

Реализация основных алгоритмических  
структур на языке С.  
*(пособие для учащихся)*

Автор: Тюленев Евгений Анатольевич

Место работы: МБОУ "Боровихинская сош"

Должность: Учитель

## Оглавление

1 Введение.....	3
2 Выражение — алгоритм.....	3
3 Определение переменных. Операция присваивания.....	4
4 Функции. Главная функция. ....	6
5 Управляющие операторы.....	10
5.1 Логические выражения.....	10
5.2 Условный оператор.....	12
5.3 Оператор выбора.....	14
5.4 Циклы.....	16
5.4.1 Цикл с предусловием.....	16
5.4.2 Цикл с постусловием.....	17
6 Массивы.....	18
6.1 Заполнение массивов.....	19
6.2 Обработка массива.....	20
7 Файлы.....	22
8 Приложения.....	24
8.1 Необходимое программное обеспечение.....	24
9 Список литературы.....	25
Послесловие:.....	25

# 1 Введение.

*Исполнитель* — это объект способный выполнять определенный набор команд.

*Алгоритм* — это последовательность инструкций, выполнение которой ведет к решению задачи.

Требования к алгоритму:

1. *Понятность*: инструкции алгоритма должны быть понятны исполнителю.
2. *Замкнутость*: в результате выполнения алгоритма должна быть решена именно данная задача.
3. *Дискретность*: выполнение очередной инструкции не начинается, пока не закончится выполнение текущей.
4. *Конечность*: выполнение инструкций алгоритма должно когда-то закончиться.

## 2 Выражение — алгоритм.

Для того, чтобы вычислить значение выражения  $3(12-6)$  нужно действовать по алгоритму:

1. из 12 вычесть 6;
2. результат первого действия умножить на 3;
3. сообщить результат второго действия.

*Программирование* можно рассматривать как процесс составления выражений.

Требования к записи выражений:

1. расставлять знаки всех операций.
2. дробная черта заменяется делением.
3. у десятичной дроби целая от дробной части отделяется точкой.
4. если необходимо, то числитель и знаменатель заключается в скобки.

Обозначение арифметических операций:

+	сложение
-	вычитание
*	умножение
/	деление (для целых чисел - нацело)
%	нахождение остатка
++	увеличение на 1

--	уменьшение на 1
----	-----------------

Например: 
$$\frac{5,6 - 11(0,12 + 4)}{6,11 + 3,2^2}$$

на языке программирования записывается так:  $(5.6 - 11 * (0.12 + 4)) / (6.11 + 3.2 * 3.2)$ .

### Упражнения.

1. Запишите на языке программирования

а)  $56,5(11 - 5,8)$

в)  $(4,6 - 1)^4 - 87,3$

б)  $\frac{23 - 1,123(3,5 + 6)}{5^3 - 6}$

г)  $1,1 + \frac{89^2}{5,6} + 7$

2. Запишите на языке алгебраических формул:

а)  $567.5 * 4 - 5 * 5$ ;

в)  $(4 - 6.78) / (23 * 5)$ ;

б)  $33.7 * (23 - 4.5 * 4.5)$ ;

г)  $5.45 - 24 / 67 + 4.5$ .

3. Вычислите.

а)  $23/5 - 8 * 3$ ;

в)  $37\%5 - 2 + 6$ ;

б)  $6 - 78 / (4 + 2)$ ;

г)  $(56 - 23\%7) + 76/4$ ;

4. Составьте выражения, и запишите их на языке программирования:

а) Найти площадь и периметр прямоугольника если его длина — 5м, а ширина 20,45см.

б) Автомобиль движется со скоростью 5.4 м/с. Сколько времени ему понадобится, чтобы проехать 55 км.

в) Сторона куба 3,4см. Сколько метров составляет объем куба.

г) С полуночи прошло 5 ч 34 м 20 с. Сколько секунд осталось до полудня.

## 3 Определение переменных. Операция присваивания.

Для хранения информации используют переменные.

*Переменная* — это область памяти.

Прежде чем использовать переменную, ее нужно определить.

Общий вид определения переменной:

```
тип имя1;
```

Обозначения основных типов:

char — символ или код символа.

int — целое число;

long — длинное целое;

float — действительное число;

double — действительное число двойной точности.

