вычисляет сумму квадратов элементов заданной строки. Номер строки передается в функцию в качестве параметра.

Написать программу, которая описывает два квадратных массива действительного типа размерностью 2*3 и 3*2. Элементы первого массива вводятся с клавиатуры, второго — задаются при описании. Требуется вычислить суммы квадратов элементов второй строки первого массива и первой строки второго массива.

Результаты вывести на экран.

Вариант 8.20.

Написать функцию, которая для заданных двухмерных массивов *A, B, C* целого типа одинаковой размерности вычисляет массив вещественного типа *D* по формуле

$$D\ i, j = Ai, j/(Bi, j + Ci, j) + Ci, j/Ai, j.$$

Написать программу, которая вводит с клавиатуры три массива целого типа: *Ml, Ml* и *МЪ* размерностью 3*2, вычисляет по указанной формуле массив *Rezult* и выводит его по строкам на экран.

Вариант 8.21.

Написать функцию, которая поэлементно сравнивает два заданных в качестве параметров двухмерных массива литерного типа одинаковой размерности и формирует массив целого типа такой же размерности. Элементы последнего массива приравнять 1, если соответствующая пара элементов сравниваемых массивов совпадает между собой, и 0 в противном случае. Функция должна возвратить в вызывающую программу количество строк сравниваемых массивов, которые полностью совпадают между собой.

В главной программе задать два массива *A* и *B* литерного типа размерностью 5*5, и ввести их с клавиатуры. Сравнить указанные массивы, результаты сравнения занести в массив целого типа *Result.*

Все результаты вывести на экран.

Вариант 8.22.

Написать функцию, получающую в качестве параметров два массива целого типа произвольного размера и возвращающую порядковый номер массива, среднее арифметическое значецие элементов которого максимально.

В главной программе описать три массива целого типа: *A* размерностью 5*6, *B* размерностью 4*4 и C размерностью 3*4.

Массивы *A и B* инициализировать при описании, а массив *C* ввести с клавиатуры. С помощью разработанной функции определить и вывести на экран имя массива, имеющего наибольшее среднее арифметическое значение элементов.

Вариант 8.23.

Написать функцию, которая располагает строки заданного в качестве параметра двухмерного массива целого типа в порядке возрастания сумм их элементов.

В главной программе описать два массива целого типа размерностью 4*4 и 3*5. Первый массив ввести с клавиатуры, второй инициализировать при описании.

С помощью разработанной функции упорядочить строки массивов и вывести массивы на экран монитора.

Вариант 8.24.

Написать функцию, которая вычисляет сумму элементов, лежащих на обеих диагоналях квадратной матрицы.

Написать программу, которая описывает два квадратных массива размерностью 2*2 и 3*3. Элементы первого массива вводятся с клавиатуры, второго — задаются при описании.

Требуется вычислить сумму диагональных элементов обоих массивов и напечатать ее на экране.

8.5. Дополнительное задание для «мастеров»

Дополнить программу таким образом, чтобы разработанная функция кроме заданной величины возвращала в вызывающую программу второе значение — общее количество элементов в массиве (передача по адресу).

8.6. Пример программы

Условие задачи. Написать функцию, которая вычисляет и возвращает сумму квадратов элементов z'-й строки заданного двухмерного массива

В главной программе описать и инициализировать двухмерный массив целого типа размерностью 4*3.

В главной программе вычислить сумму квадратов элементов его второй строки.

Программа

```
                    Лабораторная работа № 8. Суммирование
          квадратов элементов строки двухмерного
                                массива.
                        Автор Скамейкин Ян, гр. 446к
                                10.04.2006


ttinclude "conio.h"
ttinclude "stdio.h"
II Функция, вычисляющая сумму квадратов элементов к-й
Ц                          строки матрицы
  int SumStr (int *A, int Nstr, int Nstb, int K)
    {int sum, j; for (sum = 0; j = 0; j < Nstb; j
    ++)
        sum+ = *(A + K*Nstb +j) * *(A + K*Nstb +/);
    return sum: }

  void mainQ { int Mass[4][3];
    intij; int SumStr (int*, int, int,
    int);
                                //прототип функции суммирования
```

```
            // Ввод массива
 for (i = 0; / < 4; г-н-) for
  (J = 0;J< 3; j ++)
  scanf("%i",A[0} + 3*i+j); printf( "Сумма квадратов
  элементов второй строки = %d", SumStr(Mass, 4, 3, 1));
// нумерация строк начинается с нуля, // т.е. вторая строка
имеет номер 1. getch (); }'
```

8.7. Типичные ошибки при выполнении работы

В среде ВС адрес начала двухмерного массива *M* внутри функции, в которую он передается в качестве параметра, определяется в виде *M,* а в блоке, где этот массив описан, в виде *M[0]*.

