

# Computer Programming

Dr. Deepak B Phatak  
Dr. Supratik Chakraborty  
IIT Bombay

Session: Conditional Execution in C++ Programs - Part A

# Quick Recap of Some Relevant Topics

---



- Structure of a simple C++ program
- Variables and type declarations
- Assignment statements
- Arithmetic and logical expressions
- Sequential execution of statements

# Overview of This Lecture

---



- Conditional execution of statements in C++  
“if ... else ...” statement and its variants

# Recalling Some Useful Facts

---



- Program: sequence of **compiler directives**, **declarations**, **instructions**
- Normally, computer (Mr. Dumbo) executes instructions
  - In same order in which they appear in program
  - top to bottom, left to right, separated by “;”

# A Simple Problem



- Consider the problem:

Divide integer A by integer B and output the quotient Q.

- Sounds simple!

Do we really want to  
divide if B is 0?

```
int main() {  
    int A, B, Q;  
    cout << "Give A and B" << endl;  
    cin >> A >> B;  
    Q = A/B;  
    cout << "Quotient is: " << Q << endl;  
    return 0;  
}
```

# A Simple Problem

- Consider the problem:

Divide integer A by integer B and output the quotient Q if B is non-zero. Otherwise, output the string "Bad inputs!"

- We need conditional execution of (blocks of) instructions

• Read inputs A and B	Unconditionally execute first
• Divide A by B and output quotient Q	Execute next only if B is non-0
• Output the string "Bad inputs!"	Execute next only if B is 0
• Return control to caller/OS	Unconditionally execute last

# A Simple Problem

- Consider the problem:

Divide integer A by integer B and output the quotient Q if B is non-zero. Otherwise, output the string "Bad inputs!"

- If B is non-zero, sequence of execution:

- |                                       |                               |
|---------------------------------------|-------------------------------|
| • Read inputs A and B                 | Unconditionally execute first |
| • Divide A by B and output quotient Q | Execute next since B is non-0 |
| • Return control to caller/OS         | Unconditionally execute last  |

# A Simple Problem

- Consider the problem:

Divide integer A by integer B and output the quotient Q if B is non-zero. Otherwise, output the string "Bad inputs!"

- If B is zero, sequence of execution:

