# Управляющие конструкции языка С.

# Операторы

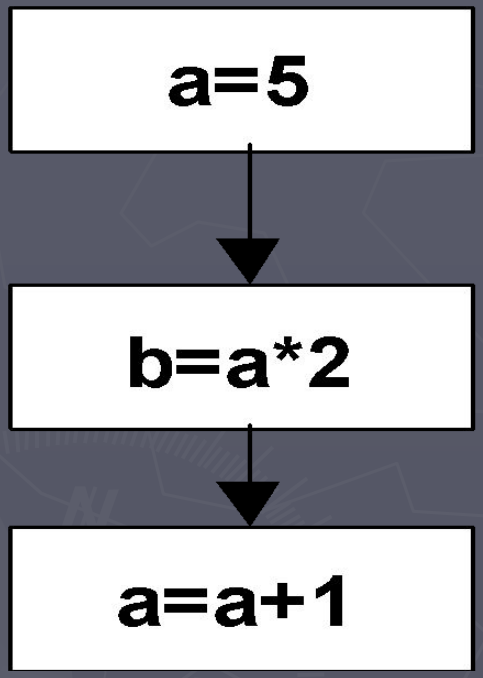
-операторы простой последовательности действий;

-условные операторы;

-операторы цикла;

-операторы перехода.

# Операторы простой последовательности




**Текст программы:**

```
a=5;
b=a*2;
a=a+1;
```

```
for (i=0; i<n-1; i++)
   {       // Составной оператор - блок
    int c;
    c = A[i];
    A[i]=A[i+1];
    A[i+1]=c;
   }
```

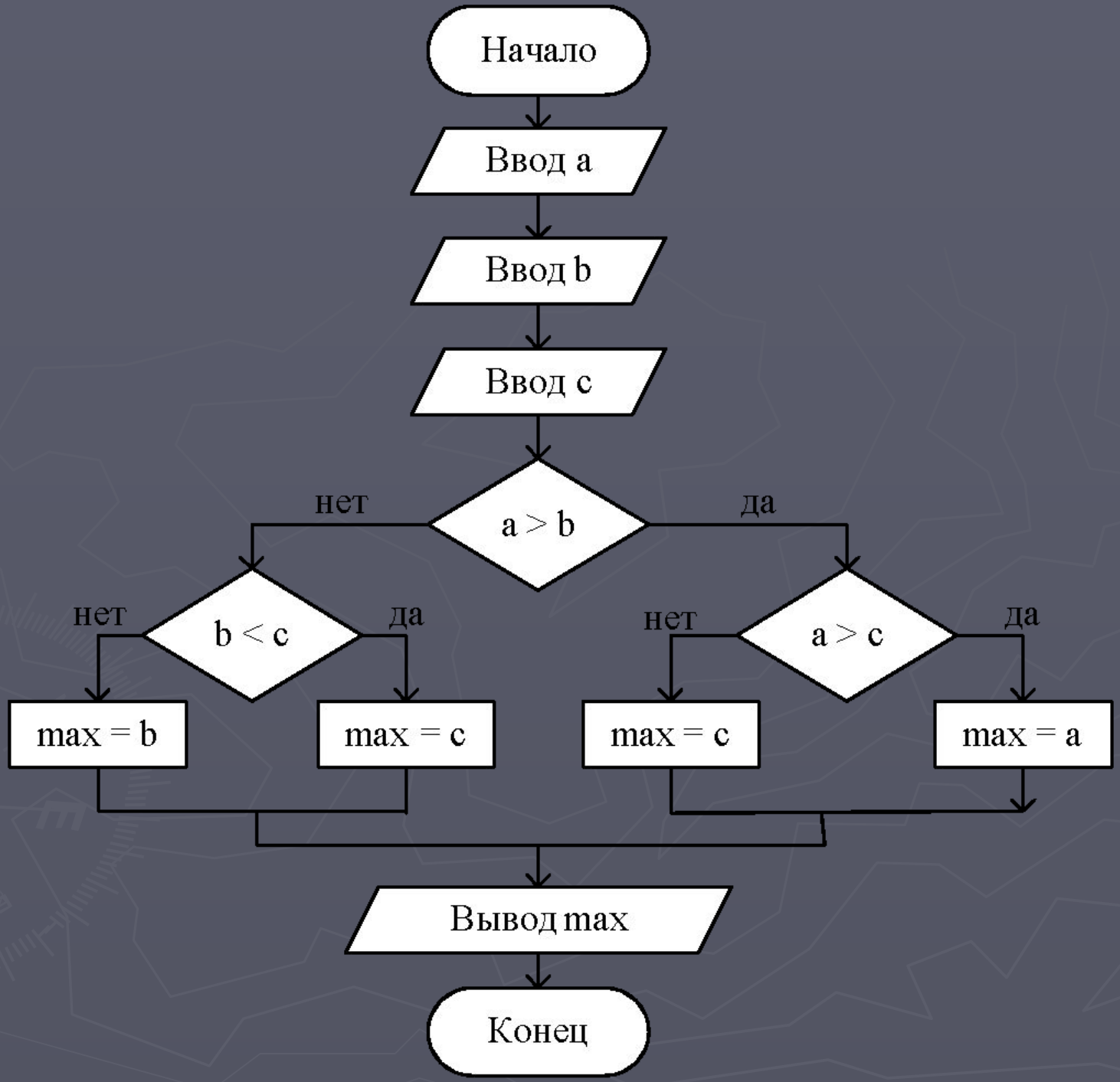
- выражение ;  - оператор
- ;  - пустой оператор
- { оператор ... оператор } - составной оператор (блок)

# Условные операторы

```
if (выражение) оператор_1;
   else            оператор_2;
```

```
if (выражение) оператор_1;
```


Условие
Ложь
Истина
операторы
операторы
операторы


Начало
Ввод a
Ввод b
Ввод c
a > b
нет
да
b < c
нет
да
a > c
нет
да
max = b
max = c
max = c
max = a
Вывод max
Конец

```
if (x>0) j=k+10;
   else m=1+10;
```

```
if(n>0)   //n=5, z=0,a=1,b=2, z-?
   if(a>b)  z=a;
    else  z=b;
```

Z=2

```
if(n>0) //n=5, z=0,a=1,b=2, z-?
   {if(a>b)  z=a; }
    else  z=b;
```

Z=0

# Организация множественного выбора

```
if (выражение1) оператор_1;
    else if (выражение2) оператор_2;
        else if (выражение3) оператор_3;
            else оператор_4;
```

- Если не используются фигурные скобки, то else соответствует ближайшему if
- Любое количество конструкций else-if

```
if  (n<0) printf (“ отрицательное\n”);
   else if (n==0) printf (“нулевое\n”);
      else printf (“положительное\n”);
```

```
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 ...
```

# Тернарный оператор и условие

c = a > b ? a : b; //присвоить с максимум из

```
if(a > b)
   c = a;
else
   c = b;
```

# Операторы перехода

- return
- continue
- goto
- break

# оператор **return**

- производит досрочный выход из текущей функции. возвращает значение результата функции.
- может быть в любом месте функции
- может быть несколько

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

# оператор **break**

- производит альтернативный выход из самого внутреннего блока, то есть переходит к первому оператору, следующему за текущим оператором блока

```
for(s = 0, i = 1; i < 100; i++)
      {
      cin >> x;
      if( x == 0) break;
   // если ввели 0, то сумм. заканчивается
      s += x;
    }
```

# оператор **continue**

- выполняет переход из тела цикла к его повторяющейся части, то есть досрочно завершает текущий шаг и переходит к следующему;
- Не может использоваться в switch

```
continue;
```

```
// обработка положительных элементов

for (i=0;i<n;i++) {
    if (a[i]<0) continue;
        ……
        }
```

# Операторы перехода

goto метка;

оператор

```
    goto mmm:
    ...
mmm:    оператор
```

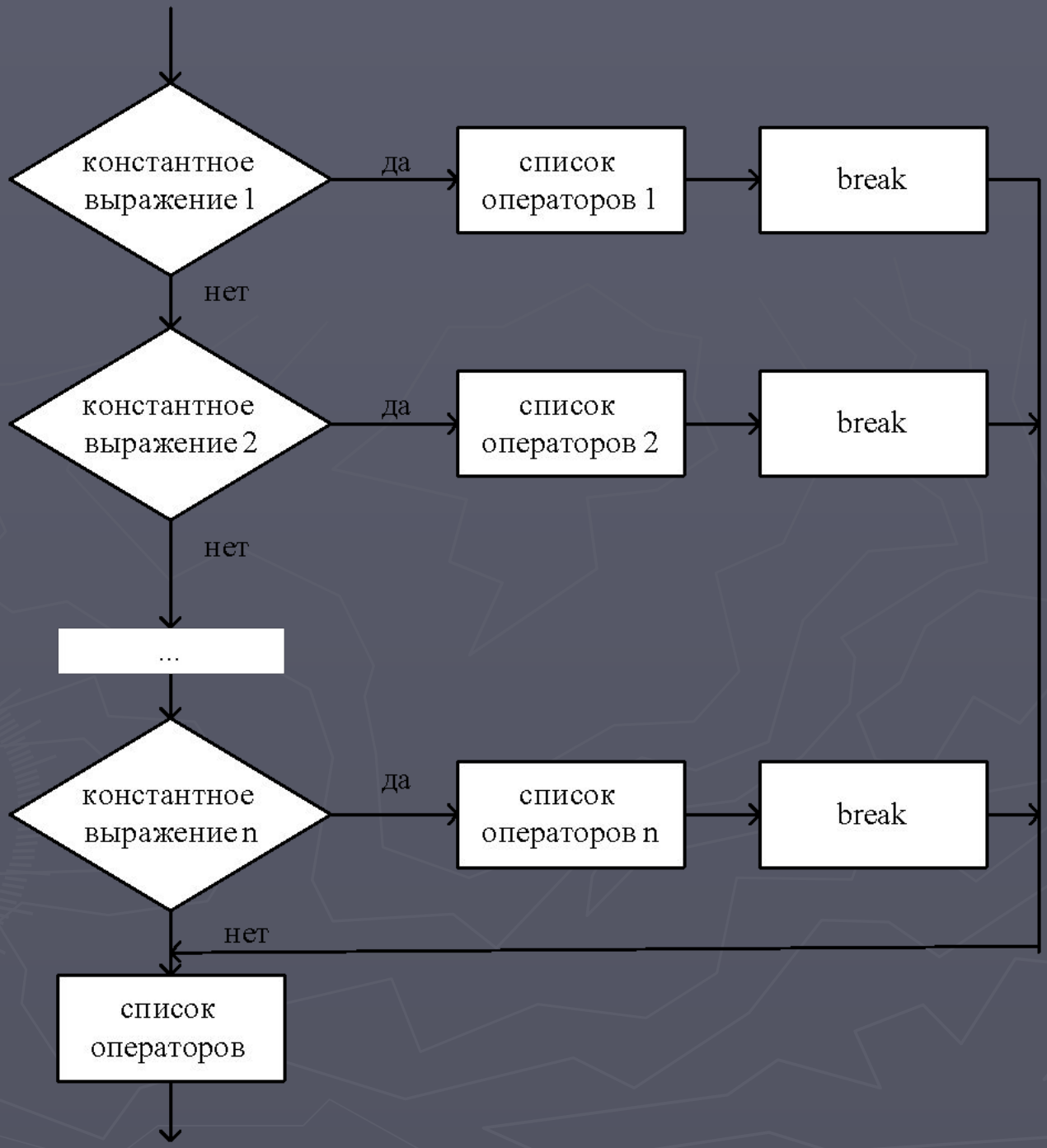
```
for (...)
   for(...)  {
        if (ошибка) goto Error;
    }
```

```
void  F()
   {
for (i=0; i<n; m1: i++)
    {
    if (A[i]==0)  continue; //goto m1;
    if (A[i]==-1) return;   //goto m2;
    if (A[i] <0)  break;    //goto m3;
    }
m3:  ... продолжение тела функции
m2:
}
```

# Оператор выбора альтернативы переключатель

```
switch (выражение)
    {
case константа1: послед операторов_1;break;
case константа2: послед операторов_2 ;break;
case константа3: послед операторов_3 ;break;
default:     последоват операторов ;break;
    }
```

1. Выражение
2. константы – целые или символьные
3. Операторы могут быть простыми и составными


константное выражение 1
да
список операторов 1
break
нет
константное выражение 2
да
список операторов 2
break
нет
...
константное выражение n
да
список операторов n
break
нет
список операторов

```
switch (n)

        {
case 1:   n=n+2; break;
case 2:   n=0;   break;
case 4:   n++;   break;
default:
        n=-1;
        }
```

Эквивалент

```
if (n==1) goto m1;
if (n==2) goto m2;
    if (n==4) goto m3;
    goto md;
    m1: n=n+2; goto m;
m2: n=0;   goto m;
m3: n++;   goto m;
md: n=-1;
   m: …
```

```
switch (c)
        {
case ' ':  ... break;
case '+':  ... break;
case '-':  ... break;
        }
```

Эквивалент

```
if (c==' ') {...}
 if (c=='+') {...}
   if (c=='-') {...}
```

```
    switch (a)
{
   case 1: b=c; break;
   case 2:
    switch (d)
        {   case 0:  f=s;  break;
                 case 1:  f=9;  break;
                  case 2:  f-=9; break;
            }
    case 3: b-=c; break;
 :
 }
```

# Операторы цикла

Цикл – организованное повторение некоторой последовательности операторов

- оператор цикла с предусловием
- оператор цикла с постусловием
- оператор цикла с предусловием и коррекцией

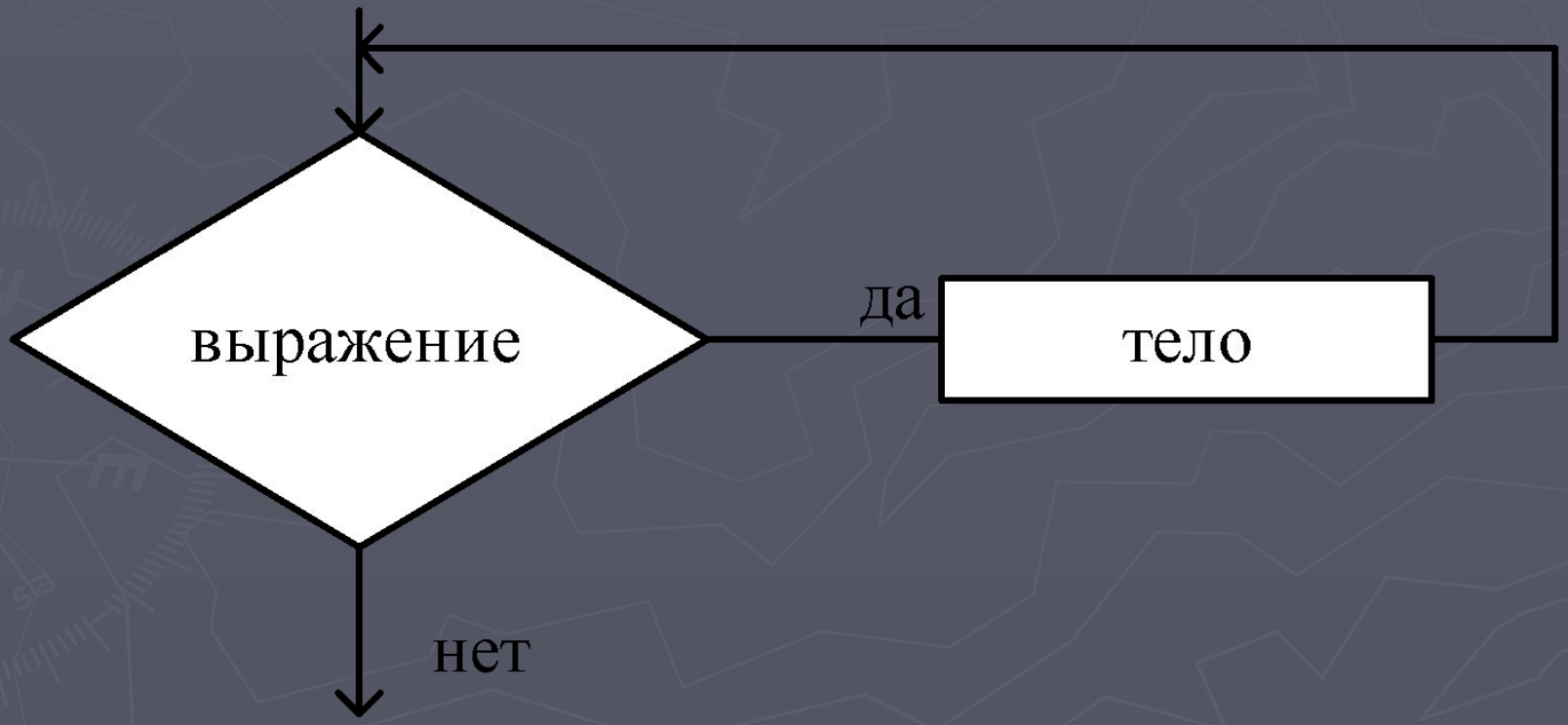
while, do while, for

код цикла

- начальных установок
- модификации параметра цикла
- проверки условия продолжения цикла

Итерация – один проход цикла

while (выражение)  оператор ;

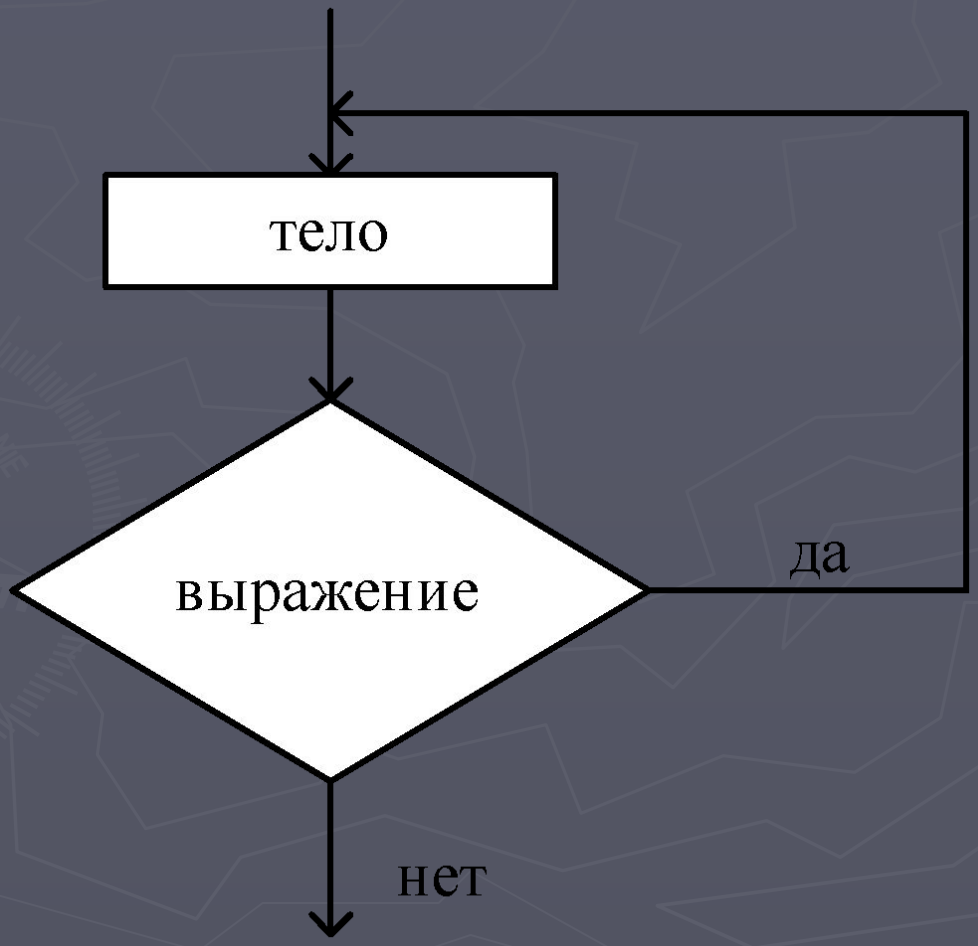


- Схема выполнения оператора while следующая:

1. Вычисляется 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.

do оператор while(выражение);

тело цикла условие продолжения



- Схема 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.

```
count=0; my_max=10;
while(count<my_max)  {   // нет изменения
        …..
        }
count=0; my_max=10;
while(--count<my_max)  {  //изм. не в ту сторону
        …..
        }
count=0; my_max=10;
while(++count<my_max)  { // правильно
        …..
        }
```

# Пример вложенности

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

# ЦИКЛ FOR

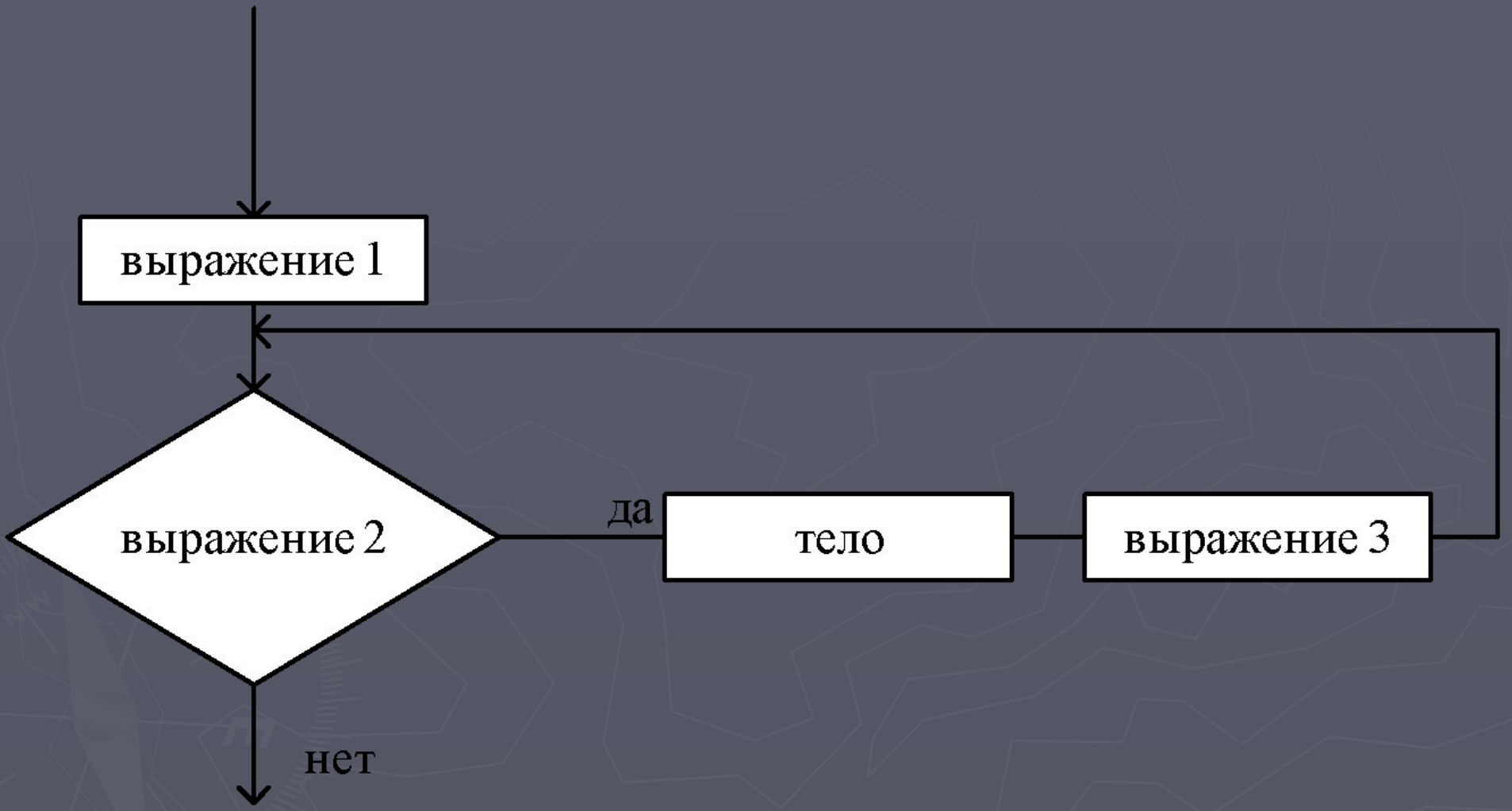
Заголовок цикла Тело цикла

for (выраж_1; выраж_2; выраж_3) оператор

инициализация условие продолжения повторяющаяся часть

Эквивалент

выраж1; while (выраже2) { оператор выраж3; }


выражение 1
выражение 2
да
тело
выражение 3
нет

- *Схема 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.

# Особенности for

1. Выполняется фиксированное число раз
2. for ( инициализация ; проверка условия ; управление циклом )

выполняется 1 раз до

выполняется перед кажд

выполняется в конце каждого

оператор простой или составной

- 3. все части могут быть пустыми . Наличие ;; обязательно

```
for (i=1; i<100000; i++)
  ;
```

# Пример:

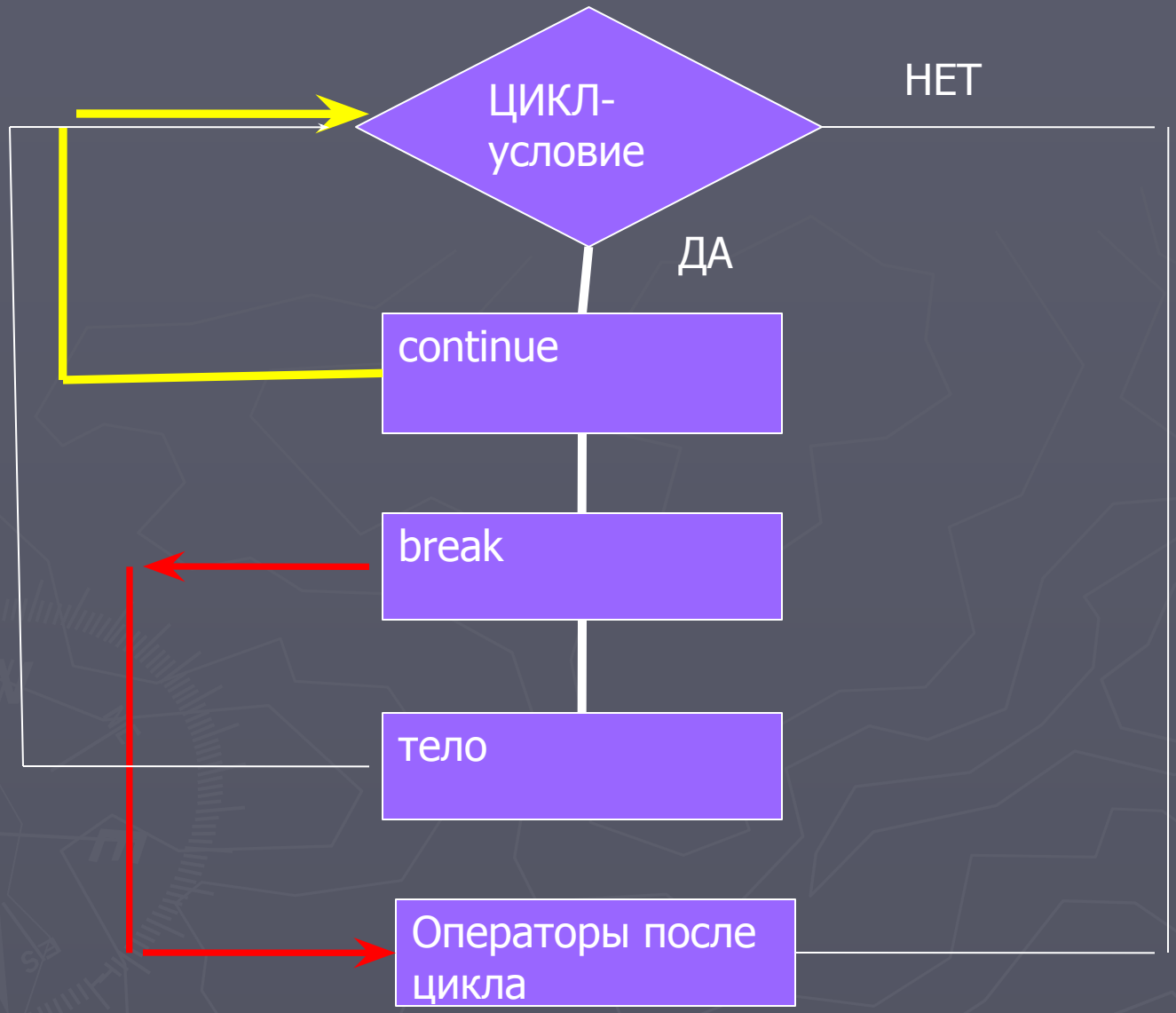
```
int i, b;
 for (i = 1; i < 10; i++) b = i * i;
   //вычисление квадратов
   //чисел от 1 до 9
 for (i = 1; i > 10; i++) b = i * i;
   //тело цикла не выполнится
```

- Желательно в разделе задания начальных значений и изменения переменных структуры for задавать только выражения, относящиеся к управляющей переменной

```
int   top,  bot;
char string[100],  temp;
      // для управления циклом
      //используются две
      // переменные top и bot
 for (top = 0, bot = 100; top < bot;
                top++, bot--)
      {  temp = string[top];
         string[top] = string[bot];
      string[bot] = temp;
           }
```

- Управлять количеством повторений цикла нужно с помощью целой переменной

```
for (int  n=0; s[i]>='0'
          &&  s[i]<'9';
          i++) ...;
for (;;)        { ...          ...
  break;          ...          }
```


ЦИКЛ-
условие
НЕТ
ДА
continue
break
тело
Операторы после
цикла

# Принцип вложенности

- Вложенность операторов -- любой сколь угодно сложный оператор (фрагмент блок-схемы) может быть представлен в другом операторе в виде элементарного действия.

# Пример:

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

# Определить значения переменных по окончании цикла

```
for(i=0,j=20;i<j; i++,j--);
```

//0..9  20..11

# Цикл не выполняется

```
for(i=0;  0;   i++);

// не выполнится
```

# Бесконечный цикл

```
for(i=0;i>=0; i++);


//бесконечно
```

# Использование ,

```
for( n=16,  i=0;  n!=1;   i++,  n/=2);

//i=4
```

# Что вычисляется в цикле?

```
for(s=0, i=1; i<100;i++)
{    cin>>x;
     if( x==0) break;
     s+=x;
}
```

# Определить численные значения всех переменных?

```
for (i=0, d=0; i<10; i++, d =!d);

// i= 9   d=1
```

# Перепишите без break

```
for (i=0; i<n; i++)
   { if (..a[i]...) break; ... }
```