Если нужно определить несколько переменных одного типа, то их имена\* перечисляют через запятую.

Примеры:

```
int a,b;//две переменных для целых чисел.  
float x,y,radius;//три переменных для действительных чисел.  
char asc,ch,c;//три переменных для символов.
```

Основной способ записи значения в переменную — *операция присваивания*.

Синтаксис операции присваивания:

```
имя = выражение;
```

Например:

```
result = 2 * ( 56.4 - 4 );//в result будет записано 104.8
```

Найти площадь круга  $S$  и длину окружности  $C$ , если радиус 4,67см.

Переменные  $S$  и  $C$  должны быть действительного типа. Расчетные формулы  $S = \pi \cdot r^2$ ;  $C = 2 \cdot \pi \cdot r$ , таким образом:

```
float s,c;  
s = 3.14 * 4.67 * 4.67;  
p = 2 * 3.14 * 4.67;
```

### Указатель на переменную.

Указатель на переменную содержит адрес этой переменной.

Определение указателя:

```
тип *имя;
```

Например:

```
int *pa;
```

После определения указателя память не выделяется. Память можно выделить вызвав функцию malloc,

Например:

```
pa = (int*)malloc(sizeof(int));
```

так же можно присвоить адрес уже существующей переменной:

```
int x;  
int *pa;  
pa = &x;
```

Для того чтобы получить или изменить значение переменной адрес которой хранит указатель нужно использовать операцию разыменования:

```
*pa = 25;
```

после использования память нужно освободить. для этого используется функция free

например:

```
free(pa);
```

Использование указателей позволяет более полно и гибко использовать память ЭВМ, а так же организовать

---

\* Имя переменной не может начинаться с цифры, в именах нельзя использовать пробел, знаки препинания, скобки и многие другие символы. Чтобы не ошибиться. используйте в именах только буквы латинского алфавита.

хранение информации в виде стеков, списков, бинарных деревьев..

### Упражнения.

5. Определите:

- а) две переменных для целых чисел;
- б) три переменных для действительных чисел и одну для символа;
- в) пять переменных для символов и две для действительных чисел;
- г) две для целых чисел, одну для действительного числа и три для символов.

6. Для хранения какой информации определены переменные:

- а) `int x,y; float a,b,c;`
- б) `char v,n,d; int d,b;`
- в) `float x,t,b; int l,m,n; char d,r,t,rr,yy;`
- г) `float x,y,z; int a,b,c,d; float f,l;`

7. Что будет записано в переменные, и какого типа они должны быть:

- а) `tt = 6 * 7 — 11;`
- б) `result = 5 % 2 + 7;`
- в) `y = 5,67 - 8/7 ;`
- г) `* r = 34 / 2 +5;`

8. Составьте выражение. ведущее к решению задачи, запишите фрагмент программы с определением необходимых переменных:

- а) Найти скорость **V**, если расстояние 2345 км, пройдено за 10 ч. (движение равномерное и прямолинейное);
- б) Определить сколько минут **ММ** содержится в 20 часах 125 секундах;
- в) Найдите остаток **r** и неполное частное **q** при делении числа 23465 на 234;
- г) За карандаш и тетрадь заплатили 15 рублей. Сколько стоит карандаш, если цена тетради составляет 70% стоимости покупки.

## 4 Функции. Главная функция.

Синтаксис определения функции:

```
тип имя(определение аргументов через запятую)
{
    //операторы
}
```

Например, можно определить функцию вычисления площади круга данного радиуса:

```
float sqrnd(float r)
{
    //посчитать значение выражения и вернуть результат
    return 2 * 3.14 * r * r;
}
```

После этого, в нужном месте достаточно написать, что-то вроде

```
sq = sqrnd(2.3);
```

и в переменную *sq* будет записана площадь круга радиуса 2,3.

Программистами всего Мира создано огромное количество функций, различных по назначению: от математических вычислений до вывода графики.

Эти функции объединены в библиотеки, которые можно подключить к своей программе.

Подключение библиотек:

```
#include <имя файла> // файл в стандартном каталоге  
#include "имя файла" // файл в каталоге с проектом
```

Особое место в программе занимает *главная функция*. Именно она вызывается первой.

