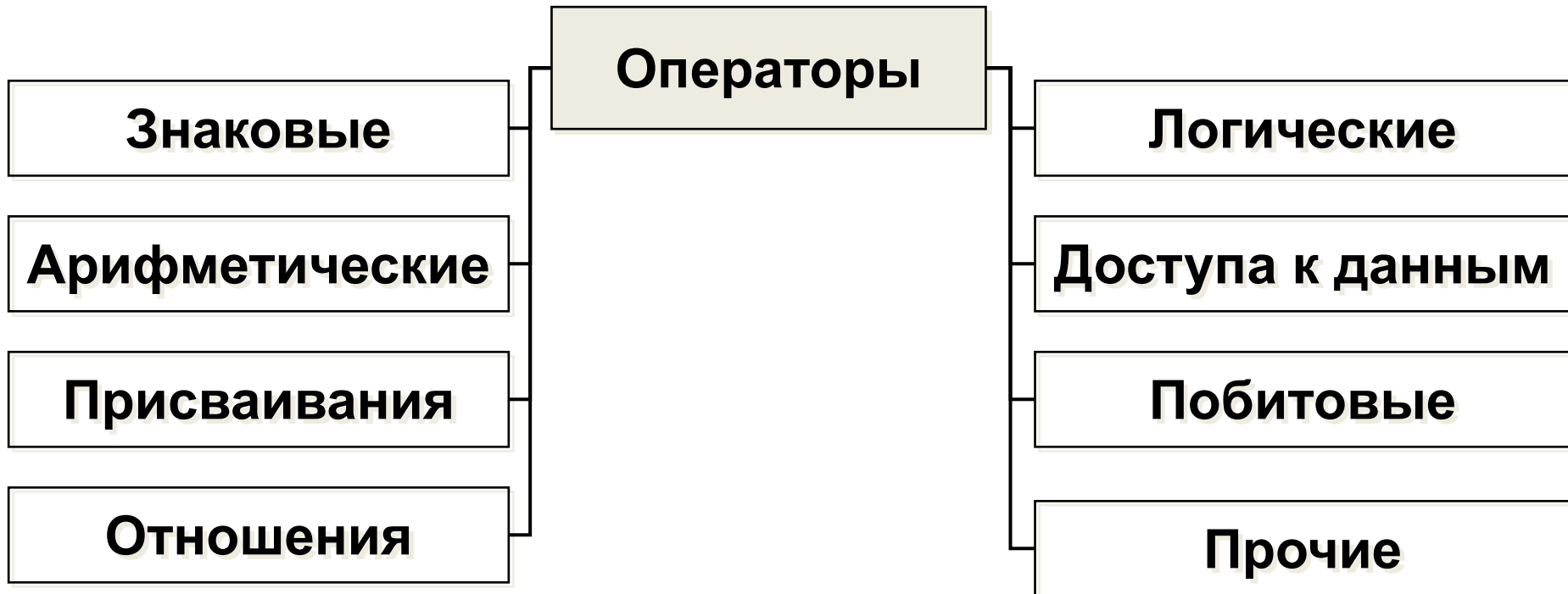


Тема 4.

Операторы

Операторы языка С

Оператор – это команда языка программирования высокого уровня.



Операторы можно разделить на группы в зависимости от того, с каким количеством операндов они работают:
унарные, бинарные и тернарные

Оператор присваивания =

Формат операции простого присваивания

операнд_1 = операнд_2

Сначала вычисляется выражение, стоящее в правой части, а затем его результат записывается в область памяти, указанную в левой части (L-значение).

Примером L-значения может быть имя переменной или разадресованный указатель.

Пример кода

```
int a;      //значение неопределенно
a = 25;     //a ← 25
int b = 5;  //b ← 5 (инициализация)
a = b = 1;  //a ← 1 и b ← 1
```

Знаковые операторы (унарный «+» и унарный «-»)

Оператор	Назначение
+	Значение операнда без изменения знака
-	Значение операнда с противоположным знаком

Унарными операторами "+" и "-" можно присваивать знаки величинам арифметических типов. Если перед величиной не указан знак, то значение по умолчанию считается положительным.

Пример кода

```
int a = -5;  
float b = +45.67; // float b = 45.67;  
double c = -b;    // c <- -45.67
```

Арифметические операторы

Бинарные операторы	Назначение
+	Суммирует два операнда
–	Вычитает из первого операнда второй
*	Умножает два операнда
/	Делит первый операнд на второй
%	Остаток от целочисленного деления первого операнда на второй

Операции умножения, деления и остатка имеют более высокий приоритет, чем операции сложения и вычитания. При равном приоритете операторы обрабатываются в последовательности слева направо. Можно изменить порядок выполнения операторов с помощью скобок.

