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

Инженерно-экономический факультет

Кафедра экономической информатики

Н. А. Кириенко, Е. А. Полоско, А. А. Ефремов

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В ДВУХ ЧАСТЯХ
ЧАСТЬ 1

ИСПОЛЬЗОВАНИЕ ЯЗЫКА СИ В ИНТЕГРИРОВАННОЙ СРЕДЕ РАЗРАБОТКИ VISUAL STUDIO

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальностей 1-28 01 01 «Экономика электронного бизнеса», 1-28 01 02 «Электронный маркетинг», 1-40 05 01 «Информационные системы и технологии (по направлениям)»

Минск БГУИР 2024

УДК [004.021+004.42](076.5)
ББК 32.973.2я73
К43

Р е ц е н з е н т ы:

кафедра управления и экономики учреждения образования
«Частный институт управления и предпринимательства»
(протокол № 2 от 27.09.2022);

заведующий лабораторией синтеза технических систем
государственного научного учреждения
«Объединенный институт проблем информатики
Национальной академии наук Беларуси»
доктор технических наук, профессор С. В. Медведев

Кириенко, Н. А.

К43 Основы алгоритмизации и программирования. Лабораторный практикум. В 2 ч. Ч. 1 : Использование языка Си в интегрированной среде разработки Visual Studio : пособие / Н. А. Кириенко, Е. А. Полоско, А. А. Ефремов. – Минск : БГУИР, 2024. – 123 с. : ил.

ISBN 978-985-543-704-9 (ч. 1).

Содержит описания лабораторных работ, выполняемых студентами, осваивающими дисциплину «Основы алгоритмизации и программирования». В описании каждой работы представлен краткий теоретический материал и подробная инструкция по разработке приложения.

УДК [004.021+004.42](076.5)
ББК 32.973.2я73

ISBN 978-985-543-704-9 (ч. 1)
ISBN 978-985-543-735-3

Содержание

Введение

Дисциплина «Основы алгоритмизации и программирования» является основной базовой частью цикла дисциплин по информационным технологиям, изучаемых студентами специальностей общего высшего образования 1-28 01 01 «Экономика электронного бизнеса», 1-28 01 02 «Электронный маркетинг», 1-40 05 01 «Информационные системы и технологии (по направлениям)» на протяжении всего курса обучения в БГУИР.

Целью данной дисциплины является формирование у студентов базовых понятий и навыков создания программных комплексов на языке программирования Си в операционной среде Windows, без которых невозможно изучение многих последующих дисциплин данного направления, а также эффективное использование информационных технологий в специальных дисциплинах.

Задачами изучения дисциплины является овладение знаниями и навыками использования языка Си и среды Microsoft Visual Studio для разработки широкого круга приложений, применяемых для автоматизации решения экономических задач.

Особенностью изучения дисциплины для студентов данных направлений специальности является рассмотрение тем и задач, наиболее тесно связанных с экономическими, экономико-математическими, логистичекими проблемами.

В процессе изучения дисциплины студенты прослушивают курс лекций и выполняют серию лабораторных работ. В пособии представлено девять лабораторных работ, даны подробные инструкции по их выполнению. Темы лабораторных работ охватывают наиболее важные разделы изучаемой дисциплины. В каждой работе приведен список индивидуальных заданий.

Лабораторная работа № 1
Структура программы на Си. Функции ввода-вывода

Цель работы – познакомиться с интегрированной средой разработки Visual Studio и средствами отладки программ. Научиться создавать простую программу на языке Си и использовать функции ввода-вывода.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Два этапа создания программ. Программа на языке Си, так же как и на большинстве современных языков программирования, создается в два этапа:

–трансляция – перевод текста программы в машинные коды;

–компоновка – сборка частей программы и подключение стандартных функций.

Исходный файл программы на языке Си имеет расширение *.c или *.cpp (расширение *.cpp говорит о том, что в программе могут быть использованы возможности языка Си++).

Структура программы. Функция вывода. Рассмотрим довольно простую программу на Си – печать строки текста:

Пример 1.1

```
/* Первая программа на С */
#include<stdio.h>
int main ()
{
  printf ("Welcome to C!\n");
  return 0;
}
```

Результат работы программы:

```
Welcome  to  C!
```

Поясним программу. Строка `/* Первая программа на С */` начинается символами `/*` и заканчивается символами `*/`, означающими, что эта строка является комментарием. Программисты вставляют в код комментарии для документирования программ и для того, чтобы сделать их более удобочитаемыми. Комментарии не оказывают никакого влияния на работу компьютера во время исполнения программы. Для однострочного комментария также используют строку `//`.

Строка `#include<stdio.h>` является директивой препроцессора. Препроцессор – это специальная программа, которая обрабатывает текст программы перед выполнением транслятора. Все команды препроцессора начинаются знаком #. Эта директива сообщает препроцессору, что необходимо включить в программу содержание стандартного заголовочного файла ввода или вывода.

Для подключения каждого нового заголовочного файла надо использовать новую команду `#include`. Заголовочный файл содержит информацию и объявления, используемые компилятором во время компиляции вызовов стандартных функций ввода/вывода, таких, как `printf()`. Кроме того, заголовочный файл содержит информацию, которая помогает компилятору определить, корректно ли написаны обращения к библиотечным функциям.

Строка `int main()` обозначает начало определения главной функции программы, с которой начинается выполнение любого проекта на языке Си. Программа может содержать произвольное количество функций, однако одна из функций обязательно должна быть `main()`.

Левая фигурная скобка должна предварять тело каждой функции. Правая фигурная скобка должна стоять в конце каждой функции. Эта пара скобок и часть программы между ними называется блоком.

Строка `printf("Welcome to C!\n");` дает компьютеру команду выполнить действие, а именно вывести на экран строку символов, находящуюся внутри кавычек. Вся строка, включая `printf`, аргументы внутри круглых скобок и точку с запятой, называется оператором. Каждый оператор должен заканчиваться точкой с запятой. Результатом выполнения данного оператора является вызов функции `printf()`, которая выводит сообщение `Welcome to C!` на экран.

Символы обычно печатаются так, как они записаны внутри двойных кавычек в функции `printf()`. Символ `\n` является служебным символом и означает перевод курсора на начало следующей строки на экране. Функция `printf()` – одна из многих функций, входящих в стандартную библиотеку ввода-вывода языка Си.

Оператор `return 0;` означает завершение программы.

Если запустить рассмотренную выше программу, то обнаружится, что программа сразу заканчивает работу и возвращается обратно в оболочку, не дав нам посмотреть результат ее работы на экране. Чтобы избежать этого, необходимо перед выходом из программы вызвать функцию `_getch()`, которая будет ожидать ввода символа с клавиатуры, и до тех пор не завершит свою работу.

Пример 1.2

```
/* Первая программа на С */
#include<stdio.h>
#include<conio.h>    // для использования _getch()
int main ()
```

```
{
  printf ("Welcome to C!\n");
  _getch();
  return 0;
}
```

Результат выполнения программы в примере 1.2 такой же, что и в примере 1.1. Каждая последующая функция `printf()` возобновляет печать с того самого места, на котором остановилась предыдущая функция `printf()`. В примере 1.3 первая функция `printf()` печатает `Welcome` и следующий за ним пробел, вторая функция `printf()` начинает печатать с позиции, следующей сразу за пробелом.

Пример 1.3

```
/* Печать в одну строку двумя вызовами printf */
#include<stdio.h>
int main ()
{
   printf("Welcome ");
   printf("to C!\n");
   return 0;
}
```

Один оператор `printf` может напечатать несколько строк, если использовать символы перехода на новую строку, как показано в примере 1.4.

Пример 1.4

```
/* Печать нескольких строк одним  вызовом  printf  */
#include<stdio.h>
int main()
{
   printf("Welcome\nto\nC!\n");
   return 0;
}
```

Здесь `\n` – управляющий символ.

Управляющие символы не выводятся на экран, а управляют расположением выводимых символов. Отличительной чертой управляющего символа является наличие обратного слэша \ перед ним.

Основные управляющие символы:

– \n – перевод на новую строку;
– \t – горизонтальная табуляция;
– \v – вертикальная табуляция;
– \b – возврат на символ;

– \r – возврат на начало строки;
– \a – звуковой сигнал.

Для подключения русского языка необходимо добавить в проект библиотеку, которая отвечает за локализацию: `#include <locale.h>`. Затем надо написать в начале программы строку `setlocale(LC_ALL,"Rus")`.

Пример 1.5

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  printf("Привет. Как дела?\n");
  return 0;
}
```

Функция ввода. Рассмотрим программу, которая использует стандартную библиотечную функцию `scanf()`, чтобы считать два целых числа, введенные пользователем с клавиатуры, вычислить сумму их значений и вывести результат, используя функцию `printf()`.

Примечание. Функция `scanf()` является функцией незащищенного ввода, т. к. она появилась в ранних версиях языка Си. Чтобы разрешить работу данной функции в современных компиляторах, необходимо в начало программы добавить строчку `#define _CRT_SECURE_NO_WARNINGS`. Другой вариант – воспользоваться функцией защищенного ввода `scanf_s()`, которая появилась несколько позже, но содержит тот же самый список параметров.

Пример 1.6

```
/* Вычислить произведение двух целых чисел */
#include<stdio.h>
#include<locale.h>
int main()
{
   setlocale(LC_ALL, "rus");
   int a, b, pr;  // объявление
   printf("Введите первое целое число\n");  // вывод на экран
   scanf_s("%d", &a);  // ввод целого числа a
   printf("Введите второе целое число\n"); // вывод на экран
   scanf_s("%d", &b);  // ввод целого числа b
   pr = a * b;  // присвоить произведение  a на b  переменной
   printf("Произведение равно %d\n", pr);  // вывод на экран
   return 0;  // завершение программы
}
```

Результат работы программы:

```
Введите первое целое число
5
Введите второе целое число
3
Произведение равно 15
```

Каждая программа начинает исполняться с функции `main()`. Строка `int a, b, pr;` является определением переменных `a, b, pr`.

Переменная – это ячейка памяти, в которую можно записывать значение, предназначенное для использования программой. Переменные `a`, `b` и `pr` принадлежат к типу `int`, в этих переменных будут храниться целые величины. Все объекты (переменные, массивы и т. д.), с которыми работает программа в Си, необходимо декларировать. При декларировании объекты можно инициализировать (задавать начальные значения).

Пример:

```
int j = 10, m = 3, n;
float c = -1.3, l = -10.23, n;
```

Именем переменной в Си может служить буквенно-цифровая последовательность символов алфавита, начинающаяся с буквы. Кроме букв и цифр можно также использовать символ нижнего подчеркивания _, который в этом случае рассматривается как буква, т. е. может быть первым в последовательности символов. Регистр имеет значение, т. е. большая и маленькая буква – это две разные переменные.

Функция `printf("Введите первое целое число\n");` выводит на экран строку приглашения.

Функция `scanf_s("%d", &a);` считывает данные со стандартного устройства ввода, которым обычно является клавиатура. В нашем случае функция `scanf_s()` имеет два аргумента, "`%d`" и `&a`. Первый аргумент – управляющая строка, задает формат считывания, тем самым определяя тип данных, которые предстоит ввести пользователю. Спецификация преобразования `%d` означает, что вводимые данные будут преобразованы в формат `int`. Второй аргумент начинается со знака & (амперсанд), которым в Си задается операция взятия адреса следующей за ним переменной.

Функция `scanf_s()` ждет, пока пользователь введет значение для переменной `a`. Завершением ввода является нажатие клавиши `Enter`.

Функция `printf("Введите второе целое число\n");` выводит на экран строку приглашения.

Функция `scanf_s("%d", &b);` получает от пользователя значение для переменной `b`. В результате выполнения оператора присваивания `pr = a * b;` вычисляется произведение переменных `a` и `b`, и его значение присваивается переменной `pr`.

Функция `printf("Произведение равно %d\n", pr);` выводит на экран текст `Произведение равно`, после которого следует численное значение переменной `pr`.

В табл. 1.1 представлены базовые типы данных, которые используются для объявления переменных.

Таблица 1.1

Базовые типы данных

Команда	Тип
`char`	символьный
`int`	целый
`float`	вещественный
`double`	вещественный двойной точности
`void`	не имеющий значения

С базовыми типами данных могут использоваться спецификаторы, представленные в табл. 1.2.

Таблица 1.2

Спецификаторы типов

Спецификатор	Тип
`unsigned`	без знака
`signed`	со знаком
`short`	короткий
`long`	длинный

В функциях `printf()` и `scanf_s()` необходимо использовать следующие спецификации преобразования:

- `%i` – для ввода/вывода целых со знаком;
- `%d` – для ввода/вывода беззнаковых целых;
- `%f` – для ввода/вывода вещественных в виде числа с плавающей точкой;
- `%c` – для ввода/вывода переменной символьного типа;
- `%s` – для ввода/вывода строки.

Количество спецификаций преобразования должно обязательно соответствовать количеству переменных, указанных после управляющей строки.

Кроме переменных в программе могут использоваться константы – данные, которые не могут быть изменены в процессе выполнения программы. Для объявления констант перед указанием типа константы и ее именем необходимо добавить ключевое слово `const`.

Примечание. В языке программирования Си существует несколько форматов вывода вещественных чисел. Наиболее распространенный – `"%f"`. По умолчанию он выводит шесть знаков после запятой. Однако можно указать желаемое количество знакомест (через точку перед буквой `f`), а также ширину поля (до точки):

```
printf("%f\n", 0.145);
printf("%.3f\n", 0.1234567);
printf("%10.3f\n", 0.1234567);
```

Результат:

```
0.145000
0.123
0.123
```

В примере 1.7 показано объявление переменных разных типов данных и примеры использования функций ввода-вывода. Обратите внимание, что переменные символьного типа берутся в одинарные кавычки.

Пример 1.7

```
#include<stdio.h>
#include<locale.h>
int main()          // определение головного модуля
{
  setlocale(LC_ALL, "rus");
  int i1, i2, i3;  // три переменные целого типа со знаком
  int ix = 5, iy = -7;
  float f1 = -1.575, f2 = 3.14; /* переменные вещественного
типа */
  char let, symb = 'z', n_str = '\n';  /* символьные перемен-
ные */
  const float pi = 3.14;     // вещественная константа
  printf("Введите 3 целых числа\n");
  scanf_s("%d%d%d", &i1, &i2, &i3);
  printf("Введенное первое число = %d\nВведенное второе число =
%d\nВведенное третье число = %d\n", i1, i2, i3);
  printf("Вещественное f1 = %.3f\nВещественное f2 = %.2f\n",
f1, f2);
  printf("Символ symb = %c\nКод символа %c = %c\n", symb,
symb, n_str);
  return 0;
}
```

Особенности выполнения арифметических операций. При использовании деления надо помнить, что при делении целого числа на целое остаток от деления отбрасывается, таким образом, 7 / 4 будет равно 1. Если же надо получить вещественное число и не отбрасывать остаток, делимое или делитель надо привести к вещественной форме.

Пример 1.8

```
#include<stdio.h>
int main()          // определение головного модуля
{
  int i = 7, n;
  float res;
  res = i / 4;           // res = 1, делится целое на целое
  res = i / 4.;         // res = 1.75, делится целое на дробное
  res = (float)i / 4; // res = 1.75, делится дробное на целое
  n = 7. / 4.;  /* n = 1, результат записывается в целую пе-
  ременную */
  return 0;
}
```

Если надо вычислить остаток от деления переменной `a` на переменную `b` и результат записать в переменную `ostatok`, то оператор присваивания выглядит как `ostatok = a%b`.

Директива `#define` определяет так называемые макросы. Если мы напишем `#define TRAM 10`, то в процессе работы препроцессора все вхождения буквосочетания `TRAM` будут в текстовом виде заменены на `10`. При этом надо отметить, что `TRAM` должно быть отдельным токеном (т. е. вокруг должны быть либо пробельные символы, либо знаки препинания). То есть `TRAM + TRAM` будет заменено на `10 + 10`, а вот `TRAMPAMPAM` останется без изменений.

Пример 1.9

```
#include<stdio.h>
#define CHARECTER '@'
#define STRING "Privet!"
#define OCTAL 122
int main()        // определение головного модуля
{
 printf("CHARACTER = %c\n", CHARECTER);
 printf("STRING = %s\n" STRING);
 printf("OCTAL - %d\n", OCTAL);
 return 0;
}
```

Результат выполнения программы:

```
CHARECTER = @
STRING = Privet!
OCTAL - 122
```

Математические функции. Для работы математических функций в Си необходимо подключить библиотеку `math.h`. В табл. 1.3 представлены наиболее часто используемые математические функции.

Таблица 1.3

Математические функции

Команда	Функция
`sqrt(x);`	Возвращает корень квадратный переменной x
`abs(n);`	Возвращает модуль целого числа n
`fabs(m);`	Возвращает модуль вещественного числа m
`sin(x);`	Возвращает синус числа x
`tan(x)`	Возвращает тангенс числа x
`cos(x);`	Возвращает косинус числа x
`log(x)`	Возвращает натуральный логарифм от x
`log10(x)`	Возвращает десятичный логарифм от x
`exp(x)`	Возвращает e^x
`pow(x,y)`	Возведение в степень x^y

Пример 1.10

Вычислить

$$h = \frac{x^{y+1} + e^{y-1}}{1 + x \cdot |y - tg(z)|} \cdot (1 + |y - x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}$$

при x = 2,444, y = 0,00869, z = –130. Должно быть получено –0,49871.

```
#include<stdio.h>
#include<locale.h>
#include<math.h>
int main()        //  определение головного модуля
{
  setlocale(LC_ALL, "rus");
  float x, y, z, h, a, b, c, d;
  printf("Введите x, y, z\n");
  scanf_s("%f%f%f", &x, &y, &z);
  a = (pow(x, y + 1) + exp(y - 1)) / (1 + x * fabs(y - tan(z)));
  b = 1 + fabs(y - x);
  c = (pow(fabs(y - x), 2)) / 2;
  d = (pow(fabs(y - x), 3)) / 3;
  h = a * b + c - d;
  printf("h = %.5f\n", h);
  return 0;
}
```

Индивидуальные задания

Создать файл проекта и разработать Си-программу.

1. Найти сумму членов геометрической прогрессии b, $b \cdot q$, ..., $b \cdot q^{(n-1)}$ по введенным значениям b, q, n, $q \neq 1$.

2. Ввести два действительных числа. Найти среднее арифметическое этих чисел и среднее геометрическое из модулей. Результат вывести с точностью до трех знаков после запятой.

3. Ввести радиус шара. Найти площадь поверхности и объем шара. Результат вывести с точностью до двух знаков после запятой.

4. Ввести длины сторон прямоугольного параллелепипеда. Найти площадь поверхности и объем параллелепипеда. Результат вывести с точностью до трех знаков после запятой.

5. Ввести длины катетов прямоугольного треугольника. Найти его гипотенузу и площадь. Результат вывести с точностью до двух знаков после запятой.

6. Ввести длины сторон равностороннего треугольника. Найти площадь этого треугольника и его высоту. Результат вывести с точностью до трех знаков после запятой.

7. Дано трехзначное число. Найти сумму и произведение цифр этого числа.

8. Треугольник задан координатами своих вершин. Найти периметр треугольника. Координаты вершин вводятся с клавиатуры.

9. Ввести длины ребер куба. Найти объем куба и площадь его боковой поверхности. Результат вывести с точностью до трех знаков после запятой.

10. Найти сумму членов арифметической прогрессии a, a + d, ..., $a + (n - 1) \cdot d$ по введенным значениям a, d, n.

11. Вычислить расстояние между двумя точками с координатами (x1, y1) и (x2, y2). Координаты точек вводятся с клавиатуры. Результат вывести с точностью до двух знаков после запятой.

12. Ввести радиус круга. Найти длину окружности и площадь круга, ограниченного этой окружностью. Результат вывести с точностью до двух знаков после запятой.

13. Дано четырехзначное число. Найти удвоенное произведение суммы его цифр.

14. Ввести радиус, длину ребра и высоту конуса. Найти площадь боковой поверхности, площадь полной поверхности и объем конуса. Результат вывести с точностью до трех знаков после запятой.

15. Ввести высоту и радиус цилиндра. Найти площадь боковой поверхности, площадь полной поверхности и объем цилиндра. Результат вывести с точностью до двух знаков после запятой.

16. В трехзначном числе x зачеркнули первую цифру. Когда оставшееся число умножили на 10, а произведение сложили с первой цифрой числа x, то получилось число 564. Найти число x.

17. Дано трехзначное число. Найти число, полученное при прочтении его цифр справа налево.

18. Даны действительные положительные числа a, b, c. По трем сторонам с длинами a, b, c можно построить треугольник. Найти углы треугольника.

19. Треугольник задан координатами своих вершин. Найти площадь треугольника. Координаты вершин вводятся с клавиатуры.

20. Из трехзначного числа x вычли его последнюю цифру. Когда результат разделили на 10, а к частному слева приписали последнюю цифру числа x, то получилось число 237. Найти число x.

21. Вычислить полное сопротивление цепи со следующими параметрами: активное сопротивление R, емкость C и индуктивность L, ω = 0,2. Значения R, L, C – ввести с клавиатуры.

22. Даны катеты прямоугольного треугольника. Найти периметр и гипотенузу треугольника.

23. Найти площадь равнобедренной трапеции с основаниями a и b и углом α при большем основании a.

24. Даны три числа a, b, c. Найти среднее арифметическое квадратов этих чисел. Значения a, b и c ввести с экрана.

25. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.

26. Идет k-я секунда суток. Определить, сколько целых часов (H) и целых минут (M) прошло с начала суток. Вывести на экран фразу «Это … часов … минут». Вместо многоточий должны стоять вычисленные значения H и M.

27. Дано трехзначное число. Найти среднее арифмитическое его цифр.

28. В трехзначном числе x зачеркнули его вторую цифру. Когда к образованному при этом двузначному числу слева приписали вторую цифру числа x, то получилось число 546. Найти число x.

29. Даны основания равнобедренной трапеции и угол при большем основании. Найти площадь трапеции.

30. Дано трехзначное число. Найти утроенную сумму его цифр.

31. Известны первый и пятый члены арифметической прогрессии. Найти величину члена прогрессии с номером N и сумму N членов.

32. Известны члены арифметической прогрессии с номерами N и M. Найти сумму членов с номерами от M до N (считать M < N).

33. Заданы первый член и знаменатель геометрической прогрессии. Найти сумму членов с номерами от заданного номера K до заданного номера P (считать K < P).

34. Повесть Рэя Брэдбери называется «451 градус по Фаренгейту». Напечатать название повести в градусах по Цельсию. Формула перевода:

$$t_c = \frac{5}{9} \cdot (t_F - 32).$$

35. В классе N учеников. После контрольной работы было получено A – пятерок, B – четверок, C – двоек, остальные – тройки. Найти процент троек.

36. Найти сумму и произведение цифр заданного трехзначного целого числа K.

37. Задано целое трехзначного число K. Найти число, полученное из исходного путем выписывания его цифр в обратном порядке.

38. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения.

39. Определить время падения камня на поверхность с высоты h.

40. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.

41. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a, b, c.

42. Вычислить объем цилиндра с радиусом основания r и высотой h.

43. Определить расстояние, пройденное физическим телом за время t, если тело движется с постоянным ускорением a и имеет в начальный момент времени скорость V_0.

44. Вычислить площадь треугольника по формуле Герона, если заданы его стороны a, b, c. Формула Герона:

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

где p – полупериметр треугольника.

45. Определить координаты вершины параболы $y = ax^2 + bx + c$ ($a \neq 0$). Коэффициенты a, b, c ввести.

46. По данным сторонам прямоугольника вычислить его периметр, площадь и длину диагонали.

47. Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей.

48. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

49. Найти длину окружности и площадь круга заданного радиуса r.

50. Даны координаты трех вершин треугольника (x_1, y_1), (x_2, y_2), (x_3, y_3). Найти его периметр и площадь.

51. Скорость первого автомобиля V_1 км/ч, скорость второго – V_2 км/ч, расстояние между ними S км. Определить расстояние между ними через t ч, если автомобили первоначально движутся навстречу друг другу.

52. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч ($U < V$). Время движения лодки по озеру t_1 ч, а по реке (против течения) – t_2 ч. Определить путь S, пройденный лодкой.

53. Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.

54. Плотность железа – 7,9 г/см^3, молярная масса – 56 г/моль, N_A = 6,02 × × 1023. Сколько атомов содержится в данном объеме (V) железа?

55. Найти площадь кольца, внутренний радиус которого равен r_1, а внешний радиус равен r_2 ($r_1 < r_2$).

56. Найти площадь и меньшее основание прямоугольника по заданной диагонали и большему основанию.

57. В квадрат вписан круг. Найти площадь квадрата и круга по заданной стороне квадрата.

58. Найти длину окружности по заданному радиусу и периметр описанного квадрата.

59. Известны диагонали ромба. Найти его площадь.

60. Заданы: r – радиус окружности с центром в точке (0, 0), k – тангенс угла наклона прямой y = kx к оси Ox. Вычислить координаты точек пересечения прямой $y = k \cdot x$ с окружностью $x^2 + y^2 = r^2$.

Лабораторная работа № 2
Операторы ветвления

Цель работы – познакомиться с правилами построения разветвляющихся алгоритмов. Научиться использовать полную и сокращенную формы оператора `if`.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Управляющие конструкции программы. Управляющие конструкции – это операторы, изменяющие последовательное выполнение программы. К ним относятся:

– составные операторы;
– операторы ветвления;
– операторы перехода;
– операторы циклов.

Составной оператор – последовательность операторов, заключенная в фигурные скобки. Блок – составной оператор, в теле которого присутствуют операторы определения переменных. Тело функции – блок.

Оператор ветвления `if` (сокращенная форма):

```
if (выражение_условие) оператор1;
```

В качестве элемента `выражение_условие` могут использоваться:
– арифметическое выражение;
– операция отношения;
– логическое выражение.

Если `выражение_условие` истинно `(true)`, выполняется оператор 1.

Если `выражение_условие` ложно `(false)`, выполняется оператор, следующий за условным.

Оператор ветвления `if` (полная форма):

```
if (выражение_условие)оператор1;
else оператор2;
```

Если `выражение_условие` истинно `(true)`, выполняется оператор 1.

Если `выражение_условие` ложно `(false)`, выполняется оператор 2.

Если при соблюдении или несоблюдении условия надо выполнить несколько операторов программы, то эти операторы следует заключить в фигурные скобки.

При помощи вложенных одна в другую нескольких инструкций `if` можно реализовать множественный выбор.

Пример 2.1

Найти максимальное из трех чисел a, b, c.

```
#include<stdio.h>
int main()
{
  int a = 2, b = 5, c = 8, max;
  if (a > b && a > c)
    max = a;
  else if (b > c)
    max = b;
  else max = c;
  printf("max = %d", max);
  return 0;
}
```

Пример 2.2

Написать программу (используя инструкцию `if`), которая запрашивает у пользователя номер месяца и затем выводит соответствующее название времени года. В случае если пользователь введет недопустимое число, программа должна вывести сообщение `Ошибка ввода данных`.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "Rus");
  int n;
  printf("Введите номер месяца\n");
  scanf_s("%d", &n);
  if (n == 1 || n == 2 || n == 12) printf("Зима");
  else if (n >= 3 && n <= 5) printf("Весна");
  else if (n >= 6 && n <= 8) printf("Лето");
  else printf("Осень");
  return 0;
}
```

Тернарная условная операция. Синтаксис тернарной операции:

```
<выражение_условие>?<выражение_1>:<выражение_2>
```

Если <выражение_условие> истинно, то тернарная операция возвращает <выражение_1>, в противном случае – <выражение_2>.

Пример 2.3
Найти минимальное из двух чисел.

```
#include<stdio.h>
int main()
{
int a, b, min;
scanf_s("%d%d", &a, &b);
min = (a > b) ? b : a;
printf("min = %d", min);
return 0;
}
```

Оператор `switch` предназначен для выбора одного из нескольких возможных направлений дальнейшего хода программы.

Первый вариант записи оператора `switch` :

```
switch (выражение)
{
case константа1: оператор1; break;
case константа2: оператор2; break;
…
case константа N: операторы; break;
default: оператор;  break;
}
```

Второй вариант записи оператора `switch`:

```
switch (выражение)
{
case константа1: оператор1; break;
case константа2: оператор2; break;
…
case константа N: операторы; break;
}
```

Выбор последовательности инструкций осуществляется в зависимости от равенства значения переменной-селектора константе, указанной после слова `case`. Если значение переменной-селектора не равно ни одной из констант, записанных после `case`, то выполняются инструкции, расположенные после слова `default`. В качестве переменной-селектора можно использовать переменную целого (`int`) или символьного (`char`) типа.

Пример 2.4

Требуется вывести на экран оценку, введенную цифрой, прописью.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  int rez;
  printf("Введите оценку\n");
  scanf("%d", &rez);
  switch(rez)
  {
    case 5: printf("Оценка – отлично"); break;
    case 4: printf("Оценка – хорошо"); break;
    case 3: printf("Оценка – удовлетворительно"); break;
    case 2: printf("Оценка – неудовлетворительно"); break;
    default: printf("Неверное значение");
  }
  return 0;
}
```

Пример 2.5

Программа выводит на экран меню, в котором пользователь может выбрать одно из следующих действий, введя соответствующее число: подсчитать сумму двух чисел, подсчитать разность двух чисел, выйти из программы.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  int n, a, b;
  printf("Меню:\n");
  printf("1 - подсчет суммы двух чисел\n");
  printf("2 - подсчет разности двух чисел\n");
  printf("3 - выход из программы\n");
  printf("Ваш выбор? \n");
  scanf("%d", &n);
  switch(n)
  {
    case 1: printf("Введите два числа\n");
            scanf("%d%d", &a, &b);
            printf("Сумма равна %d\n", a + b);
            break;
    case 2: printf("Введите два числа\n");
            scanf("%d%d", &a,&b);
            printf("Разность равна %d\n", a - b);
            break;
    case 3: return 0;
```

```
    default: printf("Неверное значение");
  }
  return 0;
}
```

Индивидуальные задания

Задание 1. Создать файл проекта и разработать Си-программу в соответствии с вариантом.

1. Даны два числа x и y. Определить, можно ли через точку (x, y) провести окружность единичного радиуса с центром в начале координат.
2. Дано трехзначное число. Определить, является ли оно палиндромом.
3. Определить, имеется ли среди трех чисел a, b и c хотя бы одна пара равных между собой чисел.
4. Определить, является ли треугольник со сторонами a, b и c равносторонним.
5. Определить, является ли треугольник со сторонами a, b и c равнобедренным.
6. Дано трехзначное число. Определить, является ли сумма его цифр двузначным числом.
7. Определить, имеется среди чисел a, b и c хотя бы одна пара взаимно противоположных чисел.
8. Составить алгоритм, определяющий по координатам вершин треугольника, является ли он тупоугольным.
9. Подсчитать количество отрицательных чисел среди чисел m, n, p.
10. Определить, имеется ли среди целых чисел a, b и c хотя бы одно четное.
11. Дано трехзначное число. Найти его максимальную цифру.
12. Определить количество положительных чисел среди чисел a, b и c.
13. Составить алгоритм, определяющий по координатам вершин треугольника, является ли он прямоугольным.
14. Дано трехзначное число. Определить, кратна ли трем сумма его цифр.
15. Найти сумму членов геометрической прогрессии b, $b \cdot q$, ..., $b \cdot q^{(n-1)}$ по введенным значениям b, q, n, $q \neq 1$.
16. Определить, имеется ли среди целых чисел a, b и c хотя бы одно нечетное.
17. Дано трехзначное число. Определить, равен ли квадрат этого числа сумме кубов его цифр.
18. Дано трехзначное число. Определить, является ли произведение его цифр трехзначным числом.
19. Даны вещественные числа x, y. Определить, принадлежит ли точка с координатами (x, y) кругу радиусом r, а также выдать сообщение `принадлежит` или `не принадлежит`. Значение r задать как константу.

20. Задано натуральное трехзначное число K. Выяснить, образуют ли цифры этого числа упорядоченную последовательность.

21. Задано натуральное трехзначное число K. Выяснить, образуют ли цифры этого числа арифметическую прогрессию.

22. Даны вещественные числа a, b, c. Если они могут быть длинами сторон треугольника, вычислить его периметр и площадь.

23. Даны целые числа k, l, n, m. Выяснить, является ли k делителем всех чисел.

24. Даны длины трех отрезков. Определить, можно ли из них построить треугольник. Будет ли он прямоугольным?

25. Для вещественных чисел m, n и знака операции s вычислить выражение n s m.

26. Выяснить, сколько точек пересечения имеют прямая y = kx + b и гипербола y = a / x. Вывести координаты этих точек. Значения k, b, a ввести с клавиатуры.

27. Вычислить корни биквадратного уравнения $ax^4 + bx^2 + c = 0$.

28. Выяснить, сколько точек пересечения имеют прямые y = kx + b и y = cx + d. Вывести координаты этих точек. Значения k, b, c, d ввести с клавиатуры.

29. Заданы координаты вершин прямоугольника. Определить площадь части прямоугольника, расположенной в первой координатной четверти.

30. На плоскости заданы точки $M_1 (x_1, y_1)$, $M_2 (x_2, y_2)$, $N_1 (x_3, y_3)$, $N_2 (x_4, y_4)$. Проверить, являются ли параллельными прямые, одна из которых проходит через точки M_1, M_2, а другая – через точки N_1, N_2. Если прямые пересекаются, то найти координаты точки пересечения.

31. Два прямоугольника со сторонами, параллельными осям координат, заданы координатами двух своих противоположных вершин: $M_1 (x_1, y_1)$, $M_2 (x_2, y_2)$ – первый прямоугольник, $N_1 (x_3, y_3)$, $N_2 (x_4, y_4)$ – второй прямоугольник. Найти площадь их пересечения.

32. Выяснить, сколько точек пересечения имеют прямая y = kx + b и окружность радиусом r с центром в начале координат. Значения k, b, r ввести с клавиатуры.

33. Заданы коэффициенты квадратного уравнения (a, b, c). Вычислить его корни, предусмотрев вариант, когда любой из коэффициентов равен 0.

34. Ввести целое число N. Выяснить, кратно ли оно трем.

35. Даны действительные числа x, y, z. Получить: L = 2max(x, z) – – 3min(x, y, z).

36. По заданным вещественным числам a, b, c вычислить $\frac{\max(a,b,c)+\min(a,b,c)}{2}$.

37. Ввести два целых числа. Выяснить, являются ли они оба четными или нечетными, либо одно является четным, а другое – нечетным.

38. Даны действительные числа a, b, c. Проверить, выполняются ли неравенства a < b < c.

39. Даны действительные числа a, b, c. Удвоить эти числа, если $a \geq b \geq c$, или заменить их абсолютными значениями, если это не так.

40. Даны два действительных числа. Заменить первое число нулем, если оно меньше второго или равно ему, оставить числа без изменения в противном случае.

41. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3).

42. Даны три числа. Определить, являются ли они последовательными членами арифметической прогрессии, и найти ее разность.

43. Заданы целые числа x, y, m, n. Если разность (x – y) меньше остатка от деления m на n, увеличить x на 1.

44. Задано целое трехзначное число K. Определить, организуют ли цифры этого числа упорядоченную последовательность, и выдать соответствующее сообщение.

45. По четырехзначному номеру трамвайного билета определить, является ли он счастливым (билет считается счастливым, если сумма первых двух цифр номера совпадает с суммой двух его последних цифр).

46. Заданы целые числа: k, l, n, m. Проверить, является ли k делителем всех чисел.

47. Заданы размеры a, b прямоугольного отверстия и размеры x, y, z кирпича. Определить, проходит ли кирпич через отверстие.

48. Заданы числа a, b, c. Напечатать эти числа в порядке убывания их абсолютных величин.

49. Выяснить, попадает ли точка с координатами (x_0, y_0) в кольцо, образованное окружностями $x^2 + y^2 = r^2$ и $x^2 + y^2 = R^2$ $(r < R)$.

50. Определить, в какой координатной четверти находится заданная точка с координатами (x, y).

51. Заданы числа x, y, R. Определить расстояние от точки с координатами (x, y) до контура полукруга радиусом R с центром в начале координат, расположенном в нижней полуплоскости.

52. Заданы координаты (x_0, y_0) точки на плоскости. Проверить, лежит ли точка в верхней полуплоскости, и выдать соответствующее сообщение.

53. Ввести три вещественных числа a, b, c – длины трех отрезков. Если отрезки могут быть сторонами треугольника, найти его периметр и площадь.

54. Определить, является ли заданный год N високосным. Год високосный, если N не кратно 100 и число, составленное из его двух последних цифр, кратно 4. Если N кратно 100, то год високосный лишь при N, кратном 400.

55. Заданы координаты вершин прямоугольника со сторонами, параллельными осям координат. Определить площадь части прямоугольника, расположенной в первой координатной четверти.

56. На плоскости заданы точки M_1 (x_1, y_1), M_2 (x_2, y_2), N_1 (x_3, y_3), N_2 (x_4, y_4). Проверить, являются ли параллельными прямые, одна из которых проходит через точки M_1, M_2, а другая – через точки N_1, N_2. Если прямые пересекаются, то найти координаты точки пересечения.

57. Найти координаты точек пересечения прямой $y = kx + b$ и окружности радиусом R с центром в начале координат. Определить, сколько точек пересечения расположены в первой координатной четверти.

58. Заданы числа k, a, b. Определить число точек пересечения прямой $y = kx + b$ с гиперболой $y = a / x$ и их координаты.

59. Заданы числа k, a, b. Проверить, попадают ли вещественные корни уравнения $x^2 + ax + b = 0$ в интервал $(-k, k)$.

60. Заданы числа a, b, x, y, z ($a < b$). Определить, какие из чисел x, y, z попадают в отрезок [a, b]. Напечатать количество таких чисел.

61. Заданы координаты двух точек на плоскости (x_1, y_1), (x_2, y_2) и числа R, P ($R > 0$). Определить, попадают ли обе точки внутрь полукруга радиусом R с центром в точке (P, 0), лежащего в верхней полуплоскости.

62. Заданы числа a, b, c, d ($a < b$, $c < d$) и x. Определить, принадлежит ли x какому-либо из отрезков [a, b], [c, d] или их общей части. Ответ вывести в виде сообщения.

63. Заданы числа k, b, c, d, e. Определить количество и координаты точек пересечения прямой $y = kx + b$ и параболы $y = cx^2 + dx + e$ и расстояние от каждой точки до начала координат.

64. Заданы числа k, b, a, l. Определить, попадают ли обе точки пересечения прямой $y = kx + b$ и параболы $y = ax^2$ в квадрат со стороной l и центром в начале координат (стороны параллельны осям). Напечатать сообщение и координаты точек.

65. Ввести N_1, N_2, N_3 – количество пропущенных часов занятий в среднем на студента за неделю для трех групп. Если $\min(N_1, N_2, N_3) < 10$, то напечатать `Есть хорошая группа`.

66. Ввести N_1, N_2, N_3 – количество рекламаций на три вида товаров. Напечатать текст `Все товары хорошие`, если $\max(N_1, N_2, N_3) < 5$, иначе – `Есть плохие товары`.

67. По порядковому номеру дня недели выдать сообщение, каким он является – рабочим или выходным.

Задание 2. Создать файл проекта и разработать Си-программу в соответствии с вариантом (использовать конструкцию `switch`).

1. Составить расписание на неделю. После ввода пользователем порядкового номера дня недели на экране должно отобразиться то, что запланировано на этот день.
2. Составить расписание занятий на неделю. После ввода пользователем порядкового номера дня недели, на экран должны выводиться названия предметов.
3. По порядковому номеру месяца определить и вывести пору года.

4. Создать список времен года: лето (1), осень (2), зима (3), весна (4). По введенному значению времени года (от 1 до 4) перечислить все месяцы этого сезона.

5. Составить программу, которая по номеру месяца выдавала бы количество дней.

6. Составить программу, которая по введенному времени года выдавала бы название месяцев, относящихся к нему.

7. Дан список дисциплин, изучаемых в БГУИР, и формы отчетности по ним. Составить программу, которая по названию дисциплины выдавала бы форму отчетности по ней.

```
Информатика (экзамен, зачет)
Культурология (зачет)
Математика (экзамен, зачет)
Иностранный язык (экзамен, зачет)
Экономика(экзамен)
```

8. Дан список дисциплин, изучаемых в БГУИР, и номер семестра, в котором они изучаются. Составить программу, которая по номеру семестра выдавала бы список изучаемых дисциплин.

```
Информатика – 2, 1
Культурология – 3, 4
Математика – 4, 3
Иностранный язык – 4, 1, 2, 3
```

9. Составить программу, которая с помощью оператора `CASE` реализовала бы все возможные операции над двумя целыми числами.

10. Составить программу, которая бы выдавала по названию страны название столицы этой страны (использовать не менее шести названий).

11. Составить программу, которая по введенному числу (до 10) выдавала бы название этой цифры.

12. Составить программу, которая по введенному названию страны выдавала бы название ее континента.

13. Составить программу, которая по значению переменной X, означающей некоторую длину в единицах измерения (дециметр, километр, метр, миллиметр, сантиметр), выдавала бы эту длину в метрах.

14. Составить программу, которая по введенному числу К (до 10) выдавала бы соответствующую ей римскую цифру.

15. Для целого числа К (от 1 до 9) напечатать фразу `мне К лет`, учитывая при этом, что при некоторых значениях К слово `лет` надо заменить на слово `год` или `года`.

16. Вывести на экран список фамилий русских писателей. Например: 1 – Пушкин, 2 – Достоевский и т. п.). Пользователь вводит порядковый номер писателя и на экране отображается название произведения соответствующего автора.

17. По порядковому номеру месяца напечатать его название.

Лабораторная работа № 3
Операторы цикла

Цель работы – познакомиться с правилами построения циклических алгоритмов. Научиться использовать различные формы операторов цикла.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Операции декремента и инкремента. Инкремент (++) – это увеличение переменной на единицу. Декремент (−−) – это уменьшение на единицу. Операции декремента и инкремента удобнее использовать, и выполняются они быстрее, чем арифметические операции.

Синтаксис операций инкремента и декремента:

```
++/*имя переменной*/; /* префиксный инкремент (преинкремент) */
/*имя переменной*/++;/* постфиксный инкремент (постинкремент) */
--/*имя переменной*/;/* префиксный декремент (предекремент) */
/*имя переменной*/--;/* постфиксный декремент (постдекремент) */
```

Когда операция инкремента или декремента ставится перед именем переменной, то такая операция называется операцией префиксного инкремента (преинкремента) или префиксного декремента (предекремента). Если операция инкремента или декремента ставится после имени переменной, то такая операция называется операцией постфиксного инкремента (постинкремента) или постфиксного декремента (постдекремента).

При использовании операции преинкремента значение переменной сначала увеличивается на 1, а затем используется в выражении. При использовании операции постинкремента значение переменной сначала используется в выражении, а потом увеличивается на 1.

При использовании операции предекремента значение переменной сначала уменьшается на 1, а затем используется в выражении. При использовании операции постдекремента значение переменной сначала используется в выражении, а потом уменьшается на 1.

Цикл `for` (параметрический цикл). Синтаксис:

```
for (Инициализация; УсловиеВыполнения; Изменение)
{
     // здесь записываются инструкции цикла
}
```

Инициализация – оператор инициализации счетчика циклов (начальное значение).

УсловиеВыполнения – выражение, значение которого определяет условие выполнения операторов цикла. Операторы цикла выполняются до тех пор, пока УсловиеВыполнения истинно, т. е. не равно нулю.

Изменение – оператор изменения параметра цикла. Как правило, эта инструкция изменяет значение переменной, которая входит в УсловиеВыполнения.

Оператор for используется для организации циклов с фиксированным, числом повторений. Количество повторений цикла определяется начальным значением переменной-счетчика и условием завершения цикла. Переменная-счетчик должна быть целого (int) типа и может быть объявлена непосредственно в операторе цикла.

Пример 3.1

Программа выводит на экран числа от 1 до 10.

```
#include<stdio.h>
void main()
{
   int counter;
   for (counter = 1; counter <= 10; counter++)
      printf("%d\n", counter);
}
```

Пример 3.2

Написать программу, которая суммирует последовательность целых чисел. Предположить, что первое целое число, считываемое с помощью scanf, определяет количество значений, которое осталось ввести. Программа должна считывать только одно значение при каждом выполнении scanf.

Типичной входной последовательностью могло бы быть

5 100 200 300 400 500,

где 5 указывает на то, что должны суммироваться последующие пять значений.

```
#include<stdio.h>
#include<locale.h>
void main()
{
  int n, i, s = 0, ch;
  setlocale(0, "rus");
  // с помощью цикла for
  printf("Введите количество чисел, которые необходимо
сложить");
  scanf_s("%d", &n);
  for (i = 1; i <= n; i++)
  {
```

```
    scanf_s("%d", &ch);
    s += ch;
  }
  printf("Сумма равна %d\n", s);
}
```

Цикл `while` (цикл с предусловием). Синтаксис:

```
while (УсловиеВыполнения)
{
// операторы цикла (тело цикла)
}
```

Сначала проверяется значение выражения `УсловиеВылолнения`. Если оно не равно нулю, т. е. условие истинно, то выполняются операторы цикла (тело цикла). Затем снова проверяется значение выражения `УсловиеВыполнения`, и если оно не равно нулю, тело цикла выполняется еще раз до тех пор, пока значение выражения `УсловиеВыполнения` не станет равным нулю.

Число повторений тела цикла `while` определяется ходом выполнения программы. Операторы цикла выполняются до тех пор, пока значение выражения `УсловиеВыполнения` не станет ложным (равным нулю). Для завершения цикла `while` в теле цикла обязательно должны быть операторы, выполнение которых влияет на условие завершения цикла. Цикл `while` – это цикл с предусловием, т. е. возможна ситуация, при которой инструкции тела цикла ни разу не будут выполнены. Цикл `while`, как правило, используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

Пример 3.3
Программа выводит на экран числа от 1 до 10.

```
#include<stdio.h>
void main()
{
int counter = 1;   /* инициализация */
while (counter <= 10) /* условие повторения */
 {
printf("%d\n", counter);
    ++counter;          /* приращение */
 }
}
```

Пример 3.4
Найти сумму чисел между числами a и b (включительно).

```
#include<stdio.h>
int main()
```

```
{
    int a, b, sum = 0;
    scanf_s("%d%d", &a, &b);
    while (a <= b)
    {
       sum += a;
       a++;
    }
    printf("sum = %d", sum);
    return 0;
}
```

Пример 3.5

Написать программу, которая выводит таблицу квадратов первых n целых положительных четных чисел. Количество чисел должно вводиться во время работы программы. После вывода результата на экран снова должен появляться запрос количества чисел n до тех пор, пока пользователь не введет 0 (когда введен 0, осуществляется выход из программы).

```
#include<stdio.h>
#include<locale.h>
void main()
{
   setlocale(0, "rus");
   int n, i;
   printf("Введите n ");
   scanf_s("%d",&n);
   while(n != 0)
   {
      for(i = 2; i <= n; i += 2)
        printf("%d%d\n", i, i * i);
      printf("Введите n ");
      scanf_s("%d",&n);
    }
}
```

Пример 3.6

Программа выводит на экран меню, в котором пользователь может выбрать одно из следующих действий, введя соответствующее число: 1) подсчитать сумму двух чисел; 2) подсчитать разность двух чисел; 3) выйти из программы. Отличие данного примера от примера 3.5 заключается в том, что программа будет выводить запрос выбора не один раз (как в примере 2.5 лабораторной работы № 2), а до тех пор, пока пользователь не выберет выход из программы.

```
#include<stdio.h>
#include<locale.h>
void main()
```

```
{
   setlocale(0, "rus");
   int n, a, b;
   while(1) // бесконечный цикл, условие цикла всегда верно
   {
    printf("Меню:\n");
    printf("1 – подсчет суммы двух чисел\n");
    printf("2 – подсчет разности двух чисел\n");
    printf("3 – выход из программы\n");
    printf("Ваш выбор? \n");
    scanf_s("%d", &n);
    switch(n)
     {
      case 1:
            printf("Введите два числа\n");
            scanf_s("%d%d", &a, &b);
            printf("Сумма равна %d\n", a + b);
            break;
      case 2:
            printf("Введите два числа\n");
            scanf_s("%d%d", &a, &b);
            printf("Разность равна %d\n", a – b);
            break;
      case 3: return 0;
      default: printf("Неверное значение");
    }
   }
}
```

Пример 3.7
Вывод на экран алфавита.

```
#include<stdio.h>
#include<locale.h>
void main()
{
  setlocale(0, "rus");
  char c;
  printf("Алфавит\n");
  c = 'А';
  while (c <= 'Я')
    {
      printf("%c", c);
      c++;
    }
}
```

Цикл `do` … `while` (цикл с постусловием). Синтаксис:

```
do
{
// операторы цикла (тело цикла)
}
while (УсловиеПовторения);
```

Сначала выполняются операторы цикла (тело цикла), затем проверяется значение выражения `УсловиеПовторения`, и если условие истинно (не равно нулю), то операторы цикла выполняются еще раз до тех пор, пока `УсловиеПовторения` не станет ложным (равным нулю).