Определение главной функции:

```
int main ( int argc , char **argv )  
{  
    //операторы;  
}
```

Теперь можно написать полноценную программу, вычисления площади круга:

```
float sqrnd(float r)  
{  
    return 2 * 3.14 * r * r;  
}  
int main ( int argc , char **argv )  
{  
    float sq;  
    sq = sqrnd(5.7);  
    return 0;  
}
```

Ее можно откомпилировать и выполнить. Беда в том, что значения площади мы не увидим, потому что, не дано указание вывести на экран значение переменной *sq*.

Выручает, то что еще создатель языка Деннис Ритчи определил функцию *printf*, которая позволяет выводить информацию на консоль. Эта функция помещена в библиотеку *stdio.h*.

Подключаем библиотеку, и вызываем функцию. Правда, прежде надо узнать каков

заголовок этой функции, чтобы правильно передать параметры.

Определение функции *printf*:

```
int printf(const char *_format, arg1, arg2, ...);
```

Первый параметр — строка формата — указывает на то *как выводить?*

Следующие параметры — это выражения значения которых нужно подставлять в строку формата вместо специальных символов — *символов формата*.

Основные символы формата:

```
%d — целое число  
%f — действительное число  
%s — строка символов  
%c — символ
```

Управляющие символы:

```
\n — начать новую строку  
\t — сместиться в следующую позицию табуляции.
```

пример вызова функции printf:

The diagram illustrates the argument passing in the `printf` function call: `printf ( "%d * %d = %d \n" , 2, 2, 2*2);`. Dashed arrows show the flow from the arguments to the format specifiers: the first `%d` is linked to the first `2`, the second `%d` to the second `2`, and the third `%d` to the expression `2*2`. A final dashed arrow points from the last argument to the end of the format string, indicating the termination of the argument list.

На консоль будет выведено

```
2 * 2 = 4
```

и начата новая строка.

Пример:

2987 секунд представить в виде hh часов mm минут ss секунд.

ясно, что  $hh = 2987 / 3600$ ;  $mm = (2987 - hh * 3600) / 60$ ;  $ss = 2987 - hh * 3600 - mm * 60$ ;

таким образом,

```
#include <stdio.h>  
int main( int argc, char **argv)
```



```

{
    int hh,mm,ss;
    hh= 2987 / 3600;
    mm = (2987 - hh * 3600) / 60;
    ss= 2987 - hh * 3600 - mm* 60;
    printf ( " %d час. %d мин. %d сек. \n", hh, mm, ss);
    return 0;
}

```

в результате выполнения на консоль будет выведено:

```
0 часов 49 минут 47 секунд
```

Существенный недостаток состоит в том, что нельзя после запуска программы ввести другое число. Проблема ввода данных в процессе выполнения программы решается использованием функции *scanf*.

Определена эта функция так:

```
int scanf(const char *_format, addr1, addr2, ...)
```

Первый параметр, строка формата, указывает на то как вводить информацию. Используются те же символы формата, что и у *printf*.

Затем, через запятую следуют адреса переменных. Операция получения адреса переменной обозначается знаком *&*.

Пример:

```
scanf("%d,%d",&a, &b);
```

Ожидается ввод двух целых чисел через запятую. Эти числа будут записаны в переменные *a* и *b* соответственно

Изменим программу примера так, чтобы можно было вводить число секунд после старта

```

#include <stdio.h>
int main( int argc, char **argv)
{
    int hh,mm,ss,dss;
    printf("Введите количество секунд:");
    scanf("%d",&dss);
    hh= dss / 3600;
    mm = (dss - hh * 3600) / 60;
    ss= dss - hh * 3600 - mm* 60;
}

```

```
printf ( " %d час. %d мин. %d сек. \n", hh, mm, ss);  
return 0;  
}
```

После запуска, организуется настоящий диалог:

Введите количество секунд:4987  
1 час. 23 мин. 7 сек.

---

### Упражнения.

Напишите программы решения задач, организовав ввод и вывод:

9. Вычислить значения выражений (вывод осуществить в виде формул):

а)  $5a + 4b$

в)  $4,6(x - y)^2 + 9$

б)  $x^3 - 4x + y$

г)  $\frac{4x - 7}{5}$

10. Найти площадь треугольника:

а) по стороне и соответствующей высоте;

в) Найти последнюю цифру целого числа.

