

Лекция №5

Операторы языка Си

Операторы языка.

Оператор-выражение

Выражение становится оператором, если за ним стоит точка с запятой « ; ».

i++ ;

A=b+3 ;

printf(..) ;

Точка с запятой в языке Си является признаком конца оператора.

Операторы языка.

Пустой оператор

;
; - пустой оператор.

Используется, когда по синтаксису должен быть хотя бы один оператор, но его нет.

Операторы языка.

Составной оператор

Составной оператор – набор операторов, выполняющихся последовательно и составляющих единое целое.

```
{  
    Оператор 1;  
    Оператор 2;  
    ...  
}
```

Операторы языка.

Условный оператор (1)

```
if (выражение)  
    Оператор 1;  
else  
    Оператор 2;
```

```
if (выражение)  
{  
    блок операторов_1  
}  
else  
{  
    блок операторов_2  
}
```

Операторы языка.

Условный оператор (2)

if (N>0)

if (A>=B)

Z=A;

else

Z=B;

if (N>0)

if (A>=B)

Z=A;

else

Z=B;

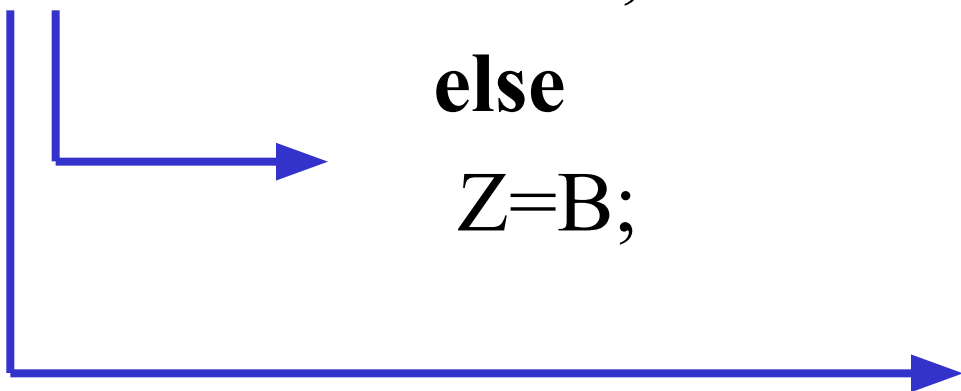
if (N>0)

if (A>=B)

Z=A;

else

Z=B;



`else` связывается с ближайшим предыдущим `if`, не содержащим `else`

Операторы языка.

Условный оператор (3)

```
if (выражение1)  
    Оператор 1;  
else if (выражение2)  
    Оператор 2;  
else  
    Оператор 3;
```

Операторы языка.

Условный оператор (4)

if (a<=3)

 Z=a+t;

else if ((a>3) && (a<100))

 Z=a*t;

else if ((a>=100) && (a<500))

 Z=a*2*t;

else

 Z=a;

Операторы языка.

Оператор выбора (1)

switch (выражение)

{

case константное_выражение_1:

оператор;

...

...

case константное_выражение_n:

оператор;

...

default:

оператор;

...

};

Операторы языка.

Оператор выбора (2)

```
char c;
```

```
...
```

```
switch (c)
```

```
{
```

```
    case '1':
```

```
        printf("one");
```

```
        break;
```

```
    case '2':
```

```
        printf("two");
```

```
    case '3':
```

```
    case '4':
```

```
        printf("others");
```

```
};
```

Операторы языка.

Цикл while (1)

while (выражение)
оператор

```
while (выражение)  
{  
    блок операторов  
};
```

выражение – условие цикла

оператор – тело цикла, в котором должно изменяться условие цикла, иначе оператор **while** будет выполняться бесконечно

Операторы языка.

Цикл while (2)

Пример «пустого» while:

```
while (выражение);
```

Пример «бесконечного» while:

```
while (1)
```

```
{
```

```
    if (выражение)
```

```
        break;
```

```
...
```

```
};
```

Операторы языка.

Цикл while (3)

```
scanf(“%d”, &a);  
while (x>0)  
{  
    //действия с участием a, x  
    scanf(“%d”, &a);  
}
```

Операторы языка.

Цикл for (1)

for (выражен_1; выражен_2; выражен_3)

тело цикла

выражен_1 – вычисляется один и только один раз перед проверкой условия цикла.

выражен_2 – задает условие продолжения цикла. Если его значение отлично от нуля, то будет выполнено тело цикла.

После этого будет вычислено **выражен_3**.