Число повторений операторов цикла `do … while` определяется ходом выполнения программы. Операторы цикла `do … while` выполняются до тех пор, пока значение выражения, записанного после слова `while`, не станет ложным (равным нулю). Для завершения цикла `do … while` в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла. Цикл `do … while` – это цикл с постусловием, т. е. операторы цикла будут выполнены хотя бы один раз. Цикл `do … while`, как правило, используется для организации приближенных вычислений в задачах поиска и обработки данных, вводимых с клавиатуры или считываемых из файла.

Пример 3.8
Вывести числа от 1 до 10.

```
#include<stdio.h>
void main()
{
  int counter = 1;
  do
  {
    printf("%d", counter);
  }
  while(++counter <= 10);
}
```

Пример 3.9
Ввести число от 0 до 10. Если пользователь ввел число больше 10 или меньше 0, запрашивать повторный ввод до того момента, пока пользователь не введет верное число.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(0, "rus");
```

```
  int num;
  do
{
    printf("Введите число от 0 до 10: ");
    scanf_s("%d", &num);
  }
  while((num < 0) || (num > 10));
  printf("Вы ввели число %d", num);
  return 0;
}
```

В языке Си существуют операторы `break` и `continue`, которые используются для управления ходом выполнения циклического процесса. Оператор `break` досрочно прерывает цикл. Оператор `continue` начинает следующий проход цикла, минуя оставшееся тело цикла.

Использование оператора `break`

```
#include<stdio.h>
 int main()
{
int i;
for (i = 0; i < 10; i++)
{
if (i == 5)
break;
printf("%d", i);
}
return 0;
}
```

Результат:

```
0 1 2 3 4
```

Использование оператора `continue`

```
#include<stdio.h>
 int main()
{
int i;
for (i = 0; i < 10; i++)
{
if (i == 5)
continue;
printf("%d", i);
}
return 0;
}
```

Результат:

```
0 1 2 3 4 6 7 8 9
```

Индивидуальные задания

1. Число a возводят в квадрат и результат увеличивают на 1. Полученное число снова возводят в квадрат и увеличивают на 1. Этот процесс продолжается до тех пор, пока не будет получено число x > 1 000 000. Найти число x.

2. Найти минимальную цифру, входящую во введенное число.

3. Составить алгоритм, определяющий количество способов, какими задуманное число n > 1 можно представить в виде суммы $n = i^3 + j^3$, считая, что перестановка слагаемых нового способа не дает.

4. Найти сумму S и произведение P:

а) четных чисел от 1 до n;

б) нечетных чисел от 1 до n;

в) чисел, кратных 3, от 1 до n.

5. Сколько слагаемых должно быть в сумме 1 + 1 / 2 + 1 / 3 + ... + 1 / n, чтобы эта сумма оказалась больше 3?

6. Найти сумму цифр введенного числа.

7. 10 000 рублей положены в сберегательный банк под 3 % годовых (процент капитализированный). Написать программу, определяющую, через какой промежуток времени первоначальная сумма увеличится в два раза.

8. В 1626 году аборигены продали остров за 20 долларов. Если бы эти деньги были помещены в банк под 4 % годовых (процент капитализированный), то какова была бы стоимость капитала сегодня?

9. Сумма R рублей положена в банк под 4 % годовых (процент капитализированный). Составить программу, определяющую, через какой промежуток времени сумма достигнет M рублей (M > R).

10. Население города ежегодно увеличивается на 1 / n жителей, где n – натуральное число. Написать программу, определяющую, через сколько лет население города утроится.

11. Можно ли разменять m рублей на рублевые, трехрублевые, пятирублевые купюры так, чтобы получить всего десять купюр (10 < m < 50)?

12. Найти максимальную цифру, входящую во введенное число.

13. Написать программу поиска четырехзначного числа, начинающегося с единицы, такого, что если переставить эту цифру в конец числа, то получится число в три раза больше искомого.

14. Искомое число больше 400 и меньше 500. Написать программу поиска этого числа, если сумма его цифр равна 9 и оно равняется 47 / 36 числа, записанного цифрами, но в обратном порядке.

15. Имеются контейнеры двух видов: по 130 кг и 160 кг. Можно ли полностью загрузить ими грузовик грузоподъемностью 3 т?

16. Сумма цифр двузначного числа равна 11. Если к этому числу прибавить 27, то получится число, записанное теми же цифрами, но в обратном порядке. Написать программу поиска этого числа, если оно существует.

17. Сумма квадратов цифр некоторого двузначного числа на 1 больше утроенного произведения этих цифр. После деления этого двузначного числа на сумму его цифр в частном получается 7 и в остатке 6. Написать программу поиска этого числа, если оно существует.

18. Дано действительное число a (1 < a < 3). Написать программу, находящую среди чисел 1, 1 + 1 / 2, 1 + 1 / 2 + 1 / 3, ... первое, которое больше a.

19. Известно, что любую целочисленную денежную сумму S > 7 рублей можно выплатить без сдачи купюрами достоинством в 3 и 5 рублей. По заданному S > 7 найти все пары целых неотрицательных чисел a и b, таких, чтобы S = 3a + 5b .

20. Даны два целых числа A и B (A < B). Найти все целые числа, расположенные между данными числами (не включая сами эти числа), в порядке их убывания, а также количество N этих чисел.

21. Дано натуральное число n. Получить все простые делители этого числа.

22. Вывести на экран 12-е простое число.

23. Найти все четырехзначные числа, у которых все цифры различны.

24. Дано натуральное число. Найти, сколько раз в нем встречается каждая цифра.

25. Даны числа от 1 до 1000 и число m. Вывести результат целочисленного деления нечетных сотен на число m.

26. Вывести на экран первые двенадцать простых чисел.

27. Вывести на экран все простые числа от 1 до n.

28. Билет называют «счастливым», если в его номере сумма первых трех цифр равна сумме последних трех. Подсчитать число тех «счастливых» билетов, у которых сумма трех цифр равна 13. Номер билета может иметь формат от 000000 до 999999.

29. Вывести на экран 1001 простое число.

30. Перевести число из десятичной системы счисления в двоичную.

31. Дано действительное число X и целое число N. Вывести N первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член X, а знаменатель 2.

32. Дано действительное число A и целое число N. Вывести все целые степени числа A от 1 до N.

33. Ежегодный прирост количества ящиков на складе составляет 5 %. Запасы на складе в начале 2015 года оценены в A тонн. Ежегодный план продажи – B тонн. Продажу следует прекратить, если останется не более 2 % от исходного запаса на складе. Составить программу, подсчитывающую, на сколько лет продаж может рассчитывать склад.

34. Найти последние три числа, кратные четырем, в диапазоне от 1 до N. Вычислить сумму этих чисел.

35. Вводится последовательность целых чисел, где 0 – конец последовательности. Определить, содержит ли последовательность хотя бы три отрицательных четных числа.

36. Вводится последовательность из N вещественных чисел. Определить среднее арифметическое среди элементов последовательности, кратных четырем.

37. При каждом делении(без остатка) числа N на 2 сделать так, чтобы выводился символ *. Цикл окончен, как только N не может быть разделено на 2 так, чтобы получилось целочисленное значение.

38. Вычислить произведение последних трех чисел не кратных семи в диапазоне от N1 до N2.

39. Найти сумму тех цифр в десятичном числе K, которые делятся на 5 без остатка.

40. Дана последовательность целых чисел, за которой следует 0. Найти разность минимального и максимального элементов в этой последовательности.

41. Написать код при помощи цикла, по результатам которого будет написано слово, а затем написано это же слово наоборот.

42. Дано число N. Определить, является ли оно числом Фибоначчи.

43. Начальный вклад в сберкассу составил A рублей. Каждые полгода вклад увеличивается на N %. Сколько лет должен находиться вклад в банке, чтобы достичь желаемой величины в K рублей?

44. Составить программу, проверяющую, является ли последовательность целых чисел, вводимых с клавиатуры, возрастающей. Введенный 0 означает конец ввода.

45. Определить, является ли число K простым.

46. Дано действительное число – цена 1 кг конфет. Вывести стоимость 1, 2, …, 15 кг конфет.

47. Даны действительные числа x, y. Вывести в порядке убывания все целые числа, расположенные между x и y, а также количество этих чисел.

48. Дана непустая последовательность натуральных чисел, за которой следует 0. Вычислить сумму тех из них, порядковые номера которых – числа Фибоначчи.

49. Написать цикл для выведения каждой третьей буквы алфавита на экран.

50. Найти значение выражения (для натуральных m и n, m < n):

а) $S = 1 + 2 + ... + n$;

б) $F = 1 \cdot 2 \cdot ... \cdot n$;

в) $A = m + (m + 1) + ... + (m + n)$;

г) $B = m \cdot (m + 1) \cdot ... \cdot (m + n)$;

д) $Y = 1 + 1 / 2 + 1 / 3 + ... + 1 / n$;

е) $X = 1 / m + 1 / (m + 1) + ... + 1 / (m + n)$;

ж) $S = 1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + 1 \cdot 2 \cdot 3 \cdot 4 + ... + 1 \cdot 2 \cdot 3 \cdot ... \cdot n$.

Лабораторная работа № 4
Одномерные массивы

Цель работы – познакомиться с созданием и обработкой одномерных массивов данных. Научиться использовать одномерные массивы в алгоритмах вычисления выражений, сортировки, поиска.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Массив – это совокупность однотипных элементов, имеющих общее имя и расположенных в памяти рядом. Чтобы использовать массивы в программировании, потребуется предварительное описание определенного типа и указание доступа к элементам. Если требуется обратиться к определенному элементу массива, то достаточно указать имя и индекс.

Массив характеризуется следующим:

– элементы массива имеют одинаковый тип, поэтому каждый элемент занимает одинаковый объем памяти;

– массив располагается в оперативной памяти, а не на внешнем устройстве, как файлы;

– элементы массива занимают подряд идущие ячейки.

Объявление массива должно содержать три параметра: тип каждого элемента, название массива, число элементов в массиве.

Общая форма для объявления массива имеет вид

```
<тип массива><имя массива>[<размер массива>],
```

т. е. массив объявляется так же, как и обычные переменные, но после имени следуют квадратные скобки, в которых указывается размер массива.

Размер – это константа (или константное выражение), которая определяет количество ячеек оперативной памяти, зарезервированной для массива.

Массивы могут иметь одну или несколько размерностей. Одномерный массив имеет одну размерность, двумерный массив (матрица) – две размерности (количество строк и столбцов). Три и более размерностей на практике используются редко, т. к. такие массивы занимают большой объем оперативной памяти.

Примеры объявления массива:

```
int a[20];     // целочисленный массив из 20 элементов
float b[5][10]; /* матрица вещественных чисел из 5 строк и 10 столбцов */
double c[10][10]; // квадратная матрица вещественных чисел
```

Доступ к элементам массива в языке Си осуществляется двумя способами.

Первый способ – с помощью порядкового номера элемента массива, который называется индексом. В качестве индекса можно использовать выражение целого или совместимого с ним типа, в том числе константу или переменную. В качестве индекса нельзя использовать выражение вещественного типа. Нумерация элементов массива начинается с 0. Индекс последнего элемента массива на единицу меньше его размерности.

Второй способ – обработка массивов с использованием указателей (адресов), т. к. в языке Си существует связь между массивами и указателями.

Примеры доступа к элементам массива с использованием индекса:

```
    a[3] = 125; /* присваивание значения 125 третьему элементу
массива */
    tmp = a[3]; /* присваивание переменной tmp значения третьего
элемента массива */
    a[0] = a[3]; /* присваивание нулевому элементу значения тре-
тьего элемента */
```

Способы инициализации элементов массивов. Возможен ввод элементов массива с экрана или из заранее подготовленного файла.

Пример 4.1

Ввод с клавиатуры одномерного массива.

```
printf("Введите элементы массива\n");
for (i = 0; i < n; i++) // n – количество элементов в массиве
{
  printf("a[%d]= ", i + 1);
  scanf_s("%d", &a[i]);
}
```

Обычно размер массива заранее неизвестен, поэтому при объявлении массива задается максимальная размерность, которая, как правило, известна. Реальная размерность N вводится и используется далее, например, в циклах и для других целей.

Значения элементов массива можно задать (проинициализировать) во время объявления следующим образом:

```
тип имя [N]={список значений};
```

В фигурных скобках записываются константы соответствующего типа, разделенные запятыми.

Пример инициализации одномерного массива:

```
const N = 5;
float a[N] = {-1.1, 22, 3, -4.4, 50};
```

При этом если в списке меньше N значений, то недостающие элементы массива примут нулевое значение и наоборот, если указать больше N значений, возникнет ошибка компиляции.

Пример 4.2

Программа запрашивает с клавиатуры десять чисел, а затем выводит их на экран в обратном порядке.

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
int main()
{
   setlocale(0, "rus");
   int array[10], i;
   printf("Введите 10 чисел\n");
   for (i = 0; i < 10; i++)
   {
      printf("Число %d: ", i + 1);
      scanf_s("%d", &array[i]);
   }
   for (i = 9; i >= 0; i--)
      printf("%d", array[i]);
   printf("\n");
   _getch();
   return 0;
}
```

Простой вывод элементов небольшого массива в одну строку экрана можно выполнить следующим образом.

Пример 4.3

Вывод одномерного массива.

```
printf("Массив: ");
for (i = 0; i < n; i++)
printf("%d", a[i]);
```

Подсчет суммы элементов одномерного массива. Алгоритм нахождения суммы элементов одномерного массива следующий:

– первоначально сумма принимается равной нулю, т. е. переменная, которая будет хранить сумму элементов, первоначально инициализируется нулем;

– далее начинается перебор элементов массива и к переменной, хранящей сумму элементов, добавляется каждый элемент массива.

Пример 4.4

```
int sum = 0;
for (i = 0; i < n; i++) // n – количество элементов в массиве
sum += a[i];    // аналогичная запись sum = sum + a[i]
printf("Сумма элементов массива = %d", sum);
```

Пример 4.5

Программа запрашивает с клавиатуры десять чисел, а затем выводит их сумму.

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
int main()
{
   setlocale(0, "rus");
   int array[10], i, sum = 0;
   printf("Введите 10 чисел\n");
   for (i = 0; i < 10; i++)
{
     printf("Число %d: ", i + 1);
     scanf_s("%d", &array[i]);
   }
   for (i = 0; i < 10; i++) // n – количество элементов в массиве
     sum += array[i];
   printf("Сумма 10 чисел = %d", sum);
   _getch();
   return 0;
}
```

Нахождение минимального (максимального) элемента одномерного массива. Алгоритм, нахождения минимального (максимального) элемента следующий:

– первоначально переменной, которая будет хранить минимальное (максимальное) значение, присваивается значение элемента с индексом 0;

– далее начинается перебор элементов массива и текущее минимальное (максимальное) значение сравнивается с текущим значением элемента. Если текущее значение элемента массива меньше (больше) текущего минимального (максимального), то минимальное (максимальное) значение становится равно текущему значению элемента массива.

Пример 4.6

Нахождение максимального элемента одномерного массива.

```
int max = a[0];
for (i = 1; i < n; i++) // n – количество элементов в массиве
if (max < a[i])
    max = a[i];
printf("Максимальный элемент массива = %d", max);
```

Пример 4.7

Пример нахождения минимального элемента одномерного массива.

```
int min = a[0];
for (i = 1; i < n; i++) // n – количество элементов в массиве
if (min > a[i])
    min = a[i];
printf("Минимальный элемент массива = %d", min);
```

Пример 4.8

Программа запрашивает с клавиатуры десять чисел, а затем находит среди них минимальный и максимальный элементы.

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
int main()
{
    setlocale(0, "rus");
    int array[10], i, sum = 0, min, max;
    printf("Введите 10 чисел\n");
    for (i = 0; i < 10; i++)
    {
        printf("Число %d:", i+1);
        scanf_s("%d", &array[i]);
    }
    min = array[0];
    for (i = 1; i < 10; i++)
         if (min > array[i])
             min = array[i];
     printf("Минимальный элемент массива = %d\n", min);
     max = array[0];
     for (i = 1; i < 10; i++)
          if (max < array[i])
              max = array[i];
     printf("Максимальный элемент массива = %d\n", max);
     _getch();
     return 0;
}
```

Пример 4.9

Заполнить одномерный массив десять случайными числами.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define SIZE 10
int main()
{
    int a[SIZE];
```

```
    srand(time(NULL));
    for (int i = 0; i < SIZE; i++)
    {
      a[i] = -10 + rand() % 20;
      printf("%d", a[i]);
    }
    return 0;
}
```

Индивидуальные задания

1. Ввести массив вещественных чисел размером n, n (вводится с клавиатуры). Найти его наибольший и наименьший элементы и поменять их местами. Найти сумму и произведение всех элементов массива.

2. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму отрицательных элементов массива;

– произведение элементов массива, расположенных между максимальным и минимальным элементами.

3. В одномерном массиве, состоящем из n целых элементов, вычислить:

– произведение элементов массива с четными номерами;

– сумму элементов массива, расположенных между первым и последним нулевыми элементами.

4. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– максимальный элемент массива;

– сумму элементов массива, расположенных до последнего положительного элемента.

5. В одномерном массиве, состоящем из n целых элементов, вычислить:

– номер максимального элемента массива;

– произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

6. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– максимальный по модулю элемент массива;

– сумму элементов массива, расположенных между первым и вторым положительными элементами.

7. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер максимального по модулю элемента массива;

– сумму элементов массива, расположенных после первого положительного элемента.

8. В одномерном массиве, состоящем из вещественных элементов, вычислить:

– количество элементов массива, больших, чем C;

– произведение элементов массива, расположенных после максимального по модулю элемента.

9. В одномерном массиве, состоящем из k целых элементов, вычислить:

– количество положительных элементов массива;

– сумму элементов массива, расположенных после последнего элемента, равного нулю.

10. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение отрицательных элементов массива;

– сумму положительных элементов массива, расположенных до максимального элемента.

11. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– минимальный элемент массива;

– сумму элементов массива, расположенных между первым и последним положительными элементами.

12. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, лежащих в диапазоне от A до B;

– сумму элементов массива, расположенных после максимального элемента.

13. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение положительных элементов массива;

– сумму элементов массива, расположенных до минимального элемента.

14. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму элементов массива с нечетными номерами;

– сумму элементов массива, расположенных между первым и последним отрицательными элементами.

15. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер минимального элемента массива;

– сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

16. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, равных 0;

– сумму элементов массива, расположенных после минимального элемента.

17. Определить в одномерном числовом массиве A, состоящем из n элементов, число соседств из двух положительных чисел.

18. Определить в одномерном числовом массиве A из n элементов число соседств из двух чисел одного знака.

19. Определить, имеется ли в одномерном числовом массиве A, состоящем из n элементов, хотя бы одна пара совпадающих по величине соседних элементов.

20. Определить в одномерном числовом массиве A, состоящем из n элементов, число соседств взаимно обратных чисел.

21. Проверить, имеется ли в одномерном числовом массиве A, состоящем из n элементов, хотя бы одна пара взаимно обратных чисел.

22. Даны массивы A и B, состоящие из n элементов. Построить массив S, каждый элемент которого равен сумме соответствующих элементов массивов A и B.

23. Задан одномерный числовой массив A, состоящий из n элементов, и число k. Найти номера всех элементов массива, которые больше, меньше и равны k.

24. Даны массивы A из n элементов и B из m элементов. Содержится ли наибольший элемент массива A в массиве B?

25. Задан массив A из n элементов. Подсчитать, сколько раз в этом массиве встречается максимальное по величине число.

26. Дан массив x, содержащий n элементов. Найти количество различных чисел среди элементов этого массива.

27. Даны два массива x и y. Найти количество одинаковых элементов в этих массивах, т. е. количество пар x[i] = y[j] для некоторых i и j.

28. Даны два массива: x, содержащий k элементов, и y, содержащий n элементов и число q. Найти сумму вида x[i] + y[j], наиболее близкую к числу q.

29. Дан массив A размером n, не содержащий нулевых элементов. Необходимо получить массив A, в котором вначале идут положительные элементы, а затем отрицательные. Дополнительные массивы не использовать.

30. Элементы заданного массива X циклически сдвинуть на k позиций вправо (влево).

31. Заданы два массива по 10 целых чисел в каждом. Найти наименьшее среди чисел первого массива, которое не входит во второй массив (считая, что хотя бы одно такое число есть).

32. В массиве из 25 элементов найти номер первого отрицательного элемента и первого положительного элемента. Значение элементов и их порядковый номер вывести на экран. Заменить найденный отрицательный элемент в массиве на произведение отрицательного и положительного, а положительный – на сумму.

33. Сформировать массив. Вывести его элементы через один в обратном порядке. В случае если есть отрицательный элемент, вывести его номер на экран.

34. В массиве из 20 целых чисел найти наибольший элемент и поменять его местами с первым элементом. В случае если есть отрицательный элемент, вывести его номер на экран.

35. В массиве R, содержащем 25 элементов, заменить значения отрицательных элементов квадратами значений, значения положительных увеличить на 7, а нулевые значения оставить без изменения. Вывести массив R.

36. Дан массив G целых чисел, содержащий 20 элементов. Вычислить и вывести количество и сумму тех элементов, которые делятся на 2 и не делятся на 5.

37. Дан массив H целых чисел, содержащий 30 элементов. Вычислить и вывести R = S + P, где S – сумма четных элементов < 3, P – произведение нечетных элементов > 1.

38. Дан массив Y, содержащий 26 элементов. Вычислить сумму элементов, стоящих до первого отрицательного элемента. Вывести исходный массив и результат вычислений.

39. Дан массив J целых чисел, содержащий 14 элементов. Все отрицательные элементы заменить на 1. Вывести исходный и полученный массивы.

40. Дан массив L целых чисел, содержащий 22 элемента. Все положительные элементы заменить на –5. Вывести полученный массив и номера измененных элементов.

41. В массиве из 20 вещественных чисел найти наибольший элемент, заменить его значение на –3 и поменять его местами с последним элементом. Вывести исходный и полученный массивы.

42. В массиве из 10 целых чисел найти наибольший элемент и поменять его местами с первым элементом. Вывести полученный массив в обратном порядке.

43.Дан массив A целых чисел, содержащий 25 элементов. Вычислить и вывести сумму тех элементов, которые нечетны и отрицательны.

44. Дан массив K, содержащий 14 элементов. Вычислить и вывести среднее арифметическое всех значений ki < 5. Вывести исходный массив.

45. В массиве A из 20 элементов все отрицательные элементы отправить в конец массива. Вывести полученный массив в обратном порядке. Вывести исходный массив.

46. В массиве A из 12 элементов удалить (преобразовать значение в 0) все отрицательные элементы, стоящие перед минимальным положительным элементом массива. Вывести исходный и полученный массивы.

47. Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые при делении на 7 дают остаток 1, 2 или 5.

48. Дан массив, содержащий 12 элементов. Все нечетные элементы сложить, вывести результат и массив в обратном порядке.

49. Дан массив T из 8 элементов, найти его первый отрицательный элемент и записать его в начало. Определить разность первых трех элементов полученного массива. Вывести полученный массив и значение разности.

50. С клавиатуры задать массив из 10 элементов. Поменять местами элементы с четными и нечетными номерами, после чего вывести исходный и полученный массивы на экран.

51. С клавиатуры задать массив из 8 элементов. Определить сумму отрицательных элементов массива и разность положительных. Вывести результат на экран.

52. С клавиатуры задать массив из 7 элементов. Поменять местами первые три элемента с последними тремя, после чего разделить сумму первых трех на сумму последних трех элементов. Вывести результат на экран.

53. Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые обладают следующим свойством: корни уравнения $q_i{}^2 + 3 \cdot q_i{}^2 - 5$ действительны и положительны.

54. Дан массив X, содержащий 20 элементов. Все положительные элементы возвести в квадрат, а отрицательные умножить на 5. Вывести исходный и полученный массивы.

55. С клавиатуры задать массив из 10 элементов. Умножить все элементы на 2. Вывести только положительные элементы полученного массива.

56. С клавиатуры задать массив из 8 элементов. Поменять местами первые два элемента с последними двумя, после чего умножить четные элементы массива на –2. Вывести исходный и полученный массивы на экран.

57. С клавиатуры задать массив из 12 элементов. К положительным элементам прибавить значение первого отрицательного элемента массива, умноженного на –1. Вывести на экран полученный массив и номер первого отрицательного числа.

58. С клавиатуры задать массив из 10 отрицательных элементов. Ко всем элементам меньше, чем –3, прибавить значение последнего элемента, умноженного на –2. Вывести исходный и полученный массивы на экран.

59. С клавиатуры задать массив из 8 элементов, где первые три числа положительных. Поменять местами первый элемент массива и первый отрицательный элемент массива. Вывести на экран номера отрицательных чисел массива.

60. С клавиатуры задать массив из 15 элементов. Положительным элементы возвести в квадрат. Вывести номера элементов массива, чье значение больше 17.

61. С клавиатуры задать массив из 10 элементов. Ко всем элементам массива прибавить значение первого положительного числа массива. Вывести на экран полученный массив и номер последнего положительного числа.

62. С клавиатуры задать массив из 16 элементов. Все отрицательные элементы массива умножить на номер первого положительного числа. Вывести на экран сумму отрицательных чисел массива и исходный массив.

Лабораторная работа № 5
Многомерные массивы

Цель работы – познакомиться с созданием и обработкой многомерных массивов данных. Научиться использовать многомерные массивы в алгоритмах вычисления выражений, сортировки, поиска.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Многомерные массивы. Массивы в Си могут иметь несколько индексов. Многомерные массивы часто применяются для представления таблиц, состоящих из значений, упорядоченных по строкам и столбцам. Двумерные массивы – это массивы массивов, т. е. массивы, элементами которых являются указатели на массивы. Двумерными массивами задаются матрицы.

Многомерные массивы объявляются точно так же, как и одномерные, только после имени массива ставится более одной пары квадратных скобок:

```
<тип><имя>[размерность1][размерность2]...[размерностьN];
```

Примеры объявления многомерных массивов:

```
double array[30][10]; // матрица с 30 строками и 10 столбцами
int buffer[1][7]; // матрица с 1 строкой и 7 столбцами
```

В первых квадратных скобках задается число строк, во вторых квадратных скобках – число столбцов.

Особенностью является то, что по своему строению многомерный массив является обыкновенным «одномерным» массивом, в котором все элементы расположены друг за другом.

Таким образом, в двумерном массиве элементы расположены «по рядам», в трехмерном – «по слоям», внутри которых элементы расположены «по рядам», и т. д.

Начальную инициализацию многомерного массива можно провести так:

```
int a[2][3] = {1, 2, 3, 4, 5, 6};
```

При этом элементы в массиве расположатся следующим образом:

```
a[0][0] = 1   a[0][1] = 2   a[0][2] = 3
a[1][0] = 4   a[1][1] = 5   a[1][2] = 6
```

Важно не забывать, что индексирование строк и столбцов начинается с 0.

С помощью фигурных скобок можно выполнить инициализацию следующим образом:

```
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

Если при объявлении массива заявлено элементов больше, чем указано при инициализации, то оставшиеся элементы заполняются нулями. Например, в матрице

```
int matr3[3][3] = {{1}, {2, 7}, {9, 0, 1}};
```

будут определены следующие элементы:

```
matr3[0][0] = 1;    matr3[0][1] = 0;    matr3[0][2] = 0;
matr3[1][0] = 2;    matr3[1][1] = 7;    matr3[1][2] = 0;
matr3[2][0] = 9;    matr3[2][1] = 0;    matr3[2][2] = 1;
```

Рассмотрим примеры использования многомерных массивов.

