Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Loops and Assignment Expressions

Quick Recap of Relevant Topics

- Iteration idioms in programming
- Necessity and convenience of iteration
- "while …", "do … while …" and "for …" loops in C++
- Use of "break" statements in loops

Overview of This Lecture

IIT Bombay

- Closer look at “for” loops

  Use of assignment expressions and its variants

  Use of “,” separated expressions

"for …" Statement: Our Simple View

IIT Bombay

Semi-colons not to denote end of executable statements
But to separate three parts inside for (…..)

Part of program before iteration

```
for (  iteration initialization     ;  loop condition ;
          instructions to execute at end of every iteration)
{
        Block of statements  ("for" loop body )
}
```

Note absence of semi-colon

Part of program after iteration

"for …" Statement: C++ Standard View

IIT Bombay

Part of program before iteration

```
for ( initialization expression ;  loop condition  ;
      update expression )
{
      Block of statements  ("for" loop body )
}
```

Part of program after iteration

Appears Nonsensical?

IIT Bombay

- We needed assignment statements to initialize variables before entering loop
- We needed assignment statements to update after each iteration

Is it meaningful to have initialization expression and update expression?

What if I write a + b*c for initialization/update expression?

Which variable is initialized/updated here?

Assignment as An Operator

IIT Bombay

- C++ allows **“=“ (assignment)** to be viewed as an **operator in an expression, with side effects**

  Assignment: **x = (y + z)**

  As a statement: **x = (y + z) ;** **Semi-colon present**

  Assign the value of expression y+z to x

  As an operator: **x = (y + z)** **Semi-colon absent**

  Side effect: Value of expression (y+z) is stored in x

  Type and value: Same as those of (y + z) … RHS of “=“

“for …” Statement: Our Enhanced View

IIT Bombay

Part of program before iteration

```
for ( count = 1.0 ; loop condition ; count = (count + 1))
{
      ock of statements (“for” loop body )
}
                                    on
```

Expressions with side-effects

More on Assignment as An Operator

IIT Bombay

- Need operator precedence

  What is (a = b + c) as an expression ?

- Precedence of = lower than that of arithmetic and logical operators we have seen so far

  (a = b + c) as an expression is (a = (b + c))

  An expression with side-effect: a is assigned the value of b+c

  Type and value of (a = b + c) is same as that of (b + c)

More on Assignment as An Operator

IIT Bombay

- Need associativity

  Right-to-left associative

  (a = b = c = a + 1) is evaluated as (a = (b = (c = (a + 1))))

    Type and value same as that of (a + 1)

Special Assignment Operators

IIT Bombay

- Increment

  Post-increment: x++

  Similar to x = x + 1

  But, value is that of x before incrementing

Value of y: 10 Value of x: 2

y = x++;

x++ as an expression

Value of y: 2 Value of x: 3

Special Assignment Operators

IIT Bombay

- Increment

  Pre-increment: ++x

  Similar to x = x + 1

  Value is that of x after incrementing

Value of y: 10 Value of x: 2

y = ++x;

++x as an expression

Value of y: 3 Value of x: 3

Special Assignment Operators

IIT Bombay

- Decrement

  Post-decrement: x--

  Similar to x = x - 1

  Value is that of x before decrementing

Value of y: 10 Value of x: 2

y = x--;

x-- as an expression

Value of y: 2 Value of x: 1

Special Assignment Operators

- Decrement

  Pre-decrement: --x

  Similar to x = x - 1

  Value is that of x after decrementing

Value of y: 10 Value of x: 2

y = --x;

--x as an expression

Value of y: 1 Value of x: 1

Compound Assignment Operators

IIT Bombay

- Increment/decrement variable by an expression

  x += (y + z) same as x = x + (y + z)

  x -= (2*w) same as x = x – (2* w)

- Can have similar operators from other arithmetic operators

  x *= 2 same as x = x*2

  x /= y   same as x = x/y

  x %= 5 same as x = x/5

Increment and Decrement Operators

IIT Bombay

- Precedence and associativity:
  - Post-increment/post-decrement same precedence, left-to-right associative
  - Pre-increment/pre-decrement same precedence, right-to-left associative
  - Pre-increment/pre-decrement has lower precedence than post-increment/post-decrement
  - All have higher precedence than other arithmetic and logical operators we have seen
    - Exception: pre-increment/pre-decrement same precedence as ! (lo
  - +=, -=, /=, %= have lowest precedence (same as that of =), right-to-left associative

Increment and Decrement Operators

IIT Bombay

• Pr

Moral of the story:

Try not to mix increment/decrement operators with other operators, if possible

Convenient idioms for increment and decrement
Use them mostly for that purpose

Use of "," Operator

IIT Bombay

- Using side effects of multiple expressions when only one is allowed
- (x++ , y = z+2 , z + 5) is one expression
  - Component expressions evaluated in left-to-right order
  - Two side-effects

    x is incremented

    y is assigned the value of z + 2
  - One type and value: Same as rightmost expression, i.e. z + 5
- Often used in initialization and update of "for" loops

"for …" Statement: Our Enhanced View

IIT Bombay

Part of program before iteration

for (count = 1.0, i = 2 ; loop condition ; count += 5, i-- , j++)

{

Block of staten ("for" loop body)

}

Part of program a

Evaluates as a single expression
Two side effects

IIT Bombay

Summary

- Assignment as a statement and as an expression
- Variants of assignment statements
- Use in loops (and other places too) in C++