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

Кафедра программного обеспечения информационных технологий

КОНСТРУИРОВАНИЕ ПРОГРАММ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие
для студентов специальности
«Программное обеспечение информационных технологий»
В 2-х частях

В.В. Бахтизин, И.М. Марина, Е.В. Шостак

Часть 1

ЯЗЫК СИ

Минск 2006

УДК 681.3.06(075.8)
ББК 32.973-018.1я73
К 64

Р е ц е н з е н т:
заведующий кафедрой информатики Минского государственного высшего радиотехнического колледжа,
кандидат технических наук, доцент Ю.А. Скудняков

К 64 **Конструирование** программ и языки программирования: Учебно-метод. пособие для студ. спец. «Программное обеспечение информационных технологий»: В 2 ч. Ч. 1: Язык Си / В.В. Бахтизин, И.М. Марина, Е.В. Шостак. – Мн.: БГУИР, 2006.– 48 с.
ISBN 985-488-023-0 (ч.1)

Пособие является дополнительным учебным материалом для студентов, изучающих программирование на языке Си. Приведены варианты заданий, позволяющие приобрести навыки в программировании различных задач.

УДК 681.3.06(075.8)
ББК 32.973-018.1я73

ISBN 985-488-023-0 (ч.1)
ISBN 985-488-024-9

СОДЕРЖАНИЕ

Общие теоретические положения

За последние годы язык Си стал наиболее популярным среди всех компьютерных языков программирования, и обусловлено это тремя вескими причинами: скоростью, переносимостью и структурированием. Многие инструкции Си адресованы непосредственно аппаратной части компьютера, поэтому программа на Си выполняется очень быстро. Фактически, она работает настолько быстро, что Си может быть использован для написания операционных систем, коммуникационных и инженерных приложений и даже компиляторов.

Алфавит и классификация данных языка Си

В языке Си используются следующие наборы символов

1. Строчные и прописные буквы латинского алфавита.
2. Арабские цифры от 0 до 9.
3. Специальные символы:

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

В языке применяются данные двух категорий: простые (скалярные) и сложные (составные) типы данных.

Основные типы простых данных: стандартный целый (int), вещественный с одинарной точностью (float) и символьный (char).

В свою очередь, данные целого типа могут быть короткими (short), длинными (long) и беззнаковыми (unsigned).

Целые и вещественные типы данных находятся в определенных числовых диапазонах и занимают разный объем оперативной памяти (в табл. 1 приведены диапазоны основных простых типов данных).

Таблица 1

Тип данных	Диапазон значений
Char	-128…127
Int	-32768…32767
Short	-32768…32767(-128…127)
Long	-2147483648…2147483647
Unsigned int	0…65535
Unsigned long	0…4294967295
Float	$3{,}14*10^{-38}$…$3{,}14*10^{38}$
Double	$1{,}7*10^{-308}$ $1{,}7*10^{308}$

Все объекты (переменные, массивы и т.д.), с которыми работает программа в Си, необходимо декларировать. При этом возможны две формы декларации: описание, не приводящее к выделению памяти, и определение, при использовании которого под объект в зависимости от его вида будет выделен определенный в соответствии с типом объем оперативной памяти.

Во втором случае при этом объект можно сразу проинициализировать, т.е. задать начальное значение.

Например: int к=50;

float a= –5.8;

При декларации объектов используются их идентификаторы.

Идентификаторы объектов программы на языке Си могут включать:

цифры 0...9;

латинские прописные и строчные буквы A...Z, a...z;

символ подчеркивания _.

Первый символ идентификатора не может быть цифрой. Длина идентификатора определяется реализацией транслятора Си и редактора связей (компоновщика). Современная тенденция – снятие ограничений длины идентификатора.

В языке Си буквы нижнего регистра отличаются от букв верхнего регистра и имена всех объектов программы по умолчанию принято писать (за редким исключением) символами нижнего регистра!

Разделители идентификаторов объектов:

пробелы;

символы табуляции, перевода строки и страницы;

комментарии.

Комментарий – любая последовательность символов, начинающаяся парой символов /* и заканчивающаяся парой символов */.

Программа, написанная на языке Си, состоит из одной или нескольких функций, причем одна функция обязательно имеет имя main() – основная, главная. Ее назначение – управление всей работой программы. Данная функция, как правило, не имеет параметров и не возвращает результат, но наличие круглых скобок (как и для других функций без параметров) обязательно.

Общая структура программы на языке Си

<директивы препроцессора>

<определение типов пользователя – typedef>

<прототипы функций>

<определение глобальных переменных>

<функции>

В свою очередь, функции имеют такую структуру:

<класс_памяти><тип><имя функции>(<объявление параметров>)

{ – начало функции

<локальные параметры>

<операции и операторы>
} – конец функции

Рассмотрим кратко основные части общей структуры программ.

Перед компиляцией программа на языке Си обрабатывается специальной программой – препроцессором, который работает под управлением директив. Директивы записываются по следующим правилам:

1) все препроцессорные директивы должны начинаться с символа #;

2) все директивы начинаются с первой позиции;

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

Препроцессор решает ряд задач по предварительной обработке программы, основной из которых является «подключение» к программе так называемых заголовочных файлов (обычных текстов) с декларацией стандартных библиотечных функций, которые используются в программе. Наименование такой директивы: #include (подключить), а общий формат ее использования

#include <файл 1.h>

…

#include <файл n.h>

где h – расширение заголовочных файлов.

Препроцессор подставляет на место этих директив тексты файлов: файл1…n, содержащих прототипы-объявления стандартных библиотечных функций и декларации некоторых дополнительных объектов соответствующей библиотеки.

Если имя файла заключено в угловые скобки < >, то поиск данного файла производится в стандартной директории с этими файлами, если же имя файла заключено в двойные кавычки ” ”, то поиск данного файла производится в текущем директории.

Например:

#include <stdio.h> – подключается файл с объявлениями стандартных функций файлового ввода-вывода;

#include <conio.h> – функции для работы с консолью;

#include <graphics.h> – содержит прототипы графических функций;

#include <math.h> – объявляет прототипы различных математических функций.

Второе основное назначение препроцессора – это обработка макроопределений, которые начинаются с #define (определить). Директива #define предписывает компилятору заменить имя константы на то, что следует за этим именем.

Макроподстановка #define имеет общий вид:

#define <идентификатор значение>

Например: #define PI 3.14

В ходе препроцессорной обработки программы появление в тексте идентификатора PI заменяется значением 3.14.

Далее идет заголовок функции и текст, который заключается в фигурные скобки.

Пример:

```
Заголовок    #include <stdio.h>     /*директива препроцессора*/
функции          void main()            /*имя функции с аргументами */

                 {
Тело               int num;                /*оператор описания*/
функции                num=25 ;             /*операция присваивания*/
                       printf("%d  это  число    \n",num);
                 }
```

Таким образом, отличительным признаком имени функции служат круглые скобки ( ), в которые заключается список аргументов. Аргументы могут отсутствовать. Здесь атрибут void – отсутствие значения (используется для нейтрализации значения, возвращаемого функцией, или для декларации указателей на область памяти без назначения типа объекта). А тело функции представляет собой набор операций и операторов, каждый из которых оканчивается символом «;» Функция printf – функция форматного вывода, %d указывает место и формат (%d – десятичное целое) вывода переменной num, \n – символ перевода строки, он служит для указания перехода на экране монитора к левому краю следующей строки.

Операторы ввода-вывода информации

Для вывода информации в языке Си используются функции: **printf**, **puts**.

Формат функции printf:

printf(управляющая строка , аргумент1, аргумент2,…);

Управляющая строка содержит объекты трех типов: *обычные символы*, которые просто выводятся на экран дисплея (копируются в стандартный выходной поток); *спецификации преобразования*, каждая из которых вызывает вывод на экран значения очередного аргумента из последующего списка, и *управляющие символьные константы*.

Каждая спецификация преобразования начинается со знака % и заканчивается некоторым символом, задающим преобразования. Между знаком % и символом преобразования могут встречаться: *знак минус*, указывающий, что преобразованный параметр должен быть выровнен влево в своем поле; *строка цифр*, задающая минимальный размер поля; *точка*, отделяющая размер поля от последующей строки цифр; *строка цифр*, задающая максимальное число символов, которые нужно вывести, или же количество цифр, которые нужно вывести справа от десятичной точки в значениях типов float или double; *символ длины* l, указывающий, что соответствующий аргумент имеет тип long.

Далее записывается один из следующих символов преобразования:

d – значением аргумента является *десятичное целое число* ;

c – значением аргумента является *символ;*

s – значением аргумента является *строка символов;*

e – значением аргумента является *вещественное десятичное число в экспоненциальной форме* ;

f – значением аргумента является *вещественное десятичное число с плавающей точкой;*

g – используется как %e или %f и исключает вывод незначащих нулей;

u – значением аргумента является *беззнаковое целое число;*

o – значением аргумента является *восьмеричное целое число;*

x – значением аргумента является *шестнадцатеричное целое число;*

p – значением аргумента является *указатель* (адрес).