Во всех примерах выполняется ввод с клавиатуры, вывод на экран и другие действия с матрицей, состоящей из n строк и m столбцов. Особых отличий от работы с одномерными массивами у многомерных массивов нет. При обращении к элементам многомерного массива необходимо указать все его индексы.

Пример 5.1

Ввести с клавиатуры элементы двумерного массива с плавающей точкой.

```
int i, j;
float b[100][100];
for (i = 0; i < 100; i++)
  for (j = 0; j < 100; j++)
  {
    printf("Введите элемент [%d, %d]\n", i + 1, j + 1);
    scanf_s("%f", &b[i][j]);
  }
```

Пример 5.2

Вывод элементов матрицы в табличном виде.

```
int i, j;
float b[100][100];
printf("Матрица:\n");
for (i = 0; i < 100; i++)
{
  for (j = 0; j < 100; j++)
    printf("%.2f", b[i][j]);
  printf("\n");
}
```

Пример 5.3
Найти сумму элементов матрицы.

```
int i,j;
float b[100][100];
float s = 0;
for (i = 0; i < 100; i++)
  for (j = 0; j < 100; j++)
    s += b[i][j]
printf("Сумма элементов матрицы = %.2f\n", s);
```

Пример 5.4
Найти значение максимального элемента матрицы.

```
int i,j;
float b[100][100];
float max = b[0][0];
for (i = 0; i < 100; i++)
  for (j = 0; j < 100; j++)
    if (b[i][j] > max)
      max = b[i][j];
printf("Максимальный эелмент матрицы = %.2f\n", max);
```

Пример 5.5
Найти сумму элементов каждой строки.

```
int i, j;
float s;
float b[100][100];
for (i = 0; i < 100; i++)
{
  s = 0;
  for (j = 0; j < 100; j++)
    s += b[i][j];
  printf("Сумма элементов %d-строки матрицы = %.2f\n", i + 1, s);
}
```

Пример 5.6
Найти произведение элементов каждого столбца.

```
int i, j;
float p;
float b[100][100];
for (j = 0; j < 100; j++)
{
  p = 1;
  for (i = 0; I < 100; i++)
    p *= b[i][j];
```

```
    printf("Произведение элементов %d-го столбца матрицы = %.2f\n",
j + 1, p);
  }
```

Пример 5.7

Найти количество положительных элементов на главной диагонали.

```
int i, j, kol = 0;
float b[100][100];
for (i = 0; i < 100; i++)
{
  if (b[i][i] > 0)
    kol++;
}
printf("Количество положительных элементов на главной диагонали =
%.d\n", kol);
```

Пример 5.8

Дана матрица вещественных чисел n × n. Количество строк и столбцов n задается пользователем. Подсчитать сумму элементов матрицы, найти минимальный и максимальный элементы матрицы, найти сумму каждой строки матрицы, произведение каждого столбца матрицы и количество положительных элементов на главной диагонали.

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
void main ()
{
setlocale(LC_ALL, "rus");
  int i, j, kol = 0, n;
  float b[30][30], s = 0, p, max, min;
  do
  {
    printf("Введите количество строк и стобцов (<30)\n");
    scanf_s("%d", &n);
  }
while (n >= 30);
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
      printf("Введите элемент [%d, %d]\n", i + 1, j + 1);
      scanf_s("%f", &b[i][j]);
    }
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      s += b[i][j];
  printf("Сумма элементов матрицы = %.2f\n", s);
  max = b[0][0];
```

```
        for (i = 0; i < n; i++)
          for (j = 0; j < n; j++)
            if (b[i][j] > max)
              max = b[i][j];
        printf("Максимальный элемент = %.2f\n", max);
        min = b[0][0];
        for (i = 0; i < n; i++)
          for (j = 0; j < n; j++)
            if (b[i][j] < min)
              min = b[i][j];
        printf("Минимальный элемент = %.2f\n", min);
        for (i = 0; i < n; i++)
        {
          s = 0;
          for (j = 0; j < n; j++)
            s += b[i][j];
        printf("Сумма элементов %d-й строки матрицы = %.2f\n",
i + 1, s);
        }
        for (j = 0; j < n; j++)
        {
          p = 1;
          for (i = 0; i < n; i++)
            p *= b[i][j];
        printf("Произведение элементов %d-го столбца матрицы = %.2f\n",
j + 1,p);
        }
        for (i = 0; i < n; i++)
        {
          if (b[i][i] > 0)
            kol++;
          }
        printf("Количество положительных элементов на главной
диагонали = %d\n", kol);
        printf("МАТРИЦА:\n");
        for (i = 0; i < n; i++)
        {
          for (j = 0; j < n; j++)
            printf("%.2f", b[i][j]);
          printf("\n");
        }
      }
```

Индивидуальные задания

1. Дана матрица вещественных чисел N × M. Количество строк N и столбцов M задается пользователем. Найти максимальный и минимальный элемент в каждой строке и поменять их местами. Найти сумму элементов на главной диагонали.

2. Двумерный массив, содержащий равное число строк и столбцов, называется магическим квадратом, если суммы чисел, записанных в каждой строке, каждом столбце и каждой из двух бóльших диагоналей, равны одному и тому же числу. Определить, является ли данный массив A из n строк и n столбцов магическим квадратом.

3. Элемент матрицы называется 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 целой матрицы размером 10 × 12 напечатать индексы всех ее седловых точек.

4. Дана матрица размером 6 × 6. Найти сумму наименьших элементов ее нечетных строк и наибольших элементов ее четных строк.

5. Дана целочисленная прямоугольная матрица. Определить количество строк, не содержащих ни одного нулевого элемента, и максимальное из чисел, встречающихся в заданной матрице более одного раза.

6. Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента.

7. Характеристикой строки целочисленной матрицы называется сумма ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

8. Дана целочисленная прямоугольная матрица. Определить количество столбцов, содержащих хотя бы один нулевой элемент, и номер строки, в которой находится самая длинная серия одинаковых элементов.

9. Дана целочисленная квадратная матрица. Определить произведение элементов в тех строках, которые не содержат отрицательных элементов, и максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

10. Для заданной матрицы размером 8 × 8 найти такие k, что k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

11. Характеристикой столбца целочисленной матрицы называется сумма модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

12. Характеристикой строки целочисленной матрицы называется сумма ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

13. Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке. Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

14. Дана целочисленная прямоугольная матрица. Определить количество строк, содержащих хотя бы один нулевой элемент, и номер столбца, в котором находится самая длинная серия одинаковых элементов.

15. Дана целочисленная квадратная матрица. Определить сумму элементов в тех строках, которые не содержат отрицательных элементов, и минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

16. Дана целочисленная квадратная матрица. Определить сумму элементов в тех столбцах, которые не содержат отрицательных элементов и минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

17. Найти в матрице первую строку, все элементы которой положительны, и сумму этих элементов. Уменьшить все элементы матрицы на эту сумму.

18. Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.

19. Найти в матрице первую строку, все элементы которой упорядочены по возрастанию. Изменить порядок элементов этой строки на обратный.

20. Найти в матрице первую строку, все элементы которой упорядочены по убыванию. Изменить порядок элементов этой строки на обратный.

21. Проверить, есть ли в матрице хотя бы одна строка, содержащая положительный элемент, и найти ее номер. Знаки элементов предыдущей строки изменить на противоположные.

22. Проверить, есть ли в матрице хотя бы одна строка, содержащая отрицательный элемент, и найти ее номер. Все элементы столбца с таким же номером уменьшить вдвое.

23. Проверить, есть ли в матрице хотя бы одна строка, содержащая элемент, равный нулю, и найти ее номер. Уменьшить все элементы матрицы на значение первого элемента найденной строки.

24. Найти в матрице первую строку, все элементы которой равны нулю. Все элементы столбца с таким же номером уменьшить вдвое.

25. Проверить, все ли строки матрицы упорядочены по убыванию. Если нет, найти первую неупорядоченную строку и упорядочить.

26. Проверить, все ли строки матрицы содержат хотя бы один положительный элемент. Если да, то изменить знаки всех элементов матрицы на обратные.

27. Проверить, все ли строки матрицы содержат хотя бы один отрицательный элемент. Если да, то изменить знаки всех элементов матрицы на обратные.

28. Проверить, все ли строки матрицы содержат хотя бы один нулевой элемент. Если нет, то заменить значения всех отрицательных элементов матрицы на нулевые.

29. Найти в матрице первый столбец, все элементы которого положительны. Знаки элементов предыдущего столбца изменить на противоположные.

30. Найти в матрице первый столбец, все элементы которого отрицательны, и среднее арифметическое этих элементов. Вычесть полученное значение из всех элементов матрицы.

31. Найти в матрице первый столбец, все элементы которого равны нулю. Знаки элементов строки с таким же номером изменить на противоположные.

32. Задана матрица C размером 5 × 5. Поменять местами максимальный элемент каждой строки с первым элементом соответствующей строки.

33. Задать с клавиатуры матрицу Q размером 7 × 7. Если на главной диагонали стоит нуль, то соответствующую строку заменить единицами.

34. Дана целочисленная матрица A размером N × T. Найти наибольшие элементы в каждом столбце этой матрице и количество нечетных чисел среди них.

35. В матрице размером 6 × 8, заданной с клавиатуры пользователем, определить элементы (их позицию), которые являются одновременно минимальными в строке и столбце. Вывести координаты данных значений.

36. Задана матрица Y размером 8 × 8. Поменять местами минимальный элемент каждой строки с элементом диагонали соответствующей строки. Вывести полученную матрицу на экран.

37. Сформировать с клавиатуры массив P размером D × R. Увеличить каждый элемент массива в три раза и поменять знак на противоположный. Массив вывести на экран в виде таблицы.

38. Дана матрица A размером 5 × 5. Найти и вывести в обратном порядке тот столбец этой матрицы, который содержит наибольшее количество отрицательных чисел.

39. Задана матрица D размером 4 × 4. Если максимальный элемент матрицы стоит на главной диагонали, то все элементы главной диагонали сделать равными максимальному.

40. Даны оценки, полученные на шести экзаменах во время сессии студентами одной группы (студентов в группе 10), по 10-балльной системе. Определить, сколько студентов сдали сессию на балл не ниже 6.

41. Дана матрица U размером 6 × 6. Сформировать одномерный массив из отрицательных элементов этой матрицы, расположенных ниже главной диагонали. Вывести исходный и одномерный массивы.

42. Дана матрица Y размером 6 × 7. Записать вместо отрицательных элементов матрицы нули, а вместо положительных – единицы. Вывести на экран матрицу в виде таблицы.

43. Дан двумерный массив размером 15 × 15, заполненный случайными числами. Подсчитать количество строк, содержащих только четные числа.

44. Дана матрица H размером P × F. Упорядочить ее столбцы по возрастанию их наименьших элементов. Вывести полученную матрицу на экран.

45. Задана матрица A размером 4 × 4. Если максимальный элемент матрицы равен сумме элементов первой строки, то поменять местами первую строку с той строкой, где находится максимальный элемент.

46. Дана целочисленная матрица размером 10 × 12. Найти для каждой строки число элементов, кратных пяти, и наибольший из полученных результатов. Вывести результаты на экран.

47. Задать матрицу размером 5 × 5 с клавиатуры. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали. Вывести исходную и полученную матрицу в табличном виде.

48. Ввести с клавиатуры матрицу P размером N × N. Сформировать одномерный массив из четных элементов этой матрицы, расположенных ниже побочной диагонали. Вывести исходный и полученный массивы.

49. Дана матрица A размером N × N. Найти и вывести ту строку в этой матрице, которая содержит наибольшее количество четных чисел.

50. Матрицу размером 8 × 10 заполнить случайными числами от 0 до 15 (можно с клавиатуры). Вывести на экран саму матрицу и номера строк, в которых число 5 встречается два и более раз.

51. Дан двумерный массив размером 4 × 4, заполненный случайными числами. Вычислить сумму всех элементов массива. Вывести не только значение суммы, но и номера положительных элементов массива

52.Сформировать двумерный массив G размером 8 × 8 по следующему правилу: Gij = 8 – (i – j) + j. Транспонировать массив (поменять местами строки и столбцы) и вывести элементы главной диагонали, разместив их в строке на экране.

53. Переписать первые элементы (> 10) каждой строки матрицы D размером 6 × 6, заданной с клавиатуры, в одномерный массив B. Вывести на экран массив D и массив B.

54. Есть группа спортсменов из 8 человек. Для каждого спортсмена приводится его рост и вес. Вес спортсмена считается нормальным, если от роста отнять 80 и полученное число отличается от веса не более чем на 3. Вывести номера тех спортсменов, чей вес превышает норму.

55. Задана матрица F размером 9 × 3. Определить, равны ли все элементы первого столбца соответствующим элементам главной диагонали. Если нет, то поменять их местами. Вывести полученную матрицу на экран.

56. Дан массив V размером 10 × 14. Вычислить и сохранить сумму и число положительных элементов каждого столбца матрицы. Результаты отпечатать в виде двух строк.

57. Для матрицы размером T × R, заданной с клавиатуры, вывести на экран все седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или наоборот.

58. Дан двумерный массив Q размером 10 × 10, заполненный случайными числами. Для каждой строки массива найти минимальный элемент. Найти сумму всех минимальных элементов строк. Вывести значения минимальных элементов и их суммы на экран.

59. Дан двумерный массив W размером 6 × 6, заполненный случайными числами. Для каждой строки массива найти максимальный элемент. Найти мак-

симальные элементы диагонали. Найти сумму всех максимальных элементов. Вывести значения суммы на экран

60. Задать с клавиатуры массив размером 5 × 5. Найти в каждой строке матрицы максимальный и минимальный элементы и поместить их на место первого и последнего элемента строки соответственно. Вывести полученную матрицу на экран.

Лабораторная работа № 6
Указатели. Динамические массивы

Цель работы – познакомиться с указателями и операциями над ними. Рассмотреть связь между указателями и массивами. Изучить способы работы с динамическими массивами. Научиться использовать указатели и динамические массивы в алгоритмах работы с матрицами.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Указатели. Указатель – особый вид переменной, которая хранит адрес элемента памяти, в котором может быть записано значение другой переменной.

Определение указателя – `имя_типа *имя_переменной;`, например:

```
int var, *pvar;
```

Здесь `var` – целочисленная переменная, `pvar` – указатель на целый тип. Символ `*` (звездочка) определяет тип «указатель».

Существует операция, неразрывно связанная с указателями. Это унарная операция – операция взятия адреса `&`. Результатом операции `&` является адрес переменной в памяти. Результат имеет тип «указатель на тип переменной». Операция `&` может использоваться практически со всеми типами данных, кроме констант и битовых полей. Например:

```
int var, *pvar;
pvar = &var;
```

Указатели часто используются для обмена данными с функциями. Функции можно передать столько аргументов, сколько требуется, но вернуть с помощью `return` можно только одно значение. Когда возникает необходимость вернуть в вызывающую функцию более одного значения, то используются указатели.

Операция доступа по указателю (разадресация) `*point`. Результатом операции будет являться содержимое ячейки памяти, на которую указывает `point`.

Пример 6.1
Операции взятия адреса и доступа по указателю.

```
#include<stdio.h>
#include<locale.h>
void main(void)
{
  setlocale(LC_ALL, "rus");
  int *p, a = 18;
  p = &a;
  printf("a = %d = %d\n", a, *p);
  printf("адрес a = %d = %d\n", &a, p);
  *p += 8;   // a = 26
  printf("a = %d = %d\n", a, *p);
}
```

Результат выполнения программы:

```
a = 18 = 18
адрес a = 2424536 = 2424536
a = 26 = 26
```

Операции над указателями. С содержимым указателя можно производить операции, в результате чего он будет указывать на другую переменную или на другой элемент массива. Рассмотрим эти операции.

1. Операция присваивания. Указателю можно присвоить либо адрес объекта того же типа, либо значение другого указателя или константу `NULL`. Если указатели разных типов, необходимо применить операцию приведения. Когда указателю присваивается другой указатель, то первый фактически начинает указывать на тот же адрес, на который указывает второй указатель. Если мы не хотим, чтобы указатель указывал на какой-то конкретный адрес, то можно присвоить ему условное нулевое значение с помощью константы `NULL`, которая определена в заголовочном файле `stdio.h`:

```
int a = 10, *p, *t;
p = &a;
t = p
p = NULL;
```

2. Операция увеличения (уменьшения) указателя. Выполняется с помощью команды `point + i; point – i`, где `i` – значение целочисленного типа.

Результат операции `point + i` – указатель, определяющий адрес `i`-го элемента после данного, а `point – i` – перед данным.

3. Операции сложного присваивания

```
point += i; point -= i;
```

4. Операции инкремента (декремента)

```
point++; point--; ++point; --point;
```

Указатель будет смещаться (увеличиваться или уменьшаться в зависимости от операции) на один элемент. Фактически указатель (адрес) изменится на количество байт, занимаемых этим элементом в памяти.

Для операций увеличения (уменьшения) указателя справедливо следующее правило. Если объявлен `type1 *pointer`, где `type1` – некоторый тип, то операция `pointer = pointer + i`, где `i` – значение целочисленного типа, изменит `pointer` на `sizeof(type1)*i` байт.

Например:

```
int a, *pi;
double f, *pf;
pi = &a;
pf = &f;
pi++;   // смещение на 4 байта
pf++;   // смещение на 8 байт
```

5. Операция индексирования `point[i]`

```
int data, *point;
int mas[10] = {5, 4, 6, 7, 9};
point = mas;
i = 0;
data = point[i];
```

Эта операция полностью аналогична выражению `*(point + i)`, т. е. извлекается из памяти и используется в выражении значение `i`-го элемента массива, адрес которого присвоен указателю `point`.

6. Операция вычитания `point1 - point2`. Операция имеет смысл только в том случае, если `point1` и `point2` являются указателями на один и тот же набор данных. Результат имеет тип `int` и равен количеству элементов, которые можно расположить между ячейками памяти, на которые указывают `point1` и `point2`.

7. Операции отношения

```
point1 == point1;
point1 >= point2; point1 > point2;
```

```
point1 <= point2; point1 < point2;
point1 != point1;
```

Любой указатель (адрес) может быть проверен на равенство (==) или неравенство (!=) со специальным значением NULL.

Связь между указателями и массивами. Имя массива – это указатель, равный адресу начального элемента массива (нулевого байта нулевого элемента массива).

При объявлении `int m[10];` одновременно с выделением памяти для десяти элементов типа `int` определяется значение указателя `m`. Значение `m` равно адресу элемента `m[0]`. Значение `m` изменить нельзя, т. к. оно является константой. Индексные и адресные выражения эквивалентны:

```
m[i] ⇔ *(m + i).
```

Для двумерных массивов имя является указателем-константой на массив указателей-констант. Элементами массива указателей являются указатели-константы на начало каждой из строк массива.

Пример 6.2

Используя указатели, ввести одномерный массив вещественных чисел из n элементов, подсчитать сумму элементов, найти минимальный и максимальный элементы.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  int i, kol = 0, n;
  float b[30], s = 0, max, min;
  do
  {
    printf("Введите количество элементов массива (<30)\n");
    scanf_s("%d", &n);
        }
  while (n >= 30);
  for (i = 0; i < n; i++)
  {
    printf("Введите элемент [%d]\n", i + 1);
    scanf_s("%f", b + i);
  }
  for (i = 0; i < n; i++)
    s += *(b + i);
  printf("Сумма элементов = %.2f\n", s);
  max = *b;
  for (i = 0; i < n; i++)
    if (*(b + i) > max) max = *(b + i);
```

```
  printf("Максимальный элемент = %.2f\n", max);
  min = *b;
  for (i = 0; i < n; i++)
    if (*(b + i) < min) min = *(b + i);
  printf("Минимальный элемент = %.2f\n", min);
}
```

Динамическое выделение памяти. Динамическое выделение памяти – это способ запроса памяти из операционной системы по мере необходимости. Память выделяется не из ограниченной памяти стека программы, а из гораздо более крупного хранилища, управляемого операционной системой – «кучи», или «хипа» (от англ. heap – куча). На современных компьютерах размер «кучи» может составлять гигабайты памяти.

Так как память выделяется по мере необходимости и освобождается, когда ее использование завершилось, то можно использовать ту же самую память в другой момент времени для других целей в другой части программы. Динамическое выделение памяти дает возможность создания динамических структур данных: списков, деревьев и др.

Динамическое выделение памяти выполняется библиотечными функциями языка Си, объявленными в заголовочном файле `stdlib.h`, представленными в табл. 6.1.

