Лекция 2
Основные операторы языка Си

Операторы языка Си

Все операторы языка Си могут быть условно разделены на следующие категории:

- условные операторы, к которым относятся оператор условия if и оператор выбора switch;
- операторы цикла (for,while,do while);
- операторы перехода (break, continue, return);
-другие операторы (оператор "выражение", пустой оператор, составной оператор).

Все операторы языка СИ, кроме составных операторов, заканчиваются точкой с запятой ";".

Составной оператор

Составной оператор представляет собой несколько операторов, заключенных в фигурные скобки:

{ оператор; [оператор]; }

Заметим, что в конце составного оператора точка с запятой не ставится.

Оператор выражение

Любое выражение, которое заканчивается точкой с запятой, является оператором.

Примеры:

```
++ i;        // оператор, который увеличивает значение переменной i на единицу.
a = cos(b * 5);  // оператор, включающий в себя операции присваивания и вызова функции.
a(x,y);      // выражение, состоящее из вызова функции.
```

Пустой оператор

Пустой оператор состоит только из точки с запятой. При выполнении этого оператора ничего не происходит. Он обычно используется в операторах **do, for, while, if** в строках,

Условный оператор

Формат оператора:

if (логическое выражение) оператор-1;

if (логическое выражение) оператор-1;
else оператор-2;

Выполнение оператора **if** начинается с вычисления *логического выражения*. Далее

- если выражение истинно (т.е. отлично от 0), то выполняется *оператор*.
- если выражение ложно (т.е. равно 0),то выполняется оператор-2 (если указано **else**), или выполняется следующий за **if** оператор (если не указано **else**).

Пример:

```
if (i < j) i++ ; else { j = i-3; i++; }
```

Логическое выражение

Логические выражения обычно имеют вид

арифм.выражение1 **операция сравнения арифм.*выражение2***

Операции сравнения:

> больше

>= больше или равно

< меньше

<= меньше или равно

== равно

!= не равно

Логические операции:

&& - логическое «и»

|| - логическое «или»

! - логическое «не»

Выражения, использующие логические операции и операции сравнения, возвращают 0 для ложного значения и 1 для истинного. Ниже приводится таблица истинности для логических операций:

exp1	exp2	exp1 && exp2	exp1 || exp2	!exp1
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Условное выражение

Условное выражение, написанное с помощью тернарной (т.е. имеющей три операнда) операции **? :** предоставляет другой способ для записи условного оператора.

Формат записи тернарной операции:

логическое выражение ? выражение 1 : выражение 2 ;

Алгоритм работы операции следующий:

1. Вычисляется логическое выражение.

Если **логическое выражение** истинно,

то вычисляется значение выражения **выражение 1**,

в противном случае — значение выражения **выражение 2**.

2. Возвращается вычисленное значение .

Например, при нахождении максимального элемента из двух чисел можно использовать следующий вариант записи:

max = (a > b) ? a : b;

Это выражение эквивалентно следующему условному оператору:

if(a > b) max=a; else max=b;

Примеры:

```
m = a > 0 ? a : -a;
y = (x < 0 )?  x :  ((x >= 0) && (x < 30)) ? 0 : x * x );  // две вложенные тернарные операции
```

Условное выражение часто помогает сократить программу. В качестве примера приведем цикл, обеспечивающий печать n элементов массива по 10 на каждой строке с одним пробелом между колонками; каждая строка, включая последнюю, заканчивается символом новой строки:

```
for (i = 0; i < n; i++)
printf( "%6d%c",  a[i],  ( i % 10 == 9 || i == n-1) ?  '\n' : ‘ ’ );
```

Пример условного оператора

Для заданного значения X вычислить значение функции F(X)=Y, которая определяется следующим образом.

$$F(X) = \begin{cases} 0, & \text{если } x >= 0 \\ 4 * x, & \text{если } x < 0 \end{cases}$$

```
#include <stdio.h>
main()
    {
   float y,x;
   puts("Введите значение x");
    scanf("%f",&x);
    if  (x>=0)   y=4*x;
            else y=0;
    printf("%f",y);
    }
```

Вложенные условные операторы

Оператор if может быть включен в конструкцию if или в конструкцию else другого оператора if. Рекомендуется группировать операторы и конструкции во вложенных операторах if, используя фигурные скобки. Если же фигурные скобки опущены, то компилятор связывает каждое ключевое слово else с наиболее близким if, для которого нет else.

Примеры:

```
int main ( )
{
  int t=2, b=7, r=3;
     if (t>b)
       {
          if (b < r) r=b;
        }
            else r=t;
  return (0);
}
```

В результате выполнения этой программы r станет равным 2.

Если же в программе опустить фигурные скобки, стоящие после оператора if, то программа будет иметь следующий вид:

```
int main ( )
{
   int t=2,b=7,r=3;
     if ( t>b )
         if ( b < r )  r=b;
                 else  r=t;
return (0);
}
```

В этом случае r получит значение равное 3, так как ключевое слово else относится ко второму оператору if, который не выполняется, поскольку не выполняется условие, проверяемое в первом операторе if.

Оператор switch

Конструкции использующие вложенные операторы if, являются довольно громоздкими и не всегда достаточно надежными:

```
char ZNAC;
int x,y,z;
if (ZNAC == '-') x = y - z;
else  if (ZNAC == '+') x = y + z;
      else  if (ZNAC == '*') x = y * z;
            else  if (ZNAC == '/') x = y / z;
                  else ...
```

В этом случае используют оператор switch. Он предназначен для организации выбора из множества различных вариантов. Формат оператора следующий:

```
switch ( выражение )
 { [ case константное-выражение1]: [ список-операторов1]
   [ case константное-выражение2]: [ список-операторов2]
                          : :
   [ default: [ список операторов ]
  }
```

Выполнение оператора **switch** начинается с вычисления выражения в круглых скобках;

- вычисленные значения последовательно сравниваются с константными выражениями, следующими за ключевыми словами case;
- если одно из константных выражений совпадает со значением выражения, то управление передается на оператор, помеченный соответствующим ключевым словом case;
- если ни одно из константных выражений не равно выражению, то управление передается на оператор, помеченный ключевым словом default, а в случае его отсутствия управление передается на следующий после switch оператор.

Пример:

```
char ZNAC;
int x,y,z;
switch (ZNAC)
    {
       case  '+':  x = y + z;  break;
       case  '-':  x = y - z;  break;
       case  '*':  x = y * z;  break;
       case  '/':  x = u / z;  break;
      default :     ;
    }
```

Оператор switch

Пример, в котором не предусмотрен выход из case с помощью оператора break :

```
int   i=2;
switch (i)
    {
        case 1: i += 2;
        case 2: i *= 3;
        case 0: i /= 2;
        case 4: i -= 5;
         default:    ;
     }
```

Выполнение данного оператора switch начинается с оператора, помеченного case 2. Таким образом, переменная i получает значение, равное 6, далее выполняется оператор, помеченный ключевым словом case 0, а затем case 4, переменная i примет значение 3, а затем значение -2. Оператор, помеченный ключевым словом default, не изменяет значения переменной.

Конструкция со словом default может быть не последней в теле оператора switch. Ключевые слова case и default в теле оператора switch существенны только при начальной проверке, когда определяется начальная точка выполнения тела оператора switch. Все операторы, между начальным оператором и концом тела, выполняются вне зависимости от ключевых слов, если только какой-то из операторов не передаст управления из тела оператора switch. Таким образом, программист должен сам позаботится о выходе из case, если это необходимо. Чаще всего для этого используется оператор break.

Операторы перехода break, continue, return

Оператор break

Оператор break обеспечивает прекращение выполнения самого внутреннего из объединяющих его операторов switch, do, for, while. После выполнения оператора break управление передается оператору, следующему за прерванным.

Оператор continue

Оператор continue, как и оператор break, используется только внутри операторов цикла, но в отличие от него выполнение программы продолжается не с оператора, следующего за прерванным оператором, а с начала прерванного оператора. Формат оператора следующий:

```
continue;
```

Пример:

```
int main()
{     int a,b;
       for (a=1, b=0; a<100; b+=a, a++)
         {   if (b%2)  continue;
                ...       /* обработка четных сумм */
          }
      return 0;
 }
```

Когда сумма чисел от **1** до **a** становится нечетной, оператор **continue** передает управление на очередную итерацию цикла **for**, не выполняя операторы обработки четных сумм.

Оператор **continue**, как и оператор **break**, прерывает самый внутренний из объемлющих его циклов.

Оператор return

Оператор return завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию, в точку, непосредственно следующую за вызовом. Функция main передает управление операционной системе. Формат оператора:

```
return [выражение] ;
```

Пример:

```
Int   sum  (int a, int b)
      {   renurn (a+b);   }
```

Функция sum имеет два формальных параметра a и b типа int, и возвращает значение типа int, о чем говорит описатель, стоящий перед именем функции. Возвращаемое оператором return значение равно сумме фактических параметров.

Операторы цикла
Оператор for

Оператор **for** имеет следующий формат:

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

Выражение 1 обычно используется для установления начального значения переменных, управляющих циклом.

Выражение 2 - это выражение, определяющее условие, при котором тело цикла будет выполняться.

Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

Правило выполнения оператора for:

1. Вычисляется выражение 1.
2. Вычисляется выражение 2.
3. Если значения выражения 2 отлично от нуля (истина), выполняется тело цикла, вычисляется выражение 3 и осуществляется переход к пункту 2, если выражение 2 равно нулю (ложь), то управление передается на оператор, следующий за оператором for.

Пример:

```
int main()
 {
    int i,b;
       for (i=1; i<10; i++)  b=i*i ;
    return 0;
 }
```

Оператор for

Некоторые варианты использования оператора for повышают его гибкость за счет возможности использования нескольких переменных, управляющих циклом.

Пример:

```
int main()
{   int top, bot;
    char string[100], temp;
        for ( top=0, bot=100 ; top < bot ; top++, bot--)
           { temp=string[top];
             string[top]=string[bot] ;
             string[bot]=temp;
            }
  return 0;
}
```

Другим вариантом использования оператора for является бесконечный цикл. Для организации такого цикла можно использовать пустое условное выражение, а для выхода из цикла обычно используют дополнительное условие и оператор break.

Пример:

```
for ( ; ; )
{
        ...
        ...
        break;
         ...
 }
```

Так как согласно синтаксису языка Си оператор может быть пустым, тело оператора for также может быть пустым. Такая форма оператора может быть использована для организации поиска.

Пример:

```
for (i=0; t[i]<10 ; i++) ;
```

Пример1 цикла for

Дано n. Определить и вывести значения x=2i – степени двойки от i=1 до n (n>1) по 5 значений в строке.

```
#include <stdio.h>   // Подключение библиотек
#include <math.h>
#include <conio.h>
main()
{
int i,n;
long int x;
clrscr ();      // Очистка экрана
printf ("Введите число - максимальную степень двойки \n");
scanf ("%i",&n);  // Ввод n
for (x=i=1; i<=n ; i++)
     {
     x*=2;                    // Определение x
     printf ("%10d",x);       // Вывод  x
     if (fmod(i,5)==0)
       printf ("\n");     // Переход на новую строку
     }
printf ("\n Для завершения программы нажмите любую клавишу\n");
getch();
}
```

Пример2 цикла for

Вычислите S= 1/20 + 2/18 + . . . + 10/2

```
#include <stdio.h>
main()
{
float S=0;
int i,j;
for (i=1,j=20; i<=10; i++, j-=2)
    S+=i/j;
printf (“%f”,S);
}
```

Операторы цикла

Оператор while

Оператор цикла while называется циклом с предусловием и имеет следующий формат:

while (логическое выражение) тело ;

В качестве выражения допускается использовать любое выражение языка Си, а в качестве тела любой оператор, в том числе пустой или составной. Правило выполнения оператора **while** :

1. Вычисляется логическое выражение.

2. Если выражение ложно, то выполнение оператора while заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполняется тело оператора while.

3. Процесс повторяется с пункта 1.

Оператор цикла вида

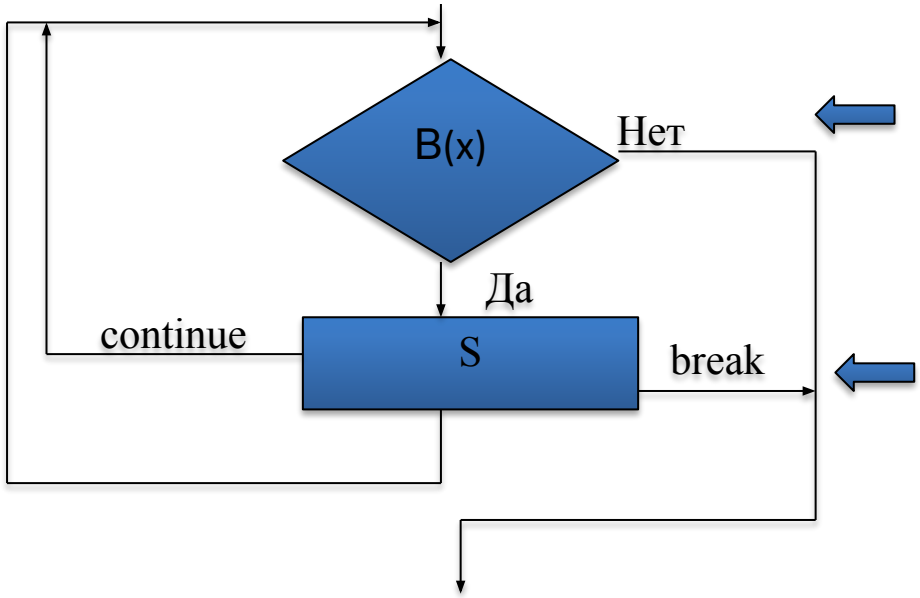
for (выражение-1; выражение-2; выражение-3) тело ;

может быть заменен оператором **while** следующим образом:

```
выражение-1;
while (выражение-2)
{
 тело выражение-3;
}
```

Так же как и при выполнении оператора for, в операторе while вначале происходит проверка условия. Поэтому оператор while удобно использовать в ситуациях, когда тело оператора не всегда нужно выполнять.

Блок-схема оператора While



Условие, при котором выполняются итерации цикла

Тело цикла

B(x) – логическое выражение . В том случае, когда это выражение будет иметь значение Ложь, произойдет выход из цикла;

S – один оператор, простой или составной; он должен включать операторы тела цикла, **в том числе** оператор изменения операторов логического выражения B(x)

Пример1 цикла while

Определить значение суммы $S=1/x_1+1/x_2+\ldots1/x_n$, где n – количество слагаемых.

```
# include <stdio.h>
void main()
    {
   float  x,s=0;
   int i=0,n;
   scanf (“%i”,&n);
    while (i<n)
      {
      i++;
       scanf (“%f”,&x);
      s+=1/x;
      }
printf (“%f”,s)
}
```

Как будет работать программа, если пользователь введет x=0?

Пример1 цикла while

Программа завершит выполнение с сообщением об ошибке (Деление на 0). Как можно этого избежать :

```
# include <stdio.h>
void main()
 {
 float  x,s=0;
 int i=0,n;
scanf (“%i”,&n);
while   (i<n)
        {
        i++;
        scanf (“%f”,&x);
        if (x==0)
                        break;
                s+=1/x;
        }
printf (“%f”,s)
}
```

```
# include <stdio.h>
void main()
 {
 float  x,s=0;
 int i=0,n;
scanf (“%i”,&n);
while  (i<n)
        {
        i++;
        scanf (“%f”,&x);
        if (x==0)
                        continue;
                s+=1/x;
        }
printf (“%f”,s)
}
```

Операторы цикла

Оператор do while

Оператор цикла **do while** называется оператором цикла с постусловием и используется в тех случаях, когда необходимо выполнить тело цикла хотя бы один раз. Формат оператора имеет следующий вид:

do тело while (выражение);

Правило выполнения оператора do while :

1. Выполняется тело цикла (которое может быть составным оператором).
2. Вычисляется выражение.
3. Если выражение ложно, то выполнение оператора do while заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполнение оператора продолжается с пункта 1.

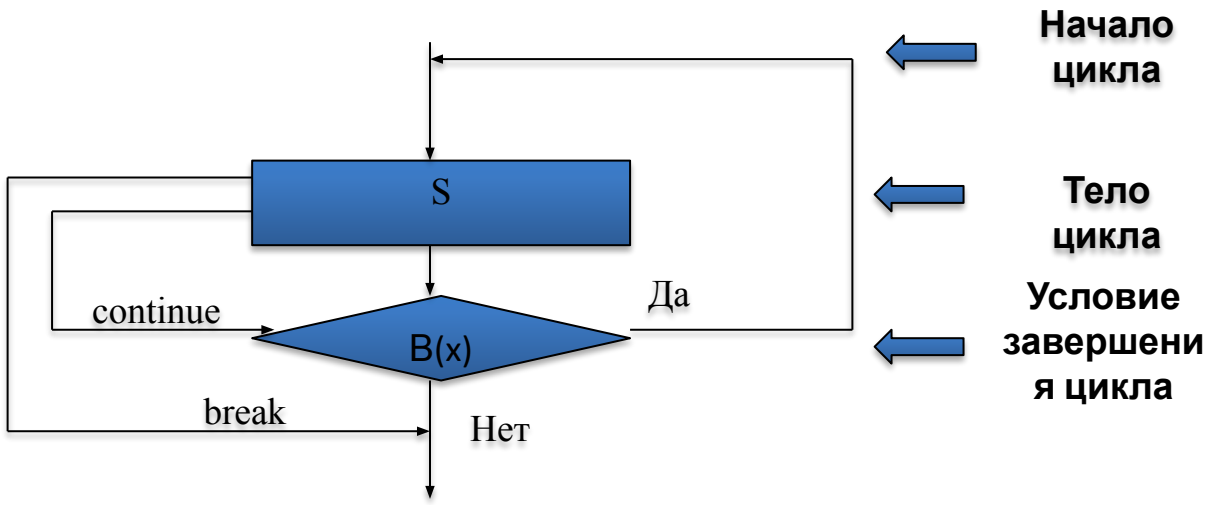
Чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор break.

Операторы **while** и **do while** могут быть вложенными.

Пример:

```
int i, j, k;
    ...
i=0;  j=0;  k=0;
   do { i++;
        j--;
        while (a[k] < i)  k++;
       }
   while (i<30 && j<-30);
```

Блок-схема оператора Do While



B(x) – логическое выражение, при истинности которого происходит выполняется следующий шаг цикла;

S – один или несколько операторов тела цикла.

Пример цикла do while

Дано x>1. Вычислить и вывести степени x; вычисления производятся до тех пор, пока вычисляемое значение станет $>10^8$

```
# include <stdio.h>
#include <math.h>
# include <conio.h>
main()
{
float x;
long int y=1;
int k=0;
clrscr();
printf (“\n Введите значение для возведения его в степень :\n”);
scanf (“%f”,&x);
do
    {
    y*=x;
    k++;
    printf (“\n %.0f в степени %i =%ld”,x,k,y);
    }
while (y<=1e8);
printf (“\n Для завершения программы нажмите любую клавишу”);
getch(); }
```

Задачи.

Задача 1.