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

Кафедра «Вычислительные методы и программирование»

ПРОГРАММИРОВАНИЕ

Лабораторный практикум

для студентов 1–2-го курсов
всех специальностей БГУИР всех форм обучения

В 2-х частях

Часть 2

Основы программирования на алгоритмическом языке С

2-е издание, исправленное и дополненное

Под общей редакцией В.Л. Бусько

Минск 2003

УДК 681.3.06 (075)
ББК 32.973 я 73
П 78

Р е ц е н з е н т
зав. кафедрой ЭИ БГУИР, канд. техн. наук В.И. Комличенко

А в т о р ы :
В.Л. Бусько, А.Г. Корбит, И.Н. Коренская,
Т.М. Кривоносова, В.И. Убийконь

Программирование: Лаб. практикум для студентов 1–2-го
П 78 курсов всех специальностей БГУИР всех форм обучения. В 2 ч. Ч. 2. Основы программирования на алгоритмическом языке С. — 2-е изд., испр. и доп. / В.Л. Бусько, А.Г. Корбит, И.Н. Коренская и др.; Под общ. ред В.Л. Бусько. — Мн.: БГУИР, 2003. — 72 с.: ил.

ISBN 985-444-485-6 (ч. 2).

В практикуме содержатся краткие теоретические сведения по основам программирования на алгоритмическом языке С, даны индивидуальные задания и контрольные вопросы к каждой лабораторной работе.

Во вторую часть практикума вошло 8 лабораторных работ. Предназначен практикум для студентов 1–2-го курсов всех специальностей всех форм обучения.

УДК 681.3.06 (075)
ББК 32.973 я 73

Часть 1: Методические указания по курсу «Вычислительная техника и программирование» для студентов всех специальностей Ч. 1. / Сост. В.Л. Бусько, А.Б. Долгий, А.Р. Живицкий. Мн.: МРТИ, 1992.

ISBN 985-444-485-6 (ч. 2)
ISBN 985-444-269-1

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА № 1

Линейные и разветвляющиеся вычислительные процессы

Цель работы:

Изучить правила составления текстов программ на языке С: базовые типы данных, ввод-вывод данных, основные математические функции; операторы безусловного перехода и разветвлений (оператор выбора по условию if, оператор-переключатель switch). Научиться программировать линейные и разветвляющиеся алгоритмы.

Краткие теоретические сведения

Язык С отражает возможности современных компьютеров. Программы на С отличаются компактностью и быстротой исполнения. Структура языка С побуждает программиста использовать в своей работе нисходящее программирование, структурное программирование, пошаговую разработку модулей.

Большинство трансляторов языка С — компиляторы. Система программирования языка включает препроцессор, компилятор, редактор связей, библиотекарь, редактор текста, отладчик и интегрированную управляющую среду.

Алфавит языка С

В языке С используются наборы символов:

1) прописные (**A, …, Z**) и строчные (**a, …, z**) буквы латинского алфавита;
2) арабские цифры от **0** до **9**;
3) специальные символы:

+ (плюс), – (минус), ***** (звездочка), ***/*** (дробная черта), **=** (равно), **>** (больше), **<** (меньше), **;** (точка с запятой), **&** (амперсант), **[ ]** (квадратные скобки), **{ }** (фигурные скобки), **( )** (круглые скобки), _ (знак подчеркивания), (пробел), **.** (точка), **,** (запятая), **:** (двоеточие), # (номер), **%** (процент), **~** (поразрядное отрицание), **?** (знак вопроса), **!** (восклицательный знак), **** (обратный слеш).

Классификация данных

В языке С применяются данные двух категорий: простые (скалярные) и сложные (составные) типы данных. К основным (базовым) типам данных относятся целый, вещественный и символьный типы. В свою очередь, данные целого типа могут быть короткими (**short**), длинными (**long**) и беззнаковыми (**unsigned**). Приведем размеры и возможные диапазоны базовых типов данных (табл. 1).

Сложные типы данных подразделяются на массивы, структуры (**struct**), объединения или смеси (**union**), перечисления (**enum**). Более

подробно они рассмотрены в других лабораторных работах данного практикума.

Таблица 1

Наименование типа	Тип данных	Объем памяти, байт	Диапазон значений
Символьный	char	1	–128 ... 127 (0 ... 255)
Целый	int	2	–32768 ... 32767
Короткий	short	2(1)	–32768 ... 32767 (–128 ... 127)
Длинный	long	4	–2147483648 ... 2147483647
Беззнаковый целый	unsigned int	2	0 ... 65535
Беззнаковый длинный	unsigned long	4	0 ... 424967295
Вещественный	float	4	$3{,}14 \cdot 10^{-38}$ ... $3{,}14 \cdot 10^{38}$
Вещественный с двойной точностью	double	8	$1{,}7 \cdot 10^{-308}$ ... $1{,}7 \cdot 10^{308}$

Декларирование объектов

Все объекты (переменные, массивы и т.д.), с которыми работает программа в С, необходимо декларировать. При декларировании объекты можно инициализировать (задавать начальные значения).

Например: int j=10, m=3, n;
float c=–1.3, l=–10.23, n;

При декларировании объектов используются их идентификаторы, которые могут включать цифры (**0...9**), латинские прописные (**A...Z**) и строчные (**a...z**) буквы, символ подчеркивания (_). Первый символ идентификатора не может быть цифрой. **В языке С буквы нижнего регистра (a...z) отличаются от букв верхнего регистра (A...Z),** т.е. PI, Pi и pi — различные идентификаторы. Принято использовать в идентификаторах переменных строчные буквы, а в именованных константах — прописные.

Например: const float PI=3.1415926;
float pi=3.14;

Длина идентификатора определяется реализацией транслятора С и редактора связей (компоновщика). Современная тенденция — снятие ограничений длины идентификатора.

Разделители идентификаторов объектов:

— пробелы;

— символы табуляции, перевода строки и страницы;

— комментарии (играют роль пробелов).

Комментарий — любая последовательность символов, начинающаяся парой символов /* и заканчивающаяся парой символов */ или начинающаяся // и до конца текущей строки.

Структура программы

Программа, написанная на языке С, состоит из директив препроцессора, объявлений глобальных переменных, одной или нескольких функций, среди которых одна главная (main) функция управляет работой всей программы.

Общая структура программы на языке С имеет вид:

<директивы препроцессора>
<определение типов пользователя — typedef>
<прототипы функций>
<определение глобальных объектов>
<функции>

Функции, в свою очередь, имеют структуру:

<класс_памяти> <тип> <имя функции> (<объявление параметров>)
{ — начало функции
<определение локальных объектов>
<операции и операторы>
} — конец функции

Работа с функциями пользователя рассматривается в лабораторной работе № 5.

Кратко рассмотрим основные части общей структуры программы.

Перед компиляцией программы на языке С автоматически выполняется предварительная (препроцессорная) обработка текста программы. С помощью директив препроцессора задаются необходимые действия по преобразованию текста программы перед компиляцией.

Директивы записываются по следующим правилам:

— все препроцессорные директивы начинаются с символа #;

— все директивы начинаются с первой позиции;

— сразу за символом # должно следовать наименование директивы, указывающее текущую операцию препроцессора.

Наиболее распространены директивы **#include** и **#define**.

Директива **#include** используется для подключения к программе заголовочных файлов (обычных текстов) с декларацией стандартных библиотечных функций. При заключении имени файла в угловые скобки **< >** поиск данного файла производится в стандартной директории с этими файлами. Если же имя файла заключено в двойные кавычки **" "**, то поиск данного файла осуществляется в текущем директории.

Например:

#include <stdio.h> — подключение файла с объявлением стандартных функций файлового ввода-вывода;
#include <conio.h> — функции работы с консолью;

#include <graphics.h> — графические функции;
#include <math.h> — математические функции.

Директива **#define** (определить) создает макроконстанту и ее действие распространяется на весь файл.

Например: #define PI 3.1415927

В ходе препроцессорной обработки программы идентификатор PI заменяется значением 3,1415927.

Пример программы:

```
#include <stdio.h>
#include < conio.h>                    // Директивы препроцессора
#define PI 3.1415927
  void main()                          // Заголовок главной функции
  {                                    // Начало функции
    int num;                           // Декларирование переменной num
    num=13;                            // Операция присваивания
    clrscr();                          // Очистка экрана
    printf(" \n    Число pi=%9.7f \n %d — это опасное число \n", PI, num);
   }                                   // Конец функции
```

В первых двух строках программы указаны директивы препроцессора #include, по которым происходит подключение заголовочных файлов, содержащих декларации функций ввода-вывода (stdio.h) для функции printf() и работы с консолью (conio.h) для функции clrscr(). Следующая директива создает макроконстанту PI и подставляет вместо ее имени значение 3,1415927 по всему тексту программы. В главной функции main декларируется переменная целого типа num. Далее этой переменной присваивается значение 13. Функция printf выводит на экран строки:

Число pi = 3.1415927
13 — это опасное число

Как видно, функция представляет собой набор операций и операторов, каждый из которых оканчивается символом **;** (точка с запятой).

В тексте программы использованы комментарии между парой символов /* */ и после пары символов ***//***.

Функции вывода информации

Для вывода информации используются следующие функции:
putchar() — вывод одиночного символа без перехода на новую строку;
puts() — вывод строки символов с переходом на начало новой строки;
printf() — форматированный вывод данных.

Ее формат:

printf (<управляющая строка>, <список аргументов>);

Управляющая строка заключается в кавычки и указывает компилятору вид выводимой информации. Она может включать спецификации преобразования и управляющие символы.

Спецификация преобразования имеет вид

% <флаг> <размер поля . точность> спецификация

где **флаг** может принимать следующие значения:

– — выравнивание влево выводимого числа (по умолчанию выполняется выравнивание вправо);

+ — выводится знак положительного числа;

размер поля — задает минимальную ширину поля, т.е. длину числа; при недостаточной ширине поля выполняется автоматическое расширение;

точность — задает точность числа, т. е. количество цифр в его дробной части;

спецификация — указывает вид выводимой информации. Ниже приведены основные форматы функции печати:

Таблица 2

Формат	Тип выводимой информации
%d	Десятичное целое число
%c	Один символ
%s	Строка символов
%e	Число с плавающей точкой (экспоненциальная запись)
%f	Число с плавающей точкой (десятичная запись)
%u	Десятичное число без знака
%o	Восьмеричное число без знака
%x	Шестнадцатеричное число без знака

Для длинных чисел (**long, double)** — используется дополнительный формат ***l***.

Например: ***%ld*** — длинное целое, ***%lf*** — вещественное число с удвоенной точностью.

При необходимости вывода управляющих символов (**%** **** и т.п.) их нужно указать 2 раза.

Например: printf("Только %d%% предприятий не работало. \n",5);

Получим: Только 5% предприятий не работало.

Управляющая строка может содержать **управляющие символы:**

\n — переход на новую строку;

\t — горизонтальная и **\v** — вертикальная табуляция;

\b — возврат назад на один символ;

\r — возврат в начало строки;

\f — прогон бумаги до начала новой страницы;
\a — звуковой сигнал;
\ddd — 8-ричный ASCII-код;
\xhhh — 16-ричный код;
\? — знак вопроса.

Список аргументов — печатаемые объекты (константы, переменные или выражения), вычисляемые перед выводом. Количество аргументов и их типы должны соответствовать спецификациям преобразования в управляющей строке.

Пример:

```
#include <stdio.h>
#define PI 3.1415926
  main()
    {
      int number=5, cost=11000, s=-777;
      float bat=255, x=12.345;
      printf ("%d студентов съели %f бутербродов. \n", number, bat);
      printf ("Значение числа pi равно %f. \n", PI);
      printf ("Стоимость этой вещи %d %s.\n", cost,"Руб.");
      printf ("x = %-8.4f s = %5d %8.2f ", x, s, x);
    }
```

В результате выполнения последней функции printf () на экране будет выведено:

x = 12.3450 s = –777 12.34

Функции ввода информации

Функция **getch ()** используется для ввода одиночных символов.

Функция **gets ()** обеспечивает ввод строки символов до нажатия клавиши ENTER.

Функция **scanf** предназначена для форматированного ввода информации любого вида. Общий вид функции:

scanf (<управляющая строка>, < список адресов>);

Для нее, как и для функции printf (), указывается управляющая строка. Однако функция scanf() в отличие от функции printf () использует в списке адресов указатели на переменные, т.е. их адреса. Для обозначения указателя перед именем переменной записывается символ &, обозначающий адрес переменной. Для ввода значений строковых переменных символ & не используется. При использовании формата %s строка вводится до первого пробела. Вводить данные можно как в одной строке через пробел, так и в разных строках.

Данную особенность иллюстрирует следующий участок программы:

```
int course;
float grant;
```

```
char name[20];
  printf ( "Укажите ваш курс, стипендию, имя \n");
  scanf ( "%d%f", &course, &grant);
  scanf ( "%s", name);                    // отсутствует & для массива символов
```

Стандартные математические функции

Декларации математических функций языка С содержатся в файле **math.h**. В последующих записях аргументы **x** и **y** имеют тип **double**, кроме abs(), параметр **n** имеет тип **int**. Аргументы тригонометрических функций задаются в радианах ($2p$ радиан = 360^o). Все приведенные математические функции возвращают значение (результат) типа **double**.

Таблица 3

Математическая функция	Имя функции в языке С	Математическая функция	Имя функции в языке С
$\sqrt{x}$	sqrt(x)	arcsin(x)	asin(x)
|x|	fabs(x)	arctg(x)	atan(x)
e^x	exp(x)	arctg(x/y)	atan2(x,y)
x^y	pow(x,y)	sh(x)=1/2 (e^x-e^{-x})	sinh(x)
ln(x)	log(x)	ch(x)=1/2 (e^x+e^{-x})	cosh(x)
lg_{10}(x)	log10(x)	tgh(x)	tanh(x)
sin(x)	sin(x)	Остаток от деления x на y	fmod(x,y)
cos(x)	cos(x)	Наименьшее целое ≥ x	ceil(x)
tg(x)	tan(x)	Наибольшее целое ≤ x	floor(x)

Операция присваивания

Операция присваивания имеет две формы записи:

1. ***Полная*** форма:

имя_переменной =выражение;

Сначала вычисляется выражение, а затем результат присваивается имени_переменной.

Например: y=(x+2)/(3*x)–5;

С помощью одного оператора можно присвоить одно значение нескольким переменным, например: x=y=z=0; /* x, y, z=0 */

или z=(x=y)*5; — сначала переменной **x** присваивается значение переменной **y**, далее вычисляется выражение **x*5** , и результат присваивается переменной **z**.

2. ***Сокращенная*** форма:

имя_переменной операция=выражение;

где **операция** — одна из арифметических операций (**+**, –, ***, /, %**);

Например: x*=5; // x=x*5;

```
s+=7;          // s=s+7;
y/=x+3;        // y=y/(x+3);
```

Сокращенная форма операции присваивания применяется тогда, когда переменная используется в обеих частях полной формы данного оператора.

В языке С существуют операции **уменьшения** (**--**) и **увеличения** (**++**) значения переменной на 1. Операции могут быть **префиксные** (++i и --i) и **постфиксные** (i++ и i--). При использовании данной операции в выражении в случае префиксной операции сначала выполняется сама операция (изменяется значение i), и только потом вычисляется выражение. В случае постфиксной операции — операция применяется после вычисления выражения.

Например:

```
b=7;
n=1;
1. c=b*++n;      // n=n+1; c=b*n;   т.е. c=14
2. c=b*n++;      // c=b*n; n=n+1;   т.е. c=7
```

Операторы перехода

Оператор безусловного перехода

goto < метка>;

Управление передается оператору с данной меткой:

<метка>: оператор;

В языке С метка не декларируется.