Таблица 6.1

Функции стандартной библиотеки `stdlib.h` для работы с динамической памятью

Имя функции	Команда	Описание
calloc	`void *calloc(unsigned n, unsigned m);`	Выделяет память для `n` элементов по `m` байт каждый и возвращает указатель на начало выделенной памяти. В случае неудачного выполнения возвращает `NULL`
coreleft	`unsigned coreleft(void);`	Возвращает значение объема неиспользованной памяти. Используется для моделей памяти `tiny`, `small`, `medium`. Уникальна для DOS. Функция несовместима с Windows

Окончание табл. 6.1

Имя функции	Команда	Описание
malloc	`void *malloc(unsigned s);`	Выделяет память длиной в `s` байт и возвращает указатель на начало выделенной памяти. В случае неудачного выполнения возвращает `NULL`
realloc	`void *realloc(void *bl, unsigned ns);`	Изменяет размер ранее выделенной динамической памяти с адресом `bl` до размера `ns` байт. Если `bl` равен `NULL`, то функция выполняется как `malloc()`
free	`void free (void *bl);`	Освобождает ранее выделенный блок динамически распределенной памяти с адресом первого байта `bl`

Рассмотрим каждую функцию в отдельности.
Функция `malloc()`:

```
void *malloc(unsigned size)
```

Функция `malloc` выделяет из «кучи» область памяти размером `size` байт. В случае успеха, `malloc` возвращает указатель на начало выделенного блока памяти. Если для выделения блока в «куче» не хватает памяти, возвращается `NULL`. Содержимое блока памяти оставляется неизменным. Если размер аргумента равен нулю, `malloc` возвращает `NULL`.

Пример 6.3

Программа выделяет динамическую память под массив из n элементов, запрашивает массив с клавиатуры и выводит его на экран.

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<stdlib.h>
int main()
{
  setlocale(LC_ALL, "rus");
  int *ptr, i, n;
  printf("Введите количество элементов массива\n");
  scanf_s("%d",&n);
  if(!(ptr = (int*)malloc(n * sizeof(int))))
  {
```

```
      puts("Not enough memory");
      _getch();
      return 0;
   }
  // ptr указывает на массив из n элементов
   for (i = 0; i < n; i++)
   {
     printf("Введите элемент [%d]\n", i + 1);
      scanf_s("%d",ptr + i);
   }
printf("\nМассив:\n");
for (i = 0; i <n; i++)
  printf("%d ", *(ptr + i));
  return 0;
}
```

Функция `calloc()`

```
void *calloc(unsigned num, unsigned size)
```

Функция `calloc` выделяет блок памяти и возвращает указатель на первый байт блока. Размер выделяемой памяти равен величине `num * size`, т. е. функция выделяет память, необходимую для хранения массива из `num` элементов по `size` байт каждый. В случае нехватки памяти для удовлетворения запроса `calloc` возвращает `NULL`. Выделяемая память инициализируется нулями.

Пример 6.4

```
int main()
{
int *ptr;
if (!(ptr = (int*)calloc(5, sizeof(int))))
// ptr указывает на массив из 5 элементов, заполненный нулями
  {
  puts("Not enough memory");
  _getch(); return 0;
  }
}
```

Функция `realloc()`

```
void *realloc(void *ptr, unsigned size)
```

Эта функция изменяет размер динамически выделяемой области памяти, на которую указывает `*ptr`, на `size` (новый размер). Если указатель не является значением, которое ранее было определено функциями `malloc`, `calloc` или `realloc`, то поведение функции не определено.

Это же справедливо, если ptr указывает на область памяти, ранее освобожденную функцией free. Значение size является абсолютным, а не относительным, т. е. задает новый размер блока, а не приращивает старый. Если size больше, чем размер ранее существовавшего блока, то новое неинициализированное пространство памяти будет выделено в конце блока, а предыдущее содержимое пространства сохранится. Если realloc не может выделить память требуемого размера, то возвращаемое значение равно NULL и содержимое пространства, на которое указывает ptr, остается нетронутым. Если ptr – не NULL, а значение size равно нулю, то функция realloc действует как free.

Из вышесказанного следует сделать вывод о том, что когда бы размер блока памяти ни подвергся изменению функцией realloc, новое пространство все равно может начинаться с адреса, отличного от исходного, даже если realloc «усекает» память. Следовательно, если используется realloc, возникает необходимость следить за указателями на изменяемое пространство. Например, если вы создаете связный список и выделяете при помощи realloc больший или меньший участки памяти для цепочки, то цепочка может оказаться перемещена. В этом случае указатели элементов будут обращаться к участкам памяти, ранее занимаемым звеньями цепочки, а не в место их реального расположения.

Функцию realloc всегда следует использовать так, как показано ниже:

```
if(p2 == realloc(p1, new_size)) p1 = p2;
```

Делая запись подобным образом, вам никогда не придется заботиться, выделялось ли для объекта новое пространство, т. к. p1 обновляется при каждом новом вызове функции.

Пример 6.5

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int *ptr, *tmp;
if(!(ptr = (int*)calloc(5, sizeof(int))))
// ptr указывает на массив из 5 элементов
  {
   puts("Not enough memory");
   return 0;
  }
if (tmp = (int*)realloc(ptr, 10*sizeof(int))) ptr = tmp;
// ptr указывает на массив из 10 элементов
else
  {
```

```
    puts("Not enough memory");
    return 0;
   }
}
```

Функция `free()`

```
void free(void *ptr)
```

Функция освобождает область памяти, ранее выделенную при помощи функций `malloc`, `calloc` или `realloc`, на которую указывает `ptr`. Если `ptr` = NULL, то `free` ничего не выполняет. Если `ptr` не является указателем, проинициализированным ранее одной из функций выделения памяти, то поведение функции не определено. Заметим, что функция `free` не располагает средствами индикации ошибки, потенциально возникающей при ее выполнении, равно как и возвращаемым значением.

Пример 6.6

```
int main()
{
   int *ptr;
    if (!(ptr = (int*)calloc(5, sizeof(int))))
   // ptr указывает на массив из 5 элементов
      {
       puts("Not enough memory");
       _getch();
       return 0;
      }
    free (ptr);
   return 0;
}
```

Пример 6.7

Программа сортирует массив из 10 элементов по возрастанию методом пузырька.

```
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
 int main()
{
  setlocale(LC_ALL, "rus");
  int *b, i, j, n, buf;
  printf("Введите количество элементов\n");
  scanf_s("%d", &n);
  b = (int*)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
```

```
  {
    printf("Введите элемент [%d]\n", i + 1);
    scanf_s("%d", b + i);
  }
  printf("\nИсходный массив:\n", i + 1);
  for (i = 0; i < n; i++)
    printf("%d", *(b + i));
     // сортировка массива методом пузырька
  for (i = 0; i < n; i++)
    for (j = 0; j < n - i - 1; j++)
      if (*(b + j) > *(b + j + 1))
        {
          buf = *(b + j);
          *(b + j) = *(b + j + 1);
          *(b + j + 1) = buf;
 }
  printf("\nОтсортированный массив:\n", i + 1);
  for (i = 0; i < n; i++)
    printf("%d", *(b + i));
  free(b);
return 0;
}
```

Пример 6.8

Дана матрица вещественных чисел n × m. Количество строк и столбцов n и m задается пользователем. Подсчитать сумму элементов матрицы, найти минимальный и максимальный элементы матрицы, найти сумму каждой строки матрицы, произведение каждого столбца матрицы.

```
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
 int main()
{
   setlocale(LC_ALL, "rus");
   int i, j, kol = 0, n, m;
   float **b, s = 0, p, max, min;
   printf("Введите количество строк\n");
   scanf_s("%d", &n);
   printf("Введите количество столбцов\n");
   scanf_s("%d", &m);
   b = (float **)calloc(n, sizeof(float *));
   for (i = 0; i < n; i++)
     *(b + i) = (float *)calloc(m, sizeof(float));
   for (i = 0; i < n; i++)
     for (j = 0; j < m; j++)
     {
       printf("Введите элемент [%d, %d]\n", i + 1, j + 1);
       scanf_s("%f", &b[i][j]);
     }
   for (i = 0; i < n; i++)
```

```
          for (j = 0; j < m; j++)
            s += b[i][j];
        printf("Сумма элементов матрицы = %.2f\n", s);
         max = b[0][0];
        for (i = 0; i < n; i++)
          for (j = 0; j < m; j++)
            if (b[i][j] > max)
              max = b[i][j];
        printf("Max элемент матрицы = %.2f\n", max);
        min = b[0][0];
        for (i = 0; i < n; i++)
          for (j = 0; j < m; j++)
            if (b[i][j] < min)
              min = b[i][j];
        printf("Min элемент матрицы = %.2f\n", min);
        for (i = 0; i < n; i++)
        {
          s = 0;
          for (j = 0; j < m; j++)
            s += b[i][j];
          printf("Сумма элементов %d-строки матрицы = %.2f\n",
i + 1, s);
        }
        for (j = 0; j < n; j++)
        {
          p = 1;
          for (i = 0; i < m; i++)
            p *= b[i][j];
          printf("Произведение элементов %d-столбца матрицы =
%.2f\n", j + 1, p);
        }
        printf("МАТРИЦА:\n");
        for (i = 0; i < n; i++)
        {
          for (j = 0; j < m; j++)
            printf("%.2f", b[i][j]);
          printf("\n");
        }
        for (i = 0; i < n; i++)
          free(b[i]);
        free(b);
     return 0;
     }
```

Индивидуальные задания

Написать программу в соответствии с вариантом. Выделить динамическую память под массив. Использовать указатель при обращении к массивам.

1. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, равных нулю;

– сумму элементов массива, расположенных после минимального элемента.

2. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение элементов массива с нечетными номерами;

– разность двух максимальных элементов массива.

3. В одномерном массиве, состоящем из n целых элементов, вычислить:

– количество отрицательных элементов массива;

– сумму элементов массива, кратных трем.

4. В одномерном массиве, состоящем из n целых элементов, вычислить:

– произведение положительных элементов массива;

– количество элементов массива, кратных пяти.

5. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение элементов массива с четными номерами;

– разность двух минимальных нечетных элементов массива.

6. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, не равных нулю;

– сумму элементов массива, расположенных после максимального элемента.

7. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– максимальный элемент массива;

– сумму элементов массива, расположенных до последнего положительного элемента.

8. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– минимальный элемент массива;

– сумму элементов массива, расположенных после первого отрицательного элемента.

9. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– максимальный по модулю элемент массива;

– сумму элементов массива, расположенных между первым и вторым положительными элементами.

10. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– минимальный по модулю элемент массива;

– сумму элементов массива, расположенных между первым и последним отрицательными элементами.

11. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер максимального по модулю элемента массива;

– сумму элементов массива, расположенных после первого положительного элемента.

12. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер минимального по модулю элемента массива;

– сумму элементов массива, расположенных перед первым отрицательным элементом.

13. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, больших, чем C;

– произведение элементов массива, расположенных после максимального по модулю элемента.

14. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива меньших, чем C;

– произведение элементов массива, расположенных перед максимальным по модулю элементом.

15. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение отрицательных элементов массива;

– сумму положительных элементов массива, расположенных до максимального элемента.

16. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение положительных элементов массива;

– сумму отрицательных элементов массива, расположенных после минимального элемента.

17. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– минимальный четный элемент массива;

– сумму элементов массива, расположенных между первым и последним положительными элементами.

18. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– максимальный нечетный элемент массива;

– сумму элементов массива, расположенных между первым и последним положительными элементами.

19. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– количество элементов массива, лежащих в диапазоне от A до B;

– сумму четных элементов массива.

20. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму элементов массива, лежащих вне диапазона от A до B;

– количество нечетных элементов массива.

21. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму элементов массива с нечетными номерами;

– сумму элементов массива, расположенных между первым и последним отрицательными элементами.

22. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму элементов массива с четными номерами;

– произведение элементов массива, расположенных между первым и последним положительными элементами.

23. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер минимального элемента массива;

– сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

24. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– номер максимального элемента массива;

– произведение элементов массива, расположенных между первым и вторым положительными элементами.

25. В одномерном массиве, состоящем из n целых элементов, вычислить:

– количество положительных элементов массива;

– сумму элементов массива, расположенных после последнего элемента, равного нулю.

26. В одномерном массиве, состоящем из n целых элементов, вычислить:

– сумму отрицательных элементов массива;

– количество элементов массива, расположенных после последнего четного элемента.

27. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– сумму отрицательных элементов массива;

– произведение элементов массива, расположенных между максимальным и минимальным элементами.

28. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение четных положительных элементов массива;

– произведение элементов массива, кратным трем и расположенных между максимальным и минимальным элементами.

29. В одномерном массиве, состоящем из n целых элементов, вычислить:

– произведение элементов массива с четными номерами;

– сумму элементов массива, расположенных между первым и последним нулевыми элементами.

30. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

– произведение положительных элементов массива;

– сумму элементов массива, расположенных до минимального элемента, кратного трем.

31. Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

32. Дана целочисленная прямоугольная матрица. Определить количество строк, не содержащих ни одного нулевого элемента.

33. Дана матрица размером 6 × 6. Найти сумму наименьших элементов ее нечетных строк и наибольших элементов ее четных строк.

34. Характеристикой строки целочисленной матрицы называется сумма ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

35. Дана целочисленная прямоугольная матрица. Определить количество столбцов, содержащих хотя бы один нулевой элемент.

36. Дана целочисленная прямоугольная матрица. Определить номер строки, в которой находится самая длинная серия одинаковых элементов.

37. Дана целочисленная квадратная матрица. Определить произведение элементов в тех строках, которые не содержат отрицательных элементов.

38. Дана целочисленная квадратная матрица. Определить максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

39. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

40. Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента.

41. Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке. Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

42. Дана целочисленная квадратная матрица. Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

43. Для заданной матрицы размером 8 × 8 найти такие k, чтобы k-я строка матрицы совпадала с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

44. Дана целочисленная прямоугольная матрица. Определить количество строк, содержащих хотя бы один нулевой элемент.

45. Дана целочисленная прямоугольная матрица. Определить номер столбца, в котором находится самая длинная серия одинаковых элементов.

46. Дана целочисленная квадратная матрица. Определить сумму элементов в тех строках, которые не содержат отрицательных элементов.

47. Дана целочисленная квадратная матрица. Определить минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

48. Двумерный массив, содержащий равное число строк и столбцов, называется магическим квадратом, если суммы чисел, записанных в каждой строке, каждом столбце и каждой из двух больших диагоналей, равны одному и тому же числу. Определить, является ли массив A из n строк и n столбцов магическим квадратом.

49. Элемент матрицы называется 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 целой матрицы размером 10 × 12 напечатать индексы всех ее седловых точек.

50. Дана целочисленная прямоугольная матрица. Определить максимальное из чисел, встречающихся в заданной матрице более одного раза.

51. Дана матрица размером 3 × 3. Найти произведение элементов с нечетными индексами.

52. Дана целочисленная прямоугольная матрица. Определить номер строки, в которой сумма ее элементов наибольшая.

53. Дана целочисленная прямоугольная матрица. Определить номер строки, в которой находится наибольшее количество отрицательных элементов.

54. Дана целочисленная прямоугольная матрица. Определить номера строк, в которой произведение ее отрицательных элементов также отрицательно.

55. Дана целочисленная квадратная матрица. Определить сумму элементов в тех строках, которые не содержат нулевых элементов.

56. Для заданной матрицы размером 6 × 6 найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

57. Для заданной матрицы размером 5 × 5 найти и поменять местами строки, содержащие максимальную и минимальную суммы их элементов.

58. Дана матрица размером 9 × 10. Расставить столбцы таким образом, чтобы элементы в первой строке были упорядочены по возрастанию.

59. Дана целочисленная матрица размером 5 × 5. Определить, сколько в ней пар соседних одинаковых элементов. Элементы считаются соседними, если их индексы в столбцах и/или в строках различаются не более чем на единицу.

60. Дана целочисленная прямоугольная матрица. Последний отрицательный элемент каждого столбца прямоугольной матрицы заменить нулем.

Лабораторная работа № 7
Функции. Работа со строками

Цель работы – изучить работу с функциями в языке Си. Рассмотреть способы определения, объявления и вызова функций. Научиться использовать функции работы со строками.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Функция – это самостоятельная единица программы, которая спроектирована для реализации конкретной подзадачи. Функция является подпрограммой, которая может содержаться в основной программе, а может быть создана отдельно (в библиотеке). Каждая функция выполняет в программе определенные действия.

Различают системные (в составе систем программирования) и собственные функции. Системные функции хранятся в стандартных библиотеках, и пользователю не нужно вдаваться в подробности их реализации, достаточно знать лишь их сигнатуру. Примером системных функций, использованных ранее, являются функции `printf()` и `scanf_s()`.

Собственные функции – это функции, написанные пользователем для решения конкретной подзадачи.

Определение функции. Каждая функция в языке Си должна быть определена, т. е. должны быть указаны:

– тип возвращаемого значения;

– имя функции;

– информация о формальных параметрах;

– тело функции.

Определение функции имеет синтаксис `<тип> <имя_функции> (<спецификация параметров>)<тело функции>`, где `<тип>` – либо `void` (для функций, не возвращающих значения), либо обозначение типа возвращаемого функцией значения; `<имя_функции>` – либо `main` для основной (главной) функции программы, либо произвольно выбираемое программистом имя (идентификатор); `<спецификация параметров>` – это либо пустое поле, либо список формальных параметров, каждый элемент которого имеет вид `<обозначение_типа> <имя_параметра>`; `<тело функции>` – это часть определения функции, представляющая программный блок, т. е. ограниченный фигурными скобками фрагмент текста на языке Си.

Для завершения работы функции используется оператор `return`.

Существует две формы этого оператора: `return;` (возврат без формирования возвращаемого значения) `return <выражение>;` (возврат с формированием возвращаемого значения).

Первая форма используется для функций, перед именем которых в определении указан тип `void`. Для функций этого типа оператор `return` в тексте программы можно не писать вовсе – компилятор встроит его автоматически перед закрывающей фигурной скобкой тела функции.

Во второй форме оператора `<выражение>` должно иметь тип, совпадающий с типом функции или допускающий автоматическое преобразование к типу возвращаемого функцией значения.

Функция может иметь несколько операторов `return`. Если функция не принимает параметров, в круглых скобках указывается `(void)`.

Имя функции – это любой правильно написанный идентификатор. Тип возвращаемого значения – это тип данных результата, возвращаемого из функции оператору ее вызова. Тип возвращаемого значения `void` указывает, что функция не возвращает никакого значения. Компилятор предполагает тип `int` для неопределенного типа возвращаемого значения.

Пропуск типа возвращаемого значения в описании функции вызывает синтаксическую ошибку, если прототип функции определяет возвращаемый тип иначе, чем `int`.

Невозвращение значения из функции, в которой предполагается возвращение результата, может привести к неожиданным ошибкам. Описание языка Си указывает, что результат такой оплошности не определенен. В этом случае обычно компиляторы Си выдают предупреждающее сообщение.

Возвращение какого-то значения из функции, для которой тип возвращаемого значения объявлен как `void`, вызывает синтаксическую ошибку.

Список параметров – это список разделенных запятыми объявлений тех параметров, которые получает функция при ее вызове. Если функция не получает никаких значений, список параметров задается как `void`. Тип должен быть указан для каждого параметра в списке. Объявление параметров `float x, y` вызовет ошибку компиляции.

Повторное определение параметра функции как локальной переменной этой функции является синтаксической ошибкой.

Не используйте одинаковые имена для аргументов, передаваемых в функцию, и соответствующих параметров в описании функции, хотя это и не является ошибкой. Использование разных имен помогает избежать путаницы.

Объявления и операторы внутри фигурных скобок образуют тело функции. Тело функции рассматривается как блок. Блок – это составной оператор, который включает объявления. Переменные могут быть объявлены в любом блоке, блоки могут быть вложенными. При любых обстоятельствах функция не может быть определена внутри другой функции. Определение функции внутри другой функции является синтаксической ошибкой.

Пример 7.1
Описание функции сложения двух вещественных чисел.

```
float function(float x, float z)
{
  float y;
  y = x + z;
  return(y);
}
```

В рассмотренном примере возвращаемое значение имеет тип `float`. В качестве возвращаемого значения в вызывающую функцию передается значение переменной `y`. Формальными параметрами являются значения переменных `x` и `z`.

Разбиение программ на функции дает следующие преимущества:

–функцию можно вызвать из различных мест программы, что позволяет избежать повторения программного кода;

–одну и ту же функцию можно использовать в разных программах;

–функции повышают уровень модульности программы и облегчают ее проектирование;

–использование функций облегчает чтение и понимание программы и ускоряет поиск и исправление ошибок.

С точки зрения вызывающей программы функцию можно представить как некий «черный ящик», у которого есть несколько входов и один выход. Неважно, каким образом производится обработка информации внутри функции. Для корректного использования функции достаточно знать лишь ее определение.

Вызов функции. Общий вид вызова функции: `<имя_функции> (<список фактических параметров>)`. Кроме того, функцию можно обозначить, разыменовав указатель на нее.

`<список фактических параметров>` – это список выражений, количество которых равно числу формальных параметров функции (исключение составляют функции с переменным количеством параметров). Соответствие между формальными и фактическими параметрами устанавливается по их взаимному расположению в списках.

Фактический параметр – это величина, которая присваивается формальному параметру при вызове функции. Таким образом, формальный параметр – это переменная в вызываемой функции, а фактический параметр – это конкретное значение, присвоенное этой переменной вызывающей функцией.

Фактический параметр может быть константой, переменной или выражением. Если фактический параметр представлен в виде выражения, то его значение сначала вычисляется, а затем передается в вызываемую функцию. Если в функцию требуется передать несколько значений, то они записываются через запятую. При этом формальные параметры заменяются значениями фактических параметров в порядке их следования в определении функции.

Возврат в вызывающую функцию. По окончании выполнения вызываемой функции осуществляется возврат значения в точку ее вызова. Это значение может быть присвоено переменной, тип которой соответствует типу возвращаемого значения функции. Функция может передать в вызывающую программу только одно значение. Для передачи возвращаемого значения в вызывающую функцию используется оператор `return` в одной из форм:

```
return(ВозвращаемоеЗначение);
return ВозвращаемоеЗначение;
```

Действие оператора следующее: значение выражения вычисляется и передается в вызывающую функцию. Возвращаемое значение может использоваться в вызывающей программе как часть некоторого выражения.

Оператор `return` также завершает выполнение функции и передает управление следующему оператору в вызывающей функции. Оператор `return` не обязательно должен находиться в конце тела функции.

Функции могут и не возвращать значения, а просто выполнять некоторые вычисления. В этом случае указывается пустой тип возвращаемого значения `void`, а оператор `return` может либо отсутствовать, либо не возвращать никакого значения.

Пример 7.2

Посчитать сумму двух чисел.

```
#include<stdio.h>
int sum(int x, int y) // определение функции
{
   int k = x + y;  // вычисляем сумму чисел и сохраняем в k
   return k;       // возвращаем значение k
}
int main()
{
   int a, r;      // описание двух целых переменных
   printf("a = ");
   scanf_s("%d", &a); // вводим a
   r = sum(a, 5);    // вызов функции: x = a, y = 5
   printf("%d + 5 = %d", a, r); // вывод: a + 5 = r
   return 0;
}
```

В языке Си нет требования, чтобы определение функции обязательно предшествовало ее вызову. Функции могут определяться как до вызывающей функции, так и после нее. Однако, если определение вызываемой функции описывается ниже ее вызова, необходимо до вызова функции определить прототип этой функции, содержащий:

– тип возвращаемого значения;

– имя функции;

– типы формальных аргументов в порядке их следования.

Прототип необходим для того, чтобы компилятор мог осуществить проверку соответствия типов передаваемых фактических аргументов типам формальных аргументов. Имена формальных аргументов в прототипе функции могут отсутствовать.

Если в примере 7.2 описание функции сложения чисел разместить после тела функции `main`, то код будет выглядеть следующим образом.

Пример 7.3

```
#include<stdio.h>
int sum(int, int);   // прототип
int main()
{
  int a,  r;
  printf("a = ");
  scanf("%d", &a);
  r = sum(a, 5);
  printf("%d + 5 = %d", a, r);
  return 0;
}
int sum(int x, int y) // определение
{
  int k;
  k = x + y;
  return k;
}
```

Рекурсивные функции. Функция, которая вызывает сама себя, называется рекурсивной функцией. Рекурсия – вызов функции из самой функции.

Пример 7.4

Функция вычисления факториала.

```
#include<stdio.h>
int fact(int num)
{
  if (num <= 1) return 1;
  else return num * fact(num - 1); // рекурсивный вызов
int main()
{
  int a, r;
  printf("a = ");
  scanf_s("%d", &a);
  r = fact(a);    // вызов функции: num = a
  printf("%d! = %d", a, r);
  return 0;
}
```

Пример 7.5

Вывести на экран квадраты натуральных чисел от 1 до 10.

```
#include<stdio.h>
int square(int);   // прототип функции square
int main()
{
  for (int x = 1; x <= 10; x++)
    printf("%d\n", square(x));
  return 0;
}
// определение функции square
int square(int y)
{
  return (y * y);
}
```

Строка `int square(int);` является прототипом функции. Тип данных `int` в круглых скобках указывает компилятору, что функция `square` ожидает в операторе вызова целое значение аргумента. Тип данных `int` слева от имени функции `square` указывает компилятору, что `square` возвращает оператору вызова целый результат. Компилятор обращается к прототипу функции для проверки того, что вызовы функции `square` содержат правильный возвращаемый тип, правильное количество аргументов, правильный тип аргументов и правильный порядок перечисления аргументов.