Все три выражения, связанные с организацией цикла (инициализация, проверка и модификация), собраны вместе.

Операторы языка.

Цикл for (2)

for (выражение_1; выражение_2; выражение_3)
оператор

for (выражение_1; выражение_2; выражение_3)
{
 блок операторов
};

Операторы языка.

Цикл for (3)

```
for (i=0; i<=n; i++)  
    if (i%2==0)  
        printf("%d\n",i);
```

```
for (i=0; ; i++)  
{  
    printf("%d\n",i);  
    if (i==n)  
        break;  
}
```


Операторы языка.

Цикл do-while (1)

do

оператор

while (выражение);

do

{

блок операторов

}

while (выражение);

Тело цикла выполняется до тех пор, пока значение выражения не станет ложным (равным нулю).

Операторы языка.

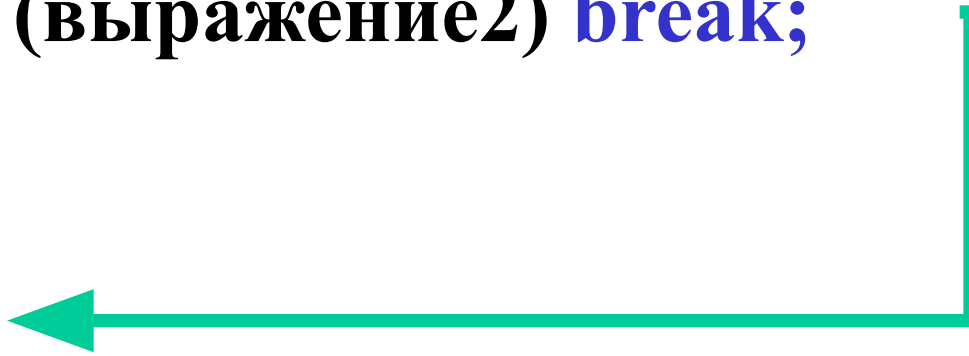
Цикл do-while (2)

Из условия известно, что как минимум первое введенное число будет положительным.

```
do  
{  
    scanf("%d",&a);  
    //действия с участием a  
}  
while (a>0);
```

Операторы языка. BREAK (1)

```
while (выражение1)  
{  
    if (выражение2) break;  
    ...  
};
```



Оператор break вызывает завершение самого внутреннего включающего его оператора while, do-while, for, switch.

Операторы языка. BREAK (2)

```
i=0;  
while (i<=(N-1))  
{  
    if (mas[i]==FindElement)  
        break;  
    i++;  
}
```

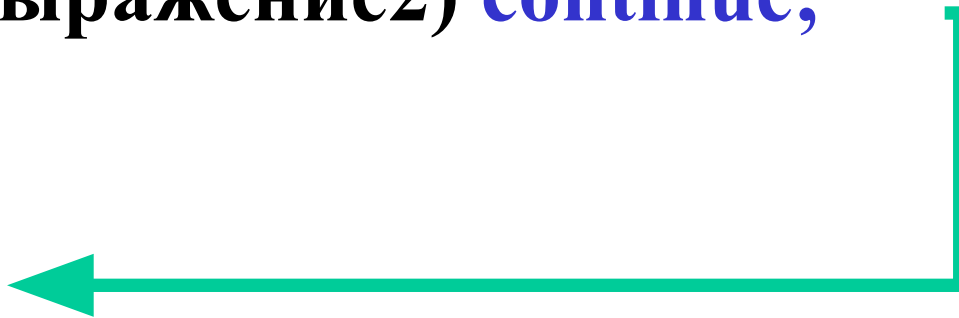
Операторы языка. CONTINUE (1)

```
while (выражение1)
```

```
{  
    if (выражение2) continue;
```

```
...
```

```
};
```



Оператор continue позволяет пропускать оставшуюся часть цикла while, do-while, for и начинать новую итерацию.

Операторы языка. CONTINUE (2)

```
for (i=0; i<=N; i++)  
{  
    if (i%2==0)  
        continue;  
    //ТОЛЬКО для нечетных чисел  
    ...  
    if (...)  
    else if (...)  
    else  
    ...  
}
```

Операторы языка.

GOTO

```
...  
goto метка;  
...  
метка:  
...
```

```
for (...)  
  for(...)  
  {  
    ...  
    if (...)  
      goto метка;  
    ...  
  }  
метка:  
...
```

Оператор перехода по метке может использоваться, например, когда нужно выйти из вложенного цикла