Оператор условного перехода **if** применяется для выбора одной из ветвей вычислений.

Общая форма записи:

```
if ( условие ) оператор_1;
  else оператор_2;
```

Например:

```
if(x>y) max=x;
  else max=y;
```

Если **оператор_1** или **оператор_2** содержит два и более операторов, то они заключаются в фигурные скобки **{ }**, т.е. применяется составной оператор. Оператор if проверяет истинность или ложность условия. Если условие истинно (не равно 0), то выполняется **оператор_1**, иначе, при ложности условия (=0) выполняется **оператор_2**.

Вторая часть оператора (**else оператор_2;**) может отсутствовать, такую форму называют «**сокращенной»**. Тогда в случае ложности условия управление передается на следующий за if оператор.

Если **оператор_1** и **оператор_2** в свою очередь являются операторами if, то такой оператор называют **вложенным**. При этом ключевое слово else принадлежит ближайшему предшествующему if.

Общий вид вложенного оператора if:

if (условие_1) оператор_1;
else if(условие_2) оператор_2;
else оператор_3;

Например: найти наибольшее значение из трех чисел x,y,z.

```
if (x>y)
        if(x>z) max=x;
          else max=z;
              else if(y>z) max=y;
                    else max=z;
```

В качестве **условий** в языке С используют следующие **операции отношений**:
< (меньше), **<=** (меньше или равно), **>** (больше), **>=** (больше или равно), **!=** (не равно), **==** (равно). Пары символов разделять нельзя.

Общий вид **условия**:

<выражение_1> <знак_операции> <выражение_2>

Операндами могут быть любые простые типы. Значения операндов после вычисления перед сравнением преобразуются к одному типу. В качестве условий используются и более сложные выражения, содержащие логические операции. Приведем их перечень в порядке убывания приоритета: **!** (отрицание или логическое НЕТ), **&&** (конъюнкция или логическое И), **||** (дизъюнкция или логическое ИЛИ). Пары символов **&&** и **||** разделять нельзя.

Например: (0<x)&&(x<=100)
((!x)&&(y>0)||((z==1)&&(k>0))

Выражения вычисляются слева направо, причем их вычисление прекращается, как только результат становится известен.

Триадный оператор (оператор условного перехода **?**). Его форма:

имя_переменной =условие ? выражение_1 : выражение_2;

Если условие истинно, то имени_переменной присваивается результат выражения_1, иначе — выражения_2.

Например: найти наибольшее из двух чисел: max=a>b ? a : b;

Оператор выбора switch

Общая форма оператора выбора:

```
switch(выражение)
 {
  case const_1: операторы;  break;
                ...
  case const_N: операторы;  break;
  default: операторы;
 }
```

где const_1…const_N — целые или символьные константы; **default** — выполняется, если результат выражения не совпал ни с одной константой;

может отсутствовать; **break** — оператор завершения работы switch. После выполнения одной из ветвей case все остальные ветви будут опущены.
Если оператор break не записан, то выполняются операторы следующей ветви case. Оператор switch проверяет, совпадает ли значение выражения с одним из значений приведенных ниже констант. При совпадении выполняются операторы, стоящие после совпавшей константы.

Например:

```
switch(i)
        {
         case 1: f=pow(x,2);
            break;
         case 2: f=fabs(x);
            break;
         case 3: f=sqrt(x);
            break;
         default: printf("Ошибка!");
            exit(1);
        }
f=f+5;
```

Пример линейного алгоритма

Вычислить $h = \frac{x^{y+1} + e^{y-1}}{1 + x * |y - tgz|} * (1 + |y - x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}$,

при x=2,444, y=0,00869, z=–130, должно быть получено: –0.49871.

Текст программы может иметь следующий вид:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define x 2.444
#define y 0.00869
#define z –130.0

void main(void)
{
double rezult,dop,a,b,c;
clrscr( );                              // Очистка экрана
puts("  Линейная программа  ");
dop=fabs(y–x);
a=pow(x,y+1)+exp(y–1);
b=1+x*fabs(y–tan(z));
c=0.5*pow(dop,2)–pow(dop,3)/3;
rezult=a/b*(1+dop)+c;
```

```
        printf("\a\n Ответ: rezult=%lf, Press any key...", rezult);
        getch( );                       // Задержка до нажатия любой клавиши
        }
```

Пример использования оператора if

Вычислить значение функции F. Предусмотреть вывод сообщения о том, по какой ветви происходило вычисление.

$$F=\begin{cases} \dfrac{a\cdot x+tgy}{5-2x} & \text{при} \quad x>0,\ y<0, \\ \max\left(\sqrt[3]{x^2},\ \cos y^2\right) & \text{при} \quad x<0,\ y>0, \\ \min\left(0{,}5x-2\sin^2 y,\ e^y\right) & \text{при} \quad x>0,\ y>0. \end{cases}$$

Текст программы может иметь следующий вид:

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define A 1
#define C 3

double max(double m,double n)    // Функция max с параметрами m и n для
 {                               // нахождения максимального значения
   if (m>n) return m;
     else return n;
 }

double min(double m, double n)  // Функция min с параметрами m и n для
 {                              // нахождения минимального значения
   if (m<n) return m;
     else return n;
 }

void main()
 {
  double x,y,f;                         // Декларирование переменных x,y,f
  clrscr();
  puts("Введите значения x и y");
  scanf("%lf %lf",&x,&y);               // Ввод значений x и y
  if ((x>0)&&(y<0))
    {
       f=(A*x+tan(C*y))/(5-2*x);
       puts("F=(a*x+tg(c*y))/(5-2*x)");
```

```
   }

else if ((x<0)&&(y>0))
   {
        f=max(pow(x,2.0/3.0),cos(y*y));                    // Вызов функции max
        puts("F=max(pow(x,2/3),cos(y*y))");
  }
else if ((x>0)&&(y>0))
  {
        f=min(0.5*x-2*pow(sin(y),2),exp(y));               // Вызов функции min
        puts("F=min(0.5*x-2*pow(sin(y),2),exp(y))");
  }
else
  {
        puts("Функция F не определена \n Press any key...");
        getch();
        exit(1);              // Принудительное завершение программы
  }
 printf("ОТВЕТ: F=%lf,\n Press any key...\n",f);
 getch();
}
```

Работа интегрированной среды программирования Borland C++, в которой происходит выполнение программы, а также порядок набора текста, компиляции, редактирования, отладки и выполнения программы приведены в **Приложении.**

Варианты индивидуальных заданий

Составить программу для определения значения функции. Предусмотреть вывод сообщения о том, по какой ветви происходило вычисление значения аргумента функции x.

1. $y=(\ln(1+x^2)+\cos(x+1))^{e^{k\cdot x}}$, где $x=\begin{cases} k\cdot z^3 & \text{при} \quad k<1, \\ z\cdot(z+1) & \text{при} \quad k\geq 1. \end{cases}$

2. $y=\dfrac{a\cdot x+b\cdot x\cdot\cos\sqrt{x}}{x+a\cdot b}$, где $x=\begin{cases} \sqrt{a^2+b^2\cdot z} & \text{при} \quad z<a\cdot b, \\ \sin^2 z+|a\cdot b\cdot z| & \text{при} \quad z\geq a\cdot b. \end{cases}$

3. $y=-p+\cos^2 x^3+\sin^3 x^2$, где $x=\begin{cases} z/b & \text{при} \quad z<1, \\ \sqrt{(z\cdot b)^3} & \text{при} \quad z\geq 1. \end{cases}$

4. $y=\cos^3 x^2+\sin^2 x^3$, где $x=\begin{cases} z^3+0,2 & \text{при} \quad z<1, \\ z+\ln z & \text{при} \quad z\geq 1. \end{cases}$

5. $y = \ln(x+0{,}5) + (e^{x} - e^{-x}),$ где $x = \begin{cases} -z/3 & \text{при } z < -1, \\ |z| & \text{при } z \ge -1. \end{cases}$

6. $y = \frac{2}{3}\sin^2 x - \frac{3}{4}\cos^2 x,$ где $x = \begin{cases} z & \text{при } z < 0, \\ \sin z & \text{при } z \ge 0. \end{cases}$

7. $y = \sin^3(c \cdot x + d^2 + k \cdot x^2),$ где $x = \begin{cases} z^2 - z & \text{при } z < 0, \\ z^3 & \text{при } z \ge 0. \end{cases}$

8. $y = \sin^2 x + \cos^5 x^3 + \ln x^{2/5},$ где $x = \begin{cases} 2z+1 & \text{при } z \ge 0, \\ \ln(z^2 - z) & \text{при } z < 0. \end{cases}$

9. $y = \frac{1}{\cos x} + \ln\left|tg\frac{x}{2}\right|,$ где $x = \begin{cases} z^b + \left|\frac{b}{2}\right| & \text{при } z \le 0, \\ \sqrt{z} & \text{при } z > 0. \end{cases}$

10. $y = \frac{e^{\sin^3 x} + \ln(x+1)}{\sqrt{x}},$ где $x = \begin{cases} z-1 & \text{при } z \ge 1, \\ z^2 + 1 & \text{при } z < 1. \end{cases}$

11. $y = \frac{2e^{-3x} - 4x^2}{\ln|x| + x},$ где $x = \begin{cases} \frac{1}{z^2 + 2z} & \text{при } z > 0, \\ 1 - z^3, & \text{при } z \le 0. \end{cases}$

12. $y = \sin^3(x^2 - 1) + \ln|x| + e^{x},$ где $x = \begin{cases} z^2 + 5 & \text{при } z \le 0, \\ \frac{1}{\sqrt{z-1}} & \text{при } z > 0. \end{cases}$

13. $y = \sin(n \cdot x) + \cos(k \cdot x) + \ln(m \cdot x),$ где $x = \begin{cases} e^{z} + z & \text{при } z > 1, \\ z^2 + 1 & \text{при } z \le 1. \end{cases}$

14. $y = \cos 5x + \sin\frac{1}{5}x + e^{x},$ где $x = \begin{cases} \sqrt{z} & \text{при } z > 0, \\ (3z^3 - z) - 5 & \text{при } z \le 0. \end{cases}$

15. $y = x(\sin x + e^{-(x+3)}),$ где $x = \begin{cases} -3z & \text{при } z > 0, \\ z^2 & \text{при } z \le 0. \end{cases}$

16. $F = \begin{cases} \min(x^2, y^2) + a & \text{при } a < 0, \\ \max(y, x, a) & \text{при } a = 0, \\ |x - y| + y(x + \sqrt{a^3}) & \text{при } a > 0. \end{cases}$

17. $F = \begin{cases} \min(x^2, y^2) + a & \text{при } a < 0, \\ \max(y, x, a) & \text{при } a = 0, \\ |x - y| + y(x + \sqrt{a^3}) & \text{при } a > 0. \end{cases}$

18. $$F=\begin{cases}\max(x,y)+\sqrt{x}, & \text{при } |x|+|y|\le 1 \text{ или } x\ge 0\\ \min(x,y)+\sin^2 x-\cos y^2, & \text{при } |x|+|y|>0 \text{ или } x<0, y<0\\ e^{x^2+|y|}, & \text{иначе}\end{cases}$$

19. $$F=\begin{cases}\max\ (x,\ y+\sqrt{x}) & \text{при } x>0,\ y\ge 0,\\ \min\ (\sin^2 x,\ \cos y^2) & \text{при } x<0,\\ 0,5x+e^y & \text{иначе.}\end{cases}$$

20. $$F=\begin{cases}\min\ (0,9y;\ e^{2x-3}) & \text{при } x\le 0,\\ \dfrac{2\cos(x-p/6)+\sqrt[3]{y}}{5-2x} & \text{при } x\ge 0,\ y>0,\\ \max\ (\sin y^2,\ \cos^2 x) & \text{иначе.}\end{cases}$$

Контрольные вопросы

1. Какие типы переменных применяются в языке С?
2. Перечислите основные отличия функций printf() и scanf().
3. Какие формы записи операторов if применяются в языке С?
4. Чем отличается оператор if от оператора switch?

ЛАБОРАТОРНАЯ РАБОТА № 2

Программирование циклических вычислительных процессов

Цель работы:

Изучить циклические операторы for, while, do–while, научиться составлять и программировать циклические алгоритмы.

Краткие теоретические сведения

Операторы циклов применяют, когда надо повторить некоторые действия (операторы и операции) несколько раз, и такие участки алгоритмов называют циклами.

Оператор цикла for

Основная форма оператора цикла for имеет вид

for (выражение_1; выражение_2; выражение_3)
оператор;

где **выражение_1** — начальное значения параметра цикла;
выражение_2 — проверка условия на продолжение цикла;
выражение_3 — изменение параметра цикла (коррекция);
оператор — простой или составной оператор языка С.

Схема работы оператора следующая: только один раз вначале вычисляется выражение_1, затем проверяется выражение_2, и если оно — «истина», то выполняется циклический участок программы, затем производится коррекция параметра, и так до тех пор, пока выражение_2 не примет значение «ложь».

Например:
```
for (k=1; k<5; k++)
    printf("\n %d", k);
```
В результате выполнения этого оператора печатаются в столбик цифры от 1 до 4.

В качестве параметра цикла можно использовать переменную любого базового типа.

Например:
```
for(ch='a';  ch<='z'; ch++)          // Вывод на экран букв
    printf(" %c",ch);                // латинского алфавита
```
Необходимо тщательно контролировать структуру циклов for в программе, чтобы не получился бесконечный цикл (из которого нет выхода).

Например:
```
for(k=10; k>6;k++)
    printf("бесконечный цикл \n");
```
Выйти из цикла досрочно можно следующими способами:
— по дополнительному условию;
— используя следующие операторы:

break; — выход из цикла, в котором находится break, управление передается на первый после цикла выполняемый оператор;

exit(int Kod); **—** выход из программы;

return; — выход из функции;

— с помощью оператора безусловного перехода **goto <метка>;**

Досрочное **завершение текущего циклического шага** возможно при помощи дополнительного условия или оператора **continue**, который прерывает выполнение текущего шага цикла, т.е. пропускает операторы оставшейся части цикла и передает управление в головной оператор цикла для коррекции параметра и проверки условия.

Передавать управление извне внутрь цикла запрещается.

Любое из выражений цикла for в круглых скобках может отсутствовать, но символ «;» опускать нельзя.

Например:

```
int i=0;
for(; i<3; i++)
    puts("Hello! ");
```

Циклические операторы while и do–while

Основная форма циклического оператора **while**:

```
while (условие)
    оператор;
```

где **оператор** — это простой, составной или пустой оператор.

Цикл выполняется до тех пор, пока условие принимает значение «истина», т.е. выражение в скобках возвращает ненулевой результат. Это цикл с предусловием — сначала проверяется условие, затем выполняется оператор. Поэтому цикл while не выполнится ни разу, если изначально результат вычисления условия будет равен 0.

Основная форма оператора **do — while**:

```
do
    оператор;
while (условие);
```

где **оператор** — это простой, составной или пустой оператор.

Оператор **do–while** — оператор цикла с постусловием, т.е. сначала выполняется оператор, а затем проверяется условие на истинность. Так как в цикле do–while условие проверяется в конце цикла, то цикл будет выполнен хотя бы один раз.

В циклах типа while и do–while допустимы те же способы досрочного выхода из цикла и досрочное завершение текущего шага цикла, как и в операторе for, но в последнем случае в отличие от цикла for управление передается на проверку условия. Для предотвращения бесконечного цикла внутри циклов while и do–while нужно предусмотреть изменение переменных, входящих в условие.