Аргументами могут быть переменные, константы, выражения, вызовы функции; главное, чтобы их значения соответствовали заданной спецификации.

Среди управляющих символьных констант наиболее часто используются следующие:

\n – переход на новую строку; \t – горизонтальная \v – вертикальная табуляция; \b – возврат назад на один символ; \r – возврат в начало строки; \f – прогон бумаги до начала новой страницы; \a – звуковой сигнал; \ddd – 8-ричный ASCII-код; \xhhh – 16-ричный код; \? – знак вопроса.

Сокращение ASCII – Американский стандартный код обмена информацией.

Функция **puts** выводит на экран дисплея строку символов, автоматически добавляя к ней символ перехода на начало новой строки. Функция **putchar** выводит на экран дисплея один символ.

Функция **scanf** предназначена для ввода исходной информации.

Общий вид этой функции такой:

scanf (управляющая строка, адрес1, адрес2,…);

Для нее так же как и для printf(), указываются управляющая строка и следующий за ней список аргументов. Но если функция printf() использует имена переменных, константы и выражения, то scanf() использует только указатели на переменные, т.е. их адреса.

Таким образом, для ввода некоторого значения и присвоения его переменной одного из основных типов перед именем переменной требуется указать символ &, обозначающий адрес переменной (указатель на переменную).

Если же нужно ввести значение строковой переменной, то использовать символ & не нужно, так как строка – это массив символов, а имя массива эквивалентно адресу его нулевого элемента.

Функция scanf() использует практически тот же набор символов спецификации преобразования, что и функция printf().

Следует отметить, что функция scanf() вводит строки через формат %s только до первого пробела. Для ввода фраз, состоящих из слов, нужно использовать функцию **gets**(имя строковой переменной).

Пример:

```
#include <stdio.h>
  void main(void)
   {
      int a;  /* объявление целой переменной a */
 char b;  /* объявление символьной переменной b */
 float z; /* объявление переменной z с плавающей точкой одинарной точности */
 double d; /* объявление переменной d с плавающей точкой двойной точности */
      scanf("%d %c %f %lf", &a,&b,&z,&d); /* ввод с клавиатуры переменных a,b,z,d */
       printf("%d %c %9.5f %19.11f", a, b, z, d);  /* вывод на экран дисплея значений переменных a, b, z, d */
      getch();
      }
```

Функция **getch** вводит с клавиатуры единичный символ. Ее также можно использовать для приостановки выполнения программы.

Стандартные математические функции

Математические функции алгоритмического языка Си определены и размещены в файле <math.h>. В последующих записях аргументы x и y имеют тип double; параметр n имеет вид int. Тригонометрические функции имеют аргумент в радианах (2π радиан = 360°). Отметим, что все приведенные здесь математические функции возвращают значение (результат) типа double (табл.2).

Таблица 2

Математическая функция	Имя функции в языке С
1	2
$\sqrt{x}$	sqrt(x)
|x|	fabs(x)
e^x	exp(x)
x^y	pow(x,y)
ln(x)	log(x)
$lg_{10}(x)$	log10(x)
sin(x)	sin(x)
cos(x)	cos(x)
tg(x)	tan(x)
arcsin(x)	asin(x)
arccos(x)	acos(x)
arctg(x/y)	atan2(x)

Окончание табл. 2

1	2
$sh(x)=1/2\ (e^{x}-e^{-x})$	sinh(x)
$ch(x)=1/2\ (e^{x}+e^{-x})$	cosh(x)
tgh(x)	tanh(x)
Остаток от деления x на y	fmod(x,y)
Наименьшее целое, которое >=x	ceil(x)
Наибольшее целое, которое <=x	floor(x)

ЛАБОРАТОРНАЯ РАБОТА №1
Разработка программ с использованием условных и циклических операторов

Цель работы: изучить условные и циклические операторы.

Краткие теоретические сведения

Условные операторы

Разветвляющиеся алгоритмы рассматривают тот или иной участок программы в зависимости от выполнения или невыполнения какого-то логического условия, и предполагают передачу управления определенным операторам программы. Для этого в Си существуют операторы условного и безусловного перехода.

Общая форма оператора безусловного перехода:

goto <метка>

Согласно этой инструкции управление будет передано оператору с данной меткой, т.е. оператору:

<метка>: оператор;

Условный оператор **if – else** применяется в языке Си для выбора одной из ветвей вычислений. Общая форма записи:

if (условие) инструкция1; else инструкция 2;

Оператор 1 и оператор 2 – это простые или составные операторы. Если в ветви операторов два и более, то используются символы {} для их объединения. Оператор if проверяет истинность или ложность условия. Если условие в скобках принимает истинное значение, то выполняется инструкция 1, а если ложное – инструкция 2.

В операторе if слово else может отсутствовать. В этом случае, если условие в скобках принимает истинное значение, выполняется инструкция 1, а если ложное, инструкция 1 пропускается и управление передается на первый после if оператор.

В качестве условий в Си используют следующие операции отношений:

== – равно или эквивалентно;
!= – не равно,
< – меньше,
<= – меньше либо равно,
> – больше,
>= – больше либо равно,

а также используются и более сложные условия – выражения, входящие в логические операции.

Оператор **switch** – оператор выбора альтернатив (переключатель). Оператор проверяет, совпадает ли значение выражения с одним из значений, входящих в

некоторое множество целых констант, и выполняет соответствующую этому значению ветвь программы. Основная форма оператора switch имеет вид

```
switch (выражение)
  { case константа1: вариант 1; break;
                … … …
    case константа n: вариант n; break;
    default: вариант n+1 }
```

Вначале выполняется выражение в скобках за ключевым словом switch, и его значение сравнивается со всеми константами (константными выражениями). При совпадении выполняется соответствующий вариант (одна или несколько инструкций). Все константы в записи оператора должны быть различны. Вариант с ключевым словом default (прочие) реализуется, если ни один другой не подошел (слово default может и отсутствовать). Если default отсутствует, а все результаты сравнения отрицательны, то ни один вариант не выполняется. Для прекращения следующих проверок после успешного выбора некоторого варианта используется оператор break, обеспечивающий немедленный выход из переключателя switch.

Операторы цикла

Операторы циклов применяют, когда надо повторить некоторые действия (операторы и операции) несколько раз. Такие участки алгоритмов называются циклами. Язык Си имеет три структуры, известные под названием циклов, которые используются для управления повторами:

1) цикл **for**;
2) цикл **do...while**;
3) цикл **while**.

Оператор цикла for

Цикл **for**(для, в течение) используется в том случае, когда известно точное количество повторов, которое нужно выполнить.

Основная форма оператора цикла for имеет следующий вид:

for (выражение 1; выражение 2; выражение 3) тело цикла;

В выражениях 1, 2, 3 фигурирует специальная переменная, называемая *управляющей*. По ее значению устанавливается необходимость повторения цикла либо выхода из него.

Выражение 1 – присваивает начальное значение управляющей переменной, выражение 3 изменяет его на каждом шаге, а выражение 2 проверяет, не достигло ли оно граничного значения, устанавливающего необходимость выхода из цикла.

Цикл for можно использовать с целью создания задержки в программе:

```
for(delay=1; delay<=10000; delay++);
```

Выполнение повторов цикла приостановит переход программы к выполнению следующих операторов.

Пример оператора цикла for:

```
for (i=1;i<10;i++)  printf("\n %d",i );
```

В результате выполнения этого оператора печатаются в столбик цифры от 1 до 9.

В качестве параметра цикла необязательно использовать переменную целого(int) типа. Фрагмент программы, которая выводит на экран дисплея буквы латинского алфавита:

```
for(ch='a';ch<='z';ch++)
        printf("  %c",ch);
```

Необходимо тщательно контролировать структуру циклов for в программе, чтобы не получился бесконечный цикл (из которого нет выхода).

Отметим, что можно выйти из цикла досрочно по дополнительному условию и управляющему оператору break, который передаст управление на первый после цикла выполняемый оператор. Можно также досрочно завершить текущий циклический шаг при помощи дополнительного условия и управляющего оператора continue, который прервет выполнение и передаст управление на коррекцию переменной цикла с последующей проверкой условия на его продолжение.

Передавать управление извне внутрь цикла запрещается.

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

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

Циклы типа while и do – while

Цикл **while** (пока) используется в том случае, когда не известно точное число повторов и при этом нет необходимости, чтобы цикл непременно был выполнен хотя бы один раз. Структура цикла while такова:

while (выражение) тело цикла;

Выражение в скобках может принимать ненулевое (истинное) или нулевое (ложное) значение. Если значение истинно, то выполняется тело цикла и выражение вычисляется снова. Если значение ложно, то цикл while заканчивается.

Цикл **do – while** (делать-пока) используется в тех случаях, когда не известно точное количество повторов, но в то же время известно, что цикл необходимо выполнить по крайней мере один раз.

Основная форма оператора do – while следующая:

do тело цикла while (выражение);

Тело цикла будет выполняться до тех пор, пока выражение в скобках не примет ложное значение. Если оно ложно при входе в цикл, то его тело выполняется ровно один раз.

