Computer Programming

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

Lectures 23, 24, 25

Recap: 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 "="

Recap: "for …" Statement with Assigns

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

Recap: 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

Recap: 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

Recap: Compound Assignments

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

Recap: Reasoning About Loops

IIT Bombay

- A loop iteratively transforms relations between variables such that
  - **Pre-condition** holds when we start iterating for first time
  - **Post-condition** holds when we exit loop
  - Pre-condition, post-condition special cases of relation that holds invariantly every time we : **loop-invariant**
  - **Desirable:** Integer valued "metric" (e.g. value of counter) monotonically changes towards fixed value : **loop-variant** (ensures loop termination)

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q1. Consider the following "for" loop in C++:

```
for (X ; Y ; Z) { W ; }
```

Which of the following is CORRECT ?

A. X, Y and Z may be empty, but W cannot be

B. X, Y, Z and W may all be empty

C. If Y is not empty, X and Z cannot be empty

D. None of the above

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q2. Which of the following is a C++ EXPRESSION?

[There may be more than one correct answer.]

A. x = y + z

B. x++

C. while (true) { x++; }

D. break;

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q3. The precedence of "=" as an operator is:

[There may be more than one correct answer.]

A. Same as that of "+" and "-"
B. Less than that of "+" and "-"
C. More than that of "*" and "/"
D. Less than that of "&&" and "||"

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q4. The "continue" statement in a "for" loop:

[There may be more than one correct answer.]

A. Specifies that the loop must execute infinitely

B. Causes the next loop iteration to start

C. Causes the program to terminate immediately

D. Causes the loop to exit immediately

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q5. Consider the following "for" loop in C++:

```
for (i = 0, j = 10;  j >= i ; j--, i++) { x = x + y; }
```

Which of the following are CORRECT?

A. "i = 0, j = 10" will cause a compilation error

B. "j--, i++" will cause a compilation error

C. Both A and B **D. None of A and B**

Recap Quiz on Lectures 23, 24, 25

IIT Bombay

Q6. Specifying loop invariants as comments serves which of the following purposes:

[There may be more than one correct answer.]

A. **Helps the compiler optimize the loop**
B. **Helps us reason about the loop's correctness**
C. **Helps the compiler ignore errors in the loop**
D. **Helps the computer skip executing the loop**

Practice Problem for Lectures 23, 24, 25

IIT Bombay

P1. Consider the following program fragment:

```
int i, j, n;  cout << "Give n: "; cin >> n;
for (i = 0, j = ?????;  ????? ; ?????) {
    cout << j << " ";
}
```

Fill in the missing parts so that the program fragment outputs the first five powers of n, i.e, n^1, n^2, n^3, n^4 and n^5

P2. What will be output on executing the following C++ program fragment?

```
int x = 10, y = 2;
cout  <<  x++  <<  “,“  <<  --y  <<  “,“;
cout  <<  x + (y ++) << “,” << y << endl;
```

Practice Problem for Lectures 23, 24, 25

IIT Bombay

What will be output on executing the following C++ code fragment?

```
int num=0, i;
for(i = 0; i < 10; i++) {
   if (i%2 == 0) {continue;}
   else { cout << num += i << "," ;}
}
```

Practice Problem for Lectures 23, 24, 25

IIT Bombay

We want to extend the problem of computing the n^{th} power of m using repeated squaring from the previous lecture.

Recall if n is a power of 2, we can compute m^n using $\log_2 n$ multiplications.

Practice Problem for Lectures 23, 24, 25

IIT Bombay

If n is not a power of 2, we would like to compute m^n using $2 \times \lceil \log_2 n \rceil$ multiplications

Write a C++ program with loops to do this.

Hint: Suppose, n (as an unsigned integer) in binary is 1101. Then m^n can be computed as $m^8 \times m^4 \times m^1$. You have already seen how to compute m^r where r is a power of 2.