б) по трем сторонам;

г) Найти гипотенузу зная катеты.

11. Найти количество теплоты необходимое для нагревания 5 литров воды от 20 до 80 градусов Цельсия.

## 5 Управляющие операторы.

Для того, чтобы указать исполнителю на необходимость принятия решения о выполнении какого-либо действия или повторения действия служат *управляющие операторы*.

К основным управляющим операторам относятся:

1. Условный оператор.
2. Оператор выбора.
3. Оператор цикла.

В управляющих операторах используют *логические выражения*.

### 5.1 Логические выражения.

*Логическое выражение* состоит из *высказываний* и логических операций над ними.

Любое элементарное высказывание или логическое выражение может принимать одно из двух значений: *истина* или *ложь*. С точки зрения программиста — 1 или 0 соответственно.

Самые распространенные высказывания - это *равенства и неравенства, их системы и совокупности*.

Например:

$$5 > 6, 7 < 90, 2 * 3 < 6, 45 = 45$$

$$\begin{cases} 5 > x \\ y \geq 12 \end{cases} \quad \begin{cases} 3x < 4 \\ m \leq 2 \end{cases}$$

Основные логические операции и их *таблицы истинности*.

Каждая строка *таблицы истинности* показывает результат логической операции при указанных значения *операндов*.

### 1. Операция "не"

A	!A
1	0
0	1

### 2. Операция "и"

A	B	A & B
0	0	0
1	0	0
0	1	0
1	1	1

### 3. Операция "или"

A	B	A    B
0	0	0
1	0	1
0	1	1
1	1	1

Примеры вычисления значений логических выражений:

$$!(4 > 5) = !0 = 1$$

$$!((45 > 6) \&\& (9 < 11)) = !(1 \&\& 1) = !1 = 0$$

Логические выражения с *переменными*.

Если в равенстве или неравенстве один из операндов — *переменная*, то значение выражения нельзя найти до тех пор, пока не задано значение этой *переменной*.

Например:

$$(x > 2) || (4 < x) = ?$$

$$\text{при } x=5, (5 > 2) || (4 < 5) = 1 || 1 = 1$$

### Контрольные вопросы.

1. Что такое логическое выражение?
2. Что, чаще всего, выступает в роли высказывания? Назовите основные логические операции.
3. Запишите, по памяти, таблицы истинности основных логических операций.

## Упражнения.

12. Запишите логические выражения на языке С:

a)  $\begin{cases} 5-x \geq 1 \\ x < 0 \end{cases}$

6)  $\begin{cases} 6x-1>7 \\ x<5 \end{cases}$

B)  $\begin{cases} 4,5a-1 \leq 4 \\ 6x > 11 \\ y < 56 \end{cases}$

$$\Gamma) \left\{ \begin{array}{l} x^2 - 1 \geq 1 \\ y > 2x \\ x < 2 \end{array} \right.$$

13. Найдите значение логического выражения, при данных значениях переменных.

- а)  $!(5x > 11)$ , при  $x = 1; 34$ ;  
 б)  $(y + x < 45) \&\& (y > x)$ , при  $x = 2; y = 17$ ;  
 в)  $!((d > 2) \&\& (d + 1 < 6))$ , при  $d = 1; 5; 0$ ;  
 г)  $(a + b + c \leq 0) \parallel !(a + b > c)$ , при  $a = -1; b = 2; -3$ .

14. Петя и Коля выше Васи, но Коля ниже Пети. Каково значение логического выражения:  $(K > V) \parallel (V > P)$ , где первой буквой имени обозначен рост.

## 5.2 Условный оператор.

Перед вычислением значения выражения  $\frac{1}{x}$  нужно убедиться в том, что значение  $x$  не равно 0, иначе вычислить нельзя. И таких ситуаций на практике очень много.

Чтобы указать исполнителю на необходимость проверки условия существует *условный оператор*.

### Синтаксис условного оператора:



Пример:

Найти значение выражения  $\frac{5a-b}{a-b}$ .

Очевидно, что значение выражения можно вычислить только при условии, что  $a-b \neq 0$ .

```

#include <stdio.h>
int main(int argc, char **argv)
{
    float a,b,res;
    printf("введите два числа через запятую:"); scanf("%f,%f",&a,&b);

    if ((a - b)!=0)
    {
        res = (5*a-b)/(a-b);
        printf("результат:%f \n",res);
    }
    else
        printf("Деление на 0!!!\n");

    return 0;
}

```

Бывает, что в качестве оператора1 или оператора2 необходимо выполнить несколько операторов, то они берутся в операторные скобки:

```

{ - скобка отрывается
} - скобка закрывается

```

Такая конструкция называется *блок*.

Если при неверном условии требуется просто перейти к выполнению операторов после условного, то else часть можно упустить. В этом случае имеет место *сокращенный условный оператор*.

Например, найти наименьшее среди трех чисел.

```

#include <stdio.h>
int main(int argc, char **argv)
{
    int a,b,c,min;
    printf("Введите три числа через запятую:");
    scanf("%d,%d,%d",&a,&b,&c);
    min=a;
    if (min>b) min=b;
    if(min>c) min = c;
    printf("min=%d",min);
    return 0;
}

```

Для организации вычислений полезно использовать библиотеку математических функций *math.h*:

Определение функции	Результат.
double sqrt(double _x)	$\sqrt{x}$
double sin(double _x)	sin(x)
double cos(double _x)	cos(x)
double tan(double _x)	tg(x)
double exp(double _x)	$e^x$
int abs(int _x)	x

Например, выражение  $\frac{\sin x}{\sqrt{x+4}}$  будет записано так: sin(x)/sqrt(x+4).

### Контрольные вопросы.

1. Запишите общий вид сокращенного и полного условного оператора.
2. Объясните: как компьютер выполняет условный оператор?
3. Перечислите основные математические функции.

### Упражнения.

15. Что будет храниться в переменной x после выполнения кода:

а) x=7; if(x>3) {x--;} else {x=x+1;}

б) x=34;x=x+5; if(x<=40) {x=0;} else {x++;}

в) x=11;y=5; if(x>2\*y) {x=x+y;} else {x=3;}

г) \* x=5; (if(x++>=6){x=0;}

16. Найти значение выражения:

а)  $\frac{r-m}{r+1}$

б)  $\sqrt{4x-5}$

в)  $\frac{2}{x} - \sqrt{x-1}$

г)  $r + \sqrt{\frac{4x-2}{x}}$

17. Найти наибольшее среди четырех чисел.

18. Определить принадлежит ли точка координатной прямой, заданному интервалу.

а) [a;b]

в) (a;b]

б) [a;b)

г) (a;b)

19. Делится ли данное число на 2?

20. Какое из трех данных чисел лежит между двумя другими.

21. Принадлежит ли точка (x;y) графику функции  $y=2x^2-5$

### 5.3 Оператор выбора

Иногда необходимо выполнять операторы в зависимости от нескольких различных значений выражения. В этом случае удобнее использовать *оператор выбора*.

Синтаксис оператора:

```
switch (выражение)
{
    зн1: оператор 1;
        break;
    зн2:оператор 2;
        break;
    ...
    знn:оператор n;
        break;
    default: оператор;
}
```

Пример: Вывести название целого числа из отрезка [1;5].

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int n;
    printf("введите целое число:"); scanf("%d",&n);
    switch(n)
    {
        1:printf("один\n");
            break;
        2:printf("два\n");
            break;
        3:printf("три\n");
            break;
        4:printf("четыре\n");
            break;
        5:printf("пять\n");
            break;
        default: printf("Не знакомое число!!!");
    }
}
```

```
    return 0;
}
```

## 5.4 Циклы.

*Цикл* — это управляющая конструкция, организующая повторение выполнения операторов, пока верно условие.

Различают циклы *с предусловием* и *постусловием*.

Чаще всего используют циклы с предусловием.

### 5.4.1 Цикл с предусловием.

Основной оператор:

```
while (условие) оператор;
```

Пример:

Вывести на консоль все числа последовательности Фибоначчи, не превосходящие заданное число.

Последовательность Фибоначчи:

1 1 2 3 5 8 13 ...

Начиная с третьего каждый элемент равен сумме двух предыдущих.

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int a,b,c,n;
    a=b=1;
    printf("Введите целое число:"); scanf("%d",&n);
    while ( b<n)
    {
        c=a+b;
        a=b;
        b=c;
    }
    printf("Искомое число:%d",a);
    return 0;
}
```

Оператор for:

```
for(инициализация; условия; изменение)оператор;
```



Пример:

Вывести на экран n первых неотрицательных чисел кратных трем.