В циклах типа while и do – while также допустимы досрочный выход из цикла по дополнительному условию и оператору break (goto); а также досрочное завершение текущего шага цикла, но в последнем случае в отличие от цикла for управление будет передано на проверку условия.

В программе можно использовать любые комбинации вложенных циклов всех типов.

Варианты индивидуальных заданий

1. Даны действительные числа **x**, **y**. Вычислить **z**:

$$z = \begin{cases} x - y, \text{если } x > y, \\ y - x + 1 \text{ в противном случае.} \end{cases}$$

2. Даны два действительных числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.
3. Даны два действительных числа. Заменить первое число на ноль, если оно меньше или равно второму, и оставить числа без изменения в противном случае.
4. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3).
5. Даны три действительных числа. Возвести в квадрат те из них, значение которых неотрицательно.
6. Даны действительные числа **a**, **b**, **c** ($a \neq 0$). Выяснить, имеет ли уравнение $\mathbf{ax^2+bx+c=0}$ действительные корни. Если действительные корни имеются, то найти их. В противном случае ответом должно служить сообщение, что действительных корней нет.
7. Дано действительное число **x**. Вычислить $\boldsymbol{f}(\mathbf{x})$, если

$$f(x) = \begin{cases} x^2 \text{ при } -2 \le x < 2, \\ 4 \text{ в противном случае.} \end{cases}$$

8. Дано действительное число **x**. Вычислить $\boldsymbol{f}(\mathbf{x})$, если

$$f(x) = \begin{cases} x^2 + 4x + 5 \text{ при } x \le 2, \\ \dfrac{1}{x^2 + 4x + 5} \text{ в противном случае.} \end{cases}$$

9. Дано действительное число **x**. Вычислить $\boldsymbol{f}(\mathbf{x})$, если

$$f(x) = \begin{cases} 0 \text{ при } x \le 0, \\ x \text{ при } 0 < x \le 1, \\ x^4 \text{ в остальных случаях.} \end{cases}$$

10. Дано натуральное число $\boldsymbol{n}$. Вычислить 2^n.
11. Даны действительное число $\boldsymbol{a}$, натуральное число $\boldsymbol{n}$. Вычислить:

а) a^n;

б) $a(a+1)...(a+n-1)$.

12. Даны действительное число ***a,*** натуральное число ***n***. Вычислить:

а) $\frac{1}{a}+\frac{1}{a(a+1)}+...+\frac{1}{a(a+1)...(a+n)}$;

б) $\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+...+\frac{1}{a^{2n}}$.

13. Вычислить $(1+\sin 0.1)(1+\sin 0.2)...(1+\sin 10)$.

14. Дано действительное число ***x***. Вычислить

$x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}+\frac{x^{13}}{13!}$.

15. Даны целые числа ***n***, ***k***. Вычислить

$\frac{n(n-1)...(n-k+1)}{k!}$.

16. Вычислить:

а) $\sum_{i=1}^{100}\frac{1}{i^2}$;

17. Вычислить:

а) $\sum_{i=1}^{50}\frac{1}{i^3}$;

18. Дано натуральное число ***n***. Вычислить:

а) $\sum_{k=1}^{n}\frac{1}{k}$;

б) $\sum_{k=1}^{n}\frac{1}{(2k+1)^2}$.

19. Дано натуральное число ***n***. Вычислить:

а) $\sum_{k=1}^{n}\frac{1}{k^5}$;

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

20. Даны натуральные числа ***n***, a_1, … ,a_n. Определить количество членов последовательности a_1, … ,a_n:

а) являющихся нечетными числами;

б) кратных трем и не кратных пяти.

21. Даны натуральные числа n, a_1, … ,a_n. Определить количество членов последовательности a_1, … ,a_n:
а) являющихся квадратами четных чисел;
б) имеющих четные порядковые номера и являющихся нечетными числами.
22. Даны натуральные числа n, a_1, … ,a_n. Найти те члены последовательности a_1, … ,a_n, которые
а) являются удвоенными нечетными числами;
б) при делении на 7 дают остаток 1, 2 или 5.
23. Даны натуральное число n, целые числа a_1, … ,a_n. Получить сумму положительных и число отрицательных членов последовательности a_1, … ,a_n..
24. Даны натуральное число n, целые числа a_1, … ,a_n. Заменить все большие семи члены последовательности a_1, … ,a_n числом 7. Вычислить количество таких членов.
25. Вычислить $\sum_{i=1}^{30} (a_i - b_i)^2$, где

$$a_i = \begin{cases} i, \text{ если i - нечетное,} \\ \dfrac{i}{2} \text{ в противном случае,} \end{cases}$$

$$b_i = \begin{cases} i^2, \text{ если i - нечетное,} \\ i^3 \text{ в противном случае.} \end{cases}$$

26. Дано натуральное число n. Получить все его натуральные делители.
27. Дано натуральное число n. Получить все такие натуральные q, что n делится на q^2 и не делится на q^3.
28. Даны натуральные числа m, n, получить:

$$\frac{n!+m!}{(m+n)!}.$$

29. Дано 50 вещественных чисел. Найти порядковый номер того из них, которое наиболее близко к какому-либо целому числу.
30. Дано 100 вещественных чисел. Вычислить разность между максимальным и минимальным из них.

ЛАБОРАТОРНАЯ РАБОТА №2
Массивы в языке Си

Цель работы: изучить работу с массивами.

Краткие теоретические сведения

Массивы – это представители структурированных типов данных. Массив состоит из нескольких элементов одного и того же типа. Ко всему массиву целиком можно обращаться по имени. Кроме того, можно выбирать любой элемент массива. Для этого необходимо задать индекс, который указывает относительную позицию элемента. Число элементов массива задается при его объявлении и в дальнейшем не меняется. Нельзя задавать массив переменного размера. Если массив объявлен, к любому его элементу можно обратиться, указав имя массива и индекс элемента в квадратных скобках. **Нумерация элементов массива всегда начинается с нуля!**

Массивы объявляются так же, как и переменные. Например:

```
int mas[5];
char b[30];
float z[45];
```

В языке Си не проверяется выход индекса за пределы массива. Корректность использования индексов элементов массива должен контролировать программист.

Работа с массивами тесно связана с применением указателей.

Указатель – это переменная, которая содержит адрес некоторого объекта в памяти компьютера.

Указатель объявляется следующим образом:

тип *<имя переменной>;

Например:
```
int *a, b, c, *d;
float *f;
char *w;
```

Здесь объявлены указатели a, d, f, w и переменные b,c типа int.

С указателями связаны две унарные операции & и *.

Операция & означает «взять адрес» операнда. Операция * имеет смысл «значение, расположенное по указанному адресу».

В языке Си принято, что имя массива – это адрес памяти, начиная с которого расположен массив, т.е. адрес первого элемента массива.

Пусть объявлены массив из 5 вещественных элементов и указатель на вещественные переменные: float m[5], *m1;

Идентификатор m является указателем на начало массива (т.е. на первый элемент массива). Тогда операции присваивания m1=m; и m1=&m[0]; приведут к одному и тому же результату: указатель m1 будет установлен на начало массива m в памяти.

Для получения 4-го элемента массива можно написать m[3] или *(m1+3). Результат будет один и тот же. Если указатели адресуют элементы одного массива, то их можно сравнивать (можно использовать отношения вида:
<, >, = =, !=). В то же время нельзя сравнивать либо применять в арифметических операциях указатели на разные массивы. Любой адрес можно проверять на равенство или неравенство со значением NULL, которое записывается вместо нуля. Указатели на элементы одного массива можно также вычитать. Тогда результатом будет число элементов массива, расположенных между уменьшаемым и вычитаемым объектами. В языке Си допускаются *массивы указателей*, которые объявляются, например, следующим образом: char *m[5]; Здесь m[5] – массив, содержащий адреса элементов типа char.

Многомерные массивы

Двухмерный массив представляется как одномерный, элементы которого являются тоже массивами. Наиболее быстро изменяется последний индекс элементов массива, поскольку многомерные массивы в языке Си размещаются в памяти компьютера в последовательности столбцов.

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

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

Следующий пример иллюстрирует определение массива целого типа, состоящего из трех строк и четырех столбцов, с одновременным присвоением его элементам (инициализацией) начальных значений:

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

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

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

Отметим, что имя двухмерного массива – это указатель на массив указателей (переменная типа указатель на указатель: int **a), поэтому выражение b[r][c] эквивалентно выражению *(*(b+r)+c).

Варианты индивидуальных заданий

1. Дана целочисленная матрица размером 6*6 . Найти матрицу, получающуюся из данной:

а) перестановкой столбцов – первого с последним, второго с предпоследним и т.д.;

б) перестановкой строк – первой с последней, второй с предпоследней и т.д.