Например:

```
int i;
for (i=1;i<=300;i++)        // Печать целых чисел, кратных 5
  {
     if (i%5!=0) continue;
     printf("%5d",i);
       }
```

Примеры бесконечных циклов:

```
1)   for( ; ; )
          оператор;
2)   while(число_не_0)          // Всегда истинно!
          оператор;
3)   do
          оператор;
     while(число_не_0);          // Всегда истинно!
```

Среди операторов цикла обязательно должно быть условие выхода.

Вложенные циклы

В случае вложенных циклов один цикл находится внутри другого, например:

```
for(i=nn;i<nk;i++)
    for(j=mn;j<mk;j++)
        оператор;
```

где **оператор** — это простой, составной или пустой оператор. Внутренний цикл будет выполняться для каждого значения параметра i, удовлетворяющего условию внешнего цикла.

Пример:

```
int i,j;
for(i=1;i<10;i++)              // Печать таблицы умножения
  {
   for(j=1;j<4;j++)
       printf("\n %d*%d=%2d", i, j, i*j);
   printf("\n");
  }
```

Пример использования оператора for

Вычислить $S = \sum_{k=1}^{N} \frac{1}{k}$. На печать программа должна выводить промежуточные и окончательный результаты.

Текст программы может иметь вид

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

```
 void main(void)
{
 float s;
 int k,N;
 clrscr( );
        puts(" Введите N");
        scanf("%d",&N);
        for (s=0, k=1; k<=N; k++)     // В заголовке цикла можно выпол-
        {                             // нять и двойное присваивание
          s+=1.0/k;
          printf(" \n k=%d s=%f ", k, s);
         }
 printf("\n ОТВЕТ: s=%f, Press any key...",s);
 getch( );
}
```

Варианты индивидуальных заданий

Составить программу для определения таблицы значений функции **y** в произвольном диапазоне [**a**,**b**] изменения аргумента **x** с произвольным шагом **h**. Значения **a, b, h** вводятся с клавиатуры. Таблица должна содержать следующие столбцы: порядковый номер, значение аргумента **x**, значение функции, сообщение о возрастании или убывании функции, разность двух соседних значений функции.

Определить максимальное и минимальное значения функции.

1. $Y(x)=\dfrac{2\sin x}{(1-x)^2}$, a=–π; b=π; h=0,4.

2. $Y(x)=-\ln\left|2\sin\dfrac{x}{2}\right|$, a=0,7; b=1,8; h=0,1.

3. $Y(x)=\dfrac{x\sin(\frac{\pi}{4})}{1-2x\cos\frac{\pi}{4}+x^2}$, a=–0,5; b=2,5; h=0,2.

4. $Y(x)=(1-\dfrac{x^2}{4})\cos x-\dfrac{x}{2}\sin x$, a=–0,9; b=2,7; h=0,3.

5. $Y(x)=\dfrac{x\cos\frac{\pi}{4}-x^2}{1-2x\cos\frac{\pi}{4}+x^2}$, a=–2; b=0,8; h=0,2.

6. $Y(x)=(\dfrac{x^2}{4}+\dfrac{x}{2}-3)\cdot e^{\frac{x}{2}}$, a=–1,9; b=2,7; h=0,3.

7. $Y(x) = x^2\sqrt{15+10\sin(x+\pi)}$, a=–0,4π; b=0,4π; h=0,5.

8. $Y(x) = e^x \sin x$, a=–0,3π; b=1,3π; h= π/10.

9. $Y(x) = x^2 \cos x \sin x$, a=–π/2; b= π/2; h=π/10.

10. $Y(x) = x\log(|x-0{,}6|)$, a=–3; b=3; h=0,5.

11. $Y(x) = \frac{x}{2}\cos x - \sin x$, a=–π; b=π; h= π/6.

12. $Y(x) = e^x + \sqrt{1+e^{2x}} - 2$, a=–0,9; b=1, h=0,3.

13. $Y(x) = (1-\frac{x^2}{4})\cos x - \frac{x}{2}\sin x$, a=–0,9; b=2,7; h=0,3.

14. $Y(x) = \frac{1}{x^2 - x + 1}$, a=–0,1; b=2; h=0,1.

15. $Y(x) = \frac{\sin x \cos x}{\sqrt{x}}$, a=π; b=2π; h= π/15.

Значение аргумента **x** изменяется от **a** до **b** с шагом **h**. Для каждого **x** найти значение суммы S(x). Значения **a**,**b**,**h** и **n** вводятся с клавиатуры. Работу программы проверить для a=0,1; b=1,0; h=0,1; n выбрать максимально возможным!

16. $S(x) = \sum_{k=0}^{n}(-1)^k \frac{x^{2k+1}}{(2k+1)!}$.

17. $S(x) = \sum_{k=1}^{n}(-1)^k \frac{(2x)^{2k}}{(2k)!}$.

18. $S(x) = \sum_{k=0}^{n}(-1)^k \frac{x^{2k}}{(2k)!}$.

19. $S(x) = \sum_{k=0}^{n}(-1)^k \frac{2k^2+1}{(2k)!}x^{2k}$.

20. $S(x) = \sum_{k=1}^{n}(-1)^{k+1} \frac{x^{2k}}{2k(2k-1)}$.

Контрольные вопросы

1. Какой процесс называется «циклическим»?
2. Чем отличаются операторы while и do–while?
3. Объясните работу оператора for.
4. Поясните понятие «вложенный цикл»?

ЛАБОРАТОРНАЯ РАБОТА № 3

Программирование циклических процессов с использованием одномерных массивов и строк

Цель работы:

Изучить правила работы с одномерными массивами, а также особенности работы со строковыми объектами как одномерными символьными массивами.

Краткие теоретические сведения

Массив — конечномерная последовательность данных одного типа. Массив — объект сложного типа. Каждый элемент массива определяется именем массива и индексом (целое число), по которому к элементу массива производится доступ. Рассмотрим одномерные массивы. ***Индексы массивов в языке С начинаются с 0.*** В программе одномерный массив объявляется следующим образом:

<Тип> <имя массива>[размер];

где размер — количество элементов одномерного массива.

Размер массива может задаваться константой или константным выражением. Нельзя задавать массив переменного размера, для этого существует отдельный механизм — динамическое выделение памяти.

Пример объявления массива целого типа:

int ***a***[5];

в массиве ***a*** первый элемент ***a***[0], второй — ***a***[1], …, пятый — ***a***[4]. В языке С не проверяется выход индекса за пределы массива. Корректность использования индексов элементов массива должен контролировать программист.

Пример работы с одномерным массивом

В массиве целых чисел найти индекс и значение максимального элемента и переставить его с первым элементом. Программа также должна подсчитать количество положительных и отрицательных элементов данного массива.

Текст программы может быть следующим:

```
#include <stdio.h>
#include <conio.h>
 void main(void)
{                                  // Объявление с инициализацией,
 int a[4]={–1,–20,4,100};          // индексы принимают значения от 0 до 3
 int i,index,max,kp=0,ko=0,zam,n=4;
      clrscr();
```

```
puts("\n Исходный массив ");
for (i=0; i<n; i++)
        printf("%d ",a[i]);  // Вывод элементов исходного массива
        max=a[0];
        for (i=1; i<n; i++)
        {
        if (a[i]>max)
                {
                max=a[i]; index=i;   }
        }
      zam=a[0];a[0]=a[index];a[index]=zam;
        for (i=0;i<n;i++)
        {
         if (a[i]<0) ko++;
                 else kp++;
        }
puts("\n Результаты работы программы ");
  for (i=0; i<n; i++)
   printf("%d ",a[i]);                          // Вывод элементов массива
printf("\n положительных элементов: %d \n",kp);
printf("\n отрицательных элементов: %d \n\n Press any key...",ko);
getch();
}
```

Строки как одномерные массивы символов

Работа со строками в языке С реализована путем использования одномерных массивов типа char, т.е. строка символов — это одномерный массив типа char, заканчивающийся нулевым байтом. Нулевой байт — это байт, каждый бит которого равен нулю, при этом для нулевого байта определена символьная константа ´\0´ (признак окончания строки или нуль-терминатор). Поэтому, если строка должна состоять из k символов, то в описании массива необходимо указать размер k+1.

Например, описание char a[7], означает, что строка содержит шесть символов, а последний байт отведен под нуль-терминатор.

Строковая константа в языке С — это набор символов, заключенных в двойные кавычки. Например: «Лабораторная работа по строкам». В конце строковой константы явно указывать символ ´\0´ не нужно, так как это сделает компилятор языка С.

Строки можно инициализировать при декларировании, например:

```
char S1[10]= "123456789", S2[ ]= "12345";
```

в последнем случае размер строки будет установлен по фактическому количеству символов.

Для ввода строки с клавиатуры дисплея используются две стандартные библиотечные функции, прототипы которых приведены в файле stdio.h.

Функция **scanf**( ) вводит значения для строковых переменных спецификатором ввода %s. Но надо помнить, что функция scanf( ) вводит символы до появления первого символа «пробел».

Библиотечная функция **gets**( ) обеспечивает ввод строки с пробелами внутри этой строки. При этом ввод строки символов завершается нажатием клавиши ENTER.

Обе функции автоматически ставят в конец строки нулевой байт. И, кроме того, так как строка — это символьный массив, а имя массива — это указатель на его начало в памяти, то символ «&» перед именами строк при использовании этих функций указывать не надо.

Вывод строк выполняют функции **printf**( ) или **puts**( ). Обе функции выводят символьный массив до первого нулевого байта. Функция printf( ) не переводит курсор после вывода на начало новой строки, программист должен предусмотреть такой перевод в строке формата. Функция puts( ) автоматически переводит курсор после вывода строковой информации в начало новой строки.

Операции над строками как объектами сложного типа, рекомендуется выполнять, используя стандартные функции. Декларации функций для работы со строками размещены в файле string.h. Рассмотрим наиболее часто используемые:

1. Функция **strcpy**(S1, S2) — копирует содержимое строки S2 в строку S1.

2. Функция **strcat**(S1, S2) — присоединяет строку S2 к строке S1 и помещает ее в массив, где находилась строка S1, при этом строка S2 не изменяется. Нулевой байт, который завершал строку S1, заменяется первым символом строки S2.

3. Функция **strcmp**(S1, S2) сравнивает строки S1 и S2 и возвращает значение =0, если строки равны, т.е. содержат одно и то же число одинаковых символов; значение <0, если S1<S2; значение >0, если S1>S2.

4. Функция **strlen**(S) возвращает длину строки, т.е. количество символов, начиная с первого (S[0]) и до нуль-терминатора, который не учитывается.

5. Функции преобразования строки S в число:

— целое: int **atoi**(S);

— длинное целое: long **atol**(S);

— действительное: double **atof**(S);

при ошибке данные функции возвращают значение 0.

6. Функции преобразования числа V в строку S:

— целое: **itoa**(int V,char S,int kod);

— длинное целое: **ltoa**(long V,char S,int kod); 2<=kod<=36, для отрицательных чисел kod=10.

Операция sizeof

Для определения размера памяти, необходимого для размещения объектов в языке С, используется унарная операция **sizeof**(параметр), параметр — тип объекта или его идентификатор (только не имя функции). Операция sizeof вычисляет размер памяти в байтах, отводимый под объект. Если указан идентификатор сложного объекта (массив, структура, объединение), то результатом является размер всего сложного объекта. Например:

sizeof(int) результат 2 байта;
int b[5]; sizeof(b) результат 10 байт;
int c[3][4]; sizeof(c) результат 24 байта.

Указатели и операции над адресами

Обращение к объектам любого типа в языке С может проводиться по имени, как мы до сих пор делали, и по ***указателю*** (косвенная адресация).

Указатель — это переменная, которая может содержать адрес некоторого объекта в памяти компьютера, например, адрес другой переменной. И через указатель, установленный на переменную, можно обращаться к участку оперативной памяти, отведенной компилятором под ее значения.

Указатель объявляется следующим образом:

<тип> ***<идентификатор>;

Например:

```
int *a, *d;
float *f;
```

Здесь объявлены указатели a, d, которые можно инициализировать адресами целочисленных переменных, и указатель f, который можно инициализировать адресами вещественных переменных.

С указателями связаны две унарные операции: & и *. Операция & означает «взять адрес». Данная операция допустима только над переменными. Операция * — «значение, расположенное по указанному адресу», и работает следующим образом:

— определяется местоположение в оперативной памяти переменной типа указатель;

— извлекается информация из этого участка памяти и трактуется как адрес переменной с типом в объявлении указателя;

— производится обращение к участку памяти по выделенному адресу для проведения некоторых действий.

Пример:

```
int x,                  // переменная типа int
  *y;                   // указатель на элемент данных типа int
       y=&x;            // y — адрес переменной x
      *y=1;             // по адресу y записать 1, в результате x = 1
```

Связь указателей и массивов

Указатели и массивы в языке С тесно связаны между собой. Имя массива является указателем на его первый элемент, т.е. для массива int **v**[10] **v** и **v**[0] имеют одинаковые значения, т.к. адрес первого (с индексом 0) элемента массива — это адрес начала последовательно расположенных элементов массива. Рассмотрим обращение к элементам массива на примере. Пусть объявлены — массив из 100 объектов типа float и указатель на объект типа float:

```
float p[100];
float *q;
int i;
```

если выполнить операцию q=p; то обращения к элементу массива **p**: p[i] , *(q+i) и *(p+i) эквивалентны.

Таким образом, для любых указателей можно использовать две эквивалентные формы выражений для доступа к элементам массива: q[i] и *(q+i). Первая форма удобнее для читаемости текста, но вторая обычно эффективнее по быстродействию программы.

Очевидна эквивалентность выражений

```
 &q[0]   <->   &(*q)   <->   q
*q       <->   q[0]
```

Пример 1: Упорядочить по алфавиту массив строк (не более 20) длиной не более 10 символов в каждой.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
   void main()
{
   char s[20][10],r[10];
   int i,j,n;
   clrscr();
   puts(" Введите количество слов ";
   scanf("%d",&n);
   for(i=0;i<n;i++)
         scanf("%s",&s[i]);
   for(i=0;i<(n-1);i++)
    for(j=(i+1);j<n;j++)
```

```
            if(strcmp(s[i],s[j])>0)
            {
             strcpy(r,s[i]);
             strcpy(s[i],s[j]);
             strcpy(s[j],r);
            }
        for(i=0;i<n;i++)
           printf("\n %s",s[i]);
        getch();
         }
```

Пример 2: Проверить, является ли введенная строка (не более 80 символов) палиндромом (справа налево читается так же, как и слева направо).

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

void main(void)
{
  char s[80],s1[80];
  int i, k;
   clrscr();
   puts(" Введите исходную строку (не более 80 символов) ");
    gets(s);
        k=strlen(s);
        puts(" Результаты работы программы ");
         for (i=0; i<k; i++)
          printf("%c",s1[i]=s[k-i-1]);          // Переворачиваем строку s
        s1[k]='\0';                             // Устанавливаем конец строки
        printf(" %s\n",strcpy(s1,s) ? " YES — Palindrom! ":" NO ! ")
        printf("\n Press any key...");
  getch();
 }
```


Массивы указателей

В языке С можно использовать массивы указателей, элементы которых содержат, как правило, указатели на строковые данные. Объявляется такой массив, например, так char *m[5]. Здесь массив m[5] — массив, который может содержать пять адресов данных типа char.