Арифметические операторы

Пример кода

```
int a = 5+3;           //a <- 8
int b = a-2;           //b <- 6
int c = a*b;           //c <- 48
int d = c/10;          //d <- 4
int e = c%10;          //e <- 8
int f = 2+3*4-5;       //f <- 9
int g = (2+3)*(4-5);   //g <- -5
double h1 = 48/10;     //h1 <- 4
double h2 = 48./10.;   //h2 <- 4.8
```

Побитовые операторы

Оператор	Назначение
&	Побитовое И
	Побитовое ИЛИ
^	Побитовое исключающее ИЛИ
~	Инвертирование битов (унарный)
>>	Сдвиг вправо на заданное кол-во разрядов
<<	Сдвиг влево на заданное кол-во разрядов

0	1	1	0	0
Оператор &				
0	0	1		
1	1	0	1	1
	1	1	1	
0	1	0	0	0
	0	0	1	

0	1	1	0	0
Оператор				
0	0	1		
1	1	0	1	1
	1	1	1	
1	1	1	1	1
	1	1	1	

0	1	1	0	0
Оператор ^				
0	0	1		
1	1	0	1	1
	1	1	1	
1	0	1	1	1
	1	1	0	

0	1	0	0	0
Оператор σ				
	0	0	1	
1	0	0	1	1
	1	1	0	

Побитовые операторы

Пример кода

```
unsigned char a = 1, b = 2;  
unsigned char c = a & b;    //c <- 0  
unsigned char d = a | b;    //d <- 3  
unsigned char e = a ^ b;    //e <- 3  
unsigned char f = ~a;       //f <- 254  
unsigned char i = b << 3;   //i <- 16  
unsigned char j = 7 >> 1;   //j <- 3
```


Составные операторы

Бинарные операторы	Назначение	Результат
+=	Суммирует два операнда	Результат присваивается первому операнду
-=	Вычитает из первого операнда второй	
*=	Умножает два операнда	
/=	Делит первый операнд на второй	
%=	Остаток от целочисленного деления первого операнда на второй	
другие	<<=, >>=, &=, ^=, =	

В языке C существуют составные операторы присваивания, с помощью которых осуществляется как преобразование данных, так и присваивание результата.

Составные операторы присваивания

Пример кода

```
int a = 5+3; //      a <- 8
a += 2;    //a = a + 2  a <- 10
a *= 7;    //a = a * 7  a <- 70
a /= 20;   //a = a / 20 a <- 3
a %= 2;    //a = a % 2   a <- 1
a -= 5;    //a = a - 5   a <- -4
```

Операторы отношения

Бинарные операторы	Назначение
<code>==</code>	Сравнение на равенство
<code>!=</code>	Сравнение на неравенство
<code><</code>	Сравнение на "меньше"
<code>></code>	Сравнение на "больше"
<code><=</code>	Сравнение на "меньше или равно"
<code>>=</code>	Сравнение на "больше или равно"

Операторы отношения сравнивают два операнда, которые также могут быть представлены выражениями.

Результатом этих операций всегда является значение **true** или **false**.

Операторы отношения

Пример кода

```
int a = 2, b = 3, c = 4, d = 5;  
bool e = (b+d == a*c); //true  
bool f = (2*a != c);    //false  
bool g = (d > b);      //true  
bool h = (d < b);      //false  
bool i = (b-a >= -2);  //true  
bool j = (-2 <= a-c);  //true
```

Логические операторы

Операторы	Назначение
&&	Логическое И (бинарный)
 	Логическое ИЛИ (бинарный)
!	Логическое НЕ (унарный)

Оператор &&		
Операнды		Результат
1	2	
true	true	true
true	false	false
false	true	false
false	false	false

Оператор		
Операнды		Результат
1	2	
true	true	true
true	false	true
false	true	true
false	false	false

Оператор !	
Операнд	Результат
true	false
false	true

Логические операторы

Пример кода

```
bool a = true, b = false;  
bool c = a && b;    //false  
bool d = a || b;    //true  
bool i = !a || b;   //false  
bool j = !(a && b); //true
```

Операторы инкремента и декремента

Унарные операторы	Назначение
L- значение++	Инкремент (увеличение на 1), постфиксная форма
++L- значение	Инкремент (увеличение на 1), префиксная форма
L-значение--	Декремент (уменьшение на 1), постфиксная форма
--L-значение	Декремент (уменьшение на 1), префиксная форма

Позиция инкремента и декремента определяет, какое действие будет выполнено сначала: присваивание значения операнда или его изменение. В префиксной форме операнд изменяется до присваивания. В постфиксной форме значением выражения является исходное значение операнда, после чего он изменяется.

Операторы инкремента и декремента

Пример программы

```
short a = 20;  
short b = a++; //b <- 20 и a <- 21  
short c = ++a; //c <- 22 и a <- 22  
short d = --a; //d <- 21 и a <- 21  
short e = a--; //d <- 21 и a <- 20
```


Оператор запятая

Если символ запятой используется не в списке элементов, то он определяет последовательность вычислений. В этом случае два выражения, разделенные запятой, вычисляются слева направо и значение левого выражения отбрасывается

Пример кода

```
float a = 5.25;
```

```
float b = (a+=1, a*2); //b <- 12.5
```

Данный оператор применяется для обработки нескольких выражений там, где синтаксис разрешает использование только одного выражения.

Приоритет и ассоциативность

Оператор	Ассоциативность
$a++$, $a--$ $++a$ $--a$ $+(унарный)$ $-(унарный)$ $!$ \sim	Слева направо Справа налево
$sizeof$ $*$ $/$ $\%$	Слева направо
$+$ $-$	Слева направо
$<<$ $>>$	Слева направо
$<$ $<=$ $>$ $>=$	Слева направо
$==$ $!=$	Слева направо
$\&$	Слева направо
\wedge	Слева направо
$ $	Слева направо
$\&\&$	Слева направо
$ $	Слева направо
$=$ $+=$ $-=$ $*=$ $/=$ $\%=$ $<<=$ $>>=$ $\&=$ $ =$ $\wedge=$	Справа налево
,	Слева направо