Пример 7.6

Определить максимальное из трех целых чисел. Определить функцию `maximum` для определения и возвращения наибольшего из трех целых чисел.

```
#include<stdio.h>
#include<locale.h>
int maximum(int, int, int); // прототип функции maximum
int main()
{
  setlocale(LC_ALL, "rus");
  int a, b, c;
  printf("Введите 3 числа: ");
  scanf_s("%d%d%d", &a, &b, &c);
  printf("Max =  %d", maximum(a, b, c));
  return 0;
}
// определение функции maximum
int maximum(int x, int y, int z)
{
  int max = x;
  if (y > max) max = y;
  if (z > max) max = z;
  return max;
}
```

Прототип функции, заголовок функции и вызовы функции должны быть согласованы между собой по количеству, типу и порядку следования аргументов и параметров, а также по типу возвращаемых результатов.

Отсутствие точки с запятой в конце прототипа функции является синтаксической ошибкой.

Вызов функции, которая не соответствует прототипу функции, ведет к синтаксической ошибке. Синтаксическая ошибка также возникает в случае отсутствия согласования между прототипом и определением функции. Например, если бы в примере 7.6 прототип функции был записан как

```
void maximum(int, int, int);
```

компилятор сообщил бы об ошибке, потому что возвращаемый тип `void` в прототипе функции отличался бы от возвращаемого типа `int` в определении функции.

Другой важной особенностью прототипов функций является автоматическое приведение типов аргументов, т. е. задание аргументам подходящего типа. Например, библиотечная математическая функция `sqrt` может быть вызвана с аргументом целого типа, даже если в `math.h` функция определяет аргумент типа `double`, функция будет работать правильно. К примеру, оператор

```
printf("Корень квадратный = %.2f", sqrt(4));
```

правильно вычислит `sqrt(4)` и напечатает значение `2.00`. Прототип функции заставляет компилятор преобразовать целое значение `4` в значение `4.0` типа `double` прежде, чем значение будет передано в `sqrt`.

Значения аргументов, которые первоначально не соответствуют типам параметров в прототипе функции, преобразуются в подходящий тип перед вызовом функции. Эти преобразования могут привести к неверным результатам, если не руководствоваться правилами приведения типов Си. Правила приведения определяют, как типы могут быть преобразованы в другие типы без потерь. В приведенном выше примере `sqrt` тип `int` автоматически преобразуется в `double` без изменения его значений. Однако `double` преобразуется в `int` с отбрасыванием дробной части значения `double`. Преобразование больших целых типов в малые целые типы (например, `long` в `short`) может также привести к изменению значений.

Отсутствие прототипа функции, когда функция не определена перед ее первым вызовом, приводит к синтаксической ошибке.

Прототип функции, размещенный вне описания другой функции, относится ко всем вызовам данной функции, появляющимся после этого прототипа в файле. Прототип функции, размещенный внутри описания функции, относится только к вызовам внутри этой функции.

Пример 7.7

Вывести на экран меню и предложить пользователю сделать выбор. В зависимости от выбора пользователя программа либо подсчитывает сумму двух чисел, либо подсчитывает длину введенной строки. В программе кроме главной функции создать еще три функции.

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<locale.h>
int menu(); // прототип функции menu
int sum(int aa, int bb); // прототип функции sum
void kol(char *s); // прототип функции kol
int main()
{
  setlocale(LC_ALL, "rus");
  int a, b, c = 0, k1 = 0;
  char s1[20];
  while(1)
  {
  // вызов функции menu
  switch(menu()) /* условное выражение в операторе switch – 
это значение, которое возвращает функция menu */
   {
    case 1: printf("Введите a и b: ");
            scanf_s("%d%d",&a, &b);
            c = sum(a, b);  // вызов функции sum
            printf("\nСумма = %d\n", c);
            break;
    case 2: fflush(stdin);
            printf("Введите строку: \n");
            getchar();
            gets_s(s1);
            kol(s1);  // вызов функции kol
            break;
    case 3: return 0; /* выполнение функции завершается после 
выполнения оператора return */
   }
  }
}

/* Функция menu выводит на экран три строки меню, и считывает 
введенное пользователем значение. Никакие параметры функции не пе-
редаются, функция возвращает введеное пользователем целочисленное 
значение */
int menu()  // определение функции menu
{
  int ch;
  do
  {
    printf("\nMenu:\n");
```

```
        printf("1. Сумма двух чисел\n");
        printf("2. Количество символов в строке\n");
        printf("3. Выход\n");
        printf("\tВаш выбор");
        scanf_s("%d", &ch);
      }
      while(ch > 3);
      return ch; /* функция возвращает значение переменной ch, т. е.
выбор пункта меню */
    }
    /* Функция sum подсчитвывает сумму двух чисел. Функции пере-
даются 2 целочисленных параметра – складываемые числа, функция
возвращает целочисленный параметр – сумму двух чисел */

    int sum(int aa, int bb) // определение функции sum
    {
      return aa + bb;
    }

    /* Функция kol подсчитывает и выводит на экран длину строки.
Функции передается введенная пользователем строка, функция ничего
не возвращает */
    void kol(char *s) // определение функции kol
    {
      printf("Длина = %d", strlen(s));
    /* оператор return отсутствует, т. к. функция не возвращает
значений */
    }
```

Передача массивов в функцию. Если в качестве передаваемого аргумента используется массив данных, то в функцию передается только указатель на массив (см. пример 7.7), т. е. адрес первого элемента. При вызове функции в списке аргументов записывается имя массива. Имя массива без индекса является адресом элемента с нулевым индексом.

Можно использовать три варианта описания массива, поступающего в функцию, в качестве параметра.

Вариант 1. Параметр в функции может быть объявлен как массив соответствующего типа с указанием его размера.

Пример 7.8

Элементы матрицы равны произведению номера столбца на номер строки. Подсчитать сумму элементов матрицы.

```
    #include<stdio.h>
    int sum(int x[5])  /* определение функции, размер массива
указан */
    // int sum(int x[]); // возможный вариант прототипа
    // int sum(int *);   // возможный вариант прототипа
    // int sum(int []);  // возможный вариант прототипа
```

```
{
  int res = 0;
  for(int i = 0; i < 5; i++)
      res += x[i];
  return res;
}
int main()
{
  int mas[3][5], i, j;
  for(i = 0; i < 3; i++)
    for(j = 0; j < 5; j++)
      mas[i][j] = i * j;
  for(i = 0; i < 3; i++)
    printf("\n res = %d\n", sum(mas[i]));
  for(i = 0; i < 3; i++)
    {for(j = 0; j < 5; j++)
       printf("%d ", mas[i][j]);
     printf("\n");
    }
}
```

В этом случае Си автоматически преобразует `mas[i]` к указателю на целый тип, и в функцию передается адрес первого элемента каждой строки матрицы, т. е. адрес элемента `mas[i][0]`. В функции вычисляется сумма элементов каждой строки матрицы, и сумма выводится на экран.

Вариант 2. Массив в качестве параметра в функции может быть объявлен без указания его размера. Так как сам массив в стек не копируется, то размер массива в общем случае компилятору не требуется. Контроль правильности использования индекса массива возлагается на программиста. В этом случае также используется указатель на первый элемент массива.

Пример 7.9

```
int sum(int x[]) // определение функции
{
  int res = 0;
  for(int i = 0; i < 5; i++)
     res += x[i];
  return res;
}
```

Вызов функции остается тем же.

Вариант 3. Наиболее распространенный способ, обычно используемый при написании профессиональных программ, – объявление параметра-массива указателем на соответствующий тип данных. Функцию можно вызывать так, как было проиллюстрировано выше.

Пример 7.10

```
int sum(int *x)  // определение функции
{
  int res = 0;
  for(int i = 0; i < 5; i++)
    res += x[i]
  return res;
}
```

Во всех случаях в функцию передается адрес нулевого элемента массива, т. е. указатель соответствующего типа.

Пример 7.11

Ввести матрицу n × m. Написать функцию, которая позволяет ввести матрицу и вывести ее на экран.

```
#include<stdio.h>
#include<locale.h>
int x[50][50];
void input(int x[50][50], int n1, int m1); /* Прототип функ-
ции input */
int output(int x[50][50], int n1, int m1); /* Прототип функ-
ции output */
int main()
{
  setlocale(LC_ALL, "rus");
  int n, m;
  printf("Введите количество строк n\n");
  scanf_s("%d", &n);
  printf("Введите количество столбцов m\n");
  scanf_s("%d", &m);
  input(x, n, m);
  output(x, n, m);
  return 0;
}
void input(int x[50][50], int n1, int m1)
{
  int z, j;
  for (z = 0; z < n1; z++)
    for (j = 0; j < m1; j++)
    {
      printf("Введите элемент [%d, %d]\n", z + 1, j + 1);
      scanf_s("%d", &x[z][j]);
    }
}
int output(int x[50][50], int n1, int m1)
{
  int z, j;
```

```
  printf("\n");
  for (z = 0; z < n1; z++)
{
  for (j = 0; j < m1; j++)
      printf("%d", x[z][j]);
    printf("\n");
  }
  return 0;
}
```

Работа со строками в языке Си. Строка – это последовательность ASCII- или UNICODE-символов. Строки в Си, как и в большинстве языков программирования высокого уровня, рассматриваются как отдельный тип, входящий в систему базовых типов языка. Так как язык Си по своему происхождению является языком системного программирования, то строковый тип данных в нем как таковой отсутствует, а в качестве строк в Си используются обычные массивы символов.

Исторически сложилось два формата строк:

–формат ANSI;

–строки с завершающим нулем (используется в Си).

Формат ANSI устанавливает, что значением первой позиции в строке является ее длина, а затем следуют сами символы строки. Например, представление строки «Моя строка!» будет следующим:

```
11 'М' 'о' 'я' ' ' 'с' 'т' 'р' 'о' 'к' 'а' '!'
```

В строках с завершающим нулем значащие символы строки указываются с первой позиции, а признаком завершения строки является значение нуль. Представление строки в этом формате имеет вид

```
'М' 'о' 'я' ' ' 'с' 'т' 'р' 'о' 'к' 'а' '!' 0
```

Объявление строк в Си. Объявление строки в Си имеет тот же синтаксис, что и объявление одномерного символьного массива. Длина строки должна представлять собой целочисленное значение.

Длина строки указывается с учетом одного символа на хранение завершающего нуля, поэтому максимальное количество значащих символов в строке на единицу меньше ее длины.

Например, строка может содержать максимально двадцать символов, если объявлена следующим образом:

```
char str[21];
```

Инициализация строки в Си осуществляется при ее объявлении с использованием следующего синтаксиса:

```
char str[длина] = строковый литерал;
```

Строковый литерал – строка ASCII-символов, заключенных в двойные кавычки. Примеры объявления строк с инициализацией:

```
char str1[20] = "Введите значение: ";
char str2[20] = "";
```

Работа с символами в строке. Так как строки на языке Си являются массивами символов, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс нуль. Например, в следующем ниже фрагменте программы в строке `str` осуществляется замена всех символов `'a'` на символы `'A'` и наоборот:

```
for(int i = 0; str[i] != 0; i++)
{
  if(str[i] == 'a') str[i] = 'A';
  else
  if(str[i] == 'A') str[i] = 'a';
}
```

Массивы строк. Объявление массивов строк в языке Си также возможно. Для этого используются двумерные массивы символов, имеющие следующий синтаксис:

```
char имя[количество][длина];
```

Первым размером матрицы указывается количество строк в массиве, а вторым – максимальная (с учетом завершающего нуля) длина каждой строки. Например, объявление массива из пяти строк максимальной длиной 30 значащих символов будет иметь вид

```
char strs[5][31];
```

При объявлении массивов строк можно производить инициализацию:

```
char имя[количество][длина] =
{строковый литерал № 1, ..., строковый литерал № N};
```

Число строковых литералов должно быть меньше или равно количеству строк в массиве. Если число строковых литералов меньше размера массива, то все остальные элементы инициализируются пустыми строками. Длина каждого строкового литерала должна быть строго меньше значения длины строки (для записи завершающего нуля).

Например:

```
char days[12][10] = {"Январь", "Февраль", "Март", "Апрель",
"Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь",
"Ноябрь", "Декабрь"};
```

При объявлении массивов строк с инициализацией допускается не указывать количество строк в квадратных скобках. В таком случае количество строк в массиве будет определено автоматически по числу инициализирующих строковых литералов.

Пример массива из семи строк:

```
char days[][12] = {"Понедельник", "Вторник", "Среда",
"Четверг", "Пятница", "Суббота", "Воскресенье"};
```

Функции для работы со строками в Си. Все библиотечные функции, предназначенные для работы со строками, можно разделить на три группы:

1) ввод и вывод строк;
2) преобразование строк;
3) обработка строк.

Ввод и вывод строк в Си. Для ввода и вывода строк в библиотеке `stdio.h` содержатся функции `gets_s` и `puts`.

Функция `gets_s` предназначена для ввода строк и имеет следующий заголовок:

```
char * gets_s(char *buffer, size_t sizeInCharacters);
```

Функция `puts` предназначена для вывода строк и имеет следующий заголовок:

```
int puts(const char *string);
```

Простейшая программа ввода и вывода строки с использованием функций `gets_s` и `puts` будет иметь вид

```
char str[100] = "";
printf("Введите строку: ");
gets_s(str, 100);
printf("Вы ввели: ");
puts(str);
```

Функция ввода символов. Для ввода символов может использоваться функция `char getchar()`.

Преобразование и обработка строк. В библиотеке `string.h` содержатся функции для различных действий над строками. Наиболее используемые представлены в табл. 7.1.

Таблица 7.1

Основные функции стандартной библиотеки `string.h`

Имя функции	Прототип	Краткое описание
`atof`	`double atof(char *str);`	Преобразует строку `str` в вещественное число типа `double`
`atoi`	`int atoi(char *str);`	Преобразует строку `str` в десятичное целое число
`atol`	`long atol(char *str);`	Преобразует строку `str` в длинное десятичное целое число
`itoa`	`char *itoa(int v, char *str, int baz);`	Преобразует целое `v` в строку `str`. При изображении числа используется основание `baz` (от 2 до 36). Для отрицательного числа и `baz = 10` первый символ – минус (–)
`ltoa`	`char *itoa(long v, char *str, int baz);`	Преобразует длинное целое v в строку `str` . При изображении числа используется основание `baz` (от 2 до 36)
`strcat`	`char * strcat(char *sp, char *si);`	Приписывает строку `si` к строке `sp` (конкатенация строк)
`strchr`	`char *strchr(char *str, int c);`	Ищет в строке `str` первое вхождение символа `c`
`strcmp`	`int strcmp(char *str1 , char *str2);`	Сравнивает строки `str1` и `str2`. Результат отрицателен, если `strl` меньше `str2`; равен нулю, если `str1` равно `str2` и положителен, если `str1` больше `str2` (сравнение беззнаковое)
`strcpy`	`char *strcpy(char *sp, char *si);`	Копирует байты строки `si` в строку `sp`
`strcspn`	`int strcspn(char *str1, char *str2);`	Определяет длину первого сегмента строки `str1`, содержащего символы, не входящие во множество символов строки `str2`
`strdup`	`char *strdup(const char *str);`	Выделяет память и переносит в нее копию строки `str`
`strlen`	`unsigned strlen(char *str);`	Вычисляет длину строки `str`

Продолжение табл. 7.1

Имя функции	Прототип	Краткое описание
strlwr	`char *strlwr(char *str);`	Преобразует буквы верхнего регистра в строке в соответствующие буквы нижнего регистра
strncat	`char *strncat(char *sp, char *si, int kol);`	Приписывает `kol` символов строки `si` к строке `sp`
strncmp	`int strncmp(char *str1, char *str2, int kol);`	Сравнивает части строк `str1` и `str2`, причем рассматриваются первые `kol` символов. Результат отрицателен, если `str1` меньше `str2;` равен нулю, если `str1` равен `str2`, и положителен, если `str1` больше `str2`
strncpy	`char *strncpy(char *sp, char *si, int kol);`	Колирует `kol` символов строки `si` в строку `sp` («хвост» отбрасывается или дополняется пробелами)
strnicmp	`char *strnicmp(char *str1, char *str2, int kol);`	Сравнивает не более `kol` символов строки `str1` и строки `str2`, не делая различия регистров
strnset	`char *strnset(char *str, int c, int kol);`	Заменяет первые `kol` символов строки `str` символом `c`
strpbrk	`char *strpbrk(char *str1, char *str2);`	Ищет в строке `str1` первое появление любого из множества символов, входящих в строку `str2`
strrchr	`char *strrchr(char *str, int c);`	Ищет в строке `str` последнее вхождение символа `c`
strset	`int strset(char *str, int c);`	Заполняет строку `str` символом `c`
strspn	`int strspn(char *str1, char *str2);`	Определяет длину первого сегмента строки `str1`, содержащего только символы из множества символов строки `str2`
strstr	`char *strstr(const char *str1, const char *str2);`	Ищет в строке `str1` подстроки `str2`. Возвращает указатель на тот элемент в строке `str1`, с которого начинается подстрока `str2`
strtod	`double *strtod(const char *str, char **endptr);`	Преобразует символьную строку `str` в число двойной точности. Если `endptr` не равен `null`, то `*endptr` возвращается как указатель на символ, при достижении которого прекращается чтение строки `str`

Окончание табл. 7.1

Имя функции	Прототип	Краткое описание
strtok	`char *strtok(char *str1, const char *str2);`	Ищет в строке `str1` лексемы, выделенные символами из второй строки
strtol	`long *strtol(const char *str, char **endptr, intbaz);`	Преобразует символьную строку `str` к значению «длинное число» с основанием `baz`(от 2 до 36). Если `endptr` не равен `null`, то `*endptr` возвращается как указатель на символ, при достижении которого прекращается чтение строки `str`
strupr	`char *strupr(char *str);`	Преобразует буквы нижнего регистра в строке в соответствующие буквы верхнего регистра
ultoa	`char *ultoa(unsigned long v, char *str, int baz);`	Преобразует беззнаковое длинное целое `v` в строку `str`

Пример 7.12

Посчитать количество символов во введенной строке.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  char s[80], sym;
  int count, i;
  puts("Введите строку: ");
  gets_s(s, 80);
  printf("Введите символ: ");
  sym = getchar();
  count = 0;
  for (i = 0; s[i] != '\0'; i++)
  {
    if (s[i] == sym) count++;
  }
  printf("В строке\n");
  puts(s);  // вывод строки
  printf("символ ");
  putchar(sym); // вывод символа
  printf("встречается %d раз", count);
  getchar();
  return 0;
}
```

Пример 7.13

Дана строка (максимально 100 символов), содержащая слова, разделенные одним или несколькими пробелами или знаками табуляции. Заменить все знаки табуляции знаком пробела, удалить двойные пробелы из строки. Функции из библиотеки `string.h` при реализации программы не использовать.

```
#include<stdio.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  char str[101];
  printf("Введите строку: ");
  gets_s(str, 101);
  for(int i = 0; str[i] != 0;i++)
    if(str[i] == '\t') str[i] = ' ';
  int j = 1;
  for(int i = 1; str[i] != 0; i++)
  {
    if((str[i] == ' ') && (str[i-1] == ' ')) continue;
    str[j++] = str[i];
  }
  str[j] = 0;
  printf("Результат: "); puts(str);
  return 0;
}
```

Пример 7.14

Дана строка (максимально 100 символов), содержащая слова, разделенные одним или несколькими пробелами или знаками табуляции. Заменить все знаки табуляции знаком пробела, удалить двойные пробелы из строки. При реализации программы использовать функции из библиотеки `string.h`.

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>
#include<locale.h>
int main()
{
  setlocale(LC_ALL, "rus");
  char str[101];
  printf("Введите строку: "); gets_s(str, 101);
  do
  {
    int ind = strcspn(str, "\t");
    if(str[ind] == 0) break;
    str[ind] = ' ';
  }
```