2. Дана действительная квадратная матрица порядка *n*, все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной

диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
3. В матрице 5*10 упорядочить элементы в каждой строке по убыванию, а строки матрицы расположить по возрастанию суммы элементов строк.
4. Дана действительная матрица размером 6*6. Найти среднее арифметическое наибольшего и наименьшего значений ее элементов.
5. В данной действительной квадратной матрице порядка 5*5 найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
6. В данной действительной матрице размером 6*6 поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны.
7. В данной квадратной целочисленной матрице 6*6 указать индексы всех элементов с наибольшим значением.
8. Дана целочисленная квадратная матрица 6*6. Найти номера строк:
а) все элементы которых – нули;
б) элементы в каждой из которых одинаковы;
в) все элементы которых четны.
9. Дана целочисленная квадратная матрица 5*5. Выяснить, имеются ли в матрице ненулевые элементы, и если имеются, то указать индексы:
а) одного из ненулевых элементов; б) всех нулевых элементов.
10. В матрице A(4,4) путем перестановки строк и столбцов поместить максимальный элемент на место элемента A(1,1).
11. Расположить элементы столбцов матрица A(4,5) в порядке возрастания, если номера столбцов четные, в порядке убывания, если нечетные.
12. Составить программу для определения координат и величины тех элементов матрицы A(10,10), которые одновременно являются максимальными в строке и минимальными в соответствующем столбце.
13. Ввести матрицу 5*5. Найти столбец с наименьшей суммой элементов и строку с наибольшей суммой элементов.
14. Ввести матрицу 6*6. Подсчитать количество четных и нечетных элементов. Умножить эту матрицу на вектор.
15. Ввести матрицу A. Среди элементов матрицы найти 5 нулевых. Построить матрицу B, удаляя столбцы с найденными нулевыми элементами.
16. В матрице A удалить строку с максимальным числом отрицательных элементов и столбец с максимальным числом положительных элементов. Из оставшихся строк и столбцов сформировать новую матрицу.
17. Ввести матрицу 6*6. Найти, сколько раз встречается каждый элемент.
18. Ввести матрицу 5*5. Упорядочить строки этой матрицы по возрастанию значений сумм их элементов.
19. Ввести матрицу 5*5. Упорядочить столбцы этой матрицы по убыванию сумм их элементов.

20. Из заданной матрицы сформировать новую, удалив строку и столбец, которым принадлежит минимальный (один любой из минимальных, если их несколько) элементов.
21. Дана матрица размером 6*6. Найти сумму наименьших элементов ее нечетных строк и наибольших элементов ее четных строк.
22. Составить программу обмена в матрице A(10,10) элементов, находящихся на главной диагонали, с соответствующими элементами первого столбца.
23. Дана действительная квадратная матрица порядка n. Найти наименьшее из значений элементов, расположенных на главной и побочной диагоналях.
24. Дана матрица размером 6*6. Среди элементов этой матрицы нет нулей и единиц. Заменить элементы, встречающиеся несколько раз, единицами, а неповторяющиеся – нулями. Подсчитать количество единиц и нулей в каждой строке и столбце.
25. Дана матрица размером 6*6. Найти сумму элементов, расположенных на главной и побочной диагоналях и их среднее арифметическое.
26. Дана матрица размером 6*6. В каждой ее строке переставить максимальный и минимальный элементы.
27. Расположить элементы матрицы A(5,5) так, чтобы на побочной диагонали были минимальные элементы столбцов.
28. Составить программу обмена в матрице A(10,10) элементов, находящихся на главной диагонали, с соответствующими элементами первого столбца.
29.Если ниже главной диагонали матрицы A нет ни одного отрицательного элемента, изменить матрицу A, умножив каждый ее элемент на находящийся с ним в одной строке элемент главной диагонали, иначе каждый элемент матрицы умножить на максимальный элемент соответствующей строки.
30. Если наибольший элемент матрицы A лежит выше главной диагонали, найти сумму элементов матрицы, лежащих выше главной диагонали. Иначе найти сумму элементов, лежащих ниже и на главной диагонали.

ЛАБОРАТОРНАЯ РАБОТА №3
Работа со строками

Цель работы: изучить особенности работы со строковыми объектами.

Краткие теоретические сведения

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

Например, описание: char a[7]; означает, что строка содержит шесть символов, а последний байт отведен под нулевой байт.

Строковая константа в языке Си – это набор символов, заключенных в двойные кавычки. Например: “Лабораторная работа”. В конце строковой константы явно указывать символ ´ \0 ´ не нужно, так как это сделает компилятор языка Си.

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

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

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

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

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

При работе со строковыми объектами допустима напрямую только одна операция: операция сцепления «+». Например:

```
char s1[]=”город”, s2[]=” Минск.”, s3[41];
s3=s1+s2; puts(s3);
```

Получим фразу: город Минск.

Остальные операции над строками выполняются с использованием стандартных функций. Функции для работы со строками размещены в библиотеке, описание которой находится в файле string.h. Вот некоторые из наиболее часто используемых:

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

3.Функция strcmp(S1, S2) сравнивает строки S1 и S2 и возвращает значение нуля, если строки равны, т.е. содержит одно и то же число одинаковых символов.

4. Функция strlen(S) возвращает длину строки, при этом завершающий нулевой байт не учитывается.

Пример работы со строками

В следующей программе значение строки формируется с клавиатуры, а затем введенная строка распечатывается в обратном порядке.

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(void)
{
  char s[100];   // объявление символьного массива
   int i, k;
    clrscr();    // очистка экрана
    puts(" Введите исходную строку");
     gets(s);
       k=strlen(s);
       puts(" Результат работы программы\n");
         for (i=k; i>=0; i--)
           printf("%c",s[i]); // вывод элементов массива в обратном порядке
  getch();
 }
```

Варианты индивидуальных заданий

1. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина четная, то удаляются 2 первых и 2 последних символа.

2. Выяснить, имеются ли среди символов некоторой строки все буквы, входящие в слово DOS.

3. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина нечетная, то удаляется символ, стоящий посередине строки.

4. Ввести с клавиатуры строку символов. Составить программу для замены в данной строке всех пробелов на символ $.

5. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина больше 10, то удаляются все цифры.

6. Проверить, является ли выражение, состоящее только из прописных букв заданной строки, палиндромом (т.е. читающееся слева направо и справа налево одинаково, например, «кабак»). Если да, то напечатать полученный палиндром. В противном случае вывести строку, состоящую из символов исходной строки с удаленными прописными символами.

7. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина кратна 4, то первая часть строки меняется местами со второй.

8. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина равна 10, то удаляются все строчные буквы.

9. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина больше 15, то удаляются все прописные буквы.

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

11. Ввести с клавиатуры строку символов. Программа должна определить длину введенной строки, и если длина больше 5, то выделяется подстрока до первого пробела.

12. С клавиатуры вводится предложение, слова в котором разделены символом подчеркивания. Напечатайте все предложения, которые получаются при перестановке слов исходного предложения.

Пример:

Исходное предложение: КОШКА_СЪЕЛА_МЫШКУ

Перестановки: СЪЕЛА_КОШКА_МЫШКУ

МЫШКУ_СЪЕЛА_КОШКА

КОШКА_МЫШКУ_СЪЕЛА

…

13. С клавиатуры вводится предложение, слова в котором разделены символом подчеркивания. Подсчитайте число вхождений в предложение используемых букв. Результат записать в строку (парами: буква-цифра) и напечатать ее.

Пример:

Введите предложение: КАРАБАС_БАРАБАС

Результат: А6Б3К1Р2С2

14. Текст состоит из слов разной длины. Определить, сколько раз в тексте встречается заданное слово.

15. С клавиатуры вводится текстовая информация (не более 100 символов). Составить программу подсчета слов и предложений в тексте, если слова разделяются пробелом, а предложения – точкой. Исходные данные и результат вывести на экран.

16. В тексте поменять слова местами (первое с последним, второе с предпоследним и т.д.). Слова имеют разную длину.

17. В заданном тексте удалить все части текста, заключенные в скобки (вместе со скобками).

18. Ввести последовательность чисел, состоящую из нулей и единиц. Требуется между всеми подряд стоящими единицами вставить нуль. На экран вывести исходную и полученную последовательности чисел.

19. Ввести текст длиной M символов (M<200). В заданном тексте везде убрать пару символов «AB», уплотнив при этом полученную информацию.

20. Вывести на экран самое длинное слово из заданного текста.

21. Имеется строка, содержащая буквы и цифры. Преобразуйте эту строку так, чтобы сначала в ней шли все буквы, встречающееся в исходной строке, но в обратном порядке, а потом – все цифры исходной строки в прямом порядке. Например,

Исходная строка	Результат
ad2e76b8	beda2768

22. Ввести строки длиной M и K символов (K< M). Удалить из первой строки все знаки, входящие во вторую строку, и сдвинуть все символы первой строки влево.

23. Даны переменные M, N и строки A и B. В строке A заменить символы, начиная с M-го, на первые N символов строки B.

24. С клавиатуры вводится строка. Выберите из нее все буквы от A(a) до I(i) (строчные преобразуйте в прописные) и отсортируйте их в алфавитном порядке. Например,

Исходная строка: SHiFROVkaOtSHPIonA

Результат: AAFHHII

25. С клавиатуры вводится строка. Выберите из нее все буквы от J(j) до S(s) (строчные преобразуйте в прописные) и отсортируйте их в алфавитном порядке. Например,

Исходная строка: SHiFROVkaOtSHPIonA

Результат: KNOOOPRSS

26. Пусть задано некоторое слово. Напечатать просмотр этого слова слева направо до тех пор, пока не встретятся повторяющиеся буквы. Если такие буквы встретились, пропустить их и продолжить просмотр с конца слова в обратном порядке (справа налево), пока снова не встретится набор повторяющихся букв. Если такой набор встретился, продолжить просмотр с того места, которое следует за первым набором повторяющихся букв и т.д. «Протокол» просмотра строки вывес-

ти на экран; вместо последовательности повторяющихся букв выводить один символ подчеркивания.

Пример:

Исходное слово: НОННИЛЛИОН

«Протокол» просмотра: НО_НОИ_И

27. Найти в исходной строке все вхождения (но не более девяти) заданной подстроки и заменить их на другую строку с указанием номера очередного вхождения. Допустимые символы – прописные русские и латинские буквы; символ-разделитель “_”.

Пример:

Исходная строка: ПОЛИЛИ_ЛИЛИЮ

Какую подстроку заменить: ЛИ

На какую подстроку заменить: СТО

Результат: ПОСТО1СТО2_СТО3СТО4Ю

28. Зашифруйте вводимое с клавиатуры предложение следующим образом: сначала выбираются два произвольных слова из базы, находящейся в тексте программы или вводимой с клавиатуры, затем слово из шифруемого предложения, потом опять два слова из базы, после чего – опять слово из предложения и т.д. “База” – набор слов, допустимых при выполнении программы, либо набор пар слов, как в приведенном ниже примере. Допустимые символы – прописные русские или латинские буквы; символ-разделитель “_”.

Пример:

Шифруемое предложение:

ДЕЛО_ЗАКОНЧЕНО_ХАДСОН_РАССКАЗАЛ_ВСЕ_БЕРЕГИТЕСЬ

База шифра:

С_ДИЧЬЮ_Я_ПОЛАГАЮ_ГЛАВА_ПРЕДПРИЯТИЯ_ПО_
СВЕДЕНИЯМ_О_МУХОБОЙКАХ_ФАЗАНЬИХ_КУРОЧЕК

Результат шифровки:

С_ДИЧЬЮ_ДЕЛО_Я_ПОЛАГАЮ_ЗАКОНЧЕНО_ГЛАВА_
ПРЕДПРИЯТИЯ_ХАДСОН_ПО_СВЕДЕНИЯМ_РАССКАЗАЛ_
О_МУХОБОЙКАХ_ВСЕ_ФАЗАНЬИХ_КУРОЧЕК_БЕРЕГИТЕСЬ

29. Имеется некоторая база данных, в которой некоторым английским словам поставлены в соответствие их русские эквиваленты, например, “THIS” – “ЭТО”, “IS” – “”, “A” – “”, “TABLE” – “СТОЛ” и т.д. С клавиатуры задается некоторое предложение на английском языке (слова разделяются символами подчеркивания). Программа ищет каждое слово у себя в базе данных и выдает его перевод. Если введенное слово в базе не обнаружено, в результирующую строку записывается исходное английское слово.

Пример выполнения программы:

Введите предложение на английском языке:

THIS_IS_A_TABLE

Перевод:

ЭТО_СТОЛ

30. С клавиатуры вводится предложение. Выведите это предложение на экран, расположив буквы “по вашей любимой функции” (экспонента, квадратный корень и т.п.).

Пример (выбрана синусоида):

```
   БАТЬ                ОЛЕБ                Л_ВЕ
  ЛЕ   СЯ              _К   АТ             ЩА  ЛИ
 О      .  ,           И      Ь            Е      К
 К          _          _      С            В      И
              К        Я        Я          А        Й
                ОЛЕ    ЬС         .        З          _С
                   БАТ                  _ТАК            ИНУС
```

ЛАБОРАТОРНАЯ РАБОТА №4
Работа со структурами

Цель работы: освоить приемы работы с составным типом данных – структурами.

Краткие теоретические сведения

Тип данных *структура* или *запись* объединяет несколько переменных, возможно, разного типа. Переменные, которые объединены структурой, называются полями структуры (или членами структуры). Пример описания структуры:

```
struct cd
        {
                char name [20];
                float cost;
                char category[12];
                int number;
        };
```

Определяя структуру, вы тем самым определяете собственный тип данных. Вы даете структуре имя и указываете компилятору имя и тип каждого элемента данных, которые должны содержаться в структуре. Список членов структуры носит название *шаблона.* Члены структуры сами по себе не являются переменными, они представляют собой компоненты одной или нескольких переменных. Шаблон определяет эти компоненты и говорит компилятору, сколько памяти следует зарезервировать для каждой структурной переменной. Для того чтобы получить доступ к картотеке cd, следует определить *структурную переменную*

```
struct  cd disc;
```

Если в программе нужно иметь несколько переменных одного структурного типа (например cd), можно определить их в одной инструкции:

```
struct  cd disc, cdrom;
```

Можно создать структуру и определить переменную в один прием.

```
    struct cd
        {
                char name [20];
                float cost;
                char category[12];
                int number;
        } disc;
```

Доступ к конкретному элементу структуры осуществляется с использованием операции «точка». Например,

```
gets(disc.name);
disc.cost=15;
```

Структуры, как и переменные других типов, могут объединяться в массивы структурных переменных. Для того чтобы объявить такой массив, надо задать шаблон структуры, а затем объявить массив.

Например: struct cd disc[20];

Для доступа к полю name 15-го элемента массива необходимо записать disc[14].name.

Если объявлены две переменные типа структуры с одним шаблоном, допустима операция присваивания.

Нельзя использовать операцию присваивания к переменным типа структуры, шаблоны которых описаны под разными номерами, пусть даже совсем идентично.

Переменные типа структуры могут быть глобальными, локальными, а также формальными. Можно создать указатель на структуру и передавать аргумент типа структуры по ссылке.

Варианты индивидуальных заданий

1. Дана ведомость абитуриентов. В каждой строке записана фамилия абитуриента, его постоянное место жительства, средний балл аттестата.
Распечатать в алфавитном порядке список абитуриентов, проживающих в Минске и определить их количество.
2. Дана ведомость абитуриентов. В каждой строке записана фамилия абитуриента, его постоянное место жительства, средний балл аттестата.
Определить средний балл аттестата по университету и распечатать список абитуриентов, средний балл которых не ниже среднего балла по университету. Список должен быть отсортирован по алфавиту.
3. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны его номер, тип самолета, пункт назначения, время вылета. Определить все номера рейсов, типы самолетов и времена их вылета для заданного пункта назначения.
4. У администратора железнодорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата выезда, конечный пункт назначения, время отправления, число свободных купейных мест, число свободных плацкартных мест. Выдать на печать информацию о поездах, следующих до Москвы.
5. Написать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать номер группы, фамилию студента, средний балл за последнюю сессию. Необходимо распечатать списки студентов по группам. В каждой группе фамилии студентов разместить в порядке убывания среднего балла.

6. Имеется список учета нуждающихся в улучшении жилищных условий. Каждая запись этого списка содержит фамилию, имя, отчество и дату постановки на учет. Список упорядочен по дате постановки на учет. В течение года выделяется 5 квартир. Вывести на экран весь список с указанием ожидаемого года получения квартиры.

7. В библиотеке имеется список книг. Каждая запись этого списка содержит фамилии авторов, название книг, год издания. Определить, имеются ли в данном списке книги, в названии которых встречается некоторое ключевое слово (например, "ПЭВМ"). Если имеются, то выдать на печать фамилии авторов, название и год издания всех таких книг.

8. Информация о сотрудниках предприятия содержит:

- Ф.И.О.;
- оклад;
- номер отдела.

Требуется по каждому отделу определить сотрудника, у которого максимальная зарплата.

9. Информация о сотрудниках предприятия содержит:

- Ф.И.О.;
- оклад;
- номер отдела.

Требуется по каждому отделу определить сумму зарплат.

10. Список товаров, имеющихся на складе, включает:

- наименование товара;
- количество единиц товара;
- дату поступления товара на склад.

Вывести в алфавитном порядке список товаров, хранящихся более одного месяца, стоимость которых превышает 100000 рублей.

11. Список товаров, имеющихся на складе, включает:

- наименование товара;
- количество единиц товара;
- дата поступления товара на склад.

Вывести в алфавитном порядке список товаров поступивших на склад в феврале месяце.

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

- Ф.И.О.;
- номер группы;
- средний балл;
- доход на одного члена семьи.

Вывести список студентов в порядке уменьшения дохода на одного члена семьи.

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

- номер рейса;
- пункт назначения;
- время отправления;
- время прибытия на конечный пункт.

Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше 21 часа.

14. На междугородной АТС информация о разговорах содержит:

- дату разговора;
- название города;
- время разговора;
- тариф.

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

15. На междугородной АТС информация о разговорах содержит:

- дату разговора;
- название города;
- время разговора;
- тариф.

Вывести общее время разговора и сумму за 30 марта.

16. Информация о сотрудниках фирмы включает:

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

Вывести размер заработной платы каждого сотрудника фирмы.

17. Информация об участниках спортивных соревнований содержит:

- название команды;
- Ф.И.О. игрока;
- возраст.

Вывести название и средний возраст самой молодой команды.

18. Информация об участниках спортивных соревнований содержит:

- Ф.И.О. игрока;
- название команды;
- рост;

Вывести название и средний рост самой рослой команды.

19. Информация об участниках спортивных соревнований содержит:

- Ф.И.О. игрока;
- название команды;
- вес.

Вывести название и средний вес самой легкой команды.

20. Для книг, хранящихся в библиотеке, задаются:
– автор;
– название;
– год издания.

Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после 2000 года.

21. Для книг, хранящихся в библиотеке, задаются:
– автор;
– название;
– год издания.

Вывести список книг заданного автора.

22. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают:
– наименование;
– количество;
– номер цеха.

Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию.

23. Информация о сотрудниках предприятия содержит:
– Ф.И.О.;
– номер отдела;
– стаж работы.

Вывести список сотрудников по отделам в порядке убывания стажа.

24. Для участия в конкурсе исполнителей необходимо заполнить анкету с данными:
– Ф.И.О.;
– год рождения
– класс музыкального инструмента (скрипка, фортепиано);
– занятое место.

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

25. Элементами структуры являются:
– фамилия студента;
– номер группы;
– место проживания (в общежитии или нет).

Вывести отсортированный по алфавиту список проживающих в общежитии студентов и их количество.

26. Информация об участниках спортивных соревнований между командами России и Беларуси содержит:
– страну участника;
– Ф.И.О.;
– занятое место.

За каждое место присуждается следующее количество очков:
1 место – 6 очков, 2 место – 5 очков, 3 место – 4 очка. Вывести списки спортсменов каждой команды, отсортировать их по алфавиту и подсчитать количество очков, набранное командами.

27. В магазине содержатся сведения об ассортименте игрушек. Структура записи:
   - название игрушки;
   - цена;
   - возрастные границы, например, 2 – 5, т.е. от 2 до 5 лет.

Вывести на печать название игрушек, которые подходят детям от 1 до 3 лет.

28. В магазине содержатся сведения об ассортименте игрушек. Структура записи:
   - название игрушки;
   - цена;
   - количество.

Вывести на печать стоимость самой дорогой игрушки и ее наименование.

29. В магазине содержатся сведения об ассортименте игрушек. Структура записи:
   - название игрушки;
   - цена;
   - количество.

Вывести на печать название игрушек, стоимость которых не превышает 10000 руб.

30. Информация о сотрудниках предприятия содержит:
   - Ф.И.О.;
   - номер отдела;
   - дату рождения.

Вычислить средний возраст сотрудников по отделам.

ЛАБОРАТОРНАЯ РАБОТА №5
Работа с файлами

Цель работы: изучить способы создания и работы с файлами.

Краткие теоретические сведения

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

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

В языке Си это делает функция fopen. Она связывает физический файл на носителе, например B:\LR5.CPP, с логическим именем в программе. Логическое имя – это указатель на файл, т.е. на область памяти, где храниться информация о файле. Указатели на файлы необходимо объявлять. Формат объявления такого указателя следующий:

FILE *указатель на файл;

Например:

```
FILE *f;
f=fopen ("B:\LR5.CPP", "w");
```

Символ "w" определяет право доступа к открываемому файлу. В данном случае открывается файл LR5.CPP на диске B:\ только для чтения.

В таблице приведены коды, устанавливающие режимы доступа к открываемым файлам.

Таблица 3

Символ	Описание
1	2
r	Файл открывается только для чтения. Если нужного файла на диске нет, то возникает ошибка.
w	Файл открывается только для записи. Если файла с заданным именем нет, то он будет создан, если же такой файл существует, то перед открытием прежняя информация уничтожается.
a	Файл открывается для записи в его конец новой информации. Если файл не существует, то он создается.
r+	Файл открывается для редактирования его данных. Возможны и запись, и чтение информации. Файл должен существовать.
w+	Файл открывается для чтения и записи. Если файл существует, то его содержимое теряется.

Окончание табл. 3

1	2
a+	Файл открывается для чтения и записи в конец файла. Если файл не существует, то он создается.
t	Файл открывается в текстовом режиме.
b	Файл открывается в бинарном режиме.

Бинарные файлы – это файлы, которые не имеют структуры текстовых файлов. Каждая программа для своих бинарных файлов определяет собственную структуру.

По умолчанию файл открывается в текстовом режиме.

После работы с файлом доступ к файлу необходимо закрыть. Закрывает файл в языке Си функция fclose.

Если требуется изменить режим доступа к файлу, то для этого сначала необходимо закрыть данный файл, а затем вновь его открыть, но с другими правами доступа. Для этого используют стандартную функцию freopen, описанную в stdio.h как FILE* freopen (char filename, chov*mode, FILE *stream). Эта функция сначала закрывает файл, связанный с потоком stream (как это делает функция fopen), а затем открывает файл с именем filename и правами доступа mode, записывая информацию об этом файле в переменную stream.

В языке Си имеется возможность работы с временными файлами, т.е. с такими, которые нужны только в процессе работы программы и которые надо удалить после выполнения части вычислений. В этом случае используют функцию tmpfile со следующим объявлением: FILE* tmpfile (void).

Функция tmpfile создает на диске временный файл с правами доступа «w+b» и возвращает указатель на управляющий блок по типу атрибута FILE. После завершения работы программы или после закрытия временного файла он автоматически удаляется из диска.

Все действия по чтению/записи данных в файл можно подразделить на три группы:

1. Операции посимвольного ввода-вывода,
2. Операции построчного ввода-вывода,
3. Операции ввода-вывода по блокам.

Наиболее часто используемые функции файловой системы Си

Таблица 4

Имя	Действие функции
1	2
fopen()	Открывает файл
fclose()	Закрывает файл
putc(), fputc()	Записывают символ в файл
getc(), fgetc()	Читают символ из файла

Окончание табл. 4

1	2
fgets()	Читает строку из файла
fputs()	Записывает строку в файл
fseek()	Устанавливает указатель текущей позиции на определенный байт файла
ftell()	Возвращает текущее значение указателя текущей позиции в файле
fprintf()	Форматированный вывод в файл
fscanf()	Форматированный ввод в файл
feof()	Возвращает значение «истина», если достигнут конец файла
rewind()	Устанавливает указатель текущей позиции в начало файла
remove()	Стирает файл
ferror()	Возвращает значение «истина», если произошла ошибка
fflush()	Дозапись потока в файл

Функции для работы с текстовыми файлами удобны для создания текстовых файлов, ведения файлов-протоколов и т.п. Но для создания 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 из файла в память соответственно. Такой способ обмена данными требует меньше времени.

Варианты индивидуальных заданий

1. Разработать программу, которая формирует файл F1, содержащий целые числа, и переписывает этот файл в другой файл - F2, помещая в него из F1 только положительные числа.
2. Разработать программу, переписывающую в текстовый файл T2 содержимое текстового файла T1, но без строк, содержащих числа.
3. Разработать программу сортировки (упорядочения по возрастанию значений элементов) файла, содержащего целые числа.
4. Разработать программу, которая формирует на основе текстового файла T1 файл T2, разбивая T1 на строки так, чтобы каждая строка оканчивалась либо точкой с запятой, либо содержала 30 любых литер, кроме точки с запятой. В качестве T1 можно использовать файл с исходным текстом разработанной программы.

5. Разработать программу формирования файла, содержащего сведения о студентах. Каждый элемент этого файла должен содержать следующие данные: номер группы; фамилию; год рождения; оценки за последнюю сессию. Предусмотреть возможность добавления, удаления и корректировки записей.
6. Разработать программу перекодировки текстового файла, заменив в нем заглавные буквы строчными.
7. Сформировать базу данных результатов экзамена студентов. Разработать программу, которая удаляет соответствующие записи из файла тех, чья оценка меньше 5 и помещает их в другой файл.
8. Разработать программу вывода на экран текстового файла. Вывод следует организовать таким образом, чтобы выполнялось выравнивание по правой границе путем вставки между словами необходимого количества пробелов. В качестве исходного файла можно использовать файл с исходным текстом разработанной программы.
9. Разработать программу слияния двух отсортированных по убыванию значений элементов файлов F1 и F2. Результатом слияния должен быть файл F3, элементы которого упорядочены по возрастанию.
10. Разработать программу, объединяющую несколько файлов, содержащих списки студенческих групп, в один результирующий файл. Запись производится в алфавитном порядке.
11. Имеется телефонный справочник. Каждая запись в нем состоит из фамилии и телефона. Список фамилий уже упорядочен по алфавиту. Организовать в файле поиск методом Хоора. Найти по фамилии телефон.
12. Написать программу для хранения информации об успеваемости студентов. Необходимо хранить номер группы, фамилию студента, оценки за последнюю сессию. Программа должна позволять корректировать указанную информацию и распечатывать списки студентов по группам с указанием среднего балла каждого за последнюю сессию.
13. Даны файлы F1 и F2. Переписать с сохранением порядка следования компоненты F1 в файл F2, а компоненты F2 – в файл F1.
14. Дан файл F. В файле F не менее двух компонент. Определить, являются ли два первых символа файла цифрами.
15. Дан файл F. Получить файл G, образованный из файла F заменой всех его прописных (больших) латинских букв одноименными строчными (малыми) русскими буквами.
16. Разработать программу, которая записывает в файл F3 сначала компонент файла F1, затем компонент файла F2 с сохранением порядка.
17. Разработать программу определения: совпадают ли компоненты файла F1 с компонентами файла F2. Если нет, то определить номера 1-го и 2-го компонента, в которых есть отличие.
18. Дан файл A. Записать в файл B компоненты файла A в обратном порядке.

19. Задан текстовый файл F1, состоящий из произвольной последовательности буквенных символов. Упорядочить символы в алфавитном порядке, при этом все повторяющиеся символы должны быть удалены, и переписать новый текст в файл F2.
20. Переписать компоненты файла F1 в файл F2, заменив при этом каждый восклицательный знак точкой, а каждое двоеточие – тремя точками.
21. Компоненты файла F1 – натуральные числа. Переписать в файл F2 все нечетные числа.
22. Составить программу записи в файл F1 всех чисел файла F2, кратных 5, а в файл F3 – всех отрицательных чисел, кратных 3.
23. Дан файл F1, который содержит данные о студентах групп (фамилия – имя – возраст). Создать файл F, который будет содержать данные о студентах, имеющих наибольший возраст.
24. Дан файл A, компоненты которого являются целыми числами. Записать в файл B все четные числа файла A, а в файл C – все нечетные. Порядок следования чисел сохраняется.
25. Дан файл A, компоненты которого являются целыми числами. Получить файл B, образованный из файла A исключением повторных вхождений одного и того же числа.
26. Дан текстовый файл A. Переписать в файл B все компоненты файла A с заменой в них символа 0 на символ 1 и наоборот.
27. Дан текстовый файл A. Исключить пробелы, стоящие в концах его строк. Результат поместить в файл A1.
28. Дан текстовый файл A. Переписать компоненты файла A в файл A1, вставляя в начало каждой строки по одному пробелу. Порядок компонент должен быть сохранен.
29. Дан файл A, компоненты которого являются целыми числами. Записать в файл A1 все повторяющиеся числа файла A, а в файл A2 – числа, которые встречаются один раз. Порядок следования чисел сохраняется.
