Объектно-ориентированное программирование на языке C++

Сокращённое присваивание

Инкремент и декремент

Приоритеты операций

Сокращённое присваивание

Вы уже знаете, что:

```
a += 1;  // эквивалентно a = a + 1;
```

Однако их намного больше:

- `+=`
- `-=`
- `*=`
- `/=`
- `%=`

Сокращённое присваивание

Вы уже знаете, что:

```
a += 1;  // эквивалентно a = a + 1;
```

Однако их намного больше:

- `+=`
- `-=`
- `*=`
- `/=`
- `%=`
- `&=`
- `|=`
- `^=`
- `>>=`
- `<<=`

Сокращённое присваивание

Вы уже знаете, что:

```
a += 1;  // эквивалентно a = a + 1;
```

Однако их намного больше:

- `+=`
- `-=`
- `*=`
- `/=`
- `%=`
- `&=`
- `|=`
- `^=`
- `>>=`
- `<<=`

Операторы инкремента и декремента

Синтаксис:

```
i++ / ++i / i-- / --i;
```

Что делает этот оператор:

`i++ / ++i` – прибавляют единицу к переменной i

`i-- / --i` – вычитают единицу от переменной i

В чем разница?

Разница во времени использования в выражениях

++i – сначала увеличивает значение переменной i, затем использует ее значение в выражении.

i++ – сначала использует значение переменной i в выражении, а затем его увеличивает на 1.

Примеры:

```
int i = 0;
cout << ++i << " "
     <<   i << endl;
// Будет выведено 1 1
```

```
int i = 0;
cout << i++ << " "
     << i   << endl;
// Будет выведено 0 1
```

Важно!

Значение, к которому применяется оператор инкремента должно быть изменяемым (т.н. lvalue)

```
(x+5)++;        ++x--;
```

Но: `(--x)++;` // Такая форма разрешена

Использование:

1. Ускорение вычислений
2. Запутывание кода

Пример

Чему равны значения переменных после выполнения программы?

```
int main() {
    int x = 5, p = 5, q = 5, w = 5, k = 5;
    p *= x++;
    q /= ++x;
    w = --x + p--;
    k += (--x)-- + 10;
    return 0;
}
```

А что, если…

```
int i = 5;
i = ++i + ++i;
```

Undefined Behavior

```
5
6  namespace ConsoleApplication12
7  {
8      class Program
9      {
10         static void Main (string[] args)
11         {
12             int i = 0;
13             int z = ++i + ++i + ++i + ++i;
14         }
15     }
16 }
```

z | 10

C#/Java/JavaScript

```
main.cpp                                   main(int, char *[])
1   #include <QtCore/QCoreApplication>
2
3   int main(int argc, char *argv[])
4   {
5       int i = 0;
6       int z = ++i + ++i + ++i + ++i;
7   }
8
```

Threads:

Level	Locals and Watchers	Breakpoints	Thread
0			

Name	Value	Type
argc	1	int
argv	0x4d2d58	char **
i	4	int
z	11	int

mingw C++/gcc

```
void main()
{
    int i = 0;
    int z = ++i + ++i + ++i + ++i;
}
```

z | 16

VS C++

Точка следования

Состояние программы, в котором все побочные эффекты (++, --, etc.) гарантированно завершены.

- `p *= x++;`
- `q /= ++x;`
- `i = i++;`
- `i = ++i + ++i;`

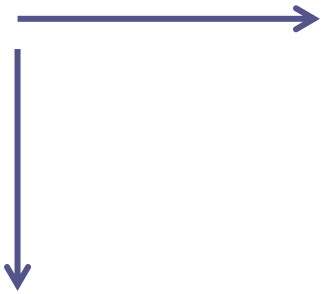
Однако:

- `a = (5>3)? 5 : 3;`
- `if ((a > b) && (c != d)) { … }`

Приоритет операций

1. ++ -- ()
2. - ! ~
3. * / %
4. + и -
5. << >>
6. < <= > >=
7. == !=

Порядок следования:



http://ru.cppreference.com/w/cpp/language/operator_precedence