```
while(1);
  do
  {
    char *ptr = strstr(str," ");
    if(!ptr) break;
    strcpy(ptr, ptr + 1);
  }
while(1);
  printf("Результат: "); puts(str);
  return 0;
}
```

Индивидуальные задания

Разработать программу, которая выполняет действия из задания 1 (работа с матрицами) и задания 2 (работа со строками). В программе должно быть реализовано меню, каждое действие должно быть реализовано в отдельной функции. Содержание меню:

1. Выполнить задание 1.
2. Выполнить задание 2.
3. Выход.

Задание 1. Работа с матрицами

1.Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.

2.Найти в матрице первую строку, все элементы которой упорядочены по убыванию. Изменить упорядоченность элементов этой строки на обратную.

3.Найти в матрице первый столбец, все элементы которого положительны. Знаки элементов предыдущего столбца изменить на противоположные.

4. Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.

5. Найти в матрице первую строку, все элементы которой упорядочены по убыванию. Изменить упорядоченность элементов этой строки на обратную.

6. Найти количество элементов в каждой строке матрицы C (8 × 8), больших, чем среднее арифметическое элементов данной строки.

7. Последний отрицательный элемент каждого столбца прямоугольной матрицы заменить нулем.

8. Определить среднее арифметическое значение элементов матрицы A (8 × 8), лежащих на главной диагонали.

9. Поменять местами максимальный и минимальный элементы данной квадратной матрицы.

10. Определить, есть ли в данном массиве размером 4 × 4 строка, содержащая больше положительных элементов, чем отрицательных.

11. Проверить, есть ли в матрице хотя бы один нечетный элемент. Если да, то умножить на него все элементы матрицы.

12. Найти в матрице первый столбец, все элементы которого положительны. Знаки элементов предыдущего столбца изменить на противоположные.

13. Найти в матрице первую строку, все элементы которой положительны, и сумму этих элементов. Уменьшить все элементы матрицы на эту сумму.

14. Проверить, есть ли в матрице хотя бы одна строка, содержащая положительный элемент, и найти ее номер. Знаки элементов предыдущей строки изменить на противоположные.

15. Дан массив A [n, m]. Удалить строки массива, не имеющие ни одного повторяющегося элемента.

16. Дана целочисленная квадратная матрица. Найти произведение элементов матрицы, лежащих ниже главной диагонали.

17. Даны две матрицы размером M × N. Найти их произведение.

18. Найти в матрице первую строку, все элементы которой равны нулю. Все элементы столбца с таким же номером уменьшить вдвое.

19. Определить, есть ли в массиве столбец, в котором содержится равное количество положительных и отрицательных элементов.

20. Проверить, все ли строки матрицы упорядочены по убыванию. Если нет, найти первую неупорядоченную строку и упорядочить.

21. Найти в матрице первую строку, все элементы которой упорядочены по возрастанию. Изменить порядок элементов этой строки на обратный.

22. Проверить, все ли строки матрицы содержат хотя бы один положительный элемент. Если да, то изменить знаки всех элементов матрицы на обратные.

23. Дан массив размером 6 × 6. Удалить строки массива, имеющие нулевые элементы.

24. Проверить, все ли строки матрицы содержат хотя бы один нулевой элемент. Если нет, то заменить значения всех отрицательных элементов матрицы на нулевые.

25. Даны две матрицы размером M × N. Найти их сумму.

26. Для произвольной матрицы найти столбец с наименьшей суммой элементов.

27. Проверить, все ли строки матрицы содержат хотя бы один отрицательный элемент. Если да, то изменить знаки всех элементов матрицы на обратные.

28. Расставить столбцы матрицы таким образом, чтобы элементы в первой строке были упорядочены по возрастанию.

29. Определить, сколько чисел в массиве равны произведению своих индексов i · j.

30. Дана матрица размером 4 × 4. Определить значение и номер последнего положительного числа.

31. Заменить все нечетные элементы матрицы на сумму максимального и минимального элементов.

32. Определить, есть ли в массиве размером 7 × 7 строка, в которой ровно два отрицательных элемента.

33. Дана матрица размером 8 × 8. Определить, есть ли в данной матрице столбец, в котором имеются одинаковые элементы.

34. Дан двумерный массив размером n × m. Найти строку с минимальной суммой и максимальный элемент в ней.

35. Найти в каждой строке двумерного массива максимальный и минимальный элементы и поменять их с первым и последним элементом соответственно.

36. Найти все отрицательные элементы матрицы. Заменить первое нечетное число на 1, второе на 3, третье на 5 и т. д.

37. Определить в двумерном массиве размером 6 × 6 строку с максимальной и столбец с минимальной суммой элементов.

38. Дан двумерный массив размером n × m. Определить в нем разность между средним арифметическим элементов массива и средним арифметическим максимального и минимального элементов.

39. Подсчитать число элементов матрицы Q (9 × 11), остаток от деления которых на 7 равен 1.

40. Определить среднее арифметическое значение элементов матрицы A, лежащих на главной диагонали.

41. Дана вещественная матрица размером 7 × 9. Упорядочить строки матрицы по возрастанию наименьших элементов строк.

42. В каждой строке матрицы сменить знак максимального по модулю элемента на противоположный.

43. Заменить все нулевые элементы матрицы на среднее арифметическое элементов строк, в которых они находятся.

Задание 2. Работа со строками

1. Написать программу, преобразующую строку, состоящую только из прописных букв, в строку, состоящую из прописных и строчных букв. Первая буква после точки – прописная, остальные – строчные.

2. Ввести массив символов, содержащий текст. Определить длину самого короткого слова и самого длинного слова.

3. Дана строка. Слова в предложении разделены одним или несколькими пробелами. Слова могут состоять только из цифр или букв. Необходимо найти сумму чисел, входящих в строку.

4. Дана строка. Подсчитать сумму кодов символов каждого слова. Слова в строке разделены пробелами.

5. Написать программу, которая вводит строки текста в массив символов `s[100]`, используя функцию `gets`. Вывести строки в верхнем и нижнем регистрах.

6. Даны два массива символов, содержащие тексты. Напечатать те слова, которые встречаются в каждом из двух заданных предложений.

7. Написать программу, которая вводит несколько строк текста и символ поиска и использует функцию `strchr`, чтобы определить суммарное число вхождений символа в текст.

8. Написать программу, которая вводит несколько строк текста и использует функцию `strtok`, чтобы сосчитать общее количество слов (слова разделяются символами новой строки или пробелами).

9. Написать программу, которая вводит ряд строк и выводит те из них, которые начинаются с буквы b.

10. Дана строка. Все русские буквы привести к верхнему регистру, латинские — заменить на «;». Вывести результат на экран.

11. Ввести текст. Подсчитать, сколько слов в тексте начинается на букву а (слова разделены пробелом).

12. Ввести текст. Подсчитать, сколько букв t содержится в последнем предложении.

13. Ввести текст. Подсчитать, сколько раз встречается в тексте слово cat.

14. Ввести массив символов. Вычислить произведение входящих в него цифр (без учета их знаков).

15. В заданном массиве символов слова зашифрованы – каждое из них записано наоборот. Расшифровать сообщение.

16. Дана строка. Подсчитать количество заглавных букв в строке.

17. Составить программу, определяющую правильность заданного скобочного выражения (количество открывающих скобок должно быть равно количеству закрывающих скобок – круглых, фигурных или квадратных).

18. Дана строка. Определить количество одинаковых слов в исходной строке.

19. Ввести текст. Убрать заданное слово из исходной строки (без учета различия строчных и прописных букв).

20. Дана строка. Убрать лишние пробелы между словами исходной строки.

21. Дан текст. Заменить заданное слово в исходной строке на указанное слово.

22. Дана строка. Определить количество слов в исходной строке, начинающихся на заданную букву

23. Вывести для каждой из двух строк строку, в которой содержатся только те символы, которые есть только в данной строке.

24. Дан текст. Вывести те слова, у которых одинаковые первые буквы.

25. Ввести строку. Заменить во всех словах каждое вхождение буквы м на мяу.

26. Ввести массив символов. Вывести те слова, в которых нет повторения букв.

27. Написать программу, которая выводит все слова заданной строки в обратном порядке.

28. Дана строка. Написать программу, которая удаляет в словах нечетной длины среднюю букву.

29. Дан текст. Вывести три самых длинных слова в этом тексте.

30. Дана символьная строка. Удалить из нее все символы, не являющиеся буквами.

31. Дана символьная строка. Удалить из нее все сочетания «ас».

32. Дана строка. Записать слова этой строки в обратном порядке.

33. Дана строка символов. Удалить из нее все слова четной длины.

34. Ввести строку символов. Подсчитать в ней количество латинских букв.

35. Дана строка. Определить, сколько слов в последовательности A оканчивается на букву N и сколько слов в последовательности B оканчивается на букву Y.

36. Дан текст. Определить число символов, отличных от букв и пробелов, встретившихся в тексте до первой точки.

37. Ввести текст. Удалить из текста все символы, отличные от цифр и пробелов.

38. Ввести текст. Определить, содержит ли букву A первое слово заданного текста.

39. Написать программу, которая во всех словах, начинающихся с p, заменяет p на сочетание ph.

Лабораторная работа № 8
Структуры

Цель работы – изучить работу со сложными типами данных в языке Си. Рассмотреть определение структурного типа, инициализацию, способы обращения к элементам, размещение в памяти. Научиться использовать структуры и массивы структур при разработке программ.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Язык программирования Си поддерживает определяемые пользователем структуры – структурированный тип данных. Он является собранием одного или более объектов (переменных, массивов, указателей, других структур и т. д.), которые для удобства работы с ними сгруппированы под одним именем.

Структуры:

– облегчают написание и понимание программ;

– помогают сгруппировать данные, объединяемые каким-либо общим понятием;

– позволяют группу связанных между собой переменных использовать как множество отдельных элементов, а также как единое целое.

Как и массив, структура представляет собой совокупность данных, но отличается от него тем, что к ее элементам (компонентам) необходимо обращаться по имени и элементы могут быть различного типа. Структуры целесообразно использовать там, где необходимо объединить данные, относящиеся к одному объекту.

Определение структуры состоит из двух шагов:

– объявление шаблона структуры (задание нового типа данных, определенного пользователем);

– определение переменных типа объявленного шаблона.

Объявление шаблонов структур. Общий синтаксис объявления шаблона структуры

```
struct имя_шаблона
{
  тип1 имя_переменной1;
  тип1 имя_переменной1;
  // другие поля структуры;
};

struct DateBase
{
  char fam[20];
```

```
  char name[15];
  long TelNumber;
  char *Adress;
  double w;
};
```

Имена шаблонов должны быть уникальными в пределах их области определения для того, чтобы компилятор мог различать различные типы шаблонов. Задание шаблона осуществляется с помощью ключевого слова `struct`, за которым следует имя шаблона структуры и список элементов, заключенных в фигурные скобки.

Имена элементов в одном шаблоне также должны быть уникальными. Однако в разных шаблонах можно использовать одинаковые имена элементов.

Задание только шаблона не влечет резервирования памяти компилятором. Шаблон представляет компилятору необходимую информацию об элементах структурной переменной для резервирования места в оперативной памяти и организации доступа к ней при определении структурной переменой и использовании отдельных элементов структурной переменной.

Кроме стандартных типов данных (`int`, `float`, `char` и т. д.) среди типов данных структуры могут также присутствовать ранее определенные типы, например:

```
/* объявление шаблона структуры типа date */
struct date
{
  int day, month, year;
};
/* шаблон структуры person */
struct person
{
 char fam[30], im[20], otch[15];
  float weight;
  int height;
  struct date birthday;
};
```

Структура `date` имеет три поля типа `int`. Шаблон структуры `person` в качестве элемента включает поле `birthday`, которое, в свою очередь, имеет ранее объявленный тип данных `struct date`. Этот элемент (`birthday`) содержит в себе все компоненты шаблона `struct date`.

Определение структур-переменных. Определение структуры-переменной ничем не отличается от объявления обычной переменной с предопределенным типом. Общий синтаксис:

```
struct имя_шаблона имя_переменной;
```

Пример 8.1

```
struct person stud[10];
```

Компилятор выделяет под каждую переменную количество байтов памяти, необходимое для хранения всех ее элементов.

Разрешается совмещать описание шаблона и определение структурной переменной.

Пример 8.2

```
struct book
{
 char title[20];
 char autor[30];
 double cast;
}
book1, book2, *ptr_bk = &book1;
```

Пример 8.3

```
struct date
{
  int day, month, year;
}
date1[5]; // объявление массива из 5 структур
```

Доступ к компонентам структуры. Доступ к полям осуществляется с помощью оператора «.» при непосредственной работе со структурой или «–>» при использовании указателей на структуру. Эти операторы называются селекторами членов класса. Общий синтаксис для доступа к компонентам структуры следующий:

```
имя_переменной_структуры.имя_поля;
имя_указателя -> имя_поля;
(*имя_указателя).имя_поля;
```

Пример 8.4

Прямой доступ к элементам.

```
1. date1[5].day = 10;
2. date1[5].year = 1991;
3. strcpy(book1.title, "Война и мир");
/* текст помещается в переменную с помощью функции копирования
strcpy()  */
4. stud[3].birthday.month = 1;
5. stud[3].birthday.year = 1980;
```

Пример 8.5

Доступ по указателю.

```
1.(date1 + 5) -> day = 10;
2.(stud + 3) -> birthday.month = 1;
/* используя доступ по указателю на структуру, присваиваем значение соответствующей переменной */
3.(*(date1 + 5)).day = 10;
4.(*(stud + 3)).birthday.month = 1;
```

Инициализация структур. При определении структурных переменных можно инициализировать их поля. Эта возможность подобна инициализации массива и следует тем же правилам:

```
имя_шаблона имя_переменной_структуры = {значение1, значение2, …};
```

Компилятор присваивает `значение1` первой переменной в структуре, `значение2` – второй переменной структуры и т. д. Здесь необходимо следовать некоторым правилам:

– присваиваемые значения должны совпадать по типу с соответствующими полями структуры;

– можно объявлять меньшее количество присваиваемых значений, чем количество полей – компилятор присвоит нули остальными полями структуры;

– список инициализации последовательно присваивает значения полям структуры, вложенным структурам и массивам.

Пример 8.6

```
struct date
{
  int day, month, year;
}
d[5] = {{1, 3, 1980}, {5, 1, 1990}, {1, 1, 1983}};
```

Копирование структур-переменных. Язык Си позволяет оператору присваивания копировать значения одной структуры-переменной в другую переменную при условии, что обе структуры-переменные относятся к одному и тому же типу. Таким образом, единственный оператор может скопировать несколько типов данных, которые включают массивы и вложенные структуры.

Однако следует учитывать, что оператор присваивания выполняет то, что называется «поверхностной копией» в применении к структурам-переменным. Поверхностная копия представляет собой копирование бит за битом значений полей переменной-источника в соответствующие поля переменной-цели. При этом может возникнуть проблема с такими типами данных, как указатели, поэтому использовать поверхностное копирование структур надо осторожно.

Пример 8.7

Создать массив структур данных о студентах группы. Для каждого студента записать: имя, фамилию, год рождения, оценки по пяти экзаменам. Определить средний балл за сессию и отсортировать список по сумме баллов.

```
#include<stdio.h>
#include<locale.h>
struct student
{
  char name[20];
  char fam[30];
  int year;
  int mark[5];
  int average;
};
struct student students[30];
struct student buffer;
int records;
int i, j;
int main()
{
  setlocale(LC_ALL, "rus");
  records = 0;
  do
  {
    printf("Студент № %d\n", records + 1);
    puts("Введите фамилию: ");
    fflush(stdin);
    gets_s(students[records].fam);
    puts("Введите имя: ");
    fflush(stdin);
    gets_s(students[records].name);
    puts("Введите возраст: ");
    scanf_s("%d", &students[records].year);
    for(i = 0; i < 5; i++)
    {
      printf("Введите оценки по экзаменам № %d: ", i + 1);
      scanf_s("%d", &students[records].mark[i]);
    }
    records++;
    puts("Завершить работу? (1/0)");
    scanf_s("%d", &i);
  }
  while(i);
  for (i = 0; i < records; i++)
  {
    students[i].average = 0;
    for (j = 0; j < 5; j++)
      students[i].average += students[i].mark[j];
  }
  for (i = 0; i < records-1; i++)
```

```
        for (j = i; j < records; j++)
          if (students[i].average > students[j].average)
          {
            Buffer = students[i];
            students[i] = students[j];
            students[j] = buffer;
          }
      for (i = 0; i < records; i++)
      {
      printf("Студент %s %s\n", students[i].name,
students[i].fam);
        printf("Возраст: %d\n", students[i].year);
        printf("Средний балл: %d\n", students[i].average);
      }
      return 0;
    }
```

Пример 8.8

Ввести массив элементов структурного типа. Отсортировать массив в алфавитном порядке фамилий, входящих в структуру.

```
    #include<stdio.h>
    #include<string.h>
    #include<locale.h>
    struct st
    {
      char name[80];
      int age;
    };
    int main()
    {
      setlocale(LC_ALL, "rus");
      int i, j, k;
      struct st m[100], t;
      printf("Введите количество студентов: ");
      scanf_s("%d", &k);
      for (i = 0; i < k; i++)
      {
        puts("Введите имя студента:");
        getchar();
        gets_s(m[i].name);
        puts("Введите возраст: ");
        scanf_s("%d", &m[i].age);
      }
      for (i = 0; i < k - 1; i++)
        for (j = i + 1; j < k; j++)
          if (strcmp(m[i].name, m[j].name) > 0)
          {
            t = m[i]; m[i] = m[j]; m[j] = t;
          }
      printf("\n\nРезультат: ");
```

```
  for (i = 0; i < k; i++)
  {
    printf("\n\n");
    puts(m[i].name);
    printf("%d years\n", m[i].age);
  }
  return 0;
}
```

Пример 8.9

Определить структурированный тип, набор функций (в виде меню) для работы с массивом структур. Структура «Склад»: наименование, единица измерения, цена единицы, количество, дата последнего прихода/расхода, тип накладной (приход или отгрузка).

В перечень обязательных функций входят:

– ввод элементов (полей) структуры с клавиатуры;

– вывод элементов (полей) структуры;

– поиск в массиве структуры с заданным значением поля;

– удаление заданного элемента;

– изменение (редактирование) заданного элемента.

Интерфейс пользователя выполнить в виде командного процессора:

– 1 – загрузить данные;

– 2 – вывести на экран.

```
#include<iostream>
#include<string.h>
#include<windows.h>
#include<locale.h>
using namespace std;
struct sklad  // объявляем шаблон структуры
{
  char name[30];      // наименование
  char ed[5];         // единица измерения
  float cena;         // цена
  int kol;            // количество
  int date;           // дата последнего завоза
  int type;           // тип накладной (приход или отгрузка)
};
struct sklad mas[30]; // объявляем глобальный массив структур
struct sklad tmp; // объявляем временную переменную структуру
типа
int sch = 0;        // счетчик полных записей
int er;             // переключатель
void enter_new();   // прототипы функций
int menu();
void out();
void del();
void change();
void find();
```

```
int main()
{
  setlocale(LC_ALL, "rus");
  while(1)
  {
    switch(menu())
    {
      case 1: del(); break;
      case 2: enter_new(); break;
      case 3: change(); break;
      case 4: out(); break;
      case 5: find(); break;
      case 6: return 0;
      default: cout<<"Неверный выбор/n";
    }
  }
}
void enter_new()  // функция ввода новой структуры
{
  if(sch < 30) /* вводим новую запись, только если счетчик
полных записей меньше максимального количества записей */
  {
    cout<<"Запись номер "<<sch + 1; // выводим номер записи
    cout<<"\nВыберите тип накладной (1 – приход, 2 –
отгрузка)\n";
    cin>>mas[sch].type;
    cout<<"\nВведите наименование\n";
    cin>>mas[sch].name;
    cout<<"Введите ед. измерения \n";
    cin>>mas[sch].ed;
    cout<<"Введите цену\n";
    cin>>mas[sch].cena;
    cout<<"Введите кол-во\n";
    cin>>mas[sch].kol;
    cout<<"Введите дату последнего поступления\n";
    cin>>mas[sch].date;
    sch++; // увеличиваем счетчик полных записей на единицу
  }
  else
    cout<<"Введено максимальное кол-во записей";
}
int menu()
{
  int er;
  cout<<"Введите:\n";
  cout<<"1 – удаление записи\n";
  cout<<"2 – ввод новой записи\n";
  cout<<"3 – изменение записи\n";
  cout<<"4 – вывод записи(ей)\n";
  cout<<"5 – поиск \n";
  cout<<"6 – выход\n";
  cin>>er;
```

```
      return er;
    }
    void out()  // функция вывода записей
    {
     int sw; // переключатель выбора: выводить все записи или одну
      int k;  // номер структуры, которую надо вывести
      if (sch == 0)
          cout<<"\nНет записей:\n";
      else
        {
          cout<<"\nВведите:\n";
          cout<<"1 - вывести какую-либо запись\n";
          cout<<"2 - вывести все записи\n";
          cin>>sw;
          if(sw == 1)
          {
            cout<<"Введите номер записи, которую нужно вывести\n";
            cin>>k;
            cout<<endl;
            if (mas[k - 1].type == 1)
              cout<<"Приход: "<<endl;
            else
              cout<<"Отгрузка: "<<endl;
              cout<<"Наименование: "<<mas[k - 1].name<<endl;
              cout<<"Ед. измер.: "<<mas[k - 1].ed<<endl;
              cout<<"Цена: "<<mas[k - 1].cena<<endl;
              cout<<"Кол-во: "<<mas[k - 1].kol<<endl;
              cout<<"Дата: "<<mas[k - 1].date<<endl;
              cout<<"____________"<<endl;
          }
          if(sw == 2)
          {
            for(int i = 0; i < sch; i++) /* выводим в цикле все
записи */
            {
              if (mas[i].type == 1)
                cout<<"Приход: "<<endl;
              else
                cout<<"Отгрузка: "<<endl;
              cout<<"Наименование: "<<mas[i].name<<endl;
    /* выводим на экран значение name i-й структуры */
              cout<<"Ед. измер.: "<<mas[i].ed<<endl;
              cout<<"Цена: "<<mas[i].cena<<endl;
              cout<<"Кол-во: "<<mas[i].kol<<endl;
              cout<<"Дата: "<<mas[i].date<<endl;
              cout<<"____________"<<endl;
            }
          }
        }
    }
    void del() // функция удаления записи
    {
```

```
       int d;  // номер записи, которую нужно удалить
       cout<<"\nВведите номер записи, которую необходимо удалить\n";
       cout<<"Если необходимо удалить все записи, введите '99'\n";
       cin>>d;
       if (d != 99)
       {
         for (int i = (d - 1); i < sch; i++) /* цикл для удаления
заданной записи, начинает цикл с удаляемой записи */
           mas[i] = mas[i + 1]; /* замещаем текущую запись следую-
щей за ней */
         sch = sch - 1; // уменьшаем счетчик полных записей на 1
       }
       if (d == 99)
       {
         for(int i = 0; i < 30; i++)
           mas[i] = tmp; /* замещаем каждую структуру в массиве
пустой структурой */
         sch = 0; /* счетчик структур обнуляем, т. к. все записи
удалены */
       }
     }
     void change() // функция для изменения записи
     {
       int c; // номер записи, которую нужно изменить
       int per;
       cout<<"\nВведите номер записи\n";
       cin>>c;
       do
       {
         cout<<"Введите:\n";
         cout<<"1 - изменение типа накладной\n";
         cout<<"2 - изменение наименования\n";
         cout<<"3 - изменение ед.измерения\n";
         cout<<"4 - изменение цены\n";
         cout<<"5 - изменение количества\n";
         cout<<"6 - изменение даты\n";
         cout<<"7 - выход\n";
         cin>>per;
         switch (per)
         {
            case 1: cout<<"\nВведите новый тип накладной
(1 - приход, 2 - отгрузка)\n";
                    cin>>mas[c - 1].type; break;
            case 2: cout<<"\nВведите новое наименование\n";
                    cin>>mas[c - 1].name; break;
            case 3: cout<<"Введите новые ед. измерения \n";
                    cin>>mas[c - 1].ed; break;
            case 4: cout<<"Введите новую цену\n";
                    cin>>mas[c - 1].cena; break;
            case 5: cout<<"Введите новое кол-во\n";
                    cin>>mas[c - 1].kol; break;
```

```
            case 6: cout<<"Введите новую дату последнего
поступления\n";
             cin>>mas[c - 1].date; break;
            case 7: return;
          }
        }
        while(1);
      }
      void find()  // функция поиска записей
      {
        int sw;     // переключатель
        if (sch == 0)
          cout<<"\nНет записей: \n";
        else
        {
          cout<<"\nВведите:\n";
          cout<<"1 - все накладные прихода\n";
          cout<<"2 - все накладные отгрузки\n";
          cin>>sw;
          for(int i = 0;i < sch;i++) /* в цикле просматриваем все
структуры из массива структур */
           if (mas[i].type == sw)
           {
             if (mas[i].type == 1)
               cout<<"Приход: "<<endl;
             else
               cout<<"Отгрузка: "<<endl;
             cout<<"Наименование: "<<mas[i].name<<endl;
             cout<<"Ед. измерения: "<<mas[i].ed<<endl;
             cout<<"Цена: "<<mas[i].cena<<endl;
             cout<<"Кол-во: "<<mas[i].kol<<endl;
             cout<<"Дата: "<<mas[i].date<<endl;
             cout<<"____________"<<endl;
           }
        }
      }
```

Индивидуальные задания

Ввести массив структур в соответствии с вариантом. Рассортировать массив в алфавитном порядке по первому полю, входящему в структуру. В программе реализовать меню, содержащее пункты, перечесленные ниже.

1. Ввод массива структур.
2. Сортировка массива структур.
3. Поиск в массиве структур по заданному параметру.
4. Изменение заданной структуры.
5. Удаление структуры из массива.
6. Вывод на экран массива структур.
7. Выход.

Варианты

1. Структура «Автосервис»: регистрационный номер автомобиля; марка; пробег; мастер, выполнивший ремонт; сумма ремонта.

2. Структура «Сотрудник»: фамилия, имя, отчество; должность; год рождения; заработная плата.

3. Структура «Государство»: название; столица; численность населения; занимаемая площадь.

4. Структура «Человек»: фамилия, имя, отчество; домашний адрес; номер телефона; возраст.

5. Структура «Читатель»: фамилия, имя, отчество; номер читательского билета; название книги; срок возврата.

6. Структура «Школьник»: фамилия, имя, отчество; класс; номер телефона; оценки по предметам (математика, физика, русский язык, литература).

7. Структура «Студент»: фамилия, имя, отчество; домашний адрес; группа; рейтинг.

8. Структура «Покупатель»: фамилия, имя, отчество; домашний адрес; номер телефона; номер дисконтной карты.

9. Структура «Пациент»: фамилия, имя, отчество; домашний адрес; номер медицинской карты; номер страхового полиса.

10. Структура «Информация»: носитель; объем; название; автор.

11. Структура «Клиент банка»: фамилия, имя, отчество; номер счета; сумма на счете; дата последнего изменения.

12. Структура «Склад»: наименование товара; цена; количество; процент торговой надбавки; дата поступления товара на склад.

13. Структура «Авиарейсы»: номер рейса; пункт назначения; время вылета; дата вылета; стоимость билета.

14. Структура «Вокзал»: номер поезда; пункт назначения; дни следования; время прибытия; время стоянки.

15. Структура «Кинотеатр»: название кинофильма; сеанс; стоимость билета; количество зрителей.

16. Структура «Фрукт»: название фрукта; вес; стоимость; цвет.

17. Структура «Мобильный телефон»: год выхода на рынок; диагональ экрана; операционная система; количество SIM-карт; оперативная память.

18. Структура «Фильм»: название фильма; год производства фильма; жанр; страна.

19. Структура «Ассортимент обуви»: артикул; наименование; количество; стоимость одной пары.

20. Структура «Книги, хранящиеся в библиотеке»: регистрационный номер книги; автор; название; год издания; издательство; количество страниц.

21. Структура «Информация о разговорах АТС»: дата разговора; время разговора; тариф; номер телефона.

22. Структура «Принтер»: марка принтера; скорость черно-белой печати; страна производства; срок службы.

23. Структура «Расписание занятий»: название предмета; время начала занятий; дата проведения занятий.

24. Структура «Исполнитель»: фамилия, имя, отчество; год рождения; название страны; музыкальный инструмент (гитара, фортепиано, скрипка, виолончель).

25. Структура «Ассортимент игрушек»: название игрушки; цена; количество; возрастные границы (например, 2–5).

26. Структура «Сведения о месячной заработной плате сотрудников отдела»: фамилия сотрудника; наименование отдела; размер заработной платы за месяц.

27. Структура «Участники спортивных соревнований»: фамилия, имя, отчество игрока; номер игрока; возраст; рост; вес.

28. Структура «Расписание движения автобусов»: номер автобуса; тип автобуса; пункт назначения; время отправления и прибытия.

29. Структура «Велосипед»: тип; размер колес; вес; материал; изготовитель.

30. Структура «Собака»: порода; вес; рост; дата рождения; окрас шерсти.

31. Структура «Ноутбук»: марка; количество ядер; объем памяти; материал корпуса; цвет корпуса.

32. Структура «Справочник по таблице Менделеева»: название химического элемента; тип элемента; валентность.

33. Структура «Грузовик»: марка; грузоподъемность; год выпуска; производитель.

34. Структура «Футболист»: фамилия; дата рождения; клуб; количество игр.

35. Структура «Музыкальный альбом»: название; год выпуска; стиль; количество треков; продолжительность звучания.

36. Структура «Огнестрельное оружие»: вид; дальность стрельбы; стоимость; вес.

37. Структура «Наручные часы»: бренд; тип часов; материал ремня/браслета; страна производства.

38. Структура «Одежда»: вид одежды; состав; цвет; материал; страна производства.

39. Структура «Торт»: тип коржей; тип начинки; форма; место изготовления.

40. Структура «Стол»: тип стола; материал столешницы; материал ножек; количество ножек; ширина; высота.

41. Структура «Палатка»: количество мест; количество входов; цвет; форма; вес.

42. Структура «Квартира»: количество комнат; местоположение; этаж; площадь; цена.

43. Структура «Компьютерная мышь»: количество кнопок; срок службы; вес; цвет; радиус действия.

44. Структура «Рюкзак»: назначение; материал; количество отделений; объем; цвет.

45. Структура «Банковская карта»: срок действия; валюта; номер карты; название банка.

46. Структура «Дом»: адрес; количество подъездов; количество этажей; наличие парковки.

47. Структура «Водитель»: график работы; отрасль компании; опыт работы; заработная плата.

48. Структура «Машина»: марка; год выпуска; максимальная скорость; количество мест; цена.

49. Структура «Кровать»: количество спальных мест; материал каркаса; цена; цвет; ширина кровати.

50. Структура «Курсы»: форма обучения; сфера; период обучения; количество свободных мест; стоимость.

Лабораторная работа № 9
Файлы

Цель работы – изучить понятие и назначение файла. Рассмотреть бинарные и текстовые файлы, библиотечные функции для работы с файлами (открытие, закрытие, ввод/вывод, организация прямого доступа). Научиться использовать файлы при разработке программ.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номер задания, коды программ, скриншоты с результатами выполнения программ.

Методические указания

Файл – это именованный объект, хранящий данные (программы или любую другую информацию) на каком-либо носителе. Файл, как и массив, – это совокупность данных.

Отличия файла от массива.

1. Файлы в отличие от массивов располагаются не в оперативной памяти, а на жестких дисках или на внешних носителях, хотя файл может располагаться и на так называемом электронном диске (в оперативной памяти).

2. Файл не имеет фиксированной длины, т. е. может увеличиваться и уменьшаться.

3. Перед работой с файлом его необходимо открыть, а после работы – закрыть.

Файловая система – это совокупность файлов и система, управляющая информацией на диске для обеспечения доступа к файлам. Или по-другому – это совокупность программных средств для доступа к файлам. Существует довольно много файловых систем и правила именования файлов в них могут незначительно отличаться.

Имена файлов состоят из двух частей, разделенных точкой: имя файла и расширение. Файлы хранятся в каталогах (директориях). Каталоги могут иметь такие же имена, что и файлы. Допускаются вложенные каталоги (подкаталоги).

Различают два вида файлов: текстовые и бинарные.

Текстовые файлы могут быть просмотрены и отредактированы с клавиатуры с помощью любого текстового редактора. Они имеют очень простую структуру – последовательность ASCII-символов. Эта последовательность символов разбивается на строки, каждая из которых заканчивается двумя символами `\r` (возврат каретки) и `\n` (новая строка). В шестнадцатеричной системе эти символы могут быть представлены кодами `0xD` и `0xA`, в десятичной – `13` и `10`.

Бинарные файлы – это файлы, которые не имеют структуры текстовых файлов. Каждая программа для своих бинарных файлов определяет собственную структуру.

Библиотека языка Си содержит функции для работы как с текстовыми, так и с бинарными файлами.

Функции работы с файлами

Функции открытия и закрытия файла. Перед работой с файлом его необходимо открыть. Функция открытия файла:

```
FILE *fopen(char *filename, char *mode);
```

Здесь `FILE` – структурный тип, который связан с физическим файлом и содержит всю необходимую информацию для работы с ним (указатель на текущую позицию в файле, тип доступа и др.).

`char *filename` – задает физическое местонахождение (путь) и имя открываемого файла;

`char *mode` – тип доступа к файлу, который может принимать значения, указанные в табл. 9.1.

Функция `fopen()`, называемая указателем на файл, при успешном открытии файла возвращает указатель на структуру типа `FILE`. Возвращаемое функцией значение нужно сохранить и использовать для ссылки на открытый файл. Если произошла ошибка при открытии файла, то возвращается `NULL`.

Таблица 9.1

Типы доступа к файлам

Команда	Действие
"r"	Открыть файл для чтения
"w"	Открыть файл для записи. Если файл существует, то его содержимое теряется
"a"	Открыть файл для записи в конец файла. Если файл не существует, то он создается
"r+"	Открыть файл для чтения и записи. Файл должен существовать
"w+"	Открыть файл для чтения и записи. Если файл существует, то его содержимое теряется
"a+"	Открыть файл для чтения и записи в конец файла. Если файл не существует, то он создается

К комбинациям вышеперечисленных литералов могут быть добавлены также "t" (открыть файл в текстовом режиме) либо "b" (открыть файл в бинарном режиме).

Возможны следующие режимы доступа: `"w+b", "wb+", "rw+", "w+t", "rt+"` и др. Если режим не указан, то файл открывается в текстовом режиме.

После работы с файлом он должен быть закрыт функцией `fclose()`.

Для этого необходимо в указанную функцию передать указатель на FILE, который был получен при открытии функцией fopen(). При завершении программы незакрытые файлы автоматически закрываются системой.

Стандартная последовательность операторов, необходимая для открытия и закрытия файла:

```
#include <stdio.h>
int main()
{
   FILE *f;
   if(!(f = fopen("readme.txt", "r + t")))
   {
     printf("Невозможно открыть файл \n");  return;
   }
   . . .          // работа с файлом
   fclose(f);    // закрытие файла
   return 0;
}
```

Примечание. Функция fopen является функцией незащищенного открытия файла, т. к. появилась в ранних версиях библиотек языка Си. Чтобы разрешить работу данной функции в современных компиляторах, необходимо в начало программы добавить строку #define _CRT_SECURE_NO_WARNINGS.

Другой вариант – воспользоваться функцией защищенного открытия fopen_s, которая имеет следующий синтаксис:

```
if (fopen_s(&f, "test.txt", "w + t") ! = 0)
{
  printf("Невозможно открыть файл\n");  return 0;
}
else
. . .           // работа с файлом
fclose(f);
```

Функции чтения из файла/записи в файл. Функции для работы с текстовым файлом: fprintf(), fscanf_s(), fgets(), fputs(). Формат параметров этих функций очень похож на формат функций printf(), scanf_s(), gets() и puts(). Схожи не только параметры, но и действия. Отличие лишь в том, что printf(), scanf_s() и другие работают по умолчанию с консолью (экран, клавиатура), а fprintf(), fscanf_s() – с файлами (в том числе и со стандартными потоками stdin, stdout и др.), поэтому в них добавлен параметр, являющийся указателем на структуру FILE, которая была рассмотрена выше.

fprintf() – это функция форматированного вывода в файл.

Пример 9.1
Записать в текстовый файл числа от 0 до 1000, кратные 3.

```
#include<stdio.h>
#include<locale.h>
int main()
{
   setlocale(LC_ALL, "rus");
   int a;
   FILE *f;
   if (fopen_s(&f, "test.txt", "w + t") == 0)
   {
     for (a = 0; a < 100; a += 3)
     {
       fprintf(f, "%d, ", a);
     }
     printf("Данные записаны в файл\n");
     fclose(f);
   }
   else
     printf("Невозможно создать файл\n");
   return 0;
}
```

Возвращаемое значение функции `fopen_s` содержит код ошибки. Эта функция возвращает нуль, если все хорошо.

```
if(fopen_s(&f, "in.txt","r") == 0)
  {
... // работа с файлом
  }
else
... // обработка ошибки
```

Пример 9.2
Записать в текстовый файл имя и возраст студента и вывести данные на консоль, используя режим `w+`.

```
#include<stdio.h>
struct student
{
   char mas[125];
   int age;
};
   int main()
   {
       char ch[125];
       struct student m[100];
       int n, i;
```

```
            FILE *f;
            fopen_s(&f, "name.txt", "w+");
            if (f == 0)
            {
               printf("Enter number of students\n");
               scanf_s("%d", &n);
               for (i = 0; i < n; i++)
               {
                  printf("name\n");
                  getchar();
                  gets_s(m[i].mas);
                  printf("age\n");
                  scanf_s("%d", &m[i].age);
               }
               for (i = 0; i < n; i++)
                  fprintf(f, "Name:  %s\t age:  %d\n", m[i].mas,
m[i].age);
               rewind(f);   // перемещаем указатель в начало файла
               while (!feof(f)) // пока не найдет конец файла
               {
                  if(fgets(ch, 255, f)) /* считываем информацию из
файла */
                  puts(ch); /* выводим на экран информацию, считанную
из файла */
               }
            }
            else printf("Не удалось открыть файл");
            fclose(f);
            return 0;
      }
```

Функции для работы с текстовыми файлами удобно использовать при их создании, ведении файлов-протоколов (log-файлов) и т. п. Однако при создании баз данных целесообразно использовать функции для работы с бинарными файлами: `fwrite()` и `fread()`. Эти функции без каких-либо изменений копируют блок данных из оперативной памяти в файл и, соответственно, из файла – в память. Такой способ обмена данными требует меньше времени. Прототипы функций:

```
unsigned fread (void *ptr, unsigned size, unsigned n, FILE
*stream);

unsigned fwrite (void *ptr, unsigned size, unsigned n, FILE
*stream);
```

Здесь `*ptr` – указатель на буфер, `size` – размер блока, `n` – количество блоков, `*stream` – указатель на структуру `FILE` открытого файла.

Пример 9.3

Записать в бинарный файл и прочитать из бинарного файла список абитуриентов. Информация об абитуриентах представлена структурой.

```
#include<stdio.h>
#include<locale.h>
struct abitur
{
  char name[32];
  int mark[3];
};
int main()
{
  setlocale(LC_ALL, "rus");
  struct abitur inf;
  int a;
  FILE *f;
  if (fopen_s(&f, "inf.dat", "w+") != 0)
  {
     printf("Ошибка открытия файла\n"); return 0;
  }
  for (;;)
  {
     printf("Введите ФИО (пустая строка – конец списка): ");
     fflush(stdin);
     gets_s(inf.name);
     if (!inf.name[0]) break;
     printf("\n Введите три оценки, полученные на экзаменах: ");
     scanf_s("%d%d%d", &inf.mark[0], &inf.mark[1],
&inf.mark[2]);
     getchar();
     fwrite(&inf, 1, sizeof(inf), f);
  }
  fclose(f);
  printf("\nСписок абитуриентов:\n");
  if (fopen_s(&f, "inf.dat", "r") != 0)
  {
     printf("Ошибка открытия файла\n"); return 0;
  }
  while (1)
  {
     if (fread(&inf, sizeof(inf), 1, f))
       printf("%s  %d  %d  %d\n", inf.name, inf.mark[0],
inf.mark[1], inf.mark[2]);
     else
       break;
  }
  fclose(f);
  return 0;
}
```

Важно понимать, что нет однозначного метода, который можно было бы использовать для отличия текстового файла от бинарного, поэтому любой бинарный файл может быть открыт для работы как текстовый, а текстовый может быть открыт как бинарный. Но такой вариант работы с файлом обычно приводит к ошибкам. Поэтому рекомендуется работать с конкретным файлом в том режиме, в котором он был создан.

Позиционирование в файле. Каждый открытый файл имеет так называемый указатель на текущую позицию в файле (нечто подобное указателю в памяти). Все операции с файлами (чтение и запись) используют данные с этой позиции. При каждом выполнении функции чтения или записи указатель смещается на количество прочитанных или записанных байт, т. е. устанавливается сразу за прочитанным или записанным блоком данных в файле. В этом случае осуществляется так называемый последовательный доступ к данным, который очень удобен, когда нам необходимо последовательно работать с данными в файле. Этот процесс демонстрируется во всех вышеприведенных примерах чтения и записи в файл. Но иногда необходимо считывать или записывать данные в произвольном порядке – это достигается путем установки указателя на некоторую заданную позицию в файле функцией `fseek()`:

```
int fseek(FILE *stream, long offset, int whence);
```

Параметр `offset` задает количество байт, на которое необходимо сместить указатель соответственно параметру `whence`. Параметр `whence` может принимать значения, представленные в табл. 9.2.

Таблица 9.2

Значения параметра `whence`

Значение	Описание
`SEEK_SET`	Смещение выполняется от начала файла
`SEEK_CUR`	Смещение выполняется от текущей позиции указателя
`SEEK_END`	Смещение выполняется от конца файла

Такой доступ к данным в файле называют произвольным. Величина смещения может быть как положительной, так и отрицательной, но указатель нельзя смещать за пределы файла.

Пример 9.4

Найти по номеру запись из бинарного файла, который был создан в примере 9.3.

```
#include<stdio.h>
#include<locale.h>
struct abitur
```

```
{
   char name[32];
   int mark[3];
};
int main()
{
   setlocale(LC_ALL, "rus");
   struct abitur inf;
   int n;
   FILE *f;
   if (fopen_s(&f, "inf.dat", "r") != 0)
   {
      printf("Ошибка открытия файла\n");
      return 0;
   }
   while (1)
   {
      printf("Введите номер записи (0 – выход): ");
      scanf_s("%d", &n);
      if (!n) break;
      fseek(f, sizeof(struct abitur)*(n – 1), SEEK_SET);
      if (sizeof(inf) != fread(&inf, 1, sizeof(inf), f))
         printf("Такой записи в файле нет\n");
      else
         printf("%s %d %d %d", inf.name, inf.mark[0],
inf.mark[1], inf.mark[2]);
    }
    fclose(f);
    return 0;
}
```

Иногда необходимо определить позицию указателя. Для этого можно воспользоваться функцией `ftell()`:

```
long ftell(FILE *stream);
```

Функция возвращает значение указателя на текущую позицию файла. В случае ошибки возвращает число –1.

Функция

```
int fsetpos(FILE *stream, const long *pos)
```

устанавливает значение указателя чтения – записи файла (указатель на текущую позицию) в позицию, заданную значением по указателю `pos`. Возвращает 0 при корректном выполнении и любое ненулевое значение при ошибке.

Функция

```
int fgetpos(FILE *stream, long *pos)
```

помещает в переменную, на которую указывает `pos`, значение указателя на текущую позицию в файле. Возвращает 0 при корректном выполнении и любое ненулевое значение при ошибке.

Индивидуальные задания

Задание 1. Разработать программу с использованием файлов в соответствии с вариантом.

Варианты

1. Компоненты файла f – целые двухзначные числа (положительные и отрицательные). Получить файл g, образованный из f включением только чисел, кратных K.

2. Компоненты файла f – целые числа. Получить файл g, образованный из f включением только чисел < K.

3. Компоненты файла f – целые двухзначные числа. Получить файл g, образованный из f включением только чисел > K.

4. Даны три файла целых чисел одинакового размера с именами NameA, NameB и NameC. Создать новый файл с именем NameD, в котором чередовались бы элементы исходных файлов с одним и тем же номером: A0, B0, C0, A1, B1, C1, A2, B2, C2 и т. д.

5. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только четные строки.

6. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только те строки, которые начинаются с буквы A.

7. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 строки, начиная с K до K + 5.

8. Даны три файла целых чисел одинакового размера с именами NameA, NameB и NameC. Создать новый файл с именем NameD, записать в него максимальные элементы исходных файлов с одним и тем же номером: max(A0, B0, C0), max(A1, B1, C1), max(A2, B2, C2) и т. д.

9. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 строки, количество символов в которых больше, чем K.

10. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только строки, которые не содержат цифр.

11. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только те строки, которые заканчиваются на букву a.

12. Компоненты файла f – целые числа. Получить файл g, образованный из f исключением повторных вхождений одного и того же числа.

13. Даны три файла целых чисел одинакового размера с именами NameA, NameB и NameC. Создать новый файл с именем NameD, записать в него максимальные элементы исходных файлов с одним и тем же номером: max(A0, B0, C0),max(A1, B1, C1), max(A2, B2, C2) и т. д.

14. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только строки, которые начинаются с цифры.

15. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только строки, которые содержат цифры.

16. Компоненты файла F1 – натуральные числа. Переписать в файл F2 все нечетные числа, увеличив их в два раза.

17. Компоненты файла F1 – целые числа (положительные и отрицательные). Получить файл F2, образованный из файла F1 путем включения всех чисел, кратных 5, и файл F3, образованный из файла F1 путем включения всех отрицательных чисел, кратных 3.

18. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Переписать компоненты символьного файла F1 в файл F2, заменив при этом каждый восклицательный знак точкой, а каждое двоеточие – тремя точками.

19. Даны два файла: F1 типа `int` и F2 типа `char`. Создать новый файл с именем F3, в котором цифра из файла F1 вставляется после двух символов из файла F2.

20. Компоненты файла f – произвольная последовательность буквенных символов. Получить файл g, образованный из файла f, в котором символы упорядочены в алфавитном порядке, при этом все повторяющиеся символы должны быть удалены.

21. Компоненты файла f – символы. Получить файл g, образованный из файла f, в котором все прописные буквы заменены строчными.

22. Компоненты файла F1 – целые числа. Создать новый файл с именем F2, в который будут записаны числа из файла F1 в обратном порядке.

23. Компоненты файла F1 – натуральные числа. Переписать из файла F1 в файл F2 все натуральные числа, которые являются полными квадратами.

24. Компоненты файла F1 – целые числа. Получить файл F2, в котором сначала записано наибольшее значение одной из первых пяти компонент файла F1, затем следующих пяти компонент и т. д.

25. Компоненты файла F1 – действительные числа. Получить файл F2, образованный из F1 включением разности между первым и последним числами файла F1.

26. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только те строки, которые начинаются и заканчиваются одним и тем же символом.

27. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только те строки, которые начинаются с k идущих подряд символов m.

28. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Строки могут начинаться либо с русских, либо с английских слов. Скопировать из файла F1 в файл F2 строки, которые начинаются с английских слов.

29. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Скопировать из файла F1 в файл F2 только те строки, которые начинаются с буквы А и заканчиваются буквой Я.

30. Создать текстовый файл F1, состоящий из не менее чем 10 строк, и записать в него информацию. Слова в строке состоят из букв латинского алфавита. Получить файл F2, образованный из файла F1 включением только тех строк, которые не имеют ни одной из следующих букв: q, t, r, s, o, p.

Задание 2. Сформировать бинарный файл из элементов заданной в варианте структуры, вывести его содержимое, выполнить добавление элементов в соответствии со своим вариантом и добавить поиск по одному из параметров (например, по фамилии, государственному номеру, году рождения и т. д.). Формирование, вывод, добавление, поиск элементов и выбор желаемого действия оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

Варианты

1. Структура «Автосервис»: регистрационный номер автомобиля; марка; пробег; мастер, выполнивший ремонт; сумма ремонта.

2. Структура «Сотрудник»: фамилия, имя, отчество; должность; год рождения; заработная плата.

3. Структура «Государство»: название; столица; численность населения; занимаемая площадь.

4. Структура «Человек»: фамилия, имя, отчество; домашний адрес; номер телефона; возраст.

5. Структура «Читатель»: фамилия, имя, отчество; номер читательского билета; название книги; срок возврата.

6. Структура «Школьник»: фамилия, имя, отчество; класс; номер телефона; оценки по предметам (математика, физика, русский язык, литература).

7. Структура «Студент»: фамилия, имя, отчество; домашний адрес; группа; рейтинг.

8. Структура «Покупатель»: фамилия, имя, отчество; домашний адрес; номер телефона; номер дисконтной карты.

9. Структура «Пациент»: фамилия, имя, отчество; домашний адрес; номер медицинской карты; номер страхового полиса.

10. Структура «Информация»: носитель; объем; название; автор.

11. Структура «Клиент банка»: фамилия, имя, отчество; номер счета; сумма на счете; дата последнего изменения.

12. Структура «Склад»: наименование товара; цена; количество; процент торговой надбавки; дата поступления товара на склад.

13. Структура «Авиарейсы»: номер рейса; пункт назначения; время вылета; дата вылета; стоимость билета.

14. Структура «Вокзал»: номер поезда; пункт назначения; дни следования; время прибытия; время стоянки.

15. Структура «Кинотеатр»: название кинофильма; сеанс; стоимость билета; количество зрителей.

16. Структура «Фрукт»: название фрукта; вес; стоимость; цвет.

17. Структура «Мобильный телефон»: год выхода на рынок; диагональ экрана; операционная система; количество SIM-карт; оперативная память.

18. Структура «Фильм»: название фильма; год производства; жанр; страна.

19. Структура «Ассортимент обуви»: артикул; наименование; количество; стоимость одной пары.

20. Структура «Книги, хранящиеся в библиотеке»: регистрационный номер книги; автор; название; год издания; издательство; количество страниц.

21. Структура «Информация о разговорах АТС»: дата разговора; время разговора; тариф; номер телефона.

22. Структура «Принтер»: марка принтера; скорость черно-белой печати; страна производства; срок службы.

23. Структура «Расписание занятий»: название предмета; время начала занятий; дата проведения занятий.

24. Структура «Исполнитель»: фамилия, имя, отчество; год рождения; название страны; музыкальный инструмент (гитара, фортепиано, скрипка, виолончель).

25. Структура «Ассортимент игрушек»: название игрушки; цена; количество; возрастные границы (например, 2–5).

26. Структура «Сведения о месячной заработной плате сотрудников отдела»: фамилия сотрудника; наименование отдела; размер заработной платы за месяц.

27. Структура «Участники спортивных соревнований»: фамилия, имя, отчество игрока; игровой номер; возраст; рост; вес.

28. Структура «Расписание движения автобусов»: номер автобуса; тип автобуса; пункт назначения; время отправления и прибытия.

29. Структура «Велосипед»: тип; размер колес; вес; материал; изготовитель.

30. Структура «Собака»: порода; вес; рост; дата рождения; окрас шерсти.

Список использованных источников

1. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – СПб. : БХВ-Петербург, 2006. – 440 с.
2. Луцик, Ю. А. Основы алгоритмизации и программирования. Язык Си : учеб.-метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск : БГУИР, 2015. – 169 с.
3. Навроцкий, А. А. Основы алгоритмизации и программирования в среде Visual C++ : учеб.-метод. пособие / А. А. Навроцкий. – Минск : БГУИР, 2014. – 160 с.
4. Культин, Н. Б. C/C++ в задачах и примерах : учеб.-метод. пособие / Н. Б. Культин. – СПб. : БХВ-Петербург, 2005. – 288 с.
5. Керниган, Б. В. Язык программирования C / Б. В. Керниган, Д. Ритчи. – 2-е изд., перераб. и доп. – М. : Вильямс, 2009. – 304 с.
6. Котов, В. М. Структуры данных и алгоритмы. Теория и практика : учеб. пособие / В. М. Котов, Е. П. Соболевская. – Минск : БГУ, 2004. – 267 с.
7. Кнут, Д. Искусство программирования. В 3 т. / Д. Кнут. – М. : Вильямс, 2014. – Т. 1 – 145 с.; Т. 2 – 801 с.; Т. 3 – 140 с.

Св. план 2023, поз. 28

Учебное издание

Кириенко Наталья Алексеевна
Полоско Екатерина Ивановна
Ефремов Андрей Александрович

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В двух частях
Часть 1

ИСПОЛЬЗОВАНИЕ ЯЗЫКА СИ В ИНТЕГРИРОВАННОЙ СРЕДЕ РАЗРАБОТКИ VISUAL STUDIO

ПОСОБИЕ

Редактор *С. Г. Девдера*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лущикова*

Подписано в печать 08.04.2024. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 7,32. Уч.-изд. л. 8,0. Тираж 80 экз. Заказ 240.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск