Общий синтаксис языка Си

Разделитель – это пробел, табулятор, перевод строки, перевод страницы. Вместо одного разделителя может использоваться любое их количество.

Идентификатор (имя) – это набор букв, цифр, символов подчеркивания, начинающийся не с цифры.
В качестве идентификатора могут использоваться любые имена, удовлетворяющие синтаксису Си, кроме ключевых слов. В имени допустимы буквы только латинского алфавита (a-z,A-Z), цифры (0-9) и символ подчеркивания (_).
Имя может быть любой длины, однако в рассматриваемой версии (2.01) различаются лишь первые 32 символа.

Примеры идентификаторов.
Правильные идентификаторы:

`kurs1  kurs_1  a12  _ab;`

неправильные идентификаторы:

`%ab` (% - недопустимый символ),

`12abc` (идентификатор начинается с цифры),

`A-1` (нельзя использовать знак -),

ИСТАС (нельзя использовать русские буквы).

Зарезервированные слова

int – целое
for – для
long – длинное
while – пока
short – короткое
do – выполнить
unsigned – без знака
signed – со знаком
break – завершить
continue – продолжить
char – символьное
float – с плавающей точкой
double – двойной точности
void – пустой
enum – перечислимый
typedef – определение типа
struct – структура
union – объединение
sizeof – размер
goto – перейти
if – если
else – иначе
switch – переключатель
case – вариант
default – по умолчанию
return – возврат
auto – автоматический
register – регистровый
static – статический
extern – внешний

Дополнительные зарезервированные слова для C++

asm	**catch**	**class**
delete	**friend**	**inline**
new	**operator**	**private**
protected	**public**	**template**
this	**throw**	**try**
	virtual	

Комментарии

1)

```
/* Короткий комментарий */
```

2)

```
/* Очень,
                очень,
                   очень длинный комментарий
   */
```

3)

```
// Комментарий в языке C++
```

Типы величин

- Числовые
  - Целые (short, int, long, unsigned)
  - С плавающей точкой (float, double)
- Текстовые
  - Одиночные символы (char)
  - Символьные строки (массив символов)
- Логические

  1-Истина (или все, что не 0)

  0-Ложь

Числовые константы

Целые числа:

1 23 –456

89L - для типа long

012 - восьмеричное число ($10_{10}=1*8^1+2*8^0$)

0x34A - шестнадцатеричное число ($842_{10}=3*16^2+4*16^1+10*16^0$)

Десятичные числа с плавающей точкой:

1.2 –3.45 -.67 (то же, что -0.67)

3e2 (т.е.$3*10^2$)

7.1e-3

6.34E-2

.21e+5

Числовой тип (Turbo C)

Тип	Размер в байтах	Диапазон значений
int	2	-32768 ... 32767 (2^{15}-1)
unsigned	2	0 … 65535 (2^{16}-1)
short	2	-32768 ... 32767
long	4	–2147483648 … 2147483647
unsigned long	4	0 … 4294967295 (2^{32}-1)
float	4	1.5e-45 …3.4e+38
double	8	5e-324 … 1.7e+308

Числовой тип (MSVC)

Тип	Размер в байтах *
short, unsigned short	2
int, unsigned int	4
long, unsigned long	4
long long	8 (**9223372036854775807** / **19** зн.)
float	4
double	8

* — зависит от версии языка (значения в таблице — для MSVS C++)

Символьные константы

Символьная константа – это один символ, заключенный в одинарные кавычки. Например, 'a' '*' 'Э' '>'.

Некоторые специальные символы языка Си:

\n перевод на новую строку. \t табуляция.

\0 код ASCII равный 0 \\ обратная косая черта.

\' одинарная кавычка. \" двойная кавычка.

\(открывающая скобка. \) закрывающая скобка.

\r возврат каретки

Символьная восьмеричная константа записывается в виде '\ddd' , где ddd – от одной до трех восьмеричных цифр, например, '\007' (звонок).

Примеры символьных строк

" ИСТАС "

что равносильно {’И’,’С’,’Т’,’А’,’С’,’\0’}

или

{’И’,’С’,’Т’,’А’,’С’,0}

""

пустая строка

Длинные символьные строки

1)

“Очень,\

очень,\

очень длинная строка! “

2)

”Очень, ”

“очень, ”

“очень длинная строка!”

Пример описания переменных стандартных типов:

```
int i,j,n=3,m=4;
int x[10],y[5]={0,1,2,3,4};
int z[]={1,2,3};
int a[2][3]={{1,2,3},{10,20,30}};
int b[2][3]={1,2,3,10,20,30};
float t;
double r=1e25;
char c,s[80];
char s1[]=”МГСУ”;
char s2[]={’М’,’Г’,’С’,’У’,’\0’};
```

Операции (15 рангов)

1. Слева направо

( ) – **вызов функции;**

[ ] – **доступ к элементу массива;**

. – **доступ к элементу структуры или объединения;**

–> – **доступ к элементу структуры или объединения, которые описаны с помощью указателя;**

Операции (2 ранг из 15) справа налево

++ -- – унарные инкремент и декремент;

sizeof – размер;

(новый_тип) – явное преобразование типа;

~ – поразрядное (побитовое) логическое отрицание;

! – логическое отрицание;

– – унарный минус;

& – унарная операция получения адреса переменной;

*** – унарная операция получения значения по адресу;**

3 ранг из 15 слева направо

*** / %**

– операции умножения, деления, нахождения остатка (при делении целых чисел);

4 ранг из 15 слева направо

+ -

– операции сложения и вычитания;

5 ранг из 15 слева направо

<< >>

– сдвиги влево и вправо;

6 ранг из 15 слева направо

< <= > >=

– операции сравнения;

7 ранг из 15 слева направо

== !=

– операции проверки равенства и неравенства ;

8 ранг из 15 слева направо

&

– побитовая операция И;

9 ранг из 15 слева направо

^

– побитовая операция исключающее ИЛИ;

10 ранг из 15 слева направо

|

– побитовая операция ИЛИ;

11 ранг из 15 слева направо

&&

– логическая операция И;

12 ранг из 15 слева направо

||

– логическая операция ИЛИ;

13 ранг из 15 **справа налево**

?:

– условная операция;

14 ранг из 15 справа налево

= += -= *= /= %=

<<= >>= &= ^= |=

– присваивание;

15 ранг из 15 слева направо

,

– операция запятая;

Таблицы логических операций.

~	0	1
	1	0

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

^	0	1
0	0	1
1	1	0

Примеры:

```
int i,j,x,y;
i=1; j=1;
x=i++;
y=++j;
========= результат ==============
i=2     j=2     x=1     y=2
```

Примеры:

```
int i,j,k,m,n;
float x,y;
i=5/2; j=5./2; m=5%2; k=1;
x=5/2; y=5/2.
k+=2; n=k; n*=5;
======== результат ============
i=2 j=2 x=2.0000000 y=2.5000000
m=1 k=3 n=15
```

Примеры:

```
int x,y,*p,*q;
x=5; p=&x, q=&y;
y=*p;
*p=7;
======== результат ==============
Объект Значение Адрес
  x        7     FF00
  y        5     FF02
  p     FF00
  q     FF02
```

Примеры:

```
int a,b,c,d,f1,f2,f3,f3,f4;
f1=5;      /* пусть   f1=0000000000000101(2)  = 5(10)    */
f2=3;      /* пусть   f2=0000000000000011(2)  = 3(10)    */
f3=f1&f2;  /* тогда   f3=0000000000000001(2)  = 1(10)    */
f4=f1>>1;  /* тогда   f4=0000000000000010(2)  = 2(10)    */
f5=f2<<2;  /* тогда   f5=0000000000001100(2)  = 12(10)   */
f1=f2^f3;  /* тогда   f1=0000000000000010(2)  = 2(10)    */
f2=f1|f3;  /* тогда   f2=0000000000000011(2)  = 3(10)    */
f2=~f2;    /* тогда   f2=1111111111111100(2)  = 65532(10)
   */

a=2; b=5; c= a & b; d= a && b;
===== результат ==============
      c=0   d=1
```

Пример 1:

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

Пример 2:

```
z = (x>y) ? x : y;
```

Пример 3:

Проверить условие: **x ∈ (-1.5 , 1.5) ∪ [5,10)**

```
float x; ...
if (fabs(x)<1.5 || x>=5 && x<10) ...
```

Упражнение 1.

```
int x,y,z,u,v,w;
x=1; y=2; z=0;
if(x==y) u=1; else u=0;
if(x=y)  v=1; else v=0;
if(x=z)  w=1; else w=0;
================================
u=?      v=?       w=?
================================
```

Упражнение 2.

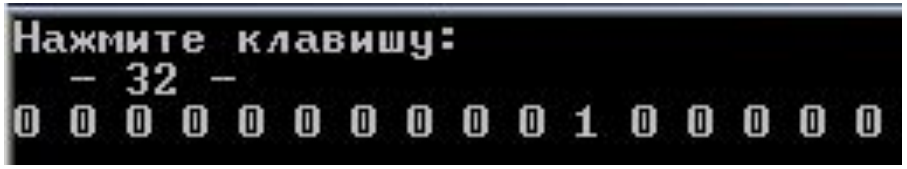
Верны ли равенства:

а) a>>4=a*16

б) a<<2=a/4

Пример

```
main()
{
int i,z[16];
char c;
unsigned u=1;
clrscr();
printf("Нажмите клавишу:");
c=getch();
for(i=15;i>=0;i--)
   {
   z[i] = u & c ? 1 : 0;
   u<<=1;
   }
printf("\n%c - %d - \n",c,c);
for(i=0;i<16;i++)
   printf("%d ",z[i]);
getch();
}
```



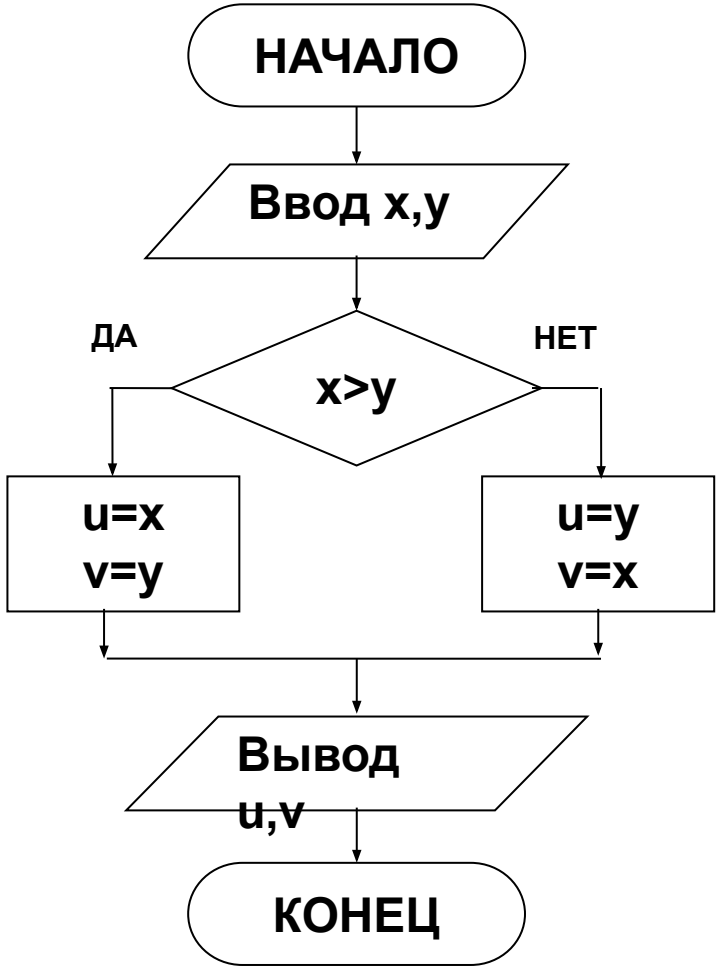
Оператор:

{ *составной_оператор* **}**
выражение **;**
if (*выражение***)** *оператор*
if (*выражение***)** *оператор* **else** *оператор*
while (*выражение***)** *оператор*
do *оператор* **while (***выражение***) ;**
for (*выражение1* **;** *выражение2* **;** *выражение3* **)** *оператор*

switch(*выражение* **){**
case *константное_выражение* **:** *оператор*
…
default : *оператор*
}
break;
continue;
return;
return *выражение* **;**
goto *идентификатор* **;**
идентификатор **:** *оператор*
; **(**пустой оператор**)**

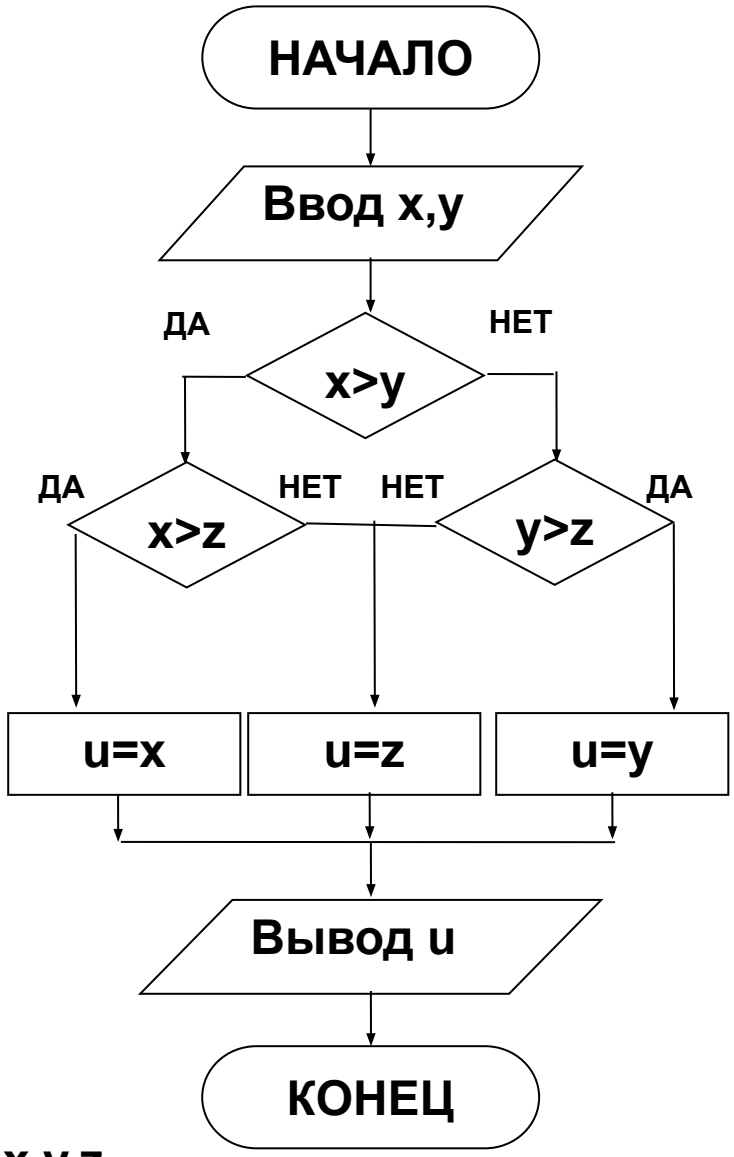
Пример: найти **u**=max{**x**,**y**}, **v**=min{**x**,**y**)

```
#include <stdio.h>
#include <conio.h>
main()
{
int x,y,u,v;
clrscr();
printf("Введите x,y:");
scanf("%d%d",&x,&y);
if(x>y)
    {
    u=x; v=y;
    }
else
    {
    u=y; v=x;
    }
printf("\n u=%d \n v=%d",u,v);
getch();
}
```



Пример: найти **u**=max{**x**,**y**,**z**}

```
#include <stdio.h>
#include <conio.h>
main()
{
int x,y,z,u;
clrscr();
printf("Введите x,y,z:");
scanf("%d%d%d",&x,&y,&z);
if(x>y)
   if(x>z)
    u=x;
   else
    u=z;
else
   if(y>z)
    u=y;
   else
    u=z;
printf("\n u=%d",u);
getch();
}
```

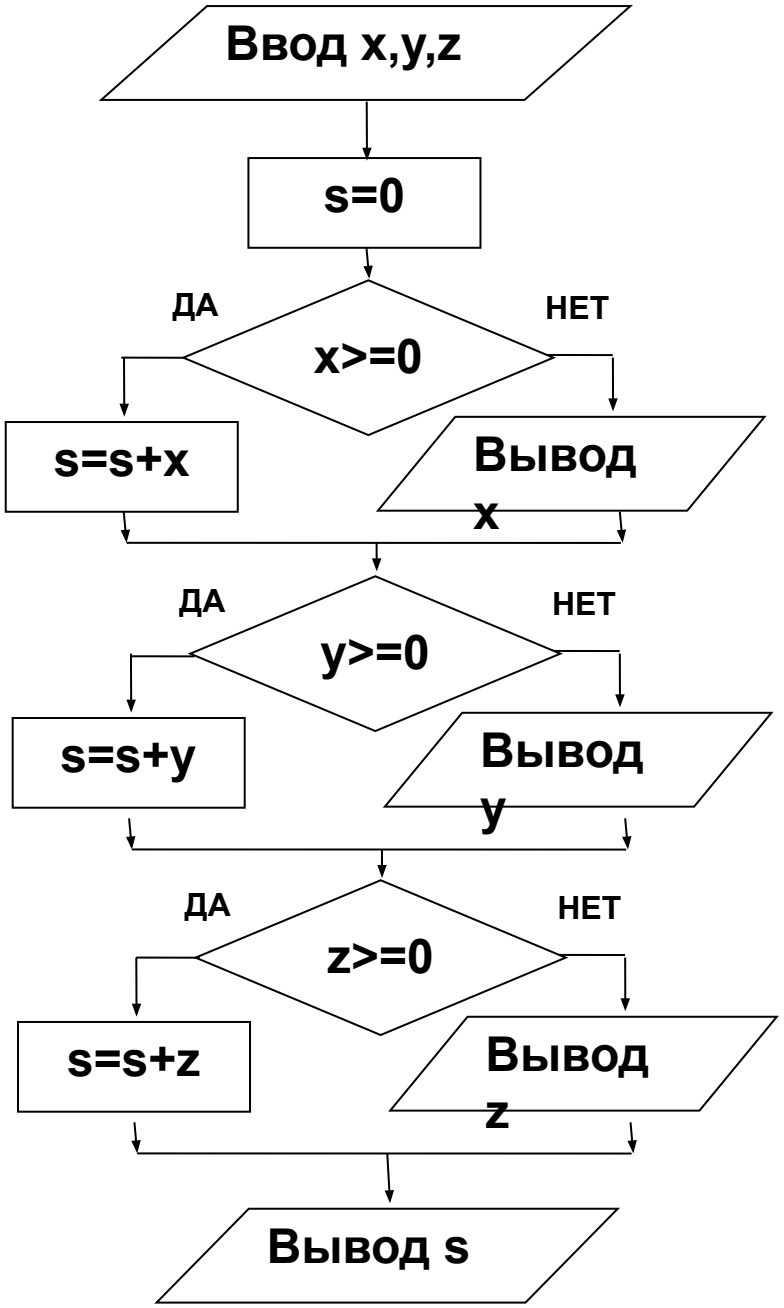


Упражнение:

Найти **u** как сумму двух наибольших среди **x,y,z**.

Пример: найти сумму положительных, отрицательные распечатать

```
#include <stdio.h>
#include <conio.h>
main()
{
int x,y,z,s;
clrscr();
printf("Введите x,y,z:");
scanf("%d%d%d",&x,&y,&z);
s=0;
if(x>=0)
   s+=x;
else
   printf("\n x=%d",x);
if(y>=0)
   s+=y;
else
   printf("\n y=%d",y);
if(z>=0)
   s+=z;
else
   printf("\n z=%d",z);
printf("\n s=%d",s);
getch();
}
```



Форматы:

Формат	Спецификация
%d	для целых десятичных чисел
%ld	для длинных целых десятичных чисел (long)
%lld	для **long long** (только для MSVS C++)
%u	для целых чисел без знака
%lu	для длинных целых чисел без знака (long)
%llu	для **unsigned long long** (только для MSVS C++)
%f	для чисел с плавающей точкой (float) разрешается вывод для типа double
%lf	для чисел с плавающей точкой (double)
%e	для чисел в экспоненциальной форме
%o	для целых восьмеричных чисел
%x	для целых шестнадцатеричных чисел
%c	для символов
%s	для строк
%p	для указателей

Форматы

Общий вид для вывода:

% [-][+] [ширина][.точность] {символ_преобразования}

[-] выравнивание влево,

[+] печать знака плюс

Точность:

для f - дробная часть,

для e - число значащих цифр

для s – число выводимых символов

Общий вид для ввода:

% [*] [ширина] {символ_преобразования}

[*] – пропуск поля во входном потоке

Примеры:

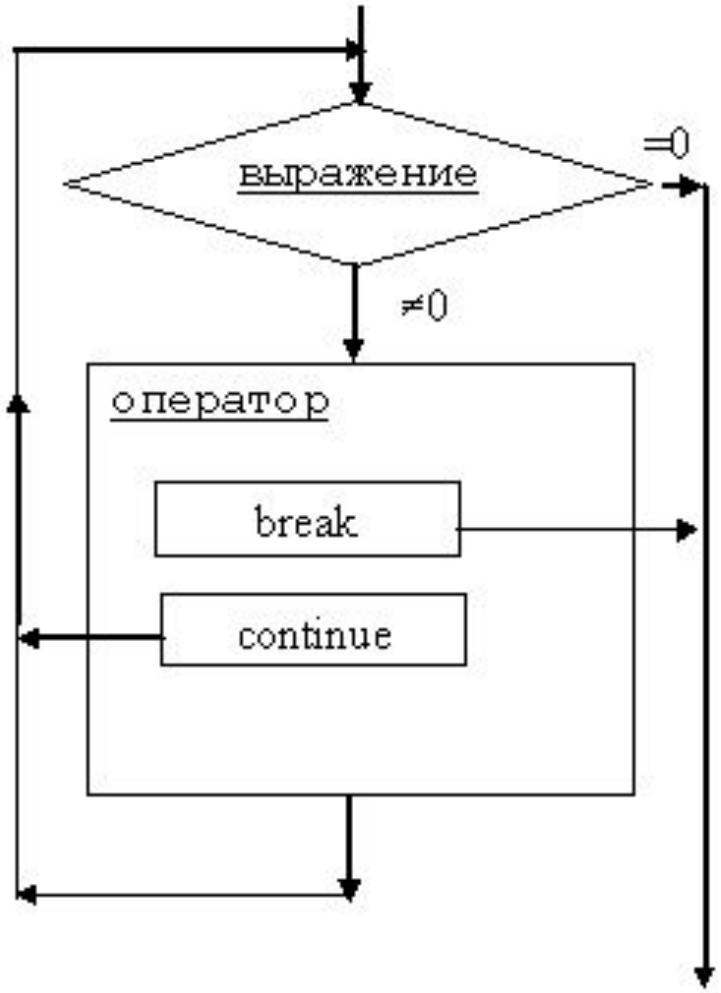
```
main()
{
int x;
double y;
clrscr();
x=15;y=12.345;
printf("\n%d",x);
printf("\n%5d",x);
printf("\n%-5d",x);
printf("\n%f",y);
printf("\n%10.2f",y);
printf("\n%1.1f",y);
printf("\n%10.3e",y);
getch();
}
```

```
Результат
1===5===10==========
15
   15
15
12.345000
     12.35
12.3
  1.23e+01
====================
```

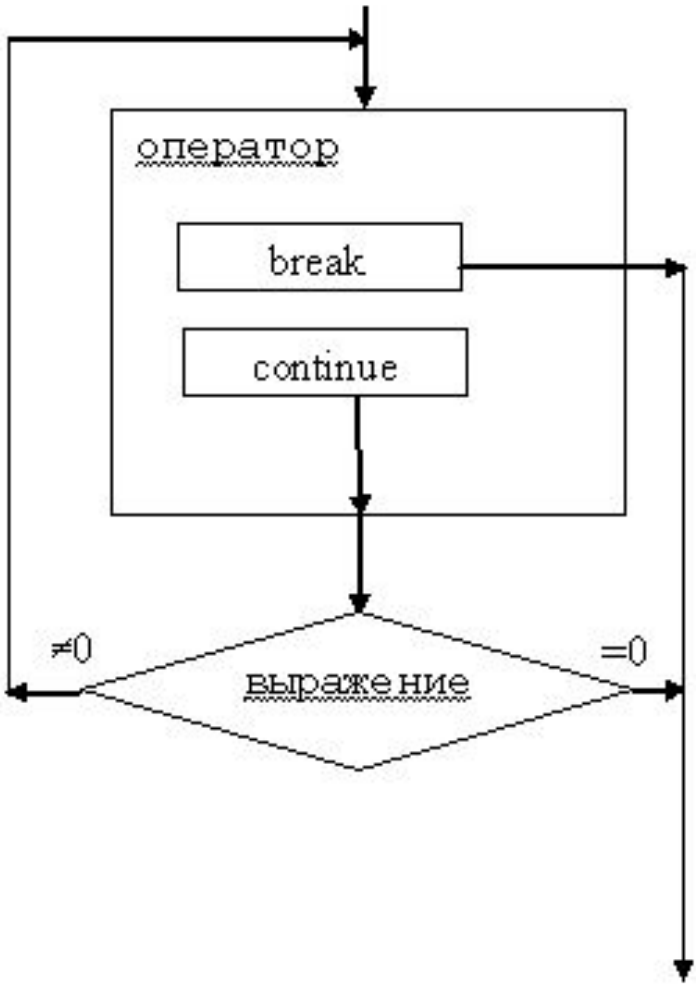
Пример для switch:

```
main()
{
int k=1;
char c;
while(k)
   {
   printf("Введите ответ Y/N?");
   c=getch();
   switch(c)
    {
    case 'y':
    case 'Y': printf("\nYes"); k=0; break;
    case 'n':
    case 'N': printf("\nNo"); k=0; break;
    default: printf("\nОшибка\nПовторите ввод\n");
    }
   getch();
   }
}
```

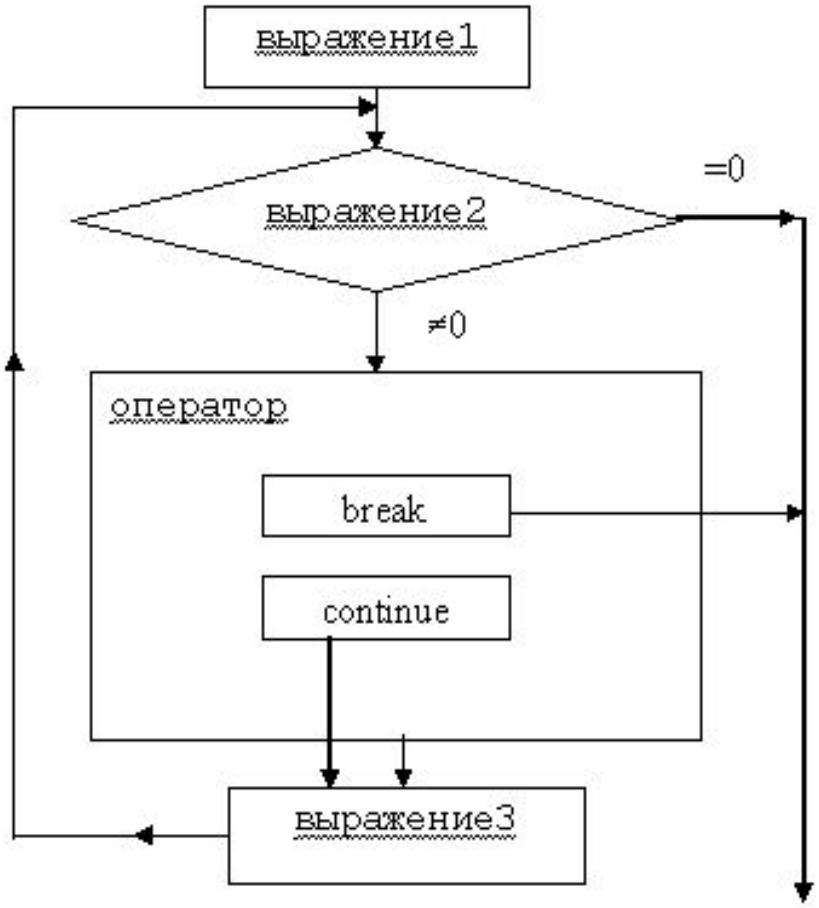
while (выражение) оператор;



do оператор while (выражение) ;



for (выражение1 ; выражение2 ; выражение3) оператор;



Примеры:

```
int x[5]={1,2,3,4,5};
int i,s;
s=0;
for(i=0; i<5; i++)
   s+=x[i];
```

x[i] равносильно *(x+i)

```
s=0; i=0;
while (i<5)
    {
    s+=x[i]; i++;
    }
```

```
s=0; i=0;
while(i<5)
    s+=x[i++];
```

```
s=0; i=0;
do   {
s+=x[i]; i++;
} while (i<5);
```

```
s=0; i=0;
do
s+=x[i];
while (++i<5);
```

Примеры:

```
#include <alloc.h>
/* #include <malloc.h> для c++ */
int i,n,*x;
printf(”Введите n:”);
scanf(”%d”,&n);
x=(int*)malloc(n*sizeof(int));
printf(”\nВведите элементы массива:”);
for(i=0;i<n;i++)
   scanf(”%d”,&x[i]);      /* scanf(”%d”,x+i); */
printf(”\nИсходный массив:\n”);
for(i=0;i<n;i++)
   printf(”%d ”,x[i]);
. . .
free(x);
```

Пример: среднее значение положительных элементов

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>  /* #include <malloc.h> для c++ */
main()
{
int i,n,k,*x;
float s;
printf(”Введите n:”);
scanf(”%d”,&n);
x=(int*)malloc(n*sizeof(int));
printf(”\nВведите элементы массива:”);
for(i=0;i<n;i++)
   scanf(”%d”,&x[i]);
printf(”\nИсходный массив:\n”);
for(i=0;i<n;i++)
   printf(”%d ”,x[i]);
```

Пример: среднее значение положительных элементов

```
s=0; k=0;
for(i=0; i<n; i++)
  if(x[i]>0){
        k++; s+=x[i];
          }
if(k)s/=k;    /* if(k>0)s=s/k; */
printf("\ns=%f",s);
getch();
```

Пример: максимальное значение и его индекс

```
k=0; r=x[0];
for(i=1; i<n; i++)
  if(x[i]>r){
        r=x[i]; k=i;
          }
printf(”\nx[%d]=%d”,k,r);
getch();
```

Пример: минимальное значение среди положительных

```
k=-1; r=MAXINT; /* <values.h> */
for(i=0; i<n; i++)
   {
   if(x[i]<=0) continue;
   if(x[i]<r){
         r=x[i]; k=i;
         }
   }
if(k==-1)
  printf(”\nПоложительных элементов нет”);
else
  printf(”\nx[%d]=%d”,k,r);
getch();
```

Пример: максимальное значение среди отрицательных

```
k=-1; r=-1e25;
for(i=0; i<n; i++)
   {
   if(x[i]>=0) continue;
   if(x[i]>r){
         r=x[i]; k=i;
         }
   }
if(k==-1)
  printf(”\nОтрицательных элементов нет”);
else
  printf(”\nx[%d]=%f”,k,r);
getch();
```

Пример: произведение до первого 0 и сумма остальных

```
s=0; p=1;
for(i=0; i<n; i++)
   {
   if(x[i]==0) break;
   p*=x[i];
   }
for(j=i+1; j<n; j++)
   s+=x[j];
printf(”\ns=%f”,s);
printf(”\np=%f”,p);
getch();
```

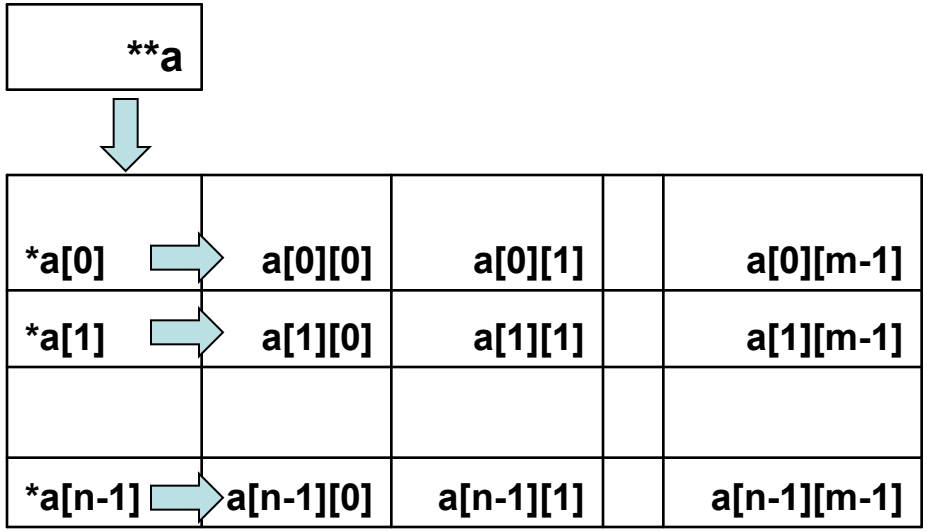
Пример: переписать массив в обратном порядке

```
printf(”\nИсходный массив:\n”);
for(i=0;i<n;i++)
   printf(”%d ”,x[i]);
for(i=0; i<n/2; i++)
   {
   r=x[i];
   x[i]=x[n-i-1];
   x[n-i-1]=r;
   }
printf(”\nПреобразованный массив:\n”);
for(i=0;i<n;i++)
   printf(”%d ”,x[i]);
getch();
```

Пример: сортировка массива по возрастанию

```
k=1;
while(k)
   {
   k=0;
   for(i=0; i<n-1; i++)
    if(x[i]>x[i+1])
              {
              k=1;
              r=x[i];
              x[i]=x[i+1];
              x[i+1]=r;
              }

   }
```

Примеры:




```
#include <alloc.h>
/* #include <malloc.h> для c++ */
int i,j,n,m,**a;
printf("Введите n и m:");
scanf("%d%d",&n,&m);
a=(int**)malloc(n*sizeof(int*));
for(i=0;i<n;i++)
   a[i]=(int*)malloc(m*sizeof(int));
printf("\nВведите элементы массива:");
for(i=0;i<n;i++)
   for(j=0;j<m;j++)
      scanf("%d",&a[i][j]);
printf("\nМатрица A:\n");
for(i=0;i<n;i++)
   {
   for(j=0;j<m;j++)
      printf("%5d",a[i][j]);
   printf("\n");
   }
```

```
for(i=n-1;i>=0;i--)
    free(a[i]);
free(a);
```

РАБОТА С ФАЙЛАМИ

Объявление файла производится следующим образом:

FILE указатель файла1[,..,указатель файлаN];

Например

```
FILE *in, *out, *f1;
```

Открытие файла:

указатель_файла=fopen(имя_файла, режим_работы);

Возможны следующие режимы работы:

r – открытие файла на чтение (при этом открываемый файл должен существовать);

w – открытие файла на запись (если открываемого файла нет, то он будет создан; если этот файл уже есть, то его содержимое стирается);

a – открытие файла на дозапись (при этом, если файла нет, то он создается);

r+ – открытие файла на чтение и запись (при этом открываемый файл должен существовать);

w+ – открытие файла на чтение и запись (при этом содержимое открываемого файла стирается);

a+ – открытие файла на чтение и дозапись (при этом, если файла нет, то он создается).

РАБОТА С ФАЙЛАМИ

Пример

```
if((in=fopen("myfile.dat","r"))==NULL)
 { printf("\n Файл myfile.dat не открыт.");
exit(1);}
```

Закрытие файла осуществляет функция

`fclose(указатель_файла);`

РАБОТА С ФАЙЛАМИ

Пример

```
#include <stdio.h>
#include <alloc.h>  /* #include <malloc.h> для c++ */

main()
{
int i,n,*x;
FILE *in,*out;
in=fopen(”mas.dat","r");
out=fopen(”mas.res","w");
scanf(”%d”,&n);
x=(int*)malloc(n*sizeof(int));
for(i=0;i<n;i++) fscanf(in,”%d”,&x[i]);
fprintf(out,”Массив X\n”);
for(i=0;i<n;i++) fprintf(out,”%d ”,x[i]);
fclose(in); fclose(out);
. . .
}
```

Работа со стандартным файлом:

scanf (управляющая строка, данные **);**

printf (управляющая строка,данные**);**

gets (имя_строки **);**

puts(строка или имя строки **);**

getchar();

getch();

getche();

putchar(символ или имя_символа)

Работа с произвольным файлом:

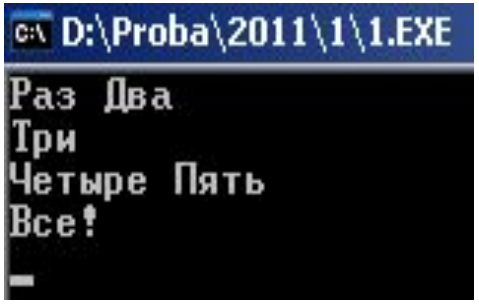
fscanf(указатель_файла,управляющая_строка, данные**);**
fprintf(указатель файла,управляющая строка, данные**);**
fgets (имя_строки,колич.симв.,указ._файла **);**
fputs (строка или имя_строки,указ._файла **);**
getc (указатель_файла**) ;**
putc(символ или имя_символа,указ._файла**);**

Чтение символов из потока до обнаружения конца

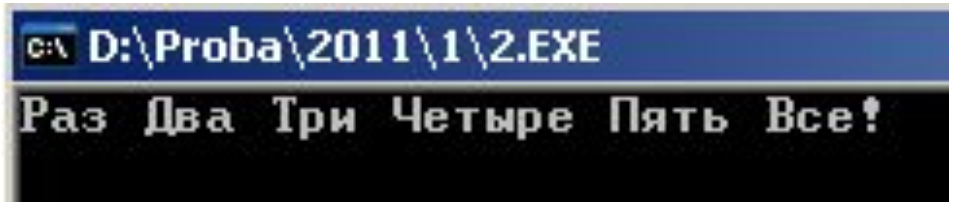
```
#include <stdio.h>
#include <conio.h>
main()
{
int c;
FILE *in;
clrscr();
in=fopen("str1.dat","r");
while((c=getc(in))!=EOF)
   putchar(c);
getch();
}
```

STR1.DAT

```
==========
Раз Два
Три
Четыре Пять
Все!
```




```
#include <stdio.h>
#include <conio.h>
main()
{
int c;
FILE *in;
clrscr();
in=fopen("str1.dat","r");
while((c=getc(in))!=EOF)
    if(c!='\n')
        putchar(c);
    else
        putchar(' ');
getch();
}
```



Двоичные файлы. Бесформатный ввод-вывод:

fread(куда, размер, сколько, указатель_файла**);**
– для чтения из файла,
fwrite (откуда, размер, сколько,указатель_файла**);**
– для записи в файл.

При этом, в **fopen**
режим дополняется опцией **”b”**
(двоичный файл), например,
"wb+"
(по умолчанию действует **”t”**
— текстовый файл).

Двоичные файлы. Бесформатный ввод-вывод:

ПРИМЕР:

```
#include<stdio.h>
main( )
{
FILE *in;
int i,a[10]={1,2,3,4,5,6,7,8,9,10},b[10];
in=fopen("D:\\ISTAS\\myfile.dat","wb+");
fwrite(a,2,10,in);
rewind(in);
fread (b,2,10,in); puts ("\nМассивВ\n");
for (i=0;i<10;i++)
    printf (" %d",b[i]);
}
```

Указание позиции при работе с файлом:

`ftell` (указатель файла);

`fseek`(указатель_файла, величина_сдвига, точка_отсчета);

- **SEEK_SET – начало файла,**
- **SEEK_CUR – текущая позиция,**
- **SEEK_END – конец файла.**

```
#include<stdio.h>
main()
{
FILE *out;
int i,a[20],b[20];
out=fopen ("fa","w+");
for (i=1;i<100;i+=5) fprintf (out," %d ",i);
fputs ("\n ИСТАС-",out); putc ('I',out);
printf ("\n Текущая позиция =%ld\n", ftell(out));
rewind (out);
printf ("\n Текущая позиция N =%ld\n", ftell(out));
for (i=0;i<19;i++) fscanf (out,"%d",&a[i]);
for (i=0;i<19;i++) printf ("%d ",a[i]);
rewind (out);
for (i=0;i<9;i++) fscanf (out,"%d",&b[i]);
fseek (out, 0, SEEK_CUR);
for (i=0;i<9;i++) fprintf (out,"%3d ",b[i]);
fseek (out, -36L, SEEK_CUR);
for (i=9;i<19;i++) fscanf (out,"%d",&b[i]);
for (i=0;i<19;i++) printf ("%d ",b[i]);
}
```

Применение двоичного файла для хранения исходных данных.

А) Чтение данных из текстового файла и запись в двоичный файл.

```
FILE *in,*out;
...
in=fopen("Vklad.dat","r");
out=fopen("VkladBin.dat","wb");
fscanf(in,"%d",&NC);
clients=(struct z*)malloc(NC*sizeof(struct z));
for(i=0;i<NC;i++)
    fscanf(in,"%s%s%ld%s",clients[i].name,
        clients[i].vid, &clients[i].summa,
        clients[i].data);
fwrite(&NC,sizeof(int),1,out);
for(i=0;i<NC;i++)
    fwrite(&clients[i],sizeof(struct z),1,out);
/*
 или одной командой
 fwrite(clients,sizeof(struct z),NC,out);
 */
fclose(in); fclose(out);
```

Применение двоичного файла для хранения исходных данных.

Б) Чтение из двоичного файла

```
FILE *in;
...
in=fopen("VkladBin.dat","rb");
fread(&NC,sizeof(int),1,in);
clients=(struct z*)malloc(NC*sizeof(struct
z));

fread(clients,sizeof(struct z),NC,in);

fclose(in);
```

Функции

Функция задаётся следующим образом:

тип_функции имя_функции (формальные аргументы)
{
текст
}

имя_функции – это идентификатор;

формальные_аргументы – это список имен аргументов,
которые будут переданы функции.

Функция не может быть описана внутри других функций.

К заданной функции можно обратиться из любой другой функции, при этом возможны две формы обращения:

1.
имя_переменной = имя_функции(фактические аргументы);
2.
имя_функции(фактические аргументы);

Фактические_аргументы должны соответствовать формальным по **количеству, типу и порядку следования.**

Функции

Пример

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

void swap(int*, int*);
void noswap(int, int);

main()
{
int x,y;
x=1; y=2;
swap(&x,&y);
printf(”\nx=%d y=%d”,x,y);
noswap(x,y);
printf(”\nx=%d y=%d”,x,y);
getch();
}
```

```
void swap(int *x, int *y)
{
int z;
z=*x; *x=*y; *y=z;
return;
}

void noswap(int x, int y)
{
int z;
z=x; x=y; y=z;
return;
}
```

Пример: сумма элементов массива

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
/* #include <malloc.h> для c++ */


int xsum(int n, int *x)
{
int i,s;
for(s=0,i=0; i<n; i++)
       s+=x[i];
return s;
}
```

```
main()
{
int i,k,*a,s;
printf("Введите k:");
scanf("%d",&k);
a=(int*)malloc(k*sizeof(int));
printf("\nВведите массив:");
for(i=0;i<k;i++)
    scanf("%d",&a[i]);
s=xsum(k,a);
printf("\ns=%d",s);
getch();
}
```

Пример: максимальный элемент и его индекс

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
/* #include <malloc.h> для c++ */
void xmax(
int n, int*x, int*r, int*k)
{
int i;
*r=x[0]; *k=0;
for(i=1;i<n;i++)
   if (x[i]>*r)
    {
    *r=x[i]; *k=i;
    }
return;
}
```

```
main()
{
int i,n,*a,k,r;
printf("Введите n:");
scanf("%d",&n);
a=(int*)malloc(n*sizeof(int));
printf("\nВведите массив:");
for(i=0;i<n;i++)
    scanf("%d",&a[i]);
xmax(n,a,&r,&k);
printf("\nx[%d}=%d",k,r);
getch();
}
```

В какой строке максимальная сумма положительных элементов?

```
#include <...>
main()
{
int i,j,n,m,k,r,*x,*s,**a;
printf("Введите n и m:");
scanf("%d%d",&n,&m);
x=(int*)malloc(m*sizeof(int));
s=(int*)malloc(n*sizeof(int));
a=(int**)malloc(n*sizeof(int*));
for(i=0;i<n;i++)
 a[i]=(int*)malloc(m*sizeof(int));
printf("\nВведите элементы A:");
for(i=0;i<n;i++)
   for(j=0;j<m;j++)
      scanf("%d",&a[i][j]);
printf("\nМатрица A:\n");
for(i=0;i<n;i++)
   {
   for(j=0;j<m;j++)
      printf("%5d",a[i][j]);
   printf("\n");
   }
```

```
for(i=0;i<n;i++)
  {
  k=0; s[i]=0;
  for(j=0;j<m;j++)
    if(a[i][j]>0)
        x[k++]=a[i][j];
   if(k>0)
     s[i]=xsum(k,x);
   }

 xmax(n,s,&r,&k);

 printf("\nk=%d",k);
 getch();
 }
```

В какой строке максимальная сумма положительных элементов?

Будет ли правильно работать такой вариант?

```
int xsum(int n, int *x)
{
int i,s;
for(s=0,i=0; i<n; i++)
      s+=x[i];
return s;
}
```

```
...
for(i=0;i<n;i++)
  {
  k=0;
  for(j=0;j<m;j++)
   if(a[i][j]>0)
      x[k++]=a[i][j];
  s[i]=xsum(k,x);
  }

 xmax(n,s,&r,&k);

 printf("\nk=%d",k);
 getch();
 }
```

Работа с объектами символьного типа

```
#include <stdio.h>
#include <conio.h>
int strlength1(char s[])
{
int i=0;
while(s[i]!='\0')i++;
   /*
   while(s[i]!=0)i++;
   или
   while(s[i])i++;
   или
   for(i=0;s[i]!=0;i++);
   или
   for(i=0;s[i];i++);
   */
return i;
}
int strlength2(char *s)
{
int i=0;
while(*s++)i++;
return i;
}
```

```
int strlength3(char *s)
{
char *p=s;
while(*p)p++;
return p-s;
}
main()
{int l;
char s[80];
printf("\nВведите строку:");
scanf("%s",s);
l=strlength1(s);
printf("\nl1=%d",l);
l=strlength2(s);
printf("\nl2=%d",l);
l=strlength3(s);
printf("\nl3=%d",l);
getch();}
```

s[0]=’М’, s[1]=’Г’, s[2]=’С’, s[3]=’У’,s[4]=’\0’,

```
#include <stdio.h>
#include <conio.h>
strcopy1(
char s[],char t[])
{
int i=0;
while(s[i]=t[i])i++;
return;
}

strcopy2(
char *s,char *t)
{
while(*s++=*t++);
return;
}
```

```
main()
{
int l;
char s[80],t[80];
printf("\nВведите строку:");
scanf("%s",t);

strcopy1(s,t);
printf("\n%s",s);
strcopy2(s,t);
printf("\n%s",s);
getch();
}
```

t=“ИСТАС” {’И’, ’С’, ’Т’, ’А’, ’С’, ’\0’,}

Некоторые функции <string.h>

- **strlen** (char* str);
  длина строки (без учета 0-символа);

- **strcpy** (char* str1,char* str2);
  strncpy (char* str1,char* str2, *число_байтов*);
  копирование строк здесь 2-я строка переписывается в первую, размер которой должен быть достаточным для такого копирования;

```
char s[80]; . . .  strcpy(s,"ИСТАС");
```

- сравнение строк **strcmp** (char* str1,char* str2)
  strncmp (char* str1,char* str2, *число_байтов*)
  эта функция возвращает
  положительное число, если первый аргумент больше второго,
  отрицательное число, если первый аргумент меньше второго,
  0, если эти аргументы равны (сравниваются числовые коды символов, из которых состоят строки);

```
if(strcmp("ЭТИЛ","ЭТАНОЛ")>0){.........}
```

Некоторые функции <string.h>

- **strcat** (char* **str1,**char* **str2**);
  strncat (char* **str1,**char* **str2**, *число_байтов*);

  сцепление строк здесь строки-аргументы соединяются в одну строку

- **strchr** (char* **str1**, int **c**); **strrchr** (char* **str1**, int **c**);
  возвращает указатель на первое (**strchr**) или последнее (**strrchr**) вхождение символа **c** в строку **str1**

- **strstr** (char* **str1**, char* **str2**);
  возвращает указатель на первое вхождение строки **str2** в строку **str1**
  При отсутствии вхождения **strchr, strrchr и strstr** возвращают **NULL**

- **memset** (char* **str1**, int **c**, int **n**);
  заполняет память, начиная с адреса **str1,**
  n - кратной вставкой символа **c**

```
char s[80];
memset(s,0,80);
memset(s,'X',5);   /* s="XXXXX" */
```

Некоторые функции <string.h>

- **strtok** (char* str1, char* str2);

  поиск лексических единиц. Возвращает указатель (char*) на первое слово, в конце которого добавляется 0. Для повторного использования вместо первого аргумента используется NULL.

  В качестве результата выдается указатель на второе слово из входной строки.

  Процесс можно повторять, пока не будет обработана вся исходная строка **str1**.

  Тогда функция вернет нулевой указатель.

  str2 задает список всех возможных разделителей (пробел, точка, запятая и т.д.)

Пример

```
char s[]="Раз два,три(четыре)пять.";
char *Temp=strtok(s, " ,.()" );
. . .
Temp=strtok(NULL, " ,.()" );
```

Применение функции **strtok()**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
char s[]="Иванов(ЭУИС)";
char *Temp=strtok(s,"(");
printf("\n%s\n",Temp);
Temp=strtok(NULL,")");
printf("%s\n",Temp);
getch();
}
```

Дано:

Иванов(ЭУИС)

Получаем:

Иванов

ЭУИС

Преобразование по формату:

```
int year;
char s[80];
strcpy(s,”2010”);

sscanf(s,"%d",&year);

year++;

sprintf(s,"%d",year);
```

Преобразование строка/число:

II способ

```
#include <stdlib.h>
int n-число,d-основание системы счисления
char s[80]-строка;
/*Преобразование числа в строку*/
itoa(n,s,d);
/*Для типа long*/ ltoa(n,s,d);
/*Преобразование строки в число*/
n=atoi(s);
/*Для типа long*/ n=atol(s);
/*Для типа double*/ x=atof(s);
```

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

int n_of_char(char c, char* s)
{
int n=0,i;
for(i=0;i<strlen(s);i++)
   if(c==s[i])n++;
return n;
}

main()
{
char c,s[80];
clrscr();
printf("\Введите символ\n");
scanf("%c",&c);
printf("\Введите строку\n");
scanf("%s",s);
printf("\n N=%d",n_of_char(c,s));
getch();
}
```

Введите символ

a

Введите строку

абракадабра

N=5

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

int n_of_char(char c, char* s)
{
int n=0,i;
for(i=0;i<strlen(s);i++)
   if(c==s[i])n++;
return n;
}

main()
{
FILE *in;
char c,s[80];
clrscr();
printf("\Введите символ\n");
scanf("%c",&c);
in=fopen("str.dat","r");
while(fscanf(in,"%s",s)!=EOF)
printf("\n %s=%d",s,n_of_char(c,s));
getch(); fclose(in);
}
```

Введите символ
Я
АРГЕНТИНА=0
РОССИЯ=1
ЯПОНИЯ=2

```
#include <stdio.h>
#include <string.h>
int test(char* s)
{
int n=0,i;
n=strlen(s)-1;
if(s[0]==s[n]) return 1;
return 0;
}
main()
{
FILE *in;
char s[80];
int k=0;
clrscr();
in=fopen("str.dat","r");
while(fscanf(in,"%s",s)!=EOF)
        if(test(s))
            {
             printf(s);
                  printf("\n");
            k++;
            }
printf("\n k=%d",k); getch();
}
```

АРГЕНТИНА

ЯПОНИЯ

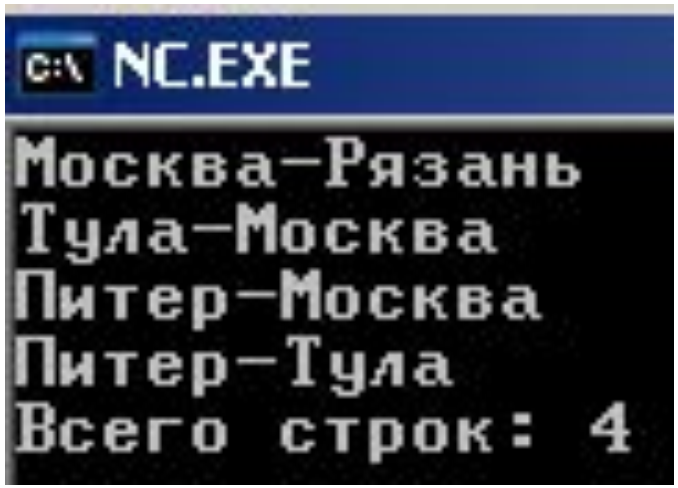
k=2

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

main()
{
FILE *in;
int c,k;

clrscr();

in=fopen("str5.dat","r");
k=0;
while((c=getc(in))!=EOF)
   {
   putchar(c);
   if(c=='\n')
    k++;
   }
printf("Всего строк: %d",k);
getch();
fclose(in);
}
```

Пример.

Посимвольное чтение из файла

\+ статистика



В предложении слова разделены одним пробелом или запятой. В конце стоит точка. Переписать предложение, заключив слова в скобки.

(Это-упрощенный вариант задачи)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char s[80],t[80];
int i,j;
printf("Введите предложение:");
gets(s);

printf("\n Заключаем в скобки:\n");

t[0]='(';
j=1;
i=0;
while(s[i]!='.')
{
if(s[i]==' '||s[i]==',')
        {
        t[j++]=')';
        t[j++]='(';
        i++;
        }
    else
        t[j++]=s[i++];
}
t[j++]=')';
t[j++]='.';
t[j]='\0';
printf(t);
getch();
}
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
char s[80];
char Znak[]=" ,.()";
char *Temp;
clrscr();
printf("Введите предложение:");
gets(s);
Temp=strtok(s,Znak);
do
  {
  /*  printf("%s\n",Temp);*/
  puts(Temp);
  Temp=strtok(NULL,Znak);
  } while(Temp);
getch();
}
```

Пример

Разобрать предложение на отдельные слова.

Введите предложение:
Раз два, три(четыре) пять.

Раз
два
три
четыре
пять

Переписать предложение, заключив слова в скобки.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char s[80],t[80]="";
char Znak[]=" ,.()";
char *Temp;

printf("Введите предложение:");
gets(s);

printf("\n Заключаем в скобки:\n");
Temp=strtok(s,Znak);
do
  {
  strcat(t,"(");
  strcat(t,Temp);
  strcat(t,")");

  Temp=strtok(NULL,Znak);
  } while(Temp);
strcat(t,".");
printf(t);
getch();
}
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
FILE *in;
char s[80],Fam[80]="";
char *Temp,*Facultet;
int Lmax,Len;
if((in=fopen("Spisok.dat","r"))==NULL)
   {printf("Файл не открыт!");exit(1);}
Lmax=0;
while(fgets(s,80,in))
{
Temp=strtok(s,"(");
Facultet=strtok(NULL,")");
if(strcmp(Facultet,"ЭУИС"))
                continue;
if((Len=strlen(Temp))>Lmax)
     {
     Lmax=Len;
     strcpy(Fam,Temp);
     }
}
fclose(in);
printf("%s-%d ", Fam,strlen(Fam));
getch();
}
```

Найти самую длинную фамилию на факультете ЭУИС

Spisok.dat
========
Иванов(ИСТАС)
Петраков(ЭУИС)
Сидоров(ЭУИС)
Иванов(ПГС)
Петров(ЭУИС)
Мисийцева(ПГС)

Фамилия и факультет выбираются в отдельные переменные
Temp -все, что до «(«
и **Facultet** – до «)»,
Значения которых анализируются программой.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
FILE *in;
char s[80],Fam[80]="";
char *Temp,*Facultet;
int Lmax,Len;
if((in=fopen("Spisok.dat","r"))==NULL)
   {printf("Файл не открыт!");exit(1);}
Lmax=0;
while(fgets(s,80,in))
{
if(!strstr(s,"(ЭУИС)")) continue;
Temp=strtok(s,"(");
if((Len=strlen(Temp))>Lmax)
    {
    Lmax=Len;
    strcpy(Fam,Temp);
    }
}
fclose(in);
printf("%s-%d ", Fam,strlen(Fam));
getch();
}
```

Найти самую длинную фамилию на факультете ЭУИС

Spisok.dat
========
Иванов(ИСТАС)
Петраков(ЭУИС)
Сидоров(ЭУИС)
Иванов(ПГС)
Петров(ЭУИС)
Мисийцева(ПГС)

В этом варианте решения сразу ищется вхождение подстроки с названием факультета в исходный текст.

```
char s[80],F[80],Fam[80]="";
char *Temp,*Facultet;
. . .
printf("Введите факультет: ");
gets(s);
F[0]='(';
strcpy(F+1,s);
strcat(F,")");

Lmax=0;
while(fgets(s,80,in))
   {
   if(!strstr(s,F))continue;
   Temp=strtok(s,"(");
   if((Len=strlen(Temp))>Lmax)
        {
        Lmax=Len;
        strcpy(Fam,Temp);
        }
   }
fclose(in);
printf("%s -%d ",Fam,strlen(Fam));
getch();
}
```

Найти самую длинную фамилию на заданном факультете

Spisok.dat
========
Иванов(ИСТАС)
Петраков(ЭУИС)
Сидоров(ЭУИС)
Иванов(ПГС)
Петров(ЭУИС)
Мисийцева(ПГС)

В этом варианте решения факультет вводится с клавиатуры.
В переменной **F** Формируется текст **(ФАКУЛЬТЕТ)**

Проблема совместного использования

scanf и gets

```
printf("Введите предложение:");
gets(s);
printf("Введите символ");
scanf("%c",&c);
```

~~**printf("Введите символ");**~~
~~**scanf("%c",&c);**~~
~~**printf("Введите предложение:");**~~
~~**gets(s);**~~

```
printf("Введите символ");
scanf("%c",&c);
getchar();
printf("Введите предложение:");
gets(s);
```

```
printf("Введите символ");
scanf("%c%*c",&c);
printf("Введите предложение:");
gets(s);
```

Структуры

```
struct z {
   char name[20];
   char vid[20];
   long summa;
   char data[11];
   } x,*a,*b;
a=(struct z*)malloc(n*sizeof(struct z));
b=&x;
x.summa=10000; b->summa=5000;
a[i].summa=200; (a+i)->summa=100;
strcpy(x.name,"Иванов");
strcpy(b->name,"Петров");
strcpy(a[i].name,"Сидоров");
strcpy((a+i)->name,"Федоров");
```

Шаблон структуры

Структуры

```
main()
{
FILE *in;
struct z {
   char name[20];
   char vid[20];
   long summa;
   char data[11];
   } *clients;
int n,int k,i;
clrscr();
in=fopen("struct.dat", "r");
fscanf(in,"%d",&n);
clients=(struct z*)malloc(n*sizeof(struct z));
for(i=0;i<n;i++)
   fscanf(in,"%s%s%ld%s",clients[i].name, clients[i].vid,
        &clients[i].summa, clients[i].data);
for(i=0;i<n;i++)
   printf("\n%-20s %-20s %7ld %s",clients[i].name,
    clients[i].vid, clients[i].summa, clients[i].data);
. . .
}
```

STRUCT.DAT
=========
5
Иванов_А_А Срочный 15000 2002-09-23
Сидоров_И_А Юбилейный 7150 1991-03-08
Петров_В_Н Пенсионный 38876 1999-12-16
Сидоров_И_А Сберегательный 12860 2008-06-23
Юдин_О_В Особый 25000 2006-12-13

Битовые поля в структурах

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

struct byte
   {
   unsigned b1:1;
   unsigned b2:1;
   unsigned b3:1;
   unsigned b4:1;
   unsigned b5:1;
   unsigned b6:1;
   unsigned b7:1;
   unsigned b8:1;
   };
union bits
   {
   unsigned char ch;
   struct byte b;
   } u;
```

```
main()
{
clrscr();
u.ch=getche();
printf("\n %d - ",u.ch);
printf("%d ",u.b.b8);
printf("%d ",u.b.b7);
printf("%d ",u.b.b6);
printf("%d ",u.b.b5);
printf("%d ",u.b.b4);
printf("%d ",u.b.b3);
printf("%d ",u.b.b2);
printf("%d ",u.b.b1);
getch();
}
```

32 – 0 0 1 0 0 0 0 0 (пробел)

Аргументы функции main

```
main(int ARGC,char **ARGV)
  {
    while (ARGC--)
       printf("\n %s ",ARGV[ARGC]);
  }
```

Допустим, программа называется

obrab_1.exe

Тогда, если ее вызвать из подкаталога TC на диске C следующим образом:

C:\TC>obrab_1 one two

то программа выдаст на экран такую информацию:

two

one

C:\TC\obrab_1.exe

КЛАССЫ ПАМЯТИ

Класс памяти	Время существования переменной	Область действия
auto	Временно	Локальная
register	Временно	Локальная
static	Постоянно	Локальная
extern	Постоянно	Глобальная
extern static	Постоянно	Глобальная (1 файл)

Классы памяти

```
#include <stdio.h>
#include <conio.h>
f()
{
static int i=0;
i++;
return;
}
main()
{
f();     /* i=1 */
f();     /* i=2 */
f();     /* i=3 */
getch();
}
```

example.h

```
int a=20;
```

example.c

```
#include <stdio.h>
#include <conio.h>
#include ”example.h”
main()
{
extern int a;
/* Здесь инициализировать нельзя */
printf(”a=%d”,a);   /* a=20 */
getch();
}
```

Классы памяти

```
int i=0,j=0;
void Print_J()
{
printf("\n j=%d",j); /* j=0  */
}
main()
{
int j=10;
i++;
printf("\n i=%d j=%d",i,j);  /* i=1  j=10 */
if (j==10)
   {
   int i=100;
   i--;
   printf("\n i=%d",i);     /* i=99 */
   }
Print_J();
printf("\n i=%d",i);     /* i=1  */
}
```

Классы памяти

Example-A.c

```
#include "MyFunc.c"
main()
{
extern int x;
x=10;
f1();
f2();
}
```

В результате будет выведено
x=11

Example-B.c

```
#include "MyFunc.c"
main()
{
int x;
x=10;
f1();
f2();
}
```

В результате будет выведено
x=1

Описатель **extern** может быть полезен, когда главная программа и все остальное компилируются из отдельных файлов.

MyFunc.c

```
#include <stdio.h>
#include <conio.h>
int x;
/* автоматически инициализируется нулем */
f1()
{
x++;
}
f2()
{
printf("x=%d",x);
getch();
}
```