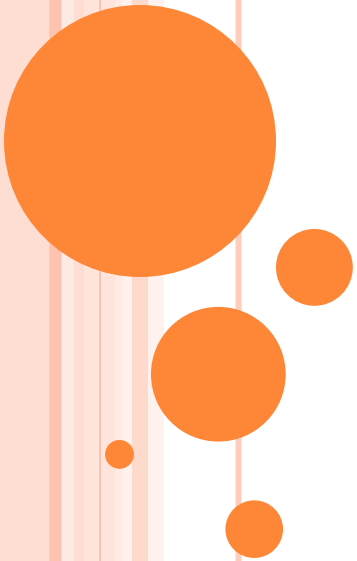


Операции и операторы

Лекция №3

План лекции

- Операции
  - Арифметические
  - Отношения
  - Логические
  - Прочие
  - Приоритеты
- Операторы
  - Оператор-выражение с ;
  - Оператор вызова функции
  - Операторы условного перехода
  - Операторы цикла
  - Вспомогательные операторы

Исходные предпосылки

- Операции – действия над операндами – переменными и константами
- Результат операции должны присваиваться переменной, передаваться на вход функции или передаваться в поток вывода, иначе он теряется
- Все операции имеют приоритеты, приоритетами можно управлять
- Операторы нужны для управления ходом выполнения программы

Операции

- Действия над переменными и константами
  - Унарные (1 операнд)
  - Бинарные (2 операнда)

- Виды операций
  - Арифметические + - * / % ++ --
  - Отношения < > <= >= == !=
  - Логические && || !
  - Адресные & [] *
  - Присваивание = += -= *= /=

Приоритеты

Ранг	Операции
1	() []
2	! -- ++ &
3	* / %
4	+ -
5	< <= >= >
6	== !=
7	&&
8	||
9	= += -= *= /=

Примеры выражений

- int a=3,b=5,c,d
- c=a+b*2-3*(a-b);//результат выражения
- // присваивается переменной - сохраняется
- b+=5;
- a++;
- d=b/a+b%a;

- b-a+c;//не имеет смысла - результат теряется

- 5 = a;//Ошибка – нельзя ничего присвоить константе
- b+3=c; //Ошибка – нельзя ничего присвоить выражению

Области применения операций

- Арифметические
  - вычисления
- Отношения, логические
  - условные операторы
  - циклы
- Адресные
  - Работа с массивами и указателями

Операторы

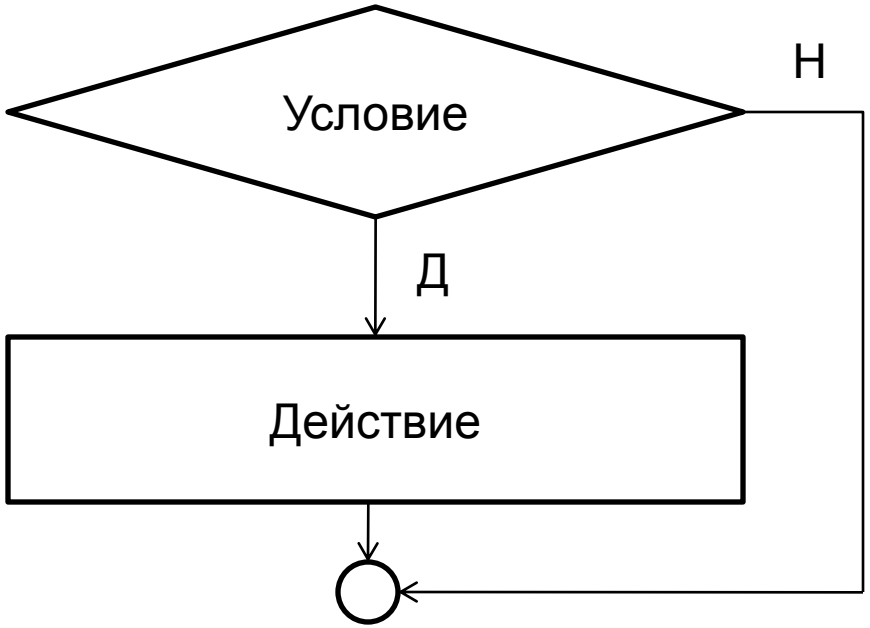
- Оператор-выражение с ; (может содержать вызовы функций, например, sin(x), log10(100))
- Условные
  - Условный переход if
  - Переключатель switch
- Циклические
  - параметрический for
  - цикл с предусловием while
  - цикл с постусловием do … while
- Прерывание цикла break
- Переход к следующему повтору цикла continue
- Возврат из функции return
- Пустой оператор ;

Назначение условного оператора

- Выбор варианта выполнения программы в зависимости от результата вычисления выражения
- Защита программы от аварийного завершения (попытка деления на ноль, попытка извлечения кв. корня из отрицательного числа и т.д.)
- Принудительный выход из цикла
- Принудительный переход к следующей итерации цикла

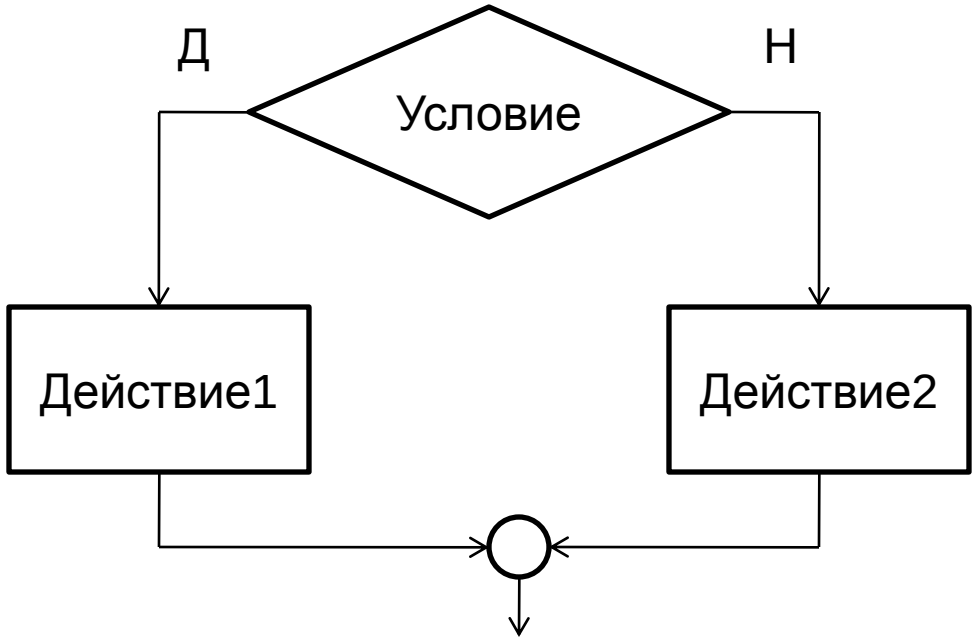
- Имеет краткую и полную форму
- Выходит за рамки программирования в режиме калькулятора

Условный переход (краткая форма)




```
if(условие)
    {
        действие;
    }
```

Условный переход (полная форма)




```
if(условие)
    {
        действие1;
    }
else
    {
        действие2;
    }
```

Пример условного перехода №1

```
int a=2,b=3,c=0;
if(a>b)
    {
        c=b;
    }
else
    {
        c=a;
    }
```

Пример условного перехода №2

```
int a=2,b=3,c=0;
if(a==0)
    {
        printf(“Div by 0!\n”);
    }
else
    {
        c=b/a;
    }
```

Примеры ошибок в операторе if

- if(a>b); //пустое тело
- {printf (“%d”,a);}//действие всегда выполняется
- //голова оторвана от тела!!!
- ; надо расставлять обдуманно только после законченных действий
- ; нельзя ставить после заголовков «на всякий случай»

- if (a=0) //условие никогда не выполняется – не имеет смысла
- { }

- if (a=1) //условие всегда выполняется – не имеет смысла
- { }
- Нельзя путать операцию присваивания с операцией проверки операндов на равенство
- При проверке на равенство операнды НЕ изменяются, а при присваивании левый операнд всегда изменяется

Логическое объединение условий

- И – одновременное выполнение условий
  - if(x>0 && x<10) { }
- ИЛИ–выполнение хотя бы одного из условий
  - if(x=<0 || x>=10) { }

- Примеры применения
  - Проверка принадлежности переменной заданному диапазону
  - Проверка нескольких признаков (является ли число одновременно четным и положительным)

Назначение операторов цикла

- Многократное повторение одних и тех же действий
  - Заранее заданное количество раз
  - При выполнении определенных условий
  - До наступления какого-либо события

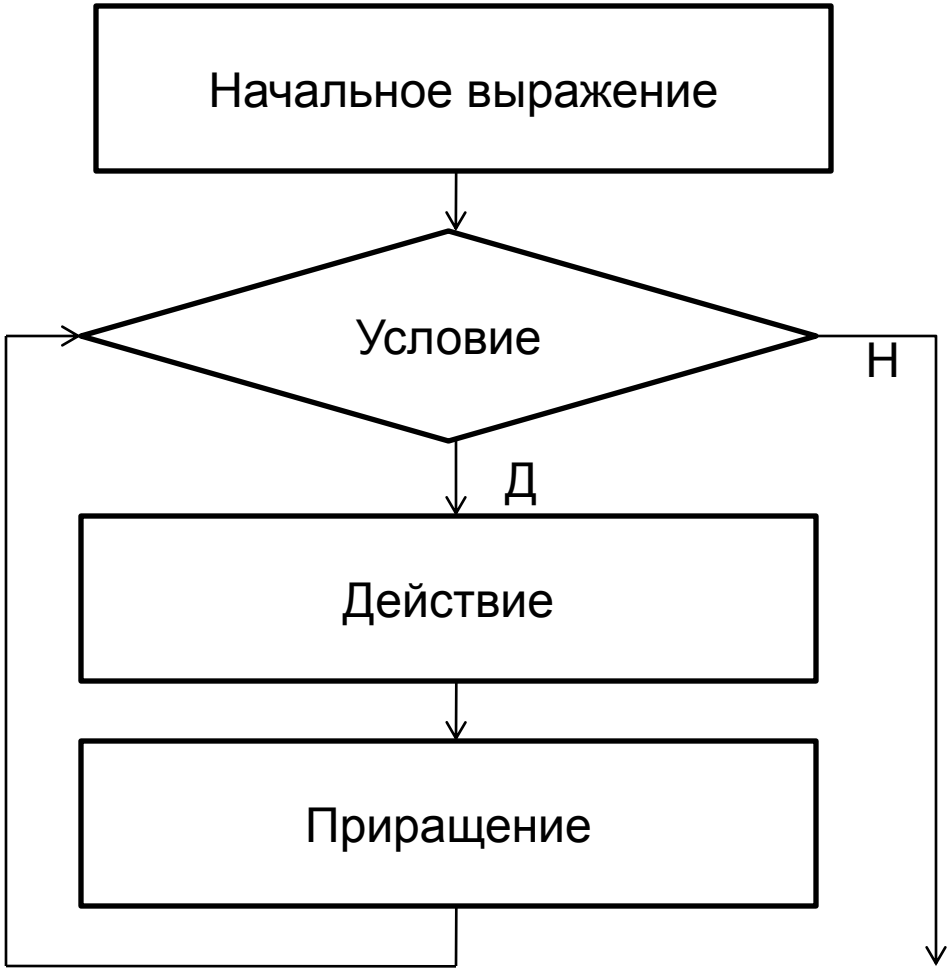
- Примеры применения
  - Обработка большого объема экспериментальных данных
  - Цифровая обработка изображения из многих точек
  - Цифровая обработка электрического сигнала из многих отсчетов
  - Анализ сообщения, принятого по цифровому каналу связи

Виды операторов цикла

- Параметрический for
  - Количество циклов заранее известно
- Цикл с предусловием while
  - Количество циклов заранее неизвестно
- Цикл с постусловием do … while
  - Количество циклов заранее неизвестно

- Операторы управления работой цикла
  - break – принудительный выход
  - continue – переход к следующей итерации

Параметрический цикл




```
for(нач_выр;усл;прир)
    {
       действие;
    }
```

Примеры параметрического цикла

```
int i=0, s=0;
for(i=1; i<10; i++)
 {
   s=s+i;
 }

for(i=3; i<10; i=i+2)
 {
   printf(“%d”,i);
 }
```

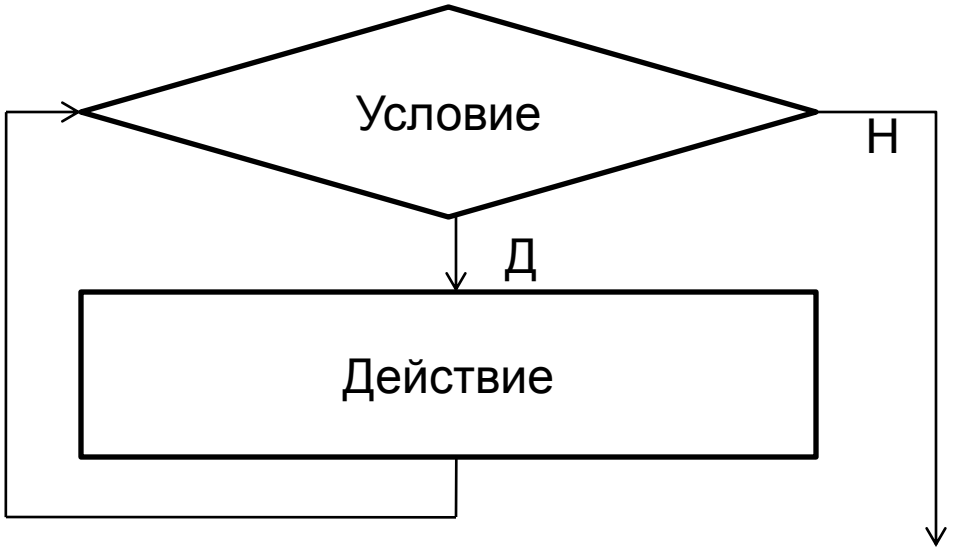
```
for(i=10; i>=1; i--)
 {
   printf(“%d ”,i);
 }

//цикл с ошибками
for(i=0, i<5; i++)
 {
   printf(“%d ,i)
```

Нестандартные варианты параметрического цикла

- Бесконечный
  - for(; ;) { if (условие) break;} //ожидание события
  - //применяется в графических приложениях при событийно-ориентированном подходе – из бесконечного цикла вызываются обработчики событий
  - for(i=0;i<10 ;i--) { }//по ошибке
- Не выполняющийся ни разу
  - for(i=0;i>10;i++) { } //по ошибке
- Пустой цикл (временная задержка)
  - for(i=0;i<10 ;i++); { } //тело оторвано, будет выполняться для значения i, при котором условие продолжения цикла не выполняется – риск получения ошибочных результатов
- Каким является цикл
  - for(i=0;i>5 ;i++) { }

Цикл с предусловием

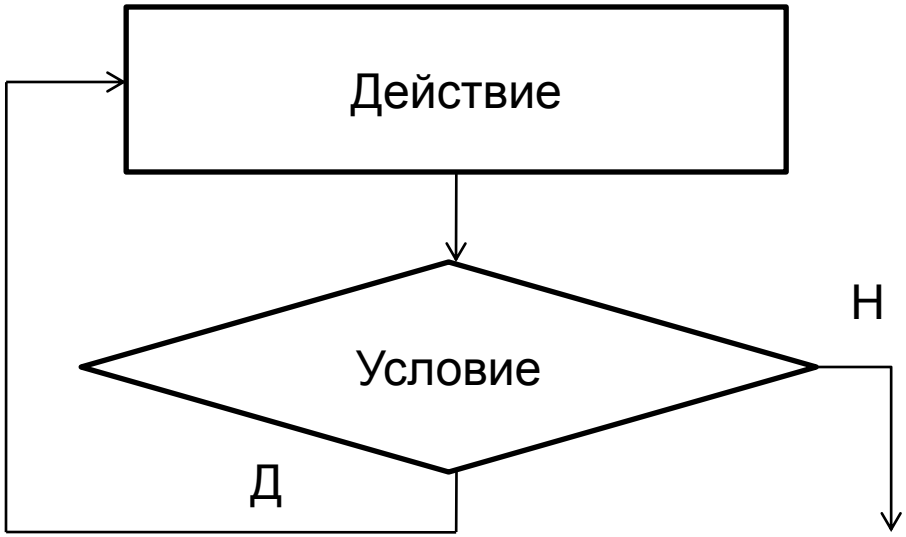



```
while(условие)
    {
        действие;
    }
```

Примеры цикла с предусловием

```
int f;
while (1)
{
   printf(“Введите f”)
   scanf(“%d”,f);
   if(f>0)
         break;
   else
         printf(“Значение f должно быть
   положительным”);
}
```

Цикл с постусловием




```
do
    {
        действие;
    }
while(условие);
```

Примеры цикла с постусловием

```
int f;
do
{
   printf(“Введите положительное f”);
   scanf(“%d”,f);
   } while (f<=0);//ожидание правильных данных
```

Цикл в цикле

```
int i,j

for (i=0;i<3;i++)
{  printf(“%d”,i);
   for(k=1;k<=4;k++)
         {//во внутреннем цикле переменная
         //внешнего цикла не изменяется
         printf(“%d ”,k);
         }
   printf(“\n”);
}
```

Переключатель

```
switch(выражение)
    {
    case 1: действия 1; break;
    case 2: действия 2; break;


    case N: действия N; break;
    default: действия;
    }
```

Пример переключателя

```
int f,a=5,b=2,c;
printf(“Введите номер операции)”);
scanf(“%d”,f);
switch(f)
{case 1: c=a+b; break;
 case 2: c=a*b; break;
 case 3: c=a-b; break;
 default: c=0;
}
```

Выводы

- В Си есть необходимый набор операторов для реализации программы любой сложности
- Операторы ветвления и цикла состоят из заголовка и тела
- Оператор if имеет краткую и полную форму
- Цикл for подходит для заранее известного числа итераций
- Циклы while и do подходят для неизвестного числа повторений
  - Цикл while может не выполниться ни разу
  - Цикл do выполняется хотя бы один раз
- Операторы ветвления и цикла могут быть вложенными