Computer Programming

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

Session: Recap

Quick Recap 1

IIT Bombay

- Assignment statements
  - A = B + C in C++ versus in maths
- Arithmetic expressions
  - Operators: +, -, *, /, %
  - Operator precedence and associativity
  - Use of parentheses

Type of An Arithmetic Expression

IIT Bombay

- Rule of thumb:

  Expression type at least as "expressive" as operand types, but no more

float a and int b

  - float "more expressive" than int
  - a + b, a – b, a * b, a/b are all float, 2 * b is int, 2.0 * b is float

- double a, float b and int c
  - double "more expressive" than float
  - float "more expressive" than int
  - a + (b * c) has type double

Operator Precedence

IIT Bombay

- What is a + b * c + d?
  - **a + (b*c) + d** or **(a + b) * (c + d)** or **((a + b) * c) + d** ?
  - Depends on operator precedence
    In C++, * has higher precedence than +: a + (b * c) + d
- What is a + b – c + d?
  - **(a + b) – (c + d)** or **(a + (b – c)) + d**?
  - In C++, + and – have same precedence: ((a + b) – c) + d
  - For now, left-to-right evaluation for same precedence operators
    Left-associative (exceptions later in course …)
- *, / and % have same precedence, and are left-associative:

  ((a % b) / c)* d

  Different from usual algebra?
- **Best practice: Use (…) to specify unambiguously**

Quick Recap 2

IIT Bombay

- Logical Expressions
  - Relational operators: <, <=, >, >=, ==, !=
  - Operator precedences and associativities
  - Use of parentheses
- Values of logical expressions
  - true, false
  - 0, non-zero
  - Printing values of logical expressions

Logical Expressions in C++

IIT Bombay

- Logical operators
  - && (two ampersands)

    Binary logical “and”, e.g. (A >= B) && (A >= C)
  - || (two verical bars)

    Binary logical “or”, e.g. (A > B) || (A > C)
  - ! (exclamation symbol)

    Unary logical “not” (negation), e.g. !((A >= B) && (A >= C))
- Complex logical expressions can be built using (…)

  ((A >= B) && (A >= C)) || ((A == 1) && (B != (A + C))
- Always evaluates to a value of type bool

Logical Operator Precedences

IIT Bombay

- Comparison operators
  - <, <=, >, >= : same precedence, lower than !, left-associative
  - ==, != : same precedence, lower than <, left-associative
  - A >= B != B < C interpretted as (A >= B) != (B < C)
- Logical operators
  - ! : highest precedence
  - && : precedence lower than ==, left-associative
  - || : precedence lower than &&, left-associative

  ! flag1 || flag2 && ! flag3 interpretted as

  (! flag1) || (flag2 && (! flag3)

Best practice: Use (…) to specify meaning unambiguously

Quick Recap 3

- Sequential execution of statements
- Declarations: not executed at run-time
- Use of ; to separate one executable statement from next
- Left-to-right and top-to-bottom execution

Simplest Programs: Another Example

- Simplest programs: linear sequence of instructions
  - Computer executes instructions in linear order

A Simple C++ Program

IIT Bombay

```
#include <iostream>
using namespace std;

// Program to find if B divides A
int main() {
  int A, B, R;   // Variable declarations
  cout << "Give A and B: " << endl;
  cin >> A >>  B;
  R = A % B;   // Remainder of A divided by B
  dividesFlag = (R == 0);  // Is the remainder 0?
  cout << "Does B divide A? " << dividesFlag << endl;
  return 0;
}
```

Note the importance of the sequence:

What happens if we swap the order of R = A%B and dividesFlag = (R == 0) ?