Последовательность: 0 3 6 9 ...

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int i, n;
    printf("Введите n:"); scanf("%d", &n);
    for(i=0 ; i<n; i++) printf(" %d", 3*i);
    return 0;
}
```

---

### Упражнения.

22. Найти значение выражения

а)  $1*2*3...*n$ ;

в)  $x+2*x+3*x+...+n*x$ ;

б)  $1+2+3+...+n$ ;

г)  $\frac{1}{2x} + \frac{2}{3x} + \dots + \frac{n}{(n+1)x}$

23. Вывести на консоль первые n нечетных чисел.

24. Вывести на консоль значения функции из данного интервала, с указанным шагом:

а)  $y=3x-1$   $[1;5]$ , шаг=1

б)  $y=2\sin(x)$   $(-\pi; \pi)$ , шаг=0,1

25. Вывести все цифры целого числа.

26. Записать цифры числа в обратном порядке.

27. Найти сумму первых n элементов последовательности, если  $a_i = \begin{cases} 2i-1 & i \text{ четное} \\ i-2 & i \text{ нечетное} \end{cases}$

28. Для последовательности Фибоначчи найти

а) сумму элементов не превосходящих

в) сумму первых n элементов

данного целого положительного числа;

г) количество четных элементов меньших

б) первые n элементов;

данного целого положительного числа.

## 5.4.2 Цикл с постусловием.

Синтаксис оператора:

```
do оператор while(условие);
```

Пример: Добиться ввода четного числа

```
#include <stdio.h>
int main(int argc, char **argv)
{
```

```

int a;
do
{
    printf("Введите четное число!:" ); scanf("%d",&a);
}
while (a%2 !=0);
printf("Наконец-то!");
return 0;
}

```

### Упражнения.

29. Добиться ввода задуманного числа.
30. Трехзначное число — пароль. Добиться ввода пароля не более чем с 4 попыток.
31. \*Реализовать игру "Угадай число".

## 6 Массивы.

Для хранения многих однотипных данных удобно использовать составную переменную — массив.

*Массив* — это последовательность однотипных элементарных переменных.

Вся последовательность имеет имя, а конкретная элементарная переменная имеет номер.

В языке C нумерация элементов массива ведется с 0.

Пример:

массив **a[11]** из 11 целых чисел

0	1	2	3	4	5	6	7	8	9	10
345	12	56	98	11	5	98	4	11	8	6

a[7]==4

Определение массива

```
тип_элементов имя[количество элементов];
```

Например:

```

int a[50]; //массив из 50-ти целочисленных переменных
float x[10]; //массив из 10-ти действительных переменных

```

```
char s[300]; //массив из 300 символов (строка).
```

Виды задач на массивы:

1. Заполнение массива;
2. Обработка массива.

## 6.1 Заполнение массивов.

*По формуле n-го элемента.*

Заполнить массив из 12 целых чисел первыми четными. (формула четного числа  $2*n$ , где  $n$  — номер четного числа)

```
#include <stdio.h>
int main(int argc, char **argv)
{
    int a[12] , i;
    for(i=0;i<12;i++) a[i]=2*i;
    for(i=0;i<12;i++) printf("a[%d] = %d\n" , i , a[i]);
    return 0;
}
```

*По рекуррентному соотношению*

*Рекуррентное соотношение* — это зависимость очередного элемента от предыдущих.

Заполнить массив из 8 действительных чисел, если  $a[0]=1,1$   $a[i]=2*a[i-1]+1$

```
#include <stdio.h>
int main(int argc, char **argv)
{
    float a[8];
    int i;
    a[0]=1.1;
    for(i=2;i<8;i++) a[i]=2*a[i-1];
    for(i=0;i<8;i++) printf("a[%d] = %f\n" , i , a[i]);
    return 0;
}
```

*Из потока ввода.*

Пример

Заполнить массив `int a[5]` со стандартного потока ввода (клавиатура).

```
#include <stdio.h>

int main(int argc, char **argv)
{
    int a[5], i;
    for(i=0;i<5;i++) scanf("%d",&a[i]);
    for(i=0;i<5;i++) printf("a[%d] = %d\n" , i , a[i]);
    return 0;
}
```

*Случайными числами.*

Случайное число получается в результате функции `rand()` из библиотеки *stdlib.h*. Это число очень большое.

Чтобы получить число из отрезка `[a;b]` можно использовать формулу  $rand() \% (b-a) + a$ ;

Пример.

Заполнить массив `int a[10]` случайными числами из отрезка `[10;99]`

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

int main(int argc, char **argv)
{
    int a[10], i;
    for(i=0;i<10;i++) a[i] = rand()%(99-10)+10;
    for(i=0;i<10;i++) printf("a[%d] = %d\n" , i , a[i]);
    return 0;
}
```

---

### Упражнения.

32. Заполнить массив:

а) `int a[17]` числами кратными 5;

б) `int b[9]` первыми элементами последовательности Фибоначчи.

в) `float x[7]` с клавиатуры.

г) `int m[10]` случайными трехзначными числами.

33. Заполнить массив `int a[20]` элементами последовательности 
$$a_i = \begin{cases} i+1 & i < 3 \\ 2*a_{i-1} & i \geq 3 \end{cases}$$

34. Заполнить массив `int a[14]` элементами последовательности:

а) 1 0 1 0 1 0 ...

б) 1 2 3 1 2 3 1 2 3 ...

в) 1 2 3 2 3 4 3 4 5 ...

г) \* 1 0 1 1 0 1 1 1 ...

## 6.2 Обработка массива

*Поиск элементов.*

Пример

Найти минимальный элемент в массиве двузначных чисел `int a[12]`

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char argv)
{
    int a[12], i, min;
    for(i=0; i<12; i++) {a[i]=rand()%89+10; printf(" %d", a[i]);}
    min=a[0];
    for(i=1; i<12; i++) if ( a[i] < min ) min = a[i];
    printf("\n min=%d", min);
    return 0;
}
```

*Поиск и замена.*

Пример.

В массиве `int a[19]` трехзначных чисел заменить все четные числа нулями.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char argv)
{
    int a[19], i;
    for(i=0; i<19; i++) {a[i]=rand()%899+100; printf(" %d", a[i]);}
    min=a[0];
    for(i=1; i<19; i++) if ( a[i]%2 == 0 ) a[i] = 0;
    printf("\n");
    for(i=0; i<19; i++) printf(" %d", a[i]);
    return 0;
}
```

*Изменение порядка элементов.*

Пример.

Упорядочить массив целых неотрицательных чисел меньших 50 int a[10] в порядке возрастания.

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

int main(int argc, char **argv)
{
    int a[10], i, j, b;
    for(i=0; i<10; i++) {a[i]=rand()%50; printf(" %d", a[i]);}
    for(i=0; i<10; i++)
        for(j=9; j>i; j--)
            if ( a[j]<a[j-1] ) {b=a[j]; a[j]=a[j-1]; a[j-1]=b;}

    printf("\n");
    for(i=0; i<10; i++) printf(" %d", a[i]);
    return 0;
}
```

---

### Упражнения.

35. Дан массив двузначных чисел int a[10]. Найти

- а) Номера четных элементов;
- б) Элементы большие 50;
- в) Элементы, оканчивающиеся на 5.
- г) Элементы, сумма цифр которых больше 10.

36. Дан массив трехзначных чисел int a[15]. Определить какие из них не могут входить в ip адрес и заменить их нулями.

37. В массиве целых неотрицательных чисел меньших 20 int a[30] найти:

- а) минимальный.
- б) максимальный
- в) сумму минимального и максимального.
- г) \* первый элемент больший минимального.

## 7 Файлы.

*Файл* — это совокупность взаимосвязанных значений. (поток. Вспомните стандартный поток ввода или вывода).

Работа с файлом состоит из:

1. Открытия файла.
2. Чтение или запись информации.
3. Закрытие файла.

Для работы с файлом понадобится файловая переменная, в которой хранится служебная информация. Эта переменная будет использоваться в функциях работы с файлом.

Определение файловой переменной:

```
FILE *имя;
```

Например:

```
FILE *f1;
```

### Открытие файла:

```
имя_фп = fopen(имя файла, режим);
```

Основные режимы:

r - чтение

w - запись

rw — чтение и запись

Например:

```
f1 = fopen("data.txt", "r");
```

### Чтение и запись:

fprintf(FILE \*f, const char \*\_format, ...); //запись в файл.

fscanf(FILE \*f, const char \*\_format, ...); //чтение из файла.

Функции аналогичны функциям чтения и записи из стандартного потока.

### Заккрытие файла:

```
fclose(фп);
```

Примеры:

В файл data.txt записать 10 первых нечетных чисел.

```
#include <stdio.h>
int main(int argc, char **argv)
{
    FILE *fd;
    int i;
    fd = fopen("data.txt", "w");
    if (!fd) return 1; // проверка удачного открытия файла.
    for(i = 0 ; i < 10 ; i++) fprintf(fd, "%d ", 2*i+1);
    fclose(fd);
}
```

```
    printf("Данные записаны!");  
    return 0;  
}
```

Из файла data.txt прочитать 7 целых чисел, увеличить их на 5 и записать в файл data1.txt

```
#include <stdio.h>  
  
int main(int argc, char **argv)  
{  
    FILE *fd, *fd1;  
    int i, a;  
    fd = fopen("data.txt", "r");  
    fd1 = fopen("data1.txt", "w");  
    if (!fd || !fd1) return 1; // проверка открытия файлов.  
    for(i = 0 ; i < 10 ; i++)  
    {  
        fscanf(fd, "%d ", &a);  
        fprintf(fd1, "%d ", a+5);  
    }  
    fclose(fd);  
    fclose(fd1);  
    printf("Данные записаны!\n");  
    return 0;  
}
```

---

### Упражнения.

1. В файл data.txt активного каталога записать 20 случайных трехзначных чисел через пробел.
2. Найти наименьшее трехзначное число из файла data.txt
3. Числа из файла data.txt умножить на 4 и записать в файл data1.txt
4. Объединить файлы data.txt и data1.txt в один файл result.txt
5. Загрузить числа из result.txt в массив целых чисел. Полученный массив упорядочить и вывести в файл result.txt

## 8 Приложения.

### 8.1 Необходимое программное обеспечение.

Для работы можно использовать любую IDE поддерживающую язык программирования C.

Можно использовать [Code::Blocks](#) (отличная кроссплатформенная штука)



## 9 Список литературы.

1. Брайан Керниган, Деннис Ритчи. Язык программирования Си — Санкт-Петербург: Невский диалект, 2001. — 352 с.
2. Березин Б.И., Березин С.К. Вводный курс С и С++ - М: ДИАЛОГ-МИФИ, 1997. -208 с.
3. Кимел П. и др. Borland C++ 5: Пер с англ. -СПб.:БХВ-Петербург, 2001. 976 с.: ил.
4. Николенко Д.В. Самоучитель по Visual C++ 6 под ред. А.А. Малышенко — СПб.: Наука и техника. 2001. 363с.
5. Borland C++ Builder 6. Для профессионалов /В. А. Шамис — Спб.: Питер. 2003. 798 с. :ил.
6. Using the GNU Compiler Collection (For gcc version 4.6.1) <http://gcc.gnu.org/onlinedocs/>

## Послесловие:

Пособие предназначено для учащихся 9 класса общеобразовательной школы.

Теоретический материал изложен максимально кратко, в силу того, что читать более страницы при подготовке к занятию ни кто из девятиклассников, как показывает опыт, не собирается.

Практические задания подобраны так, что можно пункты *а*, *в* решать на уроке *б*, *г* на дом.

Уровень заданий рассчитан на степень физико-математической подготовки среднестатистического ученика 9 класса.

Буду признателен за дельные замечания. Можно считать, что версия пособия 1.0.

С уважением, Тюленев Е.А. mail-to: jhtulen@gmail.com