

# Computer Programming

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

Session: Assignment Statement and Logical Expressions

# Quick Recap of Some Relevant Topics

---



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

# Overview of This Lecture

---



- Logical expressions in assignment statements

## Recap of Assignment Statement

---

- General form  
    `destination = expression;`
- Compute the **value** of **expression** and store in **destination**
- **Expression** has **type**

# Dealing with Logical Expressions

## Program with arithmetic expression

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

**Arithmetic  
Expression**

**Printing boolean  
value**

## Program with logical expression

```
int main() {  
    int A, B, C;  
    bool isALargest;  
    cout << "Give three numbers";  
    cin >> A >> B >> C;  
    isALargest = ((A >= B) && (A >= C));  
    cout << "Is A the largest:";  
    cout << isALargest;  
    return 0;  
}
```

**Boolean variable  
declaration**

**Logical expression**

# Logical Expressions in C++

- Simplest: Comparing **int** or **float**

- A, B, C : **int** variables

Is B greater than or equal to C ?

- $(A > B)$ ,  $(A < C)$ ,  $(B \geq C)$ ,  $(A \leq C)$

Is A less than or equal to C ?

Logical expressions, data type **bool**, evaluate to **true/false**

- $(A == B)$ ,  $(B != C)$ : equality based logical expressions

Is A equal to B ?

Is B not equal to C ?

- Note use of two = symbols in  $(A == B)$

- $A = B$  denotes an assignment of the value of B to A
- $A == B$  denotes an equality comparison of values of A and B

- Can use expressions instead of variables when comparing  
 $((A + B) * C) < ((C - A) \% B)$  makes perfect sense

# Logical Expressions in C++

- Logical operators
  - **&&** (two ampersands)  
Binary logical “and”, e.g.  $(A \geq B) \ \&\& \ (A \geq C)$
  - **||** (two vertical bars)  
Binary logical “or”, e.g.  $(A > B) \ || \ (A > C)$
  - **!** (exclamation symbol)  
Unary logical “not” (negation), e.g.  $!((A \geq B) \ \&\& \ (A \geq C))$
- Complex logical expressions can be built using ( ... )  
 $((A \geq B) \ \&\& \ (A \geq C)) \ || \ ((A == 1) \ \&\& \ (B != (A + C)))$
- Always evaluates to a value of type **bool**

# Logical Operator Precedences

- Comparison operators
  - $<$ ,  $<=$ ,  $>$ ,  $>=$  : same precedence, lower than  $!$ , left-associative
  - $==$ ,  $!=$  : same precedence, lower than  $<$ , left-associative
  - $A >= B \quad != \quad B < C$  interpreted as  $(A >= B) \quad != \quad (B < C)$
- Logical operators
  - $!$  : highest precedence
  - $\&\&$  : precedence lower than  $==$ , left-associative
  - $||$  : precedence lower than  $\&\&$ , left-associative
  - $! \text{ flag1} \quad || \quad \text{flag2} \quad \&\& \quad ! \text{ flag3}$  interpreted as  
 $(! \text{ flag1}) \quad || \quad (\text{flag2} \quad \&\& \quad (! \text{ flag3}))$

**Best practice: Use ( ... ) to specify meaning unambiguously**



# Printing Logical Expressions

---

- Logical expression has value **true** or **false**
- Internal representation
  - At least 1 byte, could be 4 bytes as well
  - **false** represented as 0
  - **true** represented as non-zero (not necessarily 1)
- Default printing
  - **true** printed as 1
  - **false** printed as 0
  - Using manipulator boolalpha, can print **true** or **false**  
`cout << boolalpha << (A > B);`

# Summary

---



- Logical expressions in C++
  - Simple expressions with comparison operators
  - Logical operators
  - Use of parentheses
  - Printing logical values