При передаче двухмерного массива в функцию следует не забывать передать в нее число строк и столбцов.

Если в функцию передаются в качестве параметров несколько массивов одинаковой размерности, то количество строк и столбцов массивов достаточно передать только один раз.

При передаче в функцию квадратного массива достаточно передать в функцию только количество его строк.

8.8. Требования к студентам

В результате выполнения работы студенты должны:

знать синтаксис описания массивов на языке С;

уметь обращаться к элементам двухмерного массива по индексу и адресу;

уметь передавать двухмерные массивы в функции в качестве аргументов.

Контрольные вопросы

1. Как описывается и инициализируется двухмерный массив?
2. Каким образом выполняется нумерация строк и столбцов двухмерного массива?
3. В каком порядке размещаются в памяти элементы двухмерного массива?
4. Как определяется адрес элемента двухмерного массива?
5. Каким образом осуществляется доступ к элементам двухмерного массива по индексам и с помощью указателя?
6. Что передается в функцию при использовании двухмерного массива в качестве параметра этой функции?

Практическая работа 9

СИМВОЛЫ И СТРОКИ

9.1. Цель работы

1. Освоить синтаксис и технику использования символьных и строковых переменных в языке С.
2. Научиться использовать библиотечные функции для работы со строками и создавать собственные функции с параметрами-строками.

9.2. Справочные сведения

Символ определяется как однобайтовая переменная, содержащая код символа, определяемый в соответствии со стандартом ANSI./ISO.

Символ описывается в программе в виде

char имя символа [= начальное значение];

причем присвоение начального значения не является обязательным элементом описания. Символьная константа представляется в форме символа, заключенного в апострофы (например, 'a,' '#,' 'R'), либо в форме восьмеричного кода символа, перед которым помещается «\» (обратная косая), причем вся комбинация также заключается в апострофы (например, '\006', '\203', '\015').

Далее приводятся коды букв английского и русского алфавитов, которые могут быть полезны при выполнении некоторых вариантов индивидуальных заданий.

Английский алфавит:

A — '\101', B — '\102', . . ., Z — '\132';
a — '\141', b — '\142', . . ., z — '\172'.

Русский алфавит:

А — '\200', Б — '\201', . . ., Я — '\237';
а — '\240', б — '\241', . . ., п — '\257';
р — '\340', с — '\341', . . ., я — '\357'.

Для управляющих символов предусмотрены следующие специальные представления:

'\a' — звуковой сигнал;
'\b' — возврат назад на одну позицию;

‘\f’ — перевод страницы при печати на принтере;
‘\n’ — перевод строки;
‘\r’ — переход к началу строки;
‘\\’ — обратная косая;
‘\” — апостроф;
‘\”’ — кавычки;
‘\?’ — вопросительный знак;
‘\0’ — нулевой символ.

Особо подчеркнем, что обратная косая изображается двумя косыми. Например, полное имя файла записывается в виде *"C: \ \dir\ \subdir\ \myfile. txt "*.

Внимание. Ошибочный вариант с одной косой *"C:\dir\subdir\myfile.txt"* компилятором не перехватывается.

Строка представляет собой массив символов, последним из которых является нулевой символ '\0'. Строковая константа изображается в форме последовательности любых символов, заключенных в кавычки, например:

" *** Чижик-пыжик, где ты был? ",

причем нулевой символ в явном виде не изображается. Он добавляется системой самостоятельно.