Массив указателей можно при объявлении инициализировать, т.е. назначать его элементам конкретные адреса. С помощью массивов

указателей типа char* удобно получать доступ к строкам в виде массивов символов.

В качестве примера приведена программа, формирующая массив указателей с одновременной инициализацией его элементов. Программа выводит на экран номер строки, ее адрес и значение.

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

void main(void)
{
        int i,k;
        char *m[]={
                "Winter",
                "Spring",
                "Summer",
                "Automn"};
        k=sizeof(m)/sizeof(*m);
        printf("\n Количество строк = %d",k);
        for(i=0; i<k; i++)
           printf("\n Номер — %d; Адрес — %p; Строка: %s", i+1, m[i], m[i]);
 getch();
}
```

В результате получим:

```
 Количество строк = 4
 Номер — 1; Адрес — 0042007C; Строка: Winter
 Номер — 2; Адрес — 00420074; Строка: Spring
 Номер — 3; Адрес — 0042006C; Строка: Summer
 Номер — 4; Адрес — 00420064; Строка: Automn
```

Конкретные значения адресов зависят от ряда причин: архитектуры компьютера, типа и размера оперативной памяти и т.д.

Замечание. Функция printf() и спецификатор преобразования %S допускают использование в качестве параметра указатель на строку, однако при этом выводится не значение указателя m[i], а содержимое адресуемой им строки, в то время как спецификатор %P выводит значение указателя m[i].

Возможности массивов указателей проявляются также в тех случаях, когда его элементы адресуют либо элементы другого массива, либо указывают на начало одномерных массивов соответствующего типа.

В качестве примера приведена программа упорядочения одномерного массива по возрастанию и убыванию без перестановки его элементов.

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

```
#define b 6
void main ()
{  float array [ ]={5.0,2.0,3.0,1.0,6.0,4.0};
   float *pmin[b], *pmax[b], *e;
   int i,j;
     for (i=0;i<b;i++)
        pmin[i]=pmax[i]=&array[i];
for(i=0;i<b-1;i++)
        for(j=i+1;j<b;j++)
        {
      if(*pmin[i]<*pmin[j])
        { e=pmin[i];
          pmin[i]=pmin[j];
          pmin[j]=e;}
      if(*pmax[i]>*pmax[j])
            { e=pmax[i];
                 pmax[i]=pmax[j];
              pmax[j]=e;}
        }
printf(" \n По убыванию: \n");
        for(i=0;i<b;i++)
                printf("\t %5.3f",*pmin[i]);
printf("\n По возрастанию: \n");
        for(i=0;i<b;i++)
                printf("\t%5.3f",*pmax[i]);
}
```

Варианты индивидуальных заданий

1. Ввести целое число N. Выделить из этого числа цифры, кратные 3, и записать их в одномерный массив.

2. Для заданного целого числа N определить цифру a, наиболее часто встречающуюся в числе. Сформировать одномерный массив из 5 элементов: a, a^2, a^3, a^4, a^5.

3. Элементы заданного массива X циклически сдвинуть на K позиций вправо (влево).

4. Дано число N целого типа. Определить, симметрично ли оно, т.е. одинаковы ли цифры слева и справа (12321). Записать 3 последние цифры в одномерный массив.

5. Упорядочить элементы массива X по возрастанию.

6. Даны координаты n точек на плоскости: x_1, y_1, x_2, y_2,…. x_n, y_n. Найти номера двух точек, расстояние между которыми наибольшее.

7. Заданы два массива по N целых чисел. Найти наименьшее среди чисел первого массива, которое не входит во второй массив.

8. Дан массив из N целых чисел. Определить количество инверсий в этом массиве (т.е. таких пар элементов, в которых большее число находится слева от меньшего: $x_i > x_j$ при $i < j$).

9. Ввести строку символов. Определить длину введенной строки L, и если длина L четная, то удаляются 2 первых и 2 последних символа.

10. Ввести строку символов. Определить длину введенной строки L, и если длина L нечетная, то удаляется символ, стоящий посередине строки.

11. Ввести строку символов. Заменить в ней каждый второй символ ! на $.

12. Ввести строку символов. Заменить в ней пробелы на символ $.

13. Ввести строку символов. Определить длину введенной строки L, и если длина L>10, то удалить все цифры.

14. Ввести строку символов. Определить длину введенной строки L, и если длина L кратна 3, то удаляются все числа, делящиеся на 3.

15. Ввести строку символов. Определить длину введенной строки L, и если длина L кратна 5, то подсчитывается количество скобок всех видов.

16. Ввести строку символов. Определить длину введенной строки L, и если длина L кратна 4, то первая часть строки меняется местами со второй.

17. Ввести строку символов. Определить длину введенной строки L, и если длина L=10, то удаляются все буквы — A... Z.

18. Ввести строку символов. Определить длину введенной строки L, и если длина L >15, то удаляются все буквы — a…z.

19. В строке символов поменять местами символы на четных и нечетных позициях.

20. Ввести строку символов. Определить длину введенной строки L, и если длина L > 6, то выделяется подстрока в { } скобках.

Контрольные вопросы

1. Укажите типы массивов, применяемых в языке С.
2. Формы (способы) работы с элементами массива.
3. Как описываются строки в языке С?
4. Чем отличаются функции scanf() и gets(), printf() и puts()?

ЛАБОРАТОРНАЯ РАБОТА № 4

Программирование циклических процессов с использованием многомерных массивов. Динамическое распределение памяти

Цель работы:

Изучить работу с многомерными массивами, освоить возможности динамического размещения данных.

Краткие теоретические сведения

Кроме одномерных массивов возможна работа с многомерными массивами. Объявление многомерного массива:

<тип><имя>[<размер 1 >][<размер 2 >]…[<размер N>]={{список начальных значений}, {список начальных значений},…};

Наиболее быстро изменяется последний индекс элементов массива, поскольку многомерные массивы размещаются в памяти компьютера в последовательности столбцов.

Например, элементы двухмерного массива b[2][1] размещаются в памяти в следующем порядке:

b[0][0], b[0][1], b[1][0], b[1][1], b[2][0], b[2][1].

Следующий пример иллюстрирует определение массива целого типа, состоящего из трех строк и четырех столбцов, с инициализацией начальных значений:

```
int a[3][4] = {{0,1,2,0},{9,–2,0,0},{–7,1,6,8}};
```

Если в какой-то группе {…} отсутствует значение, то соответствующему элементу присваивается 0. Предыдущий оператор будет эквивалентен следующему определению:

```
int a[3][4] = {{0,1,2},{9,–2},{–7,1,6,8}};
```

При обработке двухмерных массивов используются вложенные циклы. Например, ввод массива int a[5][4]:

```
for(i=0;i<5;i++)
  for(j=0;j<4;j++)
    scanf("%d",&a[i][j]);
```

Пример программы

Создать двухмерный массив целых чисел NxM (N и M не более 50), используя функцию rand, и вывести на экран в форме матрицы, N,M ввести с клавиатуры:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define rnd (rand()/ 32768.0) /* rand — генератор случайных чисел от
                                 0 до int, rnd — от 0 до 1 */
```

```
        void main(void)
{ int i,j,n,m,a[50][50];
  puts("\n Input n, m: "); scanf(“%d %d”,&n,&m);
printf("\n Array a \n");
        for(i=0; i<n; i++)          // Оформление вывода в виде матрицы
            for(j=0; j<m; j++) {
                    a[i][j]=rnd*10-5;   // случайные числа от –5 до 5
                printf("%d%c", a[i][j], (j= =m-1)?’\n’:’ ‘);
                            }
        getch();
 }
```

Указатели на указатели

Связь указателей и массивов с одним измерением справедливо и для массивов с большим числом измерений. Например, рассмотрим двухмерный массив

```
float name[5][10];
```

Если рассматривать его как массив пяти массивов размерностью по десять элементов каждый, то очевидна схема его размещения в памяти — последовательное размещение «строк» элементов. Обращению к элементам name[i][j] соответствует эквивалентное выражение *(*(name+i)+j), а объявление этого массива указателем будет:

```
float **name;
```

Таким образом, имя двухмерного массива — имя указателя на указатель. Аналогичным образом можно установить соответствие между указателями и массивами с произвольным числом измерений. Количество символов «*» определяет уровень вложенности указателей друг в друга. При объявлении указателей на указатели возможна их одновременная инициализация.

Например:

```
int a=5;
int *p1=&a;
int **pp1=&p1;
int ***ppp1=&pp1;
```

Теперь присвоим целочисленной переменной ***a*** новое значение, например, 10. Одинаковое присваивание произведут следующие операции:

```
a=10;      *p1=10;     **pp1=10;  ***ppp1=10;
```

Для доступа к области памяти, отведенной под переменную ***a***, можно использовать и индексы. Справедливы следующие аналоги:

*p1	равносильно	p1[0]
**pp1	равносильно	pp1[0][0]
***ppp1	равносильно	ppp1[0][0][0]

Таким образом, указатели на указатели — это имена многомерных массивов.

Динамическое размещение данных

Если в задаче заранее неизвестно количество объектов и объект описан указателем, удобно использовать динамическое размещение данных.

Прототипы функций работы с динамической памятью находятся в библиотеке **alloc.h**, рассмотрим основные из них:

void ***calloc**(unsigned n, unsigned m); — возвращает указатель на начало области памяти для размещения n элементов по m байт каждый, при неудачном завершении возвращает значение NULL;

void ***malloc**(unsigned n); — возвращает указатель на блок памяти длиной n байт, при неудачном завершении возвращает значение NULL;

void ***realloc**(void *bf, unsigned n); — изменяет размер ранее выделенной памяти с адресом начала bf на n байт;

void **free**(void *bf); — освобождает ранее выделенный блок памяти с адресом bf;

coreleft(void); — возвращает значение объема неиспользованной памяти (тип возвращаемого результата unsigned — для моделей памяти tiny, small, medium; unsigned long — для других моделей памяти).

Пример выделения памяти для массива действительных чисел размером *n*:

```
float *x;                          // Указатель объекта типа float — x[0]
int n;                             // Количество элементов массива
                . . .
x=(float*)calloc(n,sizeof(float));     // Захват памяти для n элементов
                . . .
free(x);                           // Освобождение памяти
```

Пример работы со строковыми данными

Проверить, является ли введенная строка палиндромом (справа налево читается так же, как и слева направо).

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <calloc.h>
      void main(void)
{
  char *s;                                 // Объявление строки указателем
  int i, k;
    clrscr();
    puts("Vvedi stroky");     gets(s);
```

```
    k=strlen(s);
    s=(char*)calloc(k,sizeof(char));   // Захват памяти для строки длиной k
        for (i=0; i<(int)(k/2); i++)
            if(s[i]!=s[k-i-1])
                {
                puts("\t\a NO!!");
                getch();
                free(s);                          // Освобождение памяти
                return;
                }
    puts("\t\a YES — Palindrom!");
    getch();
    free(s);                                      // Освобождение памяти
}
```

Пример динамического размещения одномерного массива

Ввести массив действительных чисел размером n и вывести на экран.

```
    #include<stdio.h>
    #include<stdlib.h>
    #include<conio.h>
    #include<alloc.h>

      void main()
    {
      int i,n;
      float *a;
            puts("\n Введите размер массива n: ");
            scanf("%d",&n);
            printf("\n Свободная память -%d",coreleft());
            a=(float*)calloc(n,sizeof(float));          // Захват памяти
            printf("\n Введите элементы массива a: \n");
            for(i=0; i<n; i++)
                        scanf("%f",(a+i));              // scanf("%f",&a[i]);
            printf("\n Массив a \n");
            for(i=0; i<n; i++)
                        printf(" %6.3f \n", a[i]);
                  printf("\n Память после захвата -%d",coreleft());
            free(a);                                    // Освобождение памяти
       getch();
    }
```

Пример динамического размещения двухмерного массива

```
                . . .
            void main(void)
    { int i,j,n,m;
```

```
float **a;
        puts("\n Введите n,m: ");
        scanf("%d %d",&n,&m);
        printf("\n Свободная память -%d",coreleft());
        a=(float **)calloc(n,sizeof(float*));          // Захват памяти
        for(i=0; i<n; i++)
                a[i]=(float *)calloc(m,sizeof(float));
                        . . .
        for(i=0; i<n; i++)  free(a[i]);                 // Освобождение памяти
        free(a);
  getch();
}
```

Варианты индивидуальных заданий

1. В вещественной матрице размером NxN найти максимальный и минимальный элементы. Переставить строки, в которых они находятся. Если они находятся в одной строке, выдать об этом сообщение.

2. Квадратную вещественную матрицу A размером N возвести в K-ю степень, т.е. вычислить: $A^1=A$, $A^2=A\cdot A$, $A^3=A^2\cdot A$ и т.д.

3. Дана вещественная матрица размером NxM. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (один из них) оказался в верхнем левом углу.

4. Дана вещественная матрица размером NxM. Упорядочить ее строки по возрастанию наибольших элементов в строках матрицы.

5. Задан массив размером NxN, состоящий из 0 и 1. Повернуть элементы массива на 90^o по часовой стрелке.

6. Элемент матрицы назовем седловой точкой, если он наименьший в своей строке и наибольший (одновременно) в своем столбце (или наоборот, наибольший в своей строке и наименьший в своем столбце). Для заданной целочисленной матрицы размером NxM напечатать индексы всех ее седловых точек.

7. Дана вещественная матрица размером N, все элементы которой различны. Найти скалярное произведение строки, в которой находится наибольший элемент матрицы, на столбец с наименьшим элементом.

8. Определить, является ли заданная целочисленная квадратная матрица размером N ортонормированной, т.е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1.

9. Определить, является ли заданная матрица N-го порядка магическим квадратом, т.е. такой, в которой сумма элементов во всех строках и столбцах одинакова.

10. Дана целочисленная матрица размером N. Найти сумму наименьших элементов ее нечетных строк и наибольших элементов ее четных строк.

11. Дана действительная квадратная матрица порядка N. Рассмотрим те элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти сумму тех из них, которые расположены соответственно ниже, выше и на главной диагонали матрицы.

12. Дана вещественная квадратная матрица порядка N. Получить целочисленную квадратную матрицу, в которой элемент равен 1, если соответствующий ему элемент исходной матрицы больше элемента, расположенного на главной диагонали, и равен 0 в противном случае.

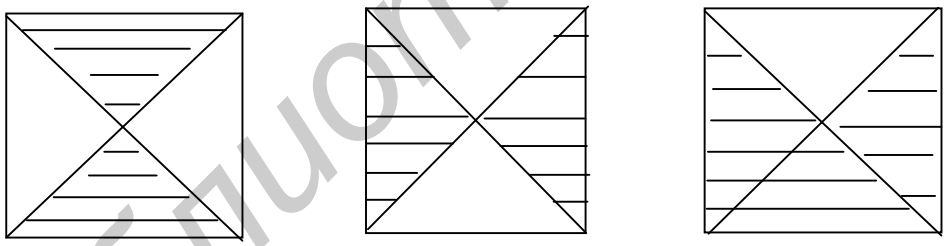
13. Дана квадратная целочисленная матрица порядка N. Упорядочить элементы в строках по возрастанию.

14. Дана действительная квадратная матрица порядка N. Найти сумму и произведение элементов, расположенных в заштрихованной части матрицы, см. рисунок «а».

15. То же, см. рисунок «б».

16. То же, см. рисунок «в».

17. Дана действительная квадратная матрица порядка N. Найти наименьшее и наибольшее из значений элементов, расположенных в заштрихованной части матрицы, см. рисунок «а».



а б в

18. То же, см. рисунок «б».

19. То же, см. рисунок «в».

20. Получить целочисленную квадратную матрицу порядка N, элементами которой являются числа 1, 2, 3, … , расположенные в ней по спирали.

Контрольные вопросы

1. Укажите способы декларирования двухмерных массивов.
2. Формы (способы) работы с элементами двухмерного массива.
3. Что такое указатель?

4. Операция sizeof() .

ЛАБОРАТОРНАЯ РАБОТА № 5

Программирование алгоритмов с использованием функций пользователя

Цель работы:

Познакомиться с механизмом составления и организации взаимодействия пользовательских функций языка С.

Краткие теоретические сведения

В алгоритмическом языке С кроме использования стандартных функций существует возможность работать с функциями пользователя. Предварительно функцию необходимо объявить. Объявление функции пользователя, т.е. ее декларация возможна в двух формах — в форме описания и в форме определения (реализации).

Описание функции — декларация ее прототипа в начале программного файла. Используется следующий способ декларации функций:

<тип_результата> <имя_функции>(<тип> <переменная>,
…<тип> <переменная>);

Идентификаторы переменных в круглых скобках прототипа указывать необязательно, так как компилятор языка их не обрабатывает.

Пример описания функции fun со списком параметров:

```
float fun(int, float, int, int);
```

Прототип функции сообщает компилятору о том, что далее в тексте программы будет приведено полное определение (полный ее текст). Полное определение функции имеет следующий вид:

```
<тип_результата> <имя_функции>(список параметров)
{
        код функции
}
```

Тип результата определяет тип значения, который возвращается функцией в точку ее вызова при помощи оператора возврата **return**. Если тип функции не указан, то по умолчанию предполагается тип **int**. Список параметров состоит из перечня типов и имен параметров, разделенных запятыми. Функция может не иметь параметров (**void**), но круглые скобки необходимы в любом случае.

Оператор return вызывает немедленный выход из данной функции и возврат в вызывающую ее функцию. Этот оператор также используется для возврата результата работы функции. Отметим, что в

коде функции может быть несколько операторов return, но может и не быть ни одного. В таких случаях возврат в вызывающую ее функцию происходит после выполнения последнего оператора.

Пример реализации функции, определяющей наименьшее из двух целых чисел:

```
int mini(int x, int y)
    { int t;
        if (x<y) t=x;
            else t=y;
    return t;
    }
```

Можно написать функцию mini и таким образом:

```
mini(int x, int y)
{
    return (x<y)? x:y;
}
```

Здесь тип возвращаемого результата по умолчанию будет int.

Все функции, возвращающие значение, должны использоваться в правой части выражений языка С, иначе возвращаемый результат будет утерян. Но они не могут использоваться в левой части операций присваивания, за исключением тех случаев, когда возвращается адрес результата работы функции.

Если функция не возвращает никакого значения, она должна быть описана как функция типа void (пустая).

Например, для вывода горизонтальной строки на экран дисплея можно использовать следующую функцию:

```
void lin(char a)                    // char a='-'; или a='=';
{
    int k;
        for(k=0; k<80; k++)
            printf("%c", a);
}
```

Если у функции отсутствует список параметров, то при декларации такой функции желательно в круглых скобках также указать ключевое слово void. Например, заголовок основной функции должен выглядеть так: void main(void).

В языке С каждая функция — это отдельный блок программы, вход в который возможен только через вызов данной функции. Например, нельзя оператором перехода goto передать управление внутрь функции.

Вызов функции имеет следующий формат:

<имя_функции>(список_аргументов);

где в качестве аргументов можно использовать константы, переменные, выражения (их значения перед вызовом функции будут компилятором

определены). Аргументы списка вызова должны полностью совпадать со списком параметров вызываемой функции по количеству, порядку следования и типам соответствующих им параметров. Отметим, что в случае отсутствия аргументов (в заголовке функции отсутствуют параметры) наличие круглых скобок у имени функции в точке ее вызова обязательно.

Область действия переменных

Область действия переменной — это правила, которые устанавливают, какие данные доступны из текущего места программы. Имеются три типа переменных: глобальные, локальные и формальные. Область действия локальных переменных — это те блоки, где локальные переменные объявлены. При выходе из блока локальная переменная и ее значение теряются.

Формальные переменные — это параметры в заголовке функции пользователя. Формальные параметры используются в функции так же, как локальные переменные. Область действия формальных параметров — блок, являющийся кодом функции.

Глобальные переменные объявляются вне какой-либо функции. Глобальные переменные могут быть использованы в любом месте программы, но перед их первым использованием они должны быть объявлены и инициализированы. Область действия глобальных переменных — вся программа с момента их объявления.

В языке С каждая переменная принадлежит к одному из четырех классов памяти — автоматическая (**auto**), внешняя (**extern**), статическая (**static**), регистровая (**register**). Тип памяти указывается ключевым словом (auto, extern, static, register), стоящим перед спецификацией типа переменной. Например, register int a;

По умолчанию переменная относится к классу auto и будет размещена в стеке.

В языке С аргументы при стандартном вызове функции передаются по значению, т.е. в функцию передаются не оригиналы аргументов, а их копии. В стеке выделяется место для формальных параметров функции и в это выделенное место при ее вызове заносятся значения фактических аргументов. Затем функция использует и может изменять эти значения в стеке. При выходе из функции измененные значения теряются. Вызванная функция не может изменить значения переменных, указанных как фактические аргументы при обращении к данной функции.

При необходимости функцию можно использовать для изменения передаваемых ей аргументов. В этом случае в качестве аргумента следует в вызываемую функцию передавать не значение переменной, а ее адрес. А для обращения к значению аргумента-оригинала использовать операцию «*».

Фрагмент программы, использующей функцию, в которой меняются местами значения аргументов x и y:

```
        . . .
void z1(int*, int*);              // Описание прототипа
     void main()
{ int a=2, b=3;
        …
printf("\n a=%d, b=%d", a, b);
z1(&a, &b);                       // Обращение к функции
printf("\n a = %d, b = %d", a, b);
        …
}

void z1(int *x, int *y)           // Описание (реализация) функции
{
   int t;
        t = *x;
        *x = *y;
        *y = t;
return;
}
```

При таком способе передачи аргументов их значения будут изменены, т.е. на экран монитора будет выведено:

a = 2, b = 3
a = 3, b = 2

В языке С функции могут вызывать сами себя. В этом случае функция называется рекурсивной.

Пример рекурсивной функции — вычисление факториала числа n!=1*2*3*…*n:

```
int fac(int n)
{
 int b;
  if (n= =1) return 1;
 b=fac(n-1)*n;
 return n;
}
```

Вызов функции в рекурсии не создает новую копию функции, а создает новые копии локальных переменных и параметров. Из рекурсивной функции надо предусмотреть выход, иначе система «зависает» через некоторое время работы с ней.

На функцию, как и на другой объект, можно создать указатель, например, указатель **p** на функцию, возвращающую значение типа type и имеющую параметры типа type1 t, type2 z , объявляется следующим образом:

```
type (*p)(type1 t1, type2 t2);
```

Пример работы с функциями

Ввести массив NxN (не больше 50) целых чисел, в функции посчитать сумму его положительных значений.

```
#include <stdio.h>
#include <conio.h>
void summa(int, int a1[ ][50]);              // Описание прототипа функции

void main(void)
 {
  int a[50][50];
  int i,j,N;
   clrscr();
   printf("\n Введите размер массива N (<50)\n");
   scanf("%d",&N);
   printf("\n Введите данные \n");
         for(i=0; i<N; i++)
          for(j=0; j<N; j++)
           {
             printf("\n a[%d][%d]=", i+1, j+1);
             scanf("%d", &a[i][j]);
           }
   summa(N,a);                               // Обращение к функции
 }

         void summa(int n, int a1[ ][50])     // Описание (реализация) функции
 {
  int i,j,s;
   printf("\n Функция summa \n");   // Вычисление суммы
          for (s=0,i=0; i<n; i++)
             for (j=0;j<n;j++)
                 if (a1[i][j]>0)  s+=a1[i][j];
   printf("\a\n Сумма = %d, Press any key... ",s);
   getch();
  }
```


Варианты индивидуальных заданий

Значение аргумента **x** изменяется от **a** до **b** с шагом **h**. Для каждого **x** найти значения функции Y(x), суммы S(x) и |Y(x)–S(x)| и вывести в виде таблицы. Значения **a**,**b**,**h** и **n** вводятся с клавиатуры. Значение S(x) является рядом разложения функции Y(x). Значения S и Y для данного

аргумента x должны совпадать в целой части и в первых двух-четырех позициях после десятичной точки.

Вычисление Y(x) и S(x) реализовать в виде функций.

В основной программе организовать ввод исходных данных, обращение к функциям и вывод результатов.

Работу программы проверить для a=0,1; b=0,8; h=0,1; n выбрать в зависимости от варианта задания (с факториалом, без факториала).

1. $S(x)=\sum_{k=1}^{n}\frac{\cos(kx)}{k}$, $Y(x)=-\ln\left|2\sin\frac{x}{2}\right|$.

2. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$, $Y(x)=\sin(x)$.

3. $S(x)=\sum_{k=0}^{n}\frac{\cos(\frac{k\pi}{4})}{k!}x^k$, $Y(x)=e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$.

4. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!}$, $Y(x)=\cos(x)$.

5. $S(x)=\sum_{k=1}^{n}x^k\sin(\frac{\pi k}{4})$, $Y(x)=\frac{x\sin(\pi/4)}{1-2x\cos\frac{\pi}{4}+x^2}$.

6. $S(x)=\sum_{k=0}^{n}\frac{x^{4k+1}}{4k+1}$, $Y(x)=\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}arctgx$.

7. $S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!}$, $Y(x)=e^{\cos x}\cos(\sin(x))$.

8. $S(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k}$, $Y(x)=(1+2x^2)e^{x^2}$.

9. $S(x)=\sum_{k=1}^{n}\frac{x^k\cos\frac{k\pi}{3}}{k}$, $Y(x)=-\frac{1}{2}\ln(1-2x\cos\frac{\pi}{3}+x^2)$.

10. $S(x)=\sum_{k=0}^{n}\frac{1}{2k+1}\left(\frac{X-1}{X+1}\right)^{2K+1}$, $Y(x)=\frac{1}{2}\ln(x)$.

11. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{\cos(kx)}{k^2}$, $Y(x)=\frac{1}{4}\left(x^2-\pi^2/3\right)$.

12. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k+1}}{4k^2-1}$, $Y(x)=\frac{1+x^2}{2}arctg(x)-x/2$.

13. $S(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}$, $Y(x)=\frac{e^x+e^{-x}}{2}$.

14. $S(x)=\sum_{k=1}^{n}\frac{\cos(2kx)}{4k^2-1}$, $Y(x)=\frac{1}{2}-\frac{\pi}{4}|\sin(x)|$.

15. $S(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}(x/2)^k$, $Y(x)=(\frac{x^2}{4}+\frac{x}{2}+1)e^{\frac{x}{2}}$.

16. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{2k^2+1}{(2k)!}x^{2k}$, $Y(x)=(1-\frac{x^2}{2})\cos(x)-\frac{x}{2}\sin(x)$.

17. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$, $Y(x)=2(\cos^2 x-1)$.

18. $S(x)=\sum_{k=1}^{n}x^k\cos(\frac{k\pi}{4})$, $Y(x)=\frac{x\cos\frac{\pi}{4}-x^2}{1-2x\cos\frac{\pi}{4}+x^2}$.

19. $S(x)=\sum_{k=1}^{n}\frac{\cos(2k-1)x}{(2k-1)^2}$, $Y(x)=\frac{\pi^2}{8}-\frac{\pi}{4}|x|$.

20. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$, $Y(x)=xarctg(x)-\ln\sqrt{1+x^2}$.

Контрольные вопросы

1. Чем функция пользователя отличается от стандартной функции?
2. Способы передачи аргументов в функцию.
3. Поясните понятие «локальные» и «глобальные» переменные.
4. Для чего и каким образом применяется оператор **return**?

ЛАБОРАТОРНАЯ РАБОТА № 6

Программирование алгоритмов с использованием структур

Цель работы:

Изучить особенности работы с составным типом данных — структурой.

Краткие теоретические сведения

Структура объединяет логически связанные данные разных типов. Структурный тип данных определяется описанием **шаблона**:

```
struct имя_структуры {
        описание полей;
            };
```

между символами «}» и «;» иногда помещают список декларируемых структурных переменных, при этом «имя_структуры» можно опустить.

Описание полей производится обычным способом. Типом элемента поля не может быть только тип **FILE**, других ограничений нет.

Пример определения структурного типа:

```
struct person
    {
      char Fio[72];
      int Nom_Gr;
    };
```

Интерпретация объекта типа struct person:

Fio	Nom_Gr
72	2

длина в байтах

Структурный тип "struct имя_структуры" можно использовать для декларации структурных переменных, массивов, функций и т.д.

```
struct person Teacher;          // структурная переменная
struct person Student[100];     // массив структур
struct person *Sved;            // указатель на структуру
```

Предыдущий пример можно записать кратко:

```
struct person {
        char Fio[72];
        int Nom_Gr;
            } Teacher, Student[100], *Sved;
```

Структурный тип данных удобно применять для группового управления манипулированием логически связанных объектов. Параметрами таких операций являются адрес и размер структуры.

Примеры групповых операций:

— захват и освобождение памяти для объекта, представленного совокупностью необязательно однотипных данных;

— запись и чтение данных, хранящихся на внешних носителях как физические и/или логические записи с известной структурой.

Обращение к элементам структур производится посредством:

а) операции принадлежности (**.**) в виде

имя_структуры **.** имя_элемента

или

(*указатель_структуры) **.** имя_элемента

б) операции косвенной адресации (**->**) в виде

указатель_структуры **->** имя_элемента

Примеры обращения к полям описанной выше структуры:

к полю Fio:	к полю Nom_Gr:
Teacher.Fio	Teacher.Nom_GR
Student[15].Fio	Student[15]. Nom_GR
Sved->Fio	Sved-> Nom_GR

Пример использования структур

Ввести сведения о студентах учебной группы (не более 50):

— фамилию и имя;

— итоги сдачи экзаменов — три оценки.

В программе рассчитывается средний балл для каждого студента, выполняется поиск по первой букве фамилии. Текст программы может иметь вид

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

struct Spisok {
        char Fio[20];
        int Ot[3];
        float S_Bal;
          } *sved;                  // Указатель на структуру
```

```
void Vvod(int nom,struct Spisok *sved)
{
        printf( "\n Vvedi svedenia %d ", (nom+1));
        puts("\n FIO   — ") ;
        fflush(stdin);
        gets(sved->Fio); // Используем операцию косвенной адресации (->)
float s=0;
  for(int i=0;i<3;i++)
        {
        puts("\n Otcenki — ");  scanf("%d", sved->Ot[i] );
        s+=sved->Ot[i];
         }
   sved->S_Bal=s/3.;
 return;
}

        void main(void)
{
 struct Spisok Stud[50];
 int i,N;
 char Bukva,Fio_p[20];
   clrscr();
   puts("\n Vvedi kol-vo < 50 ");
   scanf("%d",&N);
   for(i=0;i<N;i++)
                Vvod(i,&Stud[i]);               // Вводим поэлементно
   puts("\n Список студентов ");
   for(i=0;i<N;i++)        // Используем операцию принадлежности (.)
        printf("\n %20s %4.2f",Stud[i].Fio,Stud[i].S_Bal);
   puts("\n Поиск сведений по первой букве. \n Введите букву ");
   scanf("%c", &Bukva);
   puts("\n Сведения: ");
   int kod_p=0;
    for(i=0;i<N;i++)
        if(Stud[i].Fio[0]= =Bukva)
        {
         kod_p=1;
         printf("\n %20s %4.2f",Stud[i].Fio,Stud[i].S_Bal);
         }
  if(kod_p==0) puts( "\n Таких НЕТ!");
  getch();
 }
```

Варианты индивидуальных заданий

В программах организовать ввод информации о студентах:
— фамилия и инициалы;
— год рождения;
— номер группы;
— оценки за первый семестр: физика, математика, информатика;
— средний балл.

1. Распечатать анкетные данные студентов, сдавших сессию на 4,5.

2. Распечатать анкетные данные студентов-отличников, фамилии которых начинаются с интересующей вас буквы.

3. Распечатать анкетные данные студентов-отличников из интересующей вас группы.

4. Распечатать анкетные данные студентов, фамилии которых начинаются с буквы А, и сдавших математику на 4 и 5.

5. Распечатать анкетные данные студентов, имеющих оценку 3 по физике и оценку 5 по остальным предметам.

6. Распечатать анкетные данные студентов интересующей вас группы. Фамилии студентов начинаются с букв В, Г и Д.

7. Распечатать анкетные данные студентов, не имеющих оценок 3 и 2 по информатике и математике.

8. Вычислить общий средний балл всех студентов и распечатать список студентов со средним баллом выше общего среднего балла.

9. Вычислить общий средний балл всех студентов и распечатать список студентов интересующей вас группы, имеющих средний балл выше общего среднего балла.

10. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 2.

11. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 5 по информатике.

12. Распечатать анкетные данные студентов, имеющих оценку 4 по физике и оценку 5 по высшей математике.

13. Вычислить общий средний балл студентов интересующей вас группы и распечатать список студентов этой группы, имеющих средний балл выше общего.

14. Распечатать анкетные данные студентов-отличников интересующей вас группы.

15. Распечатать анкетные данные студентов интересующей вас группы, имеющих средний балл выше введенного с клавиатуры.

16. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 4 по физике и оценку 5 по высшей математике.

17. Распечатать анкетные данные студентов, упорядоченные по алфавиту.

18. Распечатать анкетные данные студентов, упорядоченные по году рождения.

19. Распечатать анкетные данные студентов, упорядоченные по номеру группы.

20. Распечатать анкетные данные студентов, упорядоченные по среднему баллу.

Контрольные вопросы

1. Как описываются структуры в языке С?
2. Что такое структурная переменная?
3. Поясните термин «указатель на структуру».
4. Как обратиться к полю структурной переменной?
5. Как организовать массив структурных переменных?

ЛАБОРАТОРНАЯ РАБОТА № 7

Программирование алгоритмов с использованием файлов

Цель работы:
Изучить способы создания и работы с файлами в языке С.

Краткие теоретические сведения

Файл — это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки и пересылке как единое целое.

Прежде чем работать с файлом, его нужно открыть для доступа, т.е. создать и инициализировать область данных, которая содержит информацию о файле: имя, путь и т.д.

В языке С это выполняет функция **fopen()**. Она связывает физический файл на носителе с логическим именем в программе. Логическое имя — это указатель на файл, т.е. на область памяти, где хранится информация о файле. Указатели на файлы необходимо объявлять. Формат объявления такого указателя следующий:

FILE *указатель на файл;

Формат объявления функции

fopen("строка_1" , "строка_2");

в строке_1 (заключенной в кавычки) указывается место, в которое вы собираетесь поместить файл, например «a:mas_dat.dat» — файл с именем mas_dat.dat будет находиться на дискете (a:), «d:\\work\\sved.txt» — файл с именем sved.txt будет находиться на d:, в каталоге work.

В строке_2 указываются коды режимов доступа к открываемым файлам:

w — файл открывается только для записи; если файла с заданным именем нет, то он будет создан, если же такой файл существует, то перед открытием прежняя информация уничтожается;

r — файл открывается только для чтения; если такого файла нет, то возникает ошибка;

a — файл открывается для добавления в конец новой информации;

r+ — файл открывается для редактирования данных; возможны и запись, и чтение информации;

w+ — то же, что и для r+;

a+ — то же, что и для a, только запись можно выполнять в любое место файла; доступно и чтение файла;

t — файл открывается в текстовом режиме; указывается поле r, w, a, r+, w+, a+;

b — файл открывается в двоичном режиме; указывается поле r, w, a, r+, w+, a+.

Текстовый режим отличается от двоичного тем, что при открытии файла как текстового пара символов «перевод строки», «возврат каретки» заменяется на один символ: «перевод строки» для всех функций записи данных в файл, а для всех функций вывода символ «перевод строки» теперь заменяется на два символа: «перевод строки», «возврат каретки».

По умолчанию файл открывается в текстовом режиме.

Пример:

```
FILE *f;
 f=fopen ("d:\\work\\Dat_sp.cpp", "w");
```

или

```
FILE *f=fopen ("d:\\work\\Dat_sp.cpp", "w");
```

— открывается для записи текстовый файл Dat_sp.cpp с указателем f, который будет находиться на диске d, в каталоге work.

Если при открытии файла произошла ошибка, функция **fopen**() возвращает значение **NULL**. Приведем пример стандартной последовательности операторов, необходимых для корректной работы с файлом:

```
#include <stdio.h>
    . . .
    FILE *f_my;
        If(!(f_my = fopen("rez.txt", "r+t")))
        {
            puts("\n Ошибка при открытии файла! ");
            getch(); return;
        }
    . . .                   //      Работа с файлом
    fclose(f_my);
    . . .
```

После работы доступ к файлу необходимо закрыть с помощью функции **fclose**(указатель_на_файл). Например, из предыдущего примера файл закрывается так: fclose (f);

Для закрытия нескольких файлов введена функция, объявленная следующим образом: void **fcloseall**(void);

Для работы с текстовыми файлами удобнее всего пользоваться функциями **fprintf**(), **fscanf**(). Формат параметров и выполняемые функции аналогичны известным операторам форматного ввода-вывода, только добавлен параметр — указатель на файл, к которому применяется данная функция. Рассмотрим простой пример:

```
#include<stdio.h>
    void main()
    {
    int a=2,b=3;
```

```
FILE *f1;
  if(!(f1=fopen("d:\\work\\f_rez.txt","w+t")))
        {
        puts("\n Файл не создан! ");
        getch(); return;
        }
  fprintf(f1, " Файл результатов \n");
  fprintf(f1, " %d плюс %d = %d\n",a,b,a+b);
fclose(f1);
}
```

Просмотрев содержимое файла, можно убедиться, что данные в нем располагаются так же, как на экране при использовании функции **printf**().

Функция int **fgets**(char *S,int m,FILE *fp) — выполняет чтение строки S из файла fp до тех пор, пока не встретит символ '\n' или пока не будет считано m байт.

Функция int **fputs**(char *S, FILE *fp) — записывает строку S в файл fp до тех пор, пока не встретится '\0', который в файл не переносится и на символ '\n' не заменяется.

Функции работы с текстовыми файлами удобны при создании результирующих файлов для отчетов по лабораторным и курсовым работам. Для создания баз данных удобнее пользоваться функциями работы с бинарными файлами. В основном прототипы этих функций описаны в библиотеках **stdio.h** и **io.h**. Рассмотрим некоторые из них:

1) int **fread**(void ***ptv**, int **size**, int **n**, FILE *fp) — считывает **n** блоков по **size** байт каждый из файла **fp** в область памяти, на которую указывает **ptv** (необходимо заранее отвести память под считываемый блок);

2) int **fwrite**(void ***ptv**, int **size**, int **n**, FILE *fp) — записывает **n** блоков по **size** байт каждый из области памяти, на которую указывает **ptv** в файл fp;

3) int **fileno**(указатель_файла) — возвращает дескриптор файла (число, определяющее номер файла);

4) long **filelength**(int дескриптор_файла) — возвращает длину файла в байтах;

5) int **chsize**(int дескриптор_файла, long позиция) — выполняет изменение размера файла, признак конца файла устанавливается после байта с номером «позиция»;

6) int **fseek**(указатель_файла, long количество_байт, int код) — выполняет смещение указателя на количество_байт в направлении признака код:

0 — от начала файла;
1 — от текущей позиции указателя;
2 — от конца файла;

7) long **ftell**(указатель_файла) — возвращает значение указателя на текущую позицию файла (–1 — ошибка);

8) int **feof**(указатель_файла) — возвращает ненулевое значение при правильной записи признака конца файла;

9) int **fgetpos**(указатель_файла, long*текущая_позиция) — определяет значение текущей позиции файла; возвращает 0 при успешном завершении.

Пример программы работы с файлом структур

Создать файл, содержащий сведения о студентах: фамилия, три оценки, средний балл. Организовать возможность просмотра содержимого файла и добавления новых данных в конец файла.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

 struct Sved {
              char Fam[20];
              int mark[3];
              float S_Bal;
              } zap;
 char Spis[]="d:\\work\\Sp.dat";
 FILE *F_zap;

   void main ()
{  int i,kodR,size=sizeof(Sved);
  float s;
    clrscr();
  while(1)
{
       puts("  Создание       - 1");
       puts("  Просмотр       - 2");
       puts("  Добавление     - 3");
       puts("  Выход          - 0");
        scanf("%d", &kodR);
switch(kodR)
{
  case 1:
  case 3: if(kodR==1) F_zap=fopen(Spis,"w+b");
                    else F_zap=fopen(Spis,"a+b");
        while(2) {
           puts(" Фамилия (Завершение ввода — 0)");
           scanf("%s", zap.Fam);
              if((zap.Fam[0])=='0') break;
```

```
            puts(" Marks (3) ");
            for(s=0,i=0;i<3;i++)
                        {
                        scanf("%d", &zap.mark[i]);
                        s+=zap.mark[i];
                        }
            zap.S_Bal=s/3.;
            fwrite(&zap,size,1,F_zap);
               }
         fclose(F_zap);
   break;
   case 2:  F_zap=fopen(Spis,"r+b");
        while(2)
        {
          if(size!=fread(&zap,1,size,F_zap)) break;
        /* Функция fread() возвращает количество прочитанных блоков,
           поэтому в данном случае 2-й и 3-й параметры функции
           переставлены местами */
          printf(" %20s %2d %2d %2d %5.2f\n",
                    zap.Fam,zap.mark[0],zap.mark[1],zap.mark[2],zap.S_Bal);
                 }
        fclose(F_zap);
     break;
     case 0:  return;
       }       //      Конец Switch
     }         //      Конец While(1)
}              //      Конец программы
```

Варианты индивидуальных заданий

Написать программу обработки файла типа запись, содержащую следующие пункты меню:

— создание;
— просмотр;
— коррекция — добавление новых данных или исправление старых.

1. Список товаров, имеющихся на складе, включает:
— наименование товара;
— количество единиц товара;
— цену единицы товара;
— дату поступления товара на склад.

Вывести в алфавитном порядке список товаров, хранящихся более месяца, стоимость которых превышает 1000 р.

2. Для получения места в общежитии формируется список студентов, который включает:

— Ф.И.О. студента;
— номер группы (буква и четыре цифры);
— средний балл;
— доход на одного члена семьи.

Общежитие в первую очередь предоставляется тем студентам, чьи доходы на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла.

Вывести список очередности предоставления места в общежитии.

3. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны:

— номер рейса;
— тип автобуса;
— пункт назначения;
— время отправления;
— время прибытия на конечный пункт.

Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

4. На междугородной АТС информация о разговорах содержит:

— дату разговора;
— код и название города;
— время разговора;
— тариф;
— номер телефона в этом городе;
— номер телефона абонента.

Вывести по каждому городу общее время разговора с ним и сумму.

5. Информация о сотрудниках фирмы включает:

— Ф.И.О. сотрудников;
— табельный номер;
— количество проработанных часов за месяц;
— почасовой тариф.

Рабочее время свыше 144 ч считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработной платы.

6. Информация об участниках соревнований содержит:

— Ф.И.О. игрока;
— игровой номер;
— возраст;
— рост;
— вес.

Вывести информацию о самой молодой, рослой и легкой команде.

7. Для книг, хранящихся в библиотеке, задаются:
— регистрационный номер книги;
— автор;
— название;
— год издания;
— издательство;
— количество страниц.
Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

8. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают:
— наименование;
— количество;
— номер цеха.
Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.

9. Информация о сотрудниках предприятия содержит:
— Ф.И.О.;
— номер отдела;
— должность;
— дату начала работы.
Вывести список сотрудников по отделам в порядке убывания стажа.

10. Создать файл, содержащий сведения о месячной заработной плате сотрудников отдела. Каждая запись содержит поля: фамилия сотрудника, наименование отдела, размер заработной платы за месяц.
Вычислить общую сумму выплат за месяц по отделу A, а также среднемесячный заработок сотрудникам этого отдела.
Напечатать для бухгалтерии ведомость для сотрудников этого отдела, у которых зарплата ниже введенной с клавиатуры.

11. Создать файл, содержащий сведения о количестве изделий категории A, B, C, собранных рабочими за месяц. Структура записи имеет поля: фамилия сборщика, наименование цеха, количество изделий по категории, собранных рабочими за месяц.
Считая данными (вводятся с клавиатуры) значения расценок Sa, Sb, Sc за выполненную работу по сборке единицы изделия категорий A, B, C, выдать на печать следующую информацию:
— общее количество изделий категории A, B, C, собранных рабочими цеха X;
— ведомость заработной платы рабочих цеха X;
— средний размер заработной платы работников этого цеха.

12. Создать файл, содержащий сведения о телефонах абонентов. Каждая запись имеет поля: фамилия абонентов, год установки телефона, номер телефона. На печать вывести информацию:

— по вводимой с клавиатуры фамилии абонента выдается номер телефона;

— определяется количество установленных телефонов с XXXX года (год вводится с клавиатуры).

13. Создать файл, содержащий сведения об ассортименте игрушек в магазине. Структура записи: название игрушки, цена, количество, возрастные границы, например, от 2 до 5 лет. Вывести на печать:

— название игрушек, которые подходят детям от 1 до 3 лет;

— стоимость самой дорогой игрушки и ее наименование;

— название игрушки, которая по стоимости не превышает «x» р. Значение «x» вводится с клавиатуры.

14. Создать файл, содержащий сведения о сдаче студентами сессии. Структура записи: номер группы, фамилия студента, оценки по пяти экзаменам и пяти зачетам (зачет — незачет). На печать вывести:

— фамилии неуспевающих студентов с указанием номера группы и количества задолженностей;

— средний балл, полученный каждым студентом группы X (вводится с клавиатуры), и всей группой в целом.

15. Создать файл, содержащий сведения об ассортименте обуви в магазине. Структура записи: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д для женской обуви,
М — для мужской, П — для детской. На печать вывести информацию:

— о наличии и стоимости обуви артикула X (вводится с клавиатуры);

— ассортиментный список женской обуви с указанием наименования и имеющего в наличии числа пар каждой модели.

16. Для участия в конкурсе исполнителей необходимо заполнить анкету с данными:

— Ф.И.О;

— год рождения;

— название страны;

— класс музыкального инструмента (гитара, фортепиано, скрипка).

Вывести список самых молодых лауреатов конкурса по классам инструментов в порядке занятых мест.

Контрольные вопросы

1. Укажите правила открытия, закрытия файла в языке С.
2. Какие режимы доступа к файлам вы знаете?
3. Опишите работу функций fprintf() и fscanf().
4. Опишите работу функций fwrite() и fread().

ЛАБОРАТОРНАЯ РАБОТА № 8

Использование графического режима

Цель работы:
Изучение работы дисплея в графическом режиме.

Краткие теоретические сведения

При работе в графическом режиме экран дисплея представляет собой матрицу точек (пикселов — pixel) — т.е. матрицу отображаемых точек. При этом число столбцов и строк пикселов (разрешение экрана дисплея) зависит от режима работы видеоадаптера. Можно управлять цветом каждого пиксела, задавая цвета фона, рисунка и заполнения замкнутых областей экрана дисплея, а также создавать эффект движения изображений.

За начало координат экрана дисплея в графическом режиме принимается верхний левый угол с координатами x=0 и y=0, где x — координата по горизонтали, y — координата по вертикали точки (пиксела). Во всех примерах программ, приведенных далее по тексту, нулевые координаты присваиваются верхнему левому углу создаваемого графического окна. Содержимое библиотеки графических функций в языке C подразделяется на немобильную группу функций (функции зависят от типа адаптера) и на мобильную группу функций.

Немобильная группа графических программ представляет собой BGI драйвер (Borland Graphics Interface).

Драйвер — это обработчик прерывания 10h, он должен дополнить системный обработчик до того, как будут использоваться мобильные графические функции. Перед завершением программы таблица векторов прерываний восстанавливается.

Основные функции BGI-драйвера — установка и обновление ряда внешних переменных, которые могут изменяться как функциями системного обработчика прерывания 10h (например, при переключении видеорежима или при изменении регистров палитры), так и мобильными функциями библиотеки графики языка Turbo C (TC) или C++.

Для различных типов адаптеров применяются различные драйверы: CGA.BGI — драйвер для CGA и MCGA; EGAVGA.BGI — драйвер для адаптеров EGA,VGA; HERC.BGI — драйвер для монохромных адаптеров Hercules.

Графические функции мобильной группы подразделяются на группы:

— функции подготовки графической системы и перехода в текстовый режим;

— функции получения изображений на экране дисплея;

— функции установки параметров изображения (вид штриховки, стиль линий и т.д.);

— функции определения параметров режимов и изображений.

Функции подготовки графической системы

Перед использованием графических функций необходимо инициализировать систему графики. Графические режимы, поддерживаемые библиотекой графики, задаются символическими константами, описанными в файле graphics.h в перечислимом типе graphics_mode.

Инициализация графической системы производится функцией **initgraph**(), которая загружает графический драйвер и переключает экран дисплея в требуемый графический режим. Описание функции:

initgraph(&g_driver,&g_mode," ");

В двойных апострофах (третий параметр) необходимо указать путь (маршрут) к графическому драйверу. Если указать пробел, то графический драйвер должен быть в текущем каталоге. Первый параметр &g_driver — тип графического драйвера: 1 — CGA, 3 — EGA, 9 — VGA и т.д. Второй параметр &g_mode — графический режим (рассмотрим только для VGA драйвера):

VGA	0	640x200
VGAMED	1	640x350
VGAHI	2	640x480

Запись типа 640x200 — это разрешающая способность экрана дисплея в графическом режиме (число строк умножить на число столбцов).

Для автоматического задания режима графики следует записать:

```
int g_driver=DETECT, g_mode;
```

Для завершения работы в графическом режиме необходимо применить функцию **closegraph**();

Основные функции получения изображения

Аргументами большинства функций графики являются данные целого типа.

1. Вычерчивание окружности: **circle**(x,y,r);
2. Вычерчивание закрашенного прямоугольника: **bar**(x1,y1,x2,y2);
3. Вычерчивание параллелепипеда: **bar3d**(x1,y1,x2,y2,глубина,p), p=0 или p=1 — верхняя грань отображается (не отображается);
4. Вычерчивание линии: **line**(x1,y1,x2,y2);
5. Вычерчивание точки: **putpixel**(x,y,цвет);
6. Вычерчивание прямоугольника: **rectangle**(x1,y1,x2,y2);
7. Вывод текста: **outtext**(x,y,"текст");
8. Установка указателя на экране дисплея: **moveto**(x,y);

9. Очистка экрана дисплея: **cleardevice**(void);

10. Заполнение ранее заданным наполнителем замкнутой области: **floodfill**(x,y,c); c — номер цвета линии, ограничивающей область.

Основные функции установки параметров изображения

1. Установка цвета линий: **setcolor**(цвет).
2. Установка цвета фона: **setbkcolor**(цвет).
3. Установка стиля наполнителя замкнутых линий:
   setfillstyle(номер наполнителя (0–12), цвет).
4. Установка толщины линий: **setlinestyle**(стиль линии, 0, толщина);
   0 — непрерывная, 1 — из точек, 2, 3 — штрих.
5. Установка стиля текста: **settextstyle**(шрифт 0–4, направление(0 — горизонтальное,1 — вертикальное), размер).

Некоторые функции определения параметров режимов

detectgraph() — определяет графический режим и драйвер;

getcolor(), **getbkcolor**() — возвращают номер цвета объектов или фона соответственно.

Пример программы построения на экране дисплея геометрических фигур с использованием различной цветовой палитры:

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <process.h>
    void main(void)
 {
    int g_driver=DETECT,g_mode,g_error;
    int x,y;
    initgraph(&g_driver,&g_mode,"c:\\bc31\\bgi");
    g_error=graphresult();
        if(g_error!=grOk)
        {   puts("error");
            getch();
            exit(1);
        }
    setbkcolor(1);
    setcolor(14);
    settextstyle(3,1,5);
    outtextxy(20,200,"IMEGES!!!");
    settextstyle(3,0,5);
        x=60;y=100;
        circle(x,y,50);
        outtextxy(30,180,"CIRCLE!!!");
```

```
        getch();
    bar(150,50,250,150);
    outtextxy(180,180,"BAR!!!");
    getch();
        bar3d(300,50,340,150,20,1);
        outtextxy(300,180,"BAR3d!!!");
        getch();
    setfillstyle(6,12);
    rectangle (50,250,150,350);
    floodfill(100,300,14);
    outtextxy(70,380,"IMEGES!!! — N6");
    getch();
        setfillstyle(8,12);
        rectangle (320,250,420,350);
        floodfill(350,250,14);
        outtextxy(340,380,"IMEGES!!! — N8");
        getch();
closegraph();
}
```

Некоторые функции работы с изображением на экране

Для рассматриваемых ниже функций графическая информация на экране представляет прямоугольную область с координатами:

1) unsigned **imagesize**(int x1,int y1,int x2,int y2); — возвращает значение объема буфера для сохранения графической информации в окне экрана заданного размера;

2) void **getimage**(int x1,int y1,int x2,int y2,void *buf); — получение и сохранение области экрана в буфере (указатель buf) памяти;

3) void **putimage**(int x,int y,buf,int Код); — вывод изображения из buf на экран начиная с заданной позиции; код определяет способ наложения выводимого изображения:

COPY_PUT (0) — простое перемещение;
XOR_PUT (1) — наложение с операцией «Исключающее ИЛИ»;
OR_PUT (2) — с выполнением операции «ИЛИ»;
AND_PUT (3) — с выполнением операции «И»;
NOT_PUT (4) — перемещение изображения с инверсией (отрицанием).

Рассмотрим некоторые дополнительные функции.

1) void **sound**(unsigned z); — вызывает звуковой сигнал с частотой (герц);

2) void **nosound**(void); — прекращает подачу звукового сигнала;

3) void **delay**(int x); — задерживает выполнение программы на x миллисекунд; прототипы этих функций описаны в библиотеке **dos.h**.

4) int **bioskey**(0); — ждет нажатия клавиши (int Код=0) и возвращает ее код; прототип описан в библиотеке **bios.h**.

Пример программы

В центре экрана нарисовать картинку, например такую, клавишами «стрелки» управлять движением фигуры только до границ экрана. Текст программы может иметь следующий вид:

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <process.h>
#include <dos.h>
#include <bios.h>
#include <alloc.h>

void Move(int,int,int);

void main(void)
{
 int g_driver=9,g_mode=2,g_error;
 int x,y,key;
        initgraph(&g_driver,&g_mode,"c:\\bc31\\bgi");
        g_error=graphresult();
        if(g_error!=grOk)
         {    puts("error");
              printf("\n error=%d,reason=%s\n",g_error,grapherrormsg(g_error));
              getch();
                exit(1);
         }
 setbkcolor(1);     setcolor(12);       settextstyle(3,0,4);
 x=getmaxx()/2;
 y=getmaxy()/2;
 int r=50;
          circle(x,y,r);       circle(x,y,r-5);
          outtextxy(x-28,y–17,"YES!");
                Move(x,y,r+5);
 closegraph();
```

