

Язык программирования C++

Лекция 2

- операторы отношения
- логические операторы
- символьные константы
- битовые операции
- операции присвоения
- порядок выполнения операций
- условные операции
- стиль записи

Операторы отношения

Fortran	Операция	C / C++
X .LT. Y	меньше	$x < y$
X .LE. Y	меньше или равно	$x \leq y$
X .GT. Y	больше	$x > y$
X .GE. Y	больше или равно	$x \geq y$
X .EQ. Y	равно	$x == y$
X .NE. Y	не равно	$x \neq y$

Логические операторы

Fortran		C / C++
.FALSE.	ложь	0 или false
.TRUE.	истина	любое не нулевое значение или true
.NOT. X	логическое отрицание	!x
X .AND. Y	логическое умножение	x && y
X .OR. Y	логическое сложение	x y

- Вычисления логических операторов `&&` и `||` производятся слева направо, правая часть выражения может не вычисляться, если это уже не нужно
- Деления на ноль (false) никогда не происходит:
`if (d && (x/d < 10.0)) { ... }`

СИМВОЛЬНЫЕ КОНСТАНТЫ

- Символьные константы в C++ имеют размер 1 байт
- Для их задания используются одиночные кавычки

```
char a = 'b';  
char cc = 'D';
```
- Для задания непечатаемых управляющих и некоторых других символов используются escape-последовательности:
 - '\n' – новая строка
 - '\"' – одиночная кавычка
 - '\"' – двойная кавычка
 - '\?' – вопросительный знак
 - '\ddd' – восьмеричная константа ('\047')
 - '\xdd' – шестнадцатеричная константа ('\x2F')

Битовые операции

Fortran	Операция	C / C++
NOT(I)	дополнение	$\sim i$
IAND(I,J)	умножение	$i \& j$
IEOR(I,J)	исключающее или	$i \wedge j$
IOR(I,J)	сложение	$i \mid j$
ISHFT(I,N)	сдвиг влево	$i \ll n$
ISHFT(I,N)	сдвиг вправо	$i \gg n$

Используются при программировании различной электроники (системы сбора данных, системы выработки триггерных сигналов), а также при работе с упакованными данными

Операции присвоения

Fortran	Операция	C / C++
$X = Y$	присвоение	$x = y$
$X = X + Y$	присвоение с прибавлением	$x += y$
$X = X - Y$	присвоение с вычитанием	$x -= y$
$X = X * Y$	присвоение с умножением	$x *= y$
$X = X / Y$	присвоение с делением	$x /= y$
$X = \text{MOD}(X, Y)$	присвоение с вычислением модуля	$x \% = y$
$X = \text{ISHFT}(X, -N)$	присвоение с битовым сдвигом вправо	$x >> = n$
$X = \text{ISHFT}(X, N)$	присвоение с битовым сдвигом влево	$x << = n$
$X = \text{IAND}(X, Y)$	присвоение с битовым умножением	$x \& = y$
$X = \text{IOR}(X, Y)$	присвоение с битовым сложением	$x = y$
$X = \text{IEOR}(X, Y)$	присвоение с битовым исключающим или	$x \wedge = y$

Порядок выполнения операций

$$z = a*x + b*y + c;$$

- для изменения порядка применяются круглые скобки
- скобки лишними не бывают: в случае сомнений ставьте больше скобок
- пишите формулы простыми для понимания

Условные операции - if

- оператор if аналогичен синтаксису Фортрана и С:

```
if (current_temp > maximum_safe_temp) {  
    cout << "EMERGENCY: Too hot – flushing" << endl;  
    flushWithWater();  
}
```
- допустимо любое численное выражение:

```
if ( !(channel = openChannel("temperature")) ) {  
    cout << "Could not open channel" << endl;  
    exit(1);  
}
```

Условные операции - if

- фигурные скобки не обязательны в случае единственного оператора:

```
if ( x < 0 )
```

```
    x = -x; // т.е. abs(x)
```

```
    y = -y; // а это выражение уже выполняется всегда
```

- оператор может быть записан на одной строчке:

```
if ( x < 0 ) x = -x;
```

- допустимо любое выражение, включая присвоение:

```
int i, j;
```

```
// ... вычисление i и j
```

```
if ( i = j ) { ... }
```

- **типичная ошибка:** переменной *i* присваивается значение *j*, а не производится их сравнение. Выражение в круглых скобках будет истинно при любом *j*≠0

Условные операции - if

- аналогично Фортрану и С конструкция **if – else**

```
if ( x < 0 ) {  
    y = -x;  
} else {  
    y = x;  
}
```

- конструкция **if – elseif – else**

```
if ( x < 0 ) {  
    y = -x;  
} else if ( x > 0 ) {  
    y = x;  
} else {  
    y = 0;  
}
```

Стиль записи

- С++ допускает свободный стиль записи текста программ
- Типичные стили для условного оператора if

```
if ( x < 0 ) {  
    y = -x;  
} else {  
    y = x;  
}
```

или

```
if ( x < 0 )  
{  
    y = -x;  
}  
else  
{  
    y = x;  
}
```

Кроме того, если в фигурных скобках стоит только один оператор, то эту пару скобок можно опустить

Практическая задача (№ C2)

Рассмотрим пример:

```
#include <iostream>

int main( ) {

    int i =1 ;
    std::cout << i << " , ";
    std::cout << (++i) << " , ";
    std::cout << i << " , ";
    std::cout << (i++) << " , ";
    std::cout << i << std::endl;

    return 0;
}
```

Какой получился вывод на экран ? Почему ?

Как он поменяется при замене операций инкремента на операции декремента ?