Описание строки в С-программах может быть выполнено в одной из следующих форм:

char имя строки [максимальный размер] =

= "начальное значение"; (9.1)

char имя строки [ ] = "начальное значение"; (9.2)

char * имя строки = "начальное значение". (9.3)

В форме (9.1) присвоение начального значения не является обязательным, а в формах (9.2) и (9.3) оно необходимо.

При использовании формы (9.1) следует предусмотреть позицию для нулевого символа. Например, описание
char str [5] = "Мама" является
верным, а описание *char str* [4]
= "Папа" содержит ошибку.

Для работы со строками в языке предусмотрен ряд функций, большинство из которые определены в файлах включения *string h, stdio h* и *stdlib h.*

Перечислим основные функции:

int strlen (str) — возвращение фактической длины строки (без учета символа '\0');

gets (str) — ввод строки с клавиатуры;

puts (str) — вывод строки на экран;

/gets (str, nmax, file), fputs(str, file) — чтение строки из файла и запись строки в файл *(nmax* — максимально допустимая длина строки);

strcat(strl, str2) — присоединение строки *str2* к строке *strl;*

strch (str, ch) — определение адреса первого вхождения символа *ch* в строку *str,*

strstr (strl, str2) — определение адреса первого вхождения подстроки *str2* в строку *strl;*

strcmp (strl, str2) — сравнивание строк *strl* и *str2* (при совпадении возвращение 0);

strcpy (strl, str2) — копирование строки *str2* в строку *strl;*

atoi (str), atof(str) — преобразование строки *str* соответственно в целое или действительное число.

Много других функции можно найти в литературе либо с помощью справочной системы среды ВС.

При использовании строки в качестве параметра функции в последнюю достаточно передать только адрес начала строки, так как ее длина однозначно определяется положением конечного нулевого символа.

При манипулировании со строками система не контролирует выход фактической длины строки за пределы максимально допустимого размера, задаваемого при описании. В случае ошибки дополнительные символы будут записаны поверх информации, которая располагается вслед за полем, отведенным для строки. Последнее может явиться причиной нарушения нормальной работы программы, поэтому рекомендуется следить за выходом строки за предусмотренные пределы, вводя достаточные размеры строк при их описании.

Литература: [1] с. 84—86.

9.3. Содержание работы

В процессе выполнения работы студенты должны изучить технику использования строк, библиотеку функций для работы со строками, составить и отладить программу, решающую задачу, предлагаемую в индивидуальном задании.

Необходимые для решения задачи файлы следует предварительно создать на любом из доступных для записи дисков, записав в них нужную по условиям задачи информацию.

Отчет по работе должен содержать:

формулировку задачи;

текст программы;

пример результатов ее работы.

9.4. Индивидуальные задания

Вариант 9.1.

Составить функцию, выводящую на экран слова, которые одновременно содержатся в каждой из двух заданных строк.

В главной программе ввести с клавиатуры три строки длиной до 80 символов и вывести на экран слова, которые содержатся в каждой паре строк.

Вариант 9.2.

Написать функцию, которая определяет, содержит ли заданная строка двоичное, восьмеричное, десятичное или шестнадцатеричное число.

Например:

111011 — двоичное, восьмеричное, десятичное или шестнадцатеричное число;

2173145 — восьмеричное, десятичное или шестнадцатеричное число;

927888 — десятичное или шестнадцатеричное число;

2AC378 — шестнадцатеричное число;

Aa Z318 — не число.

В главной программе ввести четыре строки с клавиатуры и две прочесть из файла *myfile.txt.* Классифицировать каждую из строк, выведя строки и результаты на экран.

Вариант 9.3.

Написать функцию, воспроизводящую работу простейшего калькулятора. В качестве параметра задается строка вида

$$a_1 Z_1 a_2 Z_2 \ldots a_n, \qquad n <= 20,$$

где a_i — целое число от 1 до 9; Z_i — знак «+» или «-».

Функция анализирует синтаксис строки и в случае отсутствия ошибок вычисляет и выводит результат, а при ошибке выводит необходимое сообщение.

В главной функции предусмотреть ввод строки и вывод результата. При вводе строки «0000» программа заканчивает работу.

Вариант 9.4.

В файле *myfile.txt* содержится информация типа

$$a = 1{,}0;\ b = 5;\ c = 25{,}2;\ d = -12{,}5.$$

Написать функцию, которая получает строку типа *«a + b =* », *«c*d =* » или *«b/d =* », выбирает из файла необходимые значения, вычисляет и возвращает результат.

В главной программе обеспечить ввод с клавиатуры строки и вывод результата.

Выход из программы обеспечить при вводе строки «О + О».

Вариант 9.5.

Составить функцию, которая выводит на экран содержимое заданного текстового файла в виде «бегущей строки». В главной программе обеспечить ввод имени файла и скорости движения строки.

Вариант 9.6.

Написать функции, которые шифруют-дешифруют текст, помещенный в файле, имя которого задается в качестве параметра, заменяя буквы цифрами в соответствии с содержащейся в файле *cod.txt* таблицей кодировки (значения кодов произвольные) вида:

А	010	а	503
Б	100	б	215
В	103	в	313

В главной программе обеспечить ввод имени исходного файла и имени файла с закодированным текстом, обеспечить кодировку и декодировку файла.

Вывести на экран исходный файл, закодированный и декодированный.

Вариант 9.7.

Написать функцию, которая шифрует содержимое заданного текстового файла, заменяя каждый его символ порядковым номером первого вхождения соответствующего символа в заданный текст, например:

«Однажды в студеную зимнюю пору» «Я
из лесу вышел», «Был сильный мороз».

При отсутствии соответствующей буквы в заданной строке оставить букву без изменения. Написать функцию, выполняющую обратное действие (дешифрацию).

В главной программе ввести имя файла и вывести на экран его содержимое, результат шифрования и дешифрования.

Вариант 9.8.

В файл *baze.txt* поместить информацию о писателях и их произведениях (20...30 наименований), например: Пушкин Евгений Онегин Тургенев Накануне Пушкин Медный всадник.

Написать программу, которая:
по заданному имени автора выводит все его произведения; по заданному названию произведения выводит имя автора; выводит начинающиеся с заданной буквы фамилии всех авторов, имеющихся в базе.

Для выбора варианта создать меню.

Вариант 9.9.

Написать программу, которая подводит итоги Олимпийских игр. Программа должна получить от пользователя для каждой из стран, перечисленных в файле *country.txt,* количество золотых, серебряных и бронзовых медалей, подсчитать полное число медалей и количество очков (золотая медаль — 10 очков, серебряная — 7 очков, бронзовая — 3 очка) и вывести таблицу результатов в порядке занятых мест.

Вариант 9.10.

Написать функцию, воспроизводящую работу простейшего калькулятора.

В качестве параметра задается строка вида

$$a_1 Z_1 a_2 Z_2 \ldots a_{nn}, \qquad n <= 20,$$

где ,a_i— целое число от 1 до 9;Z_i — знак «*» или «/».

Функция анализирует синтаксис строки и в случае отсутствия ошибок вычисляет и выводит результат, а при ошибке выводит необходимое сообщение.

Вариант 9.11.

В файле *myfile.txt* содержится информация (20...30 строк), например:

Иванов Петр
Петров Василий
Семенов Иван
Котов Василий.

Программа выбирает из файла и выводит на экран:

все фамилии, начинающиеся с заданной буквы;

всех тезок (например, всех Иванов — имя вводится с клавиатуры);

фамилии и имена всех однофамильцев (если они есть).

Для выбора варианта создать меню.

Вариант 9.12.

Составить программу, которая вводит с клавиатуры три строки, содержащие до 75 символов, и выводит на экран слова, которые являются общими для всех трех строк, а также для каждой пары строк.

Вариант 9.13.

Составить программу, которая вводит с клавиатуры строку, содержащую до 80 символов, и выводит на экран слова из этой строки, которые одинаково читаются слева направо и справа налево (например, шабаш или кок).

В случае отсутствия таких слов выводится соответствующее сообщение.

Вариант 9.14.

Составить программу, которая вводит с клавиатуры строку, содержащую до 75 символов, и выводит на экран слова из этой строки, длина которых составляет 5...8 букв.

В случае отсутствия таких слов выводится соответствующее сообщение.

Вариант 9.15.

Составить программу, которая вводит с клавиатуры строку, содержащую до 70 символов. В случае наличия в этой строке слов, которые повторяются два или более раза, вывести на экран эти слова с указанием числа их повторений.

В случае отсутствия таких слов выводится соответствующее сообщение.

Вариант 9.16.

Написать функцию, которая определяет количество слов в заданной строке, содержащих заданную букву.

В главной программе ввести с клавиатуры две строки длиной до 50... 80 символов соответственно. Определить число слов в каждой строке, содержащих русские буквы *а и р* и вывести результат на экран.

В случае отсутствия таких слов выводится соответствующее сообщение.

Вариант 9.17.

Написать программу, выводящую на экран строки, которые одновременно содержатся в двух заданных текстовых файлах. Имена файлов прочесть с клавиатуры.

Сами файлы с текстом предварительно создать на любом из доступных дисков.

Вариант 9.18.

Написать функцию, которая выводит на экран все слова заданной строки, в которые заданная буква входит Л^или более раз.

В случае отсутствия таких слов выводится соответствующее сообщение.

В главной программе ввести с клавиатуры две строки длиной до 50...80 символов соответственно, и с помощью созданной функции вывести на экран все слова первой строки, которые содержат букву *c* более одного раза, и все слова второй строки, содержащие букву *a* три или более раза.

Вариант 9.19.

Написать функцию, которая вычисляет количество разных слов в заданной строке.

В главной программе обеспечить ввод строки с клавиатуры или из файла (для выбора варианта создать меню) и с помощью функции определить количество слов в ней.

Работу программы завершить после ввода строки *Final.*

Вариант 9.20.

Написать функцию, которая определяет, содержит ли заданная в качестве параметра строка действительное число.

В главной программе обеспечить ввод строки с клавиатуры или из файла и вывести на экран число, которое в ней содержится. Если строка не является числом, выдать соответствующее сообщение.

Работу программы завершить при вводе строки *Final.*

Вариант 9.21.

Написать функцию, которая преобразует строку, содержащую произвольную последовательность нулей и единиц (двоичное число), в целое число.

В главной программе ввести с клавиатуры строку нулей и единиц и вывести на экран соответствующее целое число. Работу программы завершить, если во введенной строке содержится хотя бы один символ, отличный от нуля или единицы.

Вариант 9.22.

Написать функцию, которая получает в качестве параметров две строки и возвращает в вызывающую программу количество прописных букв русского алфавита, одновременно содержащихся в заданных строках.

В главной программе прочесть с клавиатуры две строки, содержащие русский текст, и вывести на экран количество общих прописных букв во введенных строках.

Работу завершить после ввода строки *Завершить программу.*

Вариант 9.23.

Написать программу, которая определяет и выводит на экран частоту вхождения каждой из букв русского алфавита в текст, содержащийся в заданном файле. Имя файла ввести с клавиатуры.

Соответствующий текстовый файл предварительно создать на любом из доступных для записи дисков.

Вариант 9.24.

Написать функцию, которая получает в качестве параметра строку и определяет, является ли содержимое строки правильным восьмеричным числом. В случае положительного результата функция переводит это число в десятичную форму и возвращает его, в противном случае функция генерирует соответствующее сообщение и возвращает число -77777.

В главной программе ввести с клавиатуры строку и, если она содержит восьмеричное число, вывести на экран соответствующее ему десятичное число.

Повторять выполнение программы до тех пор, пока не будет введена строка *Конец*.

9.5. Пример программы

Условие задачи. Ввести с клавиатуры имена дисковода, директории, поддиректории, файла и его расширения. Сформировать из указанной информации полное имя файла и вывести его на экран.

Программа

Лабораторная работа № 9.
Формирование полного имени файла.
Автор Щукина Элла, гр. 445
31.12.2006

```
#include "stdio.h"
Mnclude "conio.h"
#include "string.h " void
mainQ
    { char disk, directory [10], subdirectory [10], filename [10],
      dimention [3],
        FullName[50];
      clrscrQ;
      puts( "\n Введите имя диска > "); dick = getchQ;
      puts( "\n Введите имя директории > "); gets(directory);
      puts{ "\n Введите имя поддиректории > "); gets(subdirectory);
      puts("\n Введите имя файла > "); getsifilename);
```

ПО

```
 puts("\n Введите расширением "); gets(dimension);
  FullName[0] = disk;
  FullName[l] = '\0';              // копирование имени дисковода
  strcat(FullName, ":\\");
  strcat(FullName, directory); //присоединение директории
  strcat(FullName, "\\");
  strcat{FullName, subdirectory); // присоединение поддиректо-
                                  // рии
  strcat(FullName, "\\");
  strcat(FullName, filename); II присоединение имени файла
  strcat(FullName, ".");
 strcat{FullName,dimention);// присоединение расширения
 print/("Полное имя файла %s", FullName); getchQ; }
```

9.6. Типичные ошибки при выполнении работы

При объявлении строки следует предусмотреть байт для записи нулевого символа '\л\

Размер строки, сформированной в результате выполненных манипуляций, превышает количество символов, заданных при описании строки.

9.7. Требования к студентам

В результате выполнения работы студенты должны:

знать синтаксис описания, правила и приемы работы со строками;

знать и активно владеть основными библиотечными функциями для работы со строками;

уметь найти в справочной системе описание любой из функций для работы со строками;

уметь использовать строки в качестве параметров функции.

Контрольные вопросы

1. Чем отличается строка от символьного массива?
2. Как описать и инициализировать строку?
3. Как определить адрес начала строки и адрес ее г'-го символа?
4. Каким образом можно определить фактическую длину строки?
5. Как скопировать одну строку в другую?
6. Как сравнивать строки между собой?
7. Каким образом можно сцепить две строки?

8. Как найти заданный символ в строке?
9. Каким образом можно найти подстроку в строке?
10. Как передать строку в функцию в качестве параметра?
11. Как преобразовать строку в число?
12. Как прочесть строку с клавиатуры и вывести ее на экран?
13. Как прочесть строку из файла и записать ее в файл?

Приложение 1

Операции языка С и их приоритеты

Приоритет	Операция	Комментарий	Порядок выполнения
1	(), [ ]	Скобки	
Все унарные (кроме постфиксных)			
2	++	Увеличение операнда на единицу	←
	– –	Уменьшение операнда на единицу	←
	–	Изменение знака	←
	*	Определение значения по адресу	←
	&	Определение адреса	←
	!	Логическое отрицание	←
	~	Поразрядное логическое отрицание**	←
	(тип)	Явное преобразование типа	←
	sizeof	Определение размера	←
Бинарные арифметические			
3	*	Умножение	→
	/	Деление	→
	%	Остаток от деления**	→
4	+	Сложение	→
	–	Вычитание	→
5	<<	Сдвиг влево**	→
	>>	Сдвиг вправо**	→

Приоритет	Операция	Комментарий	Порядок выполнения
		Операции отношения и сравнения	
6	<	Меньше	→
	<=	Меньше или равно	→
	>	Больше	→
	>=	Больше или равно	→
7	==	Равно	→
	!=	Не равно	→
		Поразрядные операции	
8	&	Поразрядное И**	→
9	^	Поразрядное исключающее ИЛИ**	→
10	|	Поразрядное ИЛИ**	→
		Логические	
11	&&	Логическое И	→
12	||	Логическое ИЛИ	←
		Условная	
13	?	Условная	→
		Присваивание и постиксные бинарные	
14	=	Простое присваивание	←
	*=	Умножение с присваиванием	←
	/=	Деление с присваиванием	←
	%=	Остаток от деления с присваиванием**	←
	+=	Суммирование с присваиванием	←
	-=	Вычитание с присваиванием	←
	<<=	Сдвиг влево с присваиванием	←
	>>=	Сдвиг вправо с присваиванием	←

Приоритет	Операция	Комментарий	Порядок выполнения
Постфиксные унарные			
15	++	Увеличение операнда на единицу	←
	--	Уменьшение операнда на единицу	←
Операция Запятая			
16	,	Последовательность выполнения	←

Примечания: ** — операция определена только для данных целого типа;

→ — порядок выполнения операций одного приоритета слева направо;

← — порядок выполнения операций одного приоритета справа налево.

Преобразование данных при выполнении операций

При выполнении унарных арифметических операций тип результата совпадает с типом операнда.

При выполнении бинарных операций (кроме операций присваивания) автоматически действуют следующие правила:

• все операнды типа *float* преобразуются к типу *double* (это обеспечивает повышенную точность вычислений);

• все операнды типа *char* преобразуются к типу *int;*

• если один из операндов имеет тип *long double,* то второй операнд преобразуется к тому же типу, иначе

если один из операндов имеет тип *double,* второй операнд преобразуется к типу *double,* иначе

если один из операндов имеет тип *unsigned long,* второй операнд также преобразуется к типу *unsigned long,* иначе

если один из операндов имеет тип *long,* второй операнд преобразуется к типу *long,* иначе

если один из операндов имеет тип *unsigned,* второй операнд преобразуется к типу *unsigned.*

Тип полученного результата в зависимости от типов операндов задается табл. П.2.1.

При выполнении операций отношения, сравнения или логических операций преобразование операндов выполняется в соответствии с изложенными ранее правилами. Результат имеет тип *int* (1 — истина, 0 — ложь).

Таблица П.2.1

Тип первого операнда	Тип результата при типе второго опреранда							
	Id	*d*	***f***	*ul*	*I*	*u*	*i*	*c*
Id	*Id*	*Id*	*Id*	*Id*	*Id*	*Id*	*Id*	*Id*
d	*Id*	*d*	*d*	*d*	*d*	_*d*	*d*	*d*
f	*Id*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
ul	*Id*	*d*	*d*	*ul*	*ul*	*ul*	*ul*	*ul*
I	*Id*	*d*	*d*	*ul*	*I*	*I*	*I*	*I*
u	*Id*	*d*	*d*	*ul*	*I*	*u*	*u*	*u*
i	*Id*	*d*	*d*	*ul*	*I*	*u*	*i*	*i*
c	*Id*	*d*	*d*	*ul*	*I*	*u*	*i*	*i*

Примечание: *Id — long double; d — double; f — float; ul — unsigned long; I — long; u — unsigned; i — int; c — char.*

При выполнении операции присваивания сначала вычисляется выражение, стоящее справа от знака равенства. Если тип полученного результата отличается от типа переменной, стоящей слева, то тип результата автоматически преобразуется к типу левого операнда. Только после этого выполняется присваивание.

СПИСОК ЛИТЕРАТУРЫ

1. *Березин Б. И.* Начальный курс С и С++ / Б. И. Березин, С. Б. Березин. — М. : Диалог МИФИ, 1998-2000.

2. *Демидович Е. М.* Основы алгоритмизации и программирования. Язык С / Е. М. Демидович. - СПб.: БХВ — Петербург, 2006.

3. *Культин Н.* С/С++ в задачах и примерах / Н. Культин. — СПб.: БХВ — Петербург, 2001.

4. *Марченко А. Л.* С++ Бархатный путь /А. Л. Марченко. — М.: Горячая линия — Телеком, 2000.

5. *Шилъд Г.* С/С++ Справочник программиста / Г. Шильд. — М.: СПб.— Киев : Изд. дом «Вильяме», 2000.

ОГЛАВЛЕНИЕ

Предисловие 3

Практическая работа 1
Изучение интегрированной среды BC и арифметических операций языка С. Программирование и отладка простейшей задачи

1.1. Цель работы 6
1.2. Справочные сведения ..... б
1.2.1. Структура программы на языке С 7
1.2.2. Работа с консолью 8
1.2.3. Управление выводом на экран 10
1.2.4. Работа с файлами 11
1.2.5. Арифметические операции и операция присваивания 12
1.3. Содержание работы 14
1.4. Индивидуальные задания 14
1.5. Пример программы 24
1.6. Типичные ошибки при выполнении работы 25
1.7. Требования к студентам 25

Практическая работа 2
Разработка и отладка линейной программы, использующей стандартные библиотечные функции

2.1. Цель работы 27
2.2. Справочные сведения 27
2.3. Содержание работы 29
2.4. Индивидуальные задания 30
2.5. Дополнительное задание для «мастеров» 33
2.6. Пример программы 33
2.7. Типичные ошибки при выполнении работы 35
2.8. Требования к студентам 35