30. Даны два файла. Составить программу слияния этих файлов в один по алгоритму – цифра вставляется после двух символов.

ЛАБОРАТОРНАЯ РАБОТА №6
Динамические структуры данных

Цель работы: освоить работу с динамическими структурами данных.

Краткие теоретические сведения

Наиболее простой способ объединить или связать некоторое множество элементов – это «вытянуть их в линию», организовать список. Списки являются чрезвычайно гибкой структурой: их легко сделать большими или меньшими; их элементы доступны для вставки или удаления в любой позиции списка. Списки можно объединять или разбивать на меньшие. Списки могут быть односвязными и двусвязными.

В односвязном списке каждый элемент информации содержит ссылку на следующий элемент списка.

Двусвязный список состоит из элементов данных, каждый из которых содержит ссылки как на следующий, так и предыдущий элементы. Наличие двух ссылок дает возможность перемещения по списку в обоих направлениях.

Стек – это специальный тип списка, в котором все вставки и удаление выполняются только на одном конце, называемом вершиной. Стеки так же иногда называют «магазинами», а в англоязычной литературе для обозначения стеков еще используется аббревиатура LIFO (last-in-first-out – последний вошел, а первый вышел). Интуитивными моделями стека могут служить колода карт на столе при игре в покер, книги, сложенные в стопку, и т.д. В таких моделях можно взять только верхний предмет, а добавить новый объект можно только положив его на верхний. При работе со стеками основными являются операции занесения и извлечения элемента из стека. Эти операции традиционно называются «затолкать в стек» (push) и «вытолкнуть из стека» (pop).

Другой специальный тип списка – это очередь. Очередью называют упорядоченный набор элементов, где элементы удаляются с одного его конца, который называется началом, а вставляются с другого, который называется концом очереди. Очереди так же называются списками типа FIFO (аббревиатура расшифровывается как first in first out: первым вошел - первым вышел). Очередь можно представить как последовательность книг стоящих на полке, так что доступ к ним возможен с обоих концов.

Операторы, выполняемые над очередями аналогичны операторам стеков. Существенное отличие состоит в том, что вставка новых элементов осуществляется в конец списка, а не в начало, как в стеках.

Варианты индивидуальных заданий

1. Разработать программу формирования стека, содержащего последовательность чисел и его преобразования в два новых стека: первый должен содержать только положительные числа, а второй – только отрицательные.
2. Разработать программу формирования стека, содержащего целые положительные числа, и его преобразования путем удаления из него всех четных чисел.
3. Разработать программу, определяющую симметричность произвольного текста любой длины. Текст должен оканчиваться точкой. Эту задачу рекомендуется решать с помощью двух стеков. В первый стек следует поместить весь текст, затем во второй перенести его половину так, чтобы последний символ текста находился на дне стека. Далее путем поэлементного сравнения стеков получить ответ на вопрос о симметричности текста.
4. Разработать программу формирования стека, куда помещается последовательность символов, вводимых с клавиатуры. Процесс ввода символов должен прекращаться, как только среди вводимых символов появляется точка. После этого программа должна реверсировать стек, т.е. после реверсирования вершина и дно стека меняются местами.
5. Разработать программу слияния двух стеков, содержащих возрастающую последовательность целых положительных чисел, в третий стек так, чтобы его элементы располагались также в порядке возрастания.
6. Разработать программу добавления к стеку, содержащему возрастающую последовательность целых положительных чисел, нового элемента так, чтобы порядок возрастания в стеке не изменялся.
7. Разработать программу формирования стека с последующим его преобразованием в двунаправленную очередь. Двунаправленная очередь является динамической структурой данных, каждый элемент которой хранит не одну ссылку (указатель на следующий элемент, как в стеке), а две. Из них одна указывает на предыдущий элемент, другая – на следующий элемент очереди.
8. Разработать программу, реализующую следующие функции: создать очередь, удалить элемент, добавить элемент, закольцевать очередь.
9. Разработать программу формирования стека, куда помещается последовательность символов в виде отдельных слов, вводимых с клавиатуры. Каждое слово, помещенное в стек, следует вывести на экран терминала; при этом порядок вывода символов в каждом слове должен быть обратным по сравнению с последовательностью их ввода.
10. Разработать программу, определяющую число вхождений элемента E в дерево T.
11. Разработать программу, подсчитывающую сумму элементов непустого дерева.

12. Разработать программу исключения из заданного кольца среднего элемента. Элементы исключать до тех пор, пока не останется один. Значение этого элемента вывести на экран.
13. Разработать программу исключения элемента с заданным признаком из очереди. Значение признака ввести с клавиатуры.
14. Разработать программу удаления из очереди каждого второго элемента, уплотнив очередь.
15. Инвертируйте заданную очередь, т.е. первый становится последним и т.д.
16. Разработать программу поиска узла с заданным ключом в бинарном дереве.
17. Разработать программу проверки, находится ли элемент с ключом «A» в поддереве с корнем в вершине «B». Значения «A» и «B» введите.
18. Разработать программу подсчета числа узлов заданного бинарного дерева.
19. Разработать программу подсчета числа терминальных (концевых) узлов заданного бинарного дерева.
20. Разработать программу подсчета числа неполных узлов заданного бинарного дерева.
21. Разработать программу удаления узла из заданного бинарного дерева. Ключ удаляемого узла необходимо ввести.
22. Разработать программу подсчета узлов бинарного дерева, имеющих только однонаправленные связи.
23. Разработать программу подсчета узлов, имеющих два направления связи в заданном бинарном дереве.
24. Разработать программу определения максимальной длины поддерева в заданном бинарном дереве.
25. Имеется произвольное количество лунок и в каждой лунке лежит шар черного или белого цвета (белых шаров на один больше). Одним ходом разрешается менять местами два любых шара. Переставить шары так, чтобы сначала шли белые шары, а за ними черные. Если общее число лунок N, то для решения задачи достаточно сделать не более N/2 ходов. Значение N ввести. При решении задачи использовать очереди с двумя ссылками.
26. Имеется произвольное количество лунок и в каждой лунке лежит шар черного или белого цвета (белых шаров на один больше). Одним ходом разрешается менять местами два любых шара. Переставить шары так, чтобы сначала шли белые шары, а за ними черные. Если общее число лунок N, то для решения задачи достаточно сделать не более N/2 ходов. Значение N ввести. При решении задачи использовать очереди с двумя ссылками .
27. Разработать программу, которая выводит на экран элементы из всех листьев дерева.
28. В каждой лунке лежит красный, белый или синий шар. Одним ходом разрешается менять местами два любых шара. Добиться того, чтобы все красные шары шли первыми, все синие – последними, а белые – посередине.

29. Разработать программу, которая находит в непустом дереве T длину (число ветвей) пути от корня до ближайшей вершины с элементом E.
30. Имеется N лунок, в которых расставлены L черных и S белых шаров. Поменять местами черные и белые шары. Черные шары можно передвигать только вправо, а белые – только влево. Шар передвигается в соседнюю с ним лунку (пустую) либо в пустую лунку, находящуюся непосредственно за ближайшим шаром. Значения N, L, S ввести (N=L+S+1).

ЛАБОРАТОРНАЯ РАБОТА №7
Работа с графикой

Цель работы: изучение графического режима.

Краткие теоретические сведения

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

За начало координат экрана дисплея в графическом режиме принимается верхний левый угол с координатами x=0 и y=0, где x – координата по горизонтали, y - координата по вертикали точки (пиксела).

Функции для подготовки графической системы

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

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

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

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

Для задания автоматического режима графики необходимо записать:

```
int g_diver=DETECT,g_mode;
```

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

Основные функции для получения изображения:

1) вычерчивание окружности: circle(x,y,r).

2) вычерчивание закрашенного прямоугольника: bar(x1,y1,x2,y2).

3) вычерчивание параллелепипеда: bar3d(x1,y1,x2,y2,глубина,p); p=0 или p=1– верхняя грань отображается (не отображается).

4) вычерчивание линии: line(x1,y1,x2,y2).

5) вычерчивание точки: putpixel(x,y,цвет).

6) вычерчивание прямоугольника: rectangle(x1,y1,x2,y2).

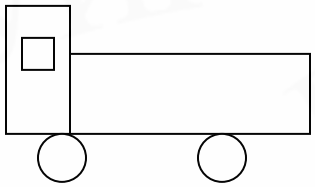
7) вывод текста: outtext(x,y,"текст").
8) установка указателя на экране дисплея: moveto(x,y).
9) очистка экрана дисплея: cleardevice(void).
10) заполнение ранее заданным наполнителем замкнутой области:
floodfill(x,y,c); c – номер цвета линии, ограничивающей область.

Основные функции для установки параметров изображения

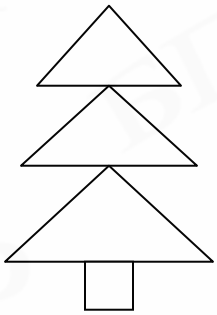
1) установка цвета линий: setcolor(n);
2) установка цвета фона: setbkcolor(n);
3) установка стиля наполнителя замкнутых линий: setfillstyle (номер наполнителя 0 – 12, цвет);
4) установка толщины линий: setlinestyle(стиль линии,0,толщина); 0 – непрерывная, 1 – из точки, 2,3 – штрих.

Варианты индивидуальных заданий

1. Построить и закрасить квадрат, прямоугольник, круг, треугольник.
2. Получить на экране и раскрасить рисунок.

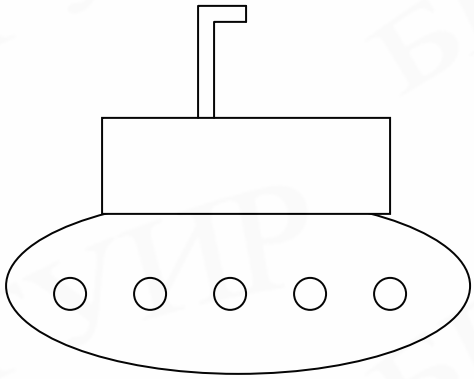


а

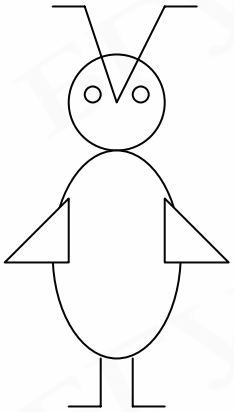


б

в



Д



Г

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

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

5. Составить программу вывода в верхней части экрана дисплея изображения облака. При запуске программы облако начинает двигаться и из него начинает идти дождь.

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

7. То же, что и в задании 6 , но из тучи идет снег и внизу растут сугробы.

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

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

10. То же, что в задании 9 , но между пушкой и мишенью есть стена высотой 0,5 экрана.

11. Составить программу вывода в верхней части экрана дисплея движущегося слева направо парусника с постоянной скоростью.

12 . Составить программу построения графика функции:

$y = x^3 + 2x^2 + x$

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

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

15. Составить программу построения графиков функций:

$y = -6x^2 + 3x$; $y = \mathrm{Sin}\ x$

16. Построить графики функций:

$$y = \frac{x}{x^2 - 3x + 1}; \qquad y = \mathrm{Tg}\ x$$

17. Составить программу вывода на экран дисплея изображения циферблата механических часов с секундной, минутной и часовой стрелками. Запуск часов осуществляется нажатием клавиши ENTER , при этом перемещается секундная стрелка.

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

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

20. Построить графики функций:

$$y = \frac{1}{x^2 + 2x + 1}; \qquad y = 3x^2$$

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

22. Составить программу вывода на экран дисплея схематичного изображения подводной лодки. При нажатии клавиши ENTER лодка начинает движение.

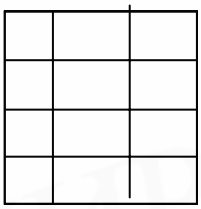
23. Составить программу вывода на экран дисплея схематичного изображения двух рыбок. По нажатии клавиши ENTER рыбки начинают движение навстречу друг другу.

24. Получить на экране схематичное изображение дома. По нажатии клавиши ENTER в окошке зажигается и гаснет свет. Окно дома при этом окрашивается в разные цвета.

25. Построить графики функций:

$y = x^5$; $y = \cos(x - 1) + |x|$

26. Построить изображение и раскрасить.



27. Составить программу для управления размерами окружности и ее положением на экране. Исходная окружность имеет центр в точке (100, 100) и радиус r = 20. Управление выполняется клавишами:

">" увеличивает радиус окружности;

"<" уменьшает радиус окружности;

клавиши управления курсором вызывают перемещение окружности в соответствующем направлении.

28.Составить программу для управления размерами прямоугольника и его положением на экране. Управление выполняется клавишами:

">" увеличивает ширину прямоугольника;

"<" уменьшает ширину прямоугольника;

"+" увеличивает высоту прямоугольника;

"–" уменьшает высоту прямоугольника;

клавиши управления курсором вызывают перемещение прямоугольника в соответствующем направлении.

29. Составить программу вывода на экран дисплея треугольника. При нажатии клавиши курсор вправо треугольник вращается по часовой стрелке.

30. То же, что в задании 26 , но добавить вращение против часовой стрелки при нажатии клавиши курсор влево.

ЛИТЕРАТУРА

1. Березин Б.И., Березин С.Б. Начальный курс С и С++.– М.: Диалог-МРТИ, 1999. –288 с.

2. Керниган Б., Ритчи Д. Язык программирования С. М.: Финансы и статистика,1992.–271 с.

3. Касаткин А.И., Вольвачев А.Н. Профессиональное программирование на языке Си : Ot Turbo –C k Borland C++: Справ. пособие. –Мн.:Выш. шк.,1992. – 240 с.

4. Фьюэр А. Задачи по языку СИ.– М.: Финансы и статистика,1985.

5. Юлин В.А., Булатова И.Р. Приглашение к СИ.– Мн.: Выш. шк., 1990.

6.Уингер Р. Язык Турбо СИ.– М.: Мир, 1991.

7. Подбельский В.В., Фомин С.С. Программирование на языке С–. М.: Финансы и статистика, 2002.

8. Бахтизин В.В., Глухова Л.А. Лабораторный практикум по курсам "Конструирование программ и языки программирования" и "Программирование" для студентов специальностей "Программное обеспечение ЭВМ и автоматизированных систем", "Вычислительные машины, комплексы, системы и сети". Ч.1. Конструирование программ с использованием процедур. - Мн.: Ротапринт МРТИ, 1993.

9. Бахтизин В.В., Глухова Л.А. Лабораторный практикум по курсам "Конструирование программ и языки программирования" и "Программирование" для студентов специальностей "Программное обеспечение ЭВМ и автоматизированных систем", "Вычислительные машины, комплексы, системы и сети". Ч.3. Конструирование программ с использованием функций. – Мн.: Ротапринт МРТИ, 1995.

Св.план 2006, поз.54

Учебное издание

Бахтизин Вячеслав Вениаминович
Марина Ирина Михайловна
Шостак Елена Викторовна

КОНСТРУИРОВАНИЕ ПРОГРАММ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие
для студентов специальности
«Программное обеспечение информационных технологий»
В 2-х частях

В.В. Бахтизин, И.М. Марина, Е.В. Шостак

Часть 1

ЯЗЫК СИ

Ответственный за выпуск Е.В. Шостак

Подписано в печать 12.07.2006. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,91.
Уч.-изд. л. 2,3. Тираж 100 экз. Заказ 5.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6