IIT Bombay

Computer Programming

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

Session: Assignment Statement and Arithmetic Expressions

Quick Recap of Some Relevant Topics

IIT Bombay

- Structure of a simple C++ program
- Variables and type declarations
- Naming conventions

Overview of This Lecture

- Assignment statement
- Arithmetic expressions

Assignment Statement

- General form

  destination = expression;

- Compute the value of expression and store in destination
- Destination
  - Variable, for now
  - Has a declared type
  - More advanced things later…
- = in C++ assignment statement

  NOT SAME AS equality in maths
  - C = C + 1 meaningful in C++, not in maths
  - A + B = C meaningful in maths, not in C++

Our friendly program:

```
int main() {
  int A, B, C;
  cout << “Give two numbers”;
  cin >> A >> B;
  C = A + B;
  cout << “Sum is” <<  C;
  return 0;
}
```

C is assigned the value of A + B

Assignment Statement

- Expression
  - Refers to values of variables
  - Refers to operators
  - Evaluates to a value
  - A value must have a type

    How much memory to store?

    How to interpret stored bits?
  - So an expression has a type
- Normally, destination and expression types match
  - C is int, A + B is int

Our friendly program:

```
int main() {
  int A, B, C;
  cout << "Give two numbers";
  cin >> A >> B;
  C = A + B;
  cout << "Sum is" << C;
  return 0;
}
```

Arithmetic Expression

Arithmetic Expressions in C++

IIT Bombay

- Usual way we write expressions in algebra
  - a, b, c : variables
  - + , - , * , / , % : operators
  - a + b, a – b, b * c, a/c , a%b : Arithmetic expressions

Integer remainder: 5 % 3 = 2

- What is the data type of a + b?
  - How many bytes to store in memory?
  - How are the stored bits interpretted?
  - Depends on data types of a and b

    a and b both int implies a + b, a – b, a * b, a/b, a%b are all int

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

Type and Value of Arithmetic Expression

```
int a, b, c;
a = 1; b = 2;
c = a/b;
```

Type of a/b: int

Value of a/b: 0

(integer part of 1/2)

```
int a;
float b, c;
a = 1; b = 2.0;
c = a/b;
```

Type of a/b: float

Value of a/b: 0.5

(float can represent fractions)

Operator Precedence

- 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**

Use of Parentheses (…)

IIT Bombay

- Can be used to override default operator precedences
  - Compare ((a + b) * c + d) with a + (b * c) + d
- Can be used to form complex expressions
  - 1 + (1 / (2 + (3 / (4 + x)))) represents $1 + \cfrac{1}{2 + \cfrac{3}{4 + x}}$
  - Evaluate from innermost parenthesized expression outwards
- Not to be confused with { … } or […]
  - a + {b * c} will give a compilation error !!!

Summary

IIT Bombay

- Assignment statement in C++
- Arithmetic expressions
  - Types
  - Values
  - Use of parentheses