Практическая работа 3
Программирование задач с разветвлениями вычислительного процесса

3.1. Цель работы 36
3.2. Справочные сведения 36
3.3. Содержание работы 39
3.4. Индивидуальные задания 41

3.5. Дополнительное задание для «мастеров» .-. 41
3.6. Пример программы 41
3.7. Типичные ошибки при выполнении работы 45
3.8. Требования к студентам 45

Практическая работа 4

Программирование задач с циклами

4.1. Цель работы 47
4.2. Справочные сведения 47
4.3. Содержание работы 49
4.4. Индивидуальные задания 50
4.5. Дополнительное задание для «мастеров» 58
4.6. Пример программы 58
4.7. Типичные ошибки при выполнении работы 59
4.8. Требования к студентам 59

Практическая работа 5

Функции в языке С

5.1. Цель работы 61
5.2. Справочные сведения 61
5.2.1. Решение трансцендентного уравнения методом последовательных приближений 62
5.2.2. Численное интегрирование методом Симпсона 63
5.2.3. Численное интегрирование дифференциального уравнения методом Рунге—Кутта 63
5.3. Содержание работы для студентов специальности 2203 64
5.4. Индивидуальные задания для студентов специальности 2203 65
5.5. Пример программы для студентов специальности 2203 71
5.6. Содержание работы для студентов специальностей 2201, 2202 72
5.7. Дополнительное задание для «мастеров» специальностей 2201, 2202 72
5.8. Пример программы для студентов специальностей 2201, 2202 73
5.9. Типичные ошибки при выполнении работы 74
5.10. Требования к студентам 75

Практическая работа 6

Многомодульный программный проект

6.1. Цель работы 76
6.2. Справочные сведения 76
6.3. Содержание работы 78
6.4. Индивидуальные задания 79
6.5. Дополнительное задание для «мастеров» 80
6.6. Пример программы 80
6.7. Типичные ошибки при выполнении работы 83
6.8. Требования к студентам 83

Практическая работа 7
Одномерные массивы

7.1. Цель работы....85
7.2. Справочные сведения....85
7.2.1. Описание массивов и работа с массивами с помощью индексов и указателей....85
7.2.2. Динамические массивы....86
7.3. Содержание работы....87
7.4. Индивидуальные задания....87
7.5. Дополнительное задание для «мастеров»....91
7.6. Пример программы....91
7.7. Типичные ошибки при выполнении работы....92
7.8. Требования к студентам....92

Практическая работа 8
Двухмерные массивы

8.1. Цель работы....93
8.2. Справочные сведения....93
8.3. Содержание работы....94
8.4. Индивидуальные задания....94
8.5. Дополнительное задание для «мастеров»....100
8.6. Пример программы....100
8.7. Типичные ошибки при выполнении работы....101
8.8. Требования к студентам....101

Практическая работа 9
Символы и строки

9.1. Цель работы....102
9.2. Справочные сведения....102
9.3. Содержание работы....104
9.4. Индивидуальные задания....105
9.5. Пример программы....ПО
9.6. Типичные ошибки при выполнении работы....111
9.7. Требования к студентам....111

Приложение 1. Операции языка С и их приоритеты....113
Приложение 2. Преобразование данных при выполнении операций....116
Список литературы....118

Учебное издание

Эпштейн Марк Семенович

Практикум по программированию
на языке С

Учебное пособие

Редактор *И. В. Могилевец*
Технический редактор *О.Н.Крайнова*
Компьютерная верстка: *Л.М.Беляева*
Корректоры *Н.П.Захарова, Н.Л.Котелина*

Изд. № 101110429. Подписано в печать 16.04.2007. Формат 60x90/16.
Гарнитура «Тайме». Печать офсетная. Бумага тип. № 2. Усл. печ. л. 8,0.
Тираж 3 000 экз. Заказ № 18915.

Издательский центр «Академия», www.academia-moscow.ru
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.0047963.07.04 от 20.07.2004
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (495)330-1092, 334-8337.

Отпечатано в ОАО «Саратовский полиграфический комбинат», www.sarpk.ru
410004, г. Саратов, ул. Чернышевского, 59.