```
 }

        void Move(int x,int y,int d)
{
        int Fon,Color_Fon, x1,y1,Key;
        void *bufC;
        unsigned sizeC;
        sizeC=imagesize(x-d,y-d,x+d,y+d);
       bufC=malloc(sizeC);
       getimage(x–d,y–d,x+d,y+d,bufC);
             Color_Fon=getbkcolor();
 while(1)
 {
       Key=bioskey(0);
       setfillstyle(0,Color_Fon);
       bar(x–d,y–d,x+d,y+d);
  switch(Key)
  {
        case 283: free(bufC);
                  return;                    // Нажата клавиша <Esc>
        case 19712: x+=20;
             if(x+d>getmaxx()) x=d; break;     // — >
        case 19200: x-=20;
             if(x–d<0)x=getmaxx()-d;break;     // <-
        case 20480: y+=10;
             if(y+d>getmaxy())y=d; break;      // Down
        case 18432: y-=10;
             if(y–d<0)y=getmaxy()-d;break;     // Up
       default: sound(1000); delay(500);
                nosound(); break;
    }
  putimage((x–d),(y–d),bufC,COPY_PUT);
  }
 return;
}
```


Варианты индивидуальных заданий

1. Составить программу вывода на экран дисплея схематичного изображения велосипедиста. При запуске программы велосипедист начинает движение, вращая ногами педали велосипеда.

2. Составить программу вывода на экран дисплея схематичного изображения человека. При запуске программы человек начинает идти, размахивая в такт движения руками.

3. Составить программу вывода в верхней части экрана дисплея изображения облака. При запуске программы облако начинает двигаться и из него начинает идти дождь. При этом размер облака постепенно уменьшается.

4. Составить программу вывода в верхнюю часть экрана дисплея изображения тучи, а в нижнюю часть экрана дисплея — емкость для воды. При запуске программы начинает идти дождь. При этом размер тучи уменьшается, а емкость наполняется водой.

5. То же, что и в задании 4, но из тучи идет снег и внизу растут сугробы.

6. Составить программу вывода на экран дисплея изображения летящего самолета.

7. Составить программу вывода на экран дисплея изображения пушки. В правой части экрана появляется и исчезает (случайным образом) мишень. Нажатием клавиши ВВОД производится выстрел из пушки. Момент попадания фиксируется в виде взрыва.

8. Составить программу вывода в верхней части экрана дисплея движущегося слева направо парусника с постоянной скоростью. Ее значение всякий раз задается генератором случайных чисел. В нижней части экрана дисплея расположена пушка. При нажатии клавиши ВВОД происходит выстрел торпедой с постоянной скоростью. При попадании торпеды в пушку смоделировать взрыв парусника и его исчезновение. При промахе парусник достигает правой границы экрана дисплея и начинает движение сначала с новой постоянной скоростью.

9. Составить программу вывода на экран дисплея схематичного изображения лыжника. При нажатии клавиши ВВОД он начинает движение классическим стилем.

10. Составить программу вывода на экран дисплея схематичного человека в положении готовности осуществить прыжок в длину. При нажатии клавиши ВВОД спортсмен начинает разбег и выполняет прыжок в длину.

11. Составить программу вывода изображения циферблата механических часов с секундной, минутной и часовой стрелками. Запуск часов осуществляется нажатием клавиши ВВОД, при этом перемещение секундной стрелки сопровождается характерным для часов звуком.

12. То же, что в задании 11, но предусмотреть режим будильника.

13. То же, что в задании 11, но в 6 и 12 часов на экране появляется изображение кукушки, затем подается соответствующее число сигналов.

14. То же, что в задании 11, но предусмотреть коррекцию времени путем ускоренного перемещения стрелок при нажатии клавиши курсор вправо и курсор влево.

15. Составить программу вывода на экран дисплея настольных электронных часов и изображения метронома. При нажатии клавиши ВВОД стрелка метронома начинает колебательное движение, синхронно с которым начинает изменяться показание электронных часов.

16. Составить программу вывода на экран дисплея песочных часов. При нажатии клавиши ВВОД моделируется процесс падения песчинок, уменьшение уровня песка в верхней части колбы и увеличение в нижней части колбы.

17. Составить программу вывода на экран дисплея треугольника. При нажатии клавиши курсор вправо треугольник вращается по часовой стрелке, клавиши курсор влево — против часовой стрелки.

18. Составить программу вывода на экран дисплея схематичного изображения бабочки. При нажатии клавиши ВВОД бабочка начинает полет, взмахивая крыльями.

19. Составить программу вывода на экран дисплея трех вложенных друг в друга окружностей, представляющих собой беговые дорожки. На линию старта выходят три спортсмена (произвольные фигуры). При нажатии клавиши ВВОД участники стартуют с одинаковой угловой скоростью. После старта угловые скорости участников забега изменяются по случайному закону. На финише указать место, занятое каждым участником забега.

20. То же, что в задании 19, но беговые дорожки представляют собой три одинаковых рядом расположенных окружностей.

Контрольные вопросы

1. Каким образом производится инициализация графического режима?
2. Назовите основные функции установки параметров изображения.
3. Как задать автоматический режим графики?
4. Какими функциями можно создать эффект движения изображения по экрану дисплея?

ЛИТЕРАТУРА

1. Березин Б.И., Березин С.Б. Начальный курс С и С++. — М.: Диалог-МРТИ, 1999. — 288 с.
2. Керниган Б., Ритчи Д. Язык программирования Си. — М.: Финансы и статистика, 1992. — 271 с.
3. Касаткин А.И., Вольвачев А.Н. Профессиональное программирование на языке Си: От Turbo–C к Borland C++: Справочное пособие. — Мн.: Выш.шк., 1992. — 240 с.
4. Страуструп Б. Язык программирования С++. 2-е изд.: В 2 т. Киев: ДиаСофт, 1993.
5. Фьюэр А. Задачи по языку СИ. — М.: Финансы и статистика, 1985.
6. Хэнкок Л., Кригер М. Введение в программирование на языке СИ. — М.: Радио и связь, 1986.
7. Берри В., Микинз Б. Язык СИ: введение для программистов. — М.: Финансы и статистика, 1988.
8. Уэйт М., Прама С., Мартин Д. Язык СИ. Руководство для начинающих. — М.: Мир, 1988.
9. Больски М.Н. Язык программирования СИ. Справочник. — М.: Радио и связь, 1988.
10. Юлин В.А., Булатова И.Р. Приглашение к СИ. — Мн.: Высш.шк., 1990.
11. Уингер Р. Язык Турбо СИ. — М.: Мир, 1991.
12. Романовская Л.М., Русс Т.В., Свитковский С.Г. Программирование в среде СИ для ПЭВМ ЕС. — М.: Финансы и статистика, 1992.
13. Демидович Е.М. Основы алгоритмизации и программирования. Язык СИ. — Мн.: Бестпринт, 2001. — 440 с.
14. Аксенкин М.А., Целобенок О.Н. Язык С. — Мн.: Універсітэцкае,, 1995. — 302 с.
15. Котлинская Г.П., Галиновский О.И. Программирование на языке СИ. — Мн.: Выш.шк., 1991. — 155 с.
16. Подбельский В.В. Язык С++. — М.: ФиС, 2001. — 559 с.
17. Климова Л.И. С++. Практическое программирование. — М.: Кудиц-Образ, 2001. — 587 с.
18. Шилд Г. Программирование на Borland C++. — Мн.: ПОПУРРИ, 1999. — 800 с.
19. Тимофеев В.В. Программирование в среде С++ Builder 5. — М.: БИНОМ, 2000.

Приложение

Интегрированная среда программирования Borland C++

Интегрированная среда (Integrated Development Environment — IDE) — это программа, имеющая встроенный редактор текстов, подсистему работы с файлами, систему помощи, встроенный отладчик, подсистему управления компиляцией и редактированием связей, а также компилятор и редактор связей. Другими словами, IDE дает возможность получить EXE-файл, не используя другие программы. IDE запускается файлом BC.EXE.

После запуска на исполнение файла BC.EXE на экране отображается основное окно IDE (рис. 1).

Верхняя строка окна — это главное меню. Опции меню позволяют обратиться к подменю и выбрать соответствующую команду.

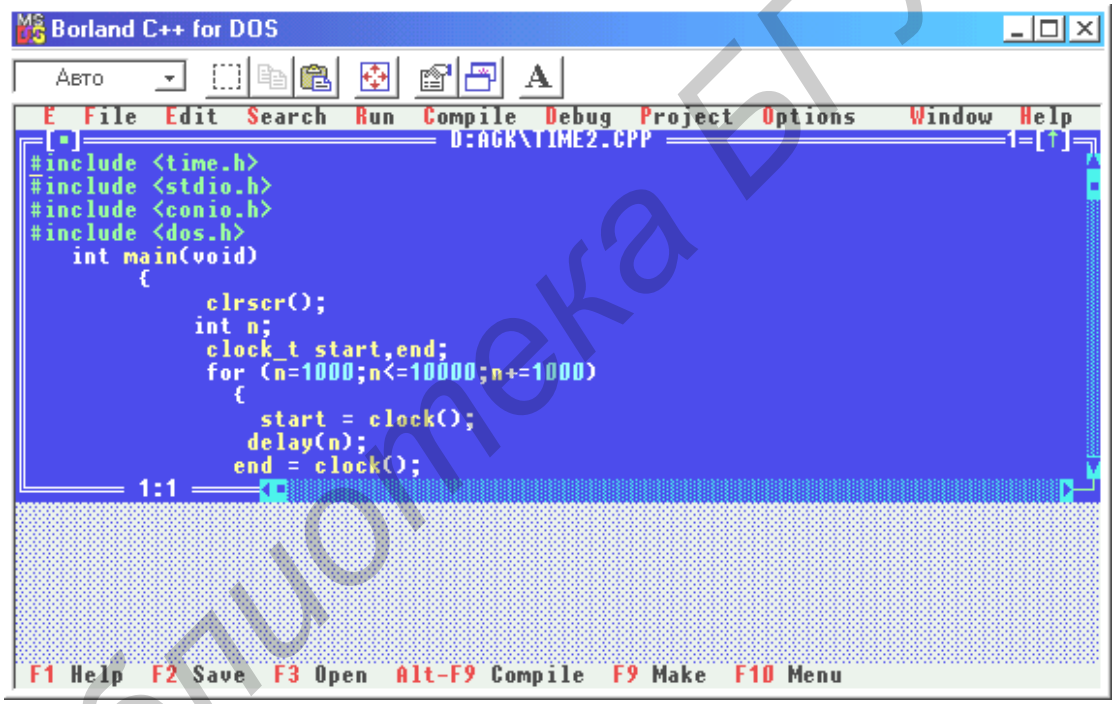


Рис. 1

Нижняя строка экрана отведена под строку состояния, где выделены назначения «горячих» клавиш, воспринимаемых на данном этапе работы.

Выбрать любую из команд меню можно одним из трех способов:

1. Нажать клавишу F10 и с помощью клавиш со стрелками выбрать необходимую команду.
2. Установить курсор мыши на любое ключевое слово меню и нажать левую кнопку мыши.
3. Использовать «горячие» клавиши (метод скорейшего вызова команды).

Одновременно нажатие клавиши Alt и «горячей» (клавиша, подсвеченная другим цветом).

Окно — это ограниченная рамкой область экрана. Его можно открыть, переместить, покрыть другими окнами, изменить размеры, закрыть. Активное окно (то, которое воспринимает нажатия клавиш) обозначается двойной линией. Активное окно изображается всегда поверх других окон. В любой момент только одно окно может быть активным. Каждое окно имеет заголовок и номер, показанные в его верхней строке. Для активного окна там же расположены два управляющих поля, заключенных в квадратные скобки: поле справа используется для раскрытия окна мышью на полный экран, а поле слева — для закрытия окна. Многие окна для управления положение текста имеют по правой и нижней границам вертикальную и горизонтальную полосы прокрутки (скроллинга). Поля изменений размеров окна — это символы нижних углов рамки окна.

Операции с окнами могут выполняться тремя способами:

1. Через команды главного меню Window.
2. С помощью манипулятора мышь.
3. При помощи «горячих» клавиш.

Закрыть можно только активное окно. Для этого либо выбирается в меню Windows команда Close, либо нажимается «горячая» клавиша Alt+F3, либо мышью выбирается поле [•] на окне.

Переключение между окнами осуществляется так. Выбирается команда List… из меню Window или нажимается «горячая» клавиша Alt+0. Открывается окно диалога, в котором можно выбрать любое из открытых и ранее закрытых окон. Если на экране отображена хотя бы небольшая часть необходимого окна, достаточно в эту область установить мышь и нажать ее левую кнопку, чтобы окно стало активным. Циклический просмотр окон возможен при помощи клавиши F6.

Активное окно может быть раскрыто на весь экран либо выбором Window-Zoom, либо нажатием клавиши F5, либо выбором мышью поля [↑].

Для просмотра результатов выполнения программы, если вывод выполняется в текстовом режиме, используется переключение в окно вывода (Window-Output). Для просмотра результатов как в текстовом, так и графическом режимах, следует активизировать окно экрана пользователя (Window- User screen) или воспользоваться одновременным нажатием клавиш Alt+F5. Возврат в среду происходит при нажатии любой клавиши.

Переключение в режим редактирования выполняется автоматически при выборе команды New в меню File или при открытии файла. Для возврата из меню в режим редактирования достаточно нажать клавишу Esc.

Команды вставки и удаления

Под блоком понимается выделенное подмножество символов.

Ins — режим вставки/замены;

Del — удалить символ в позиции курсора;

Backspase — удалить символ слева от курсора;
Ctrl+Y — удалить строку;
Ctrl+N — вставить строку.

Команды работы с блоками

Shift+клавиши со стрелками — выделение блока текста;
Ctrl+Ins — копировать блок в буфер обмена;
Shift_Ins — копировать блок из буфера обмена в текущую позицию курсора;
Ctrl+Del — удалить блок;
Shift+Del — вырезать блок в буфер обмена.

Система программирования Borland C++ включает:

1. Интегрированную среду программирования (IDE).
2. Компилятор исходного текста программы.
3. Редактор связей (компоновщик).
4. Библиотеки заголовочных файлов.
5. Библиотеки функций.
6. Программы-утилиты.

Задание опций интегрированной среды

При работе с IDE необходима настройка нужных опций (дополнительных параметров). Все опции имеют значения по умолчанию. Рассмотрим основные опции, настраиваемые с помощью команд меню Options.

Первым шагом является задание директорий, используемых текстовым редактором, компилятором и компоновщиком (рис. 2).

Рис. 2

Для этого используется команда Options\Directories (мы будем использовать формат записи Меню\Меню\…\Команда для экономии места). После ввода Include Directories используется для задания директориев заголовочных файлов. В поле ввода разрешается указывать несколько директориев, разделяемых символом «;». После

ввод Library Directories задает директории, содержащие объектный файл загрузчика (CO?.OBJ, где ? — это буквы M, S, H, T, L, C в зависимости от используемой модели памяти) и файлы библиотек функций (*.LIB). Поле ввода Output Directory задает директорий, в котором помещаются файлы с расширениями *.OBJ, *.EXE, *.MAP. Если в поле — пустая строка, используется текущий директорий.

При выборе строки Options\Compileer открывается еще одно меню для настройки опций компилятора. Наиболее важные опции задаются при выборе команды Code generation. Опция считается выбранной, если она помечена символом (•), и включенной, если помечена символом [X]. Самым важным пунктом в окне Code generation является выбор модели памяти. Для большинства программ, разрабатываемых для ОС MS-DOS, нужно выбрать SMALL модель памяти.

Набор текста программы

Следующим шагом является ввод программы с использованием текстового редактора и сохранение исходного текста программы в файле.

Набор текста программы можно выполнить встроенным или любым другим текстовым редактором. Файлы исходных текстов программ на языке С имеют расширение *.cpp.

Не следует начинать компиляцию, компоновку или запуск программы без сохранения сделанного набора! Запущенная на выполнение программа может вызвать «зависание» компьютера, и сделанный набор будет потерян.

К программам-утилитам относят ассемблер, препроцессор, отладчик, программу профилирования и многие другие полезные программы.

Компиляция, редактирование связей, запуск программы на выполнение

Borland C++ включает богатейшие библиотеки функций для управлениями файлами, выполнения ввода-вывода и многих других действий. Прототипы (заголовки функций с описанием типов формальных параметров и типа возвращаемого функцией значения), символические константы и другие макро, связанные с библиотечными функциями, объединяют в заголовочные файлы, которые по традиции имеют расширение .h. Необходимые для компиляции файлы включаются в текст программы при помощи препроцессорной директивы #include. При запуске на компиляцию текст программы сначала обрабатывается препроцессором, который обрабатывает только «свои» директивы (в частности, вместо директивы «#include имя_файла» встраивается из библиотеки INCLUDE файл, имя которого задано в директиве), а затем текст программы передается непосредственно на обработку компилятору.

Компиляция исходного текста программы инициируется либо через команду Compile\Compile to OBJ, либо нажатием «горячих» клавиш Alt+F9. Команда Make EXE file также запускает программу на компиляцию и при отсутствии синтаксических ошибок автоматически запускает компоновщик для получения *.EXE — файла. Еще одна возможность для запуска программы на компиляцию — команда Run\Run (Ctrl+F9). После успешной компиляции и компоновки запускается полученный *.EXE-файл на выполнение.

Все сообщения и предупреждения IDE помещает в окно по имени Message. Это окно активно после завершения компиляции. Если в программе обнаружены ошибки, включаются средства трассировки ошибок, которые связывают строки текста программы в окне редактора со строками окна Message. Перемещение высвечивания клавишами со стрелками в окне Message синхронно сопровождается высвечиванием соответствующих ошибочных строк в тексте программы. При нажатии клавиши Enter активизируется окно редактора и курсор устанавливается на ошибочную строку. Нажатие клавиши F6 (переход или активизация следующего окна) вновь делает активным окно Message.

Многофайловая компиляция

При модульном программировании не обойтись без многофайловой компиляции. При работе с большими программами намного удобнее размещать части программы не в одном, а в нескольких файлах. Каждый файл должен включать целиком одну или несколько функций. Имена этих файлов записываются в специальный файл — файл проекта, из которого IDE узнает, какие из текстовых файлов следует объединять в исполняемый (*.EXE) файл.

Все необходимые для работы с файлами проектов команды включены в меню Project.

Для организации файла проекта необходимо открыть файл проекта. Для этого команда Project\Open Project… IDE активизирует специальное окно Project в нижней части экрана и открывает окно диалога, позволяющее загрузить нужный файл проекта или создать новый с заданным именем.

Если создан новый файл проекта, окно Project первоначально будет пустым. Включение файлов в проект и их удаление выполняются либо через команды Project\Add item… и Project\Delete item, либо нажатием клавиш Ins и Del, в случае если курсор размещен в окне Project. При добавлении файлов в проект открывается окно диалога, позволяющее выбрать нужный файл.

Окно Project упрощает переход от одного файла, включенного в проект, к другому при их редактировании. Для этого высвечивание перемещается на нужную строку окна Project и нажимается клавиша ENTER.

При работе со средой Borland C++ рекомендуется использовать проект, даже если программа состоит из одного файла.

Отладка программы

В процессе отладки вы можете:

1. Осуществлять пошаговое выполнение программы. После перехода каждой ее строки будет производиться приостановка, позволяющая проанализировать промежуточные результаты.
2. Проверять значение и местоположение (адрес) некоторой переменной в ходе выполнения программы.
3. Просмотреть последовательность вызова функций в программе.

Существует два режима пошагового выполнения программы:

1. Трассировка с заходом в тело функции, при встрече ее вызова в тексте программы (F7);
2. Пошаговое выполнение функции (как обычной команды без захода в тело функции), при встрече ее вызова в тексте программы (F8).

Команда Run\Trace into (F7) запускает программу на отладку. Интегрированная среда высвечивает строку программы, содержащую точку входа main(). После этого нажатием клавиши F7 вызывается выполнение кода, соответствующего одной строке текста программы. Если в строке записана ссылка на функцию, начинается трассировка по тексту функции. При необходимости выполнения строки функции за один шаг, используется клавиша F8 (команда Run\Step over).

Для ускорения процесса отладки используется команда Run\Go to cursor (F4). Программа выполняется до строки, в которой в данный момент располагается текстовый курсор. Можно также задать режим выполнения до точки останова (через опцию подменю «Debug\Toggle breakpoint» или одновременным нажатием клавиш Ctrl и F8, в дальнейшем будем использовать запись «Ctrl+F4»). При этом строка в точке останова подсвечивается обычно красным фоном. Снять установку точки останова можно повторным выполнением описанной команды, размещая курсор на подсвеченной строке останова.

Для наблюдения за изменением значений переменных в ходе выполнения программы используется подменю Debug\Watches\Add watch или «Ctrl+F7». В появившемся окне Add Watch (вызов окна Add Watch можно также получить, если нажать клавишу Ins, предварительно сделав активным окно Watch) необходимо ввести имя переменной, значение которой необходимо просмотреть и нажать Ввод.

Указанная переменная размещается в окне Watch, создаваемом в нижней части экрана, и в процессе отладки через это окно можно наблюдать за изменением размещенных в нем переменных. Удалить переменную из окна Watch можно при помощи клавиши «Del», предварительно выделив ее подсветкой.

Используя опцию меню Evaluate\modify или «Ctrl+F4», можно изменить значение переменной в процессе выполнения отладки, чтобы протестировать алгоритм с новым заданным значением. Окно этой опции «Evaluate and Modify» можно использовать и в качестве

калькулятора, если записать выражения с переменными в строке «Expression» и нажать клавишу «Evaluate» для получения результата в строке Result.

Св. план 2003, поз. 27

Учебное издание

Бусько Виталий Леонидович,
Корбит Анатолий Григорьевич,
Коренская Ирина Николаевна и др.

ПРОГРАММИРОВАНИЕ

Лабораторный практикум
для студентов 1–2-го курсов
всех специальностей БГУИР всех форм обучения

В 2-х частях

Часть 2

Основы програмирования на алгоритмическом языке С

Редактор Е.Н. Батурчик
Компьютерная верстка Т.В. Шестакова

Подписано в печать 19.02.2003. Формат 60x84 1/16. Бумага офсетная.
Печать ризографическая. Гарнитура «Ариал». Усл. печ. л. 4,42.
Уч.-изд. л. 3,8. Тираж 500 экз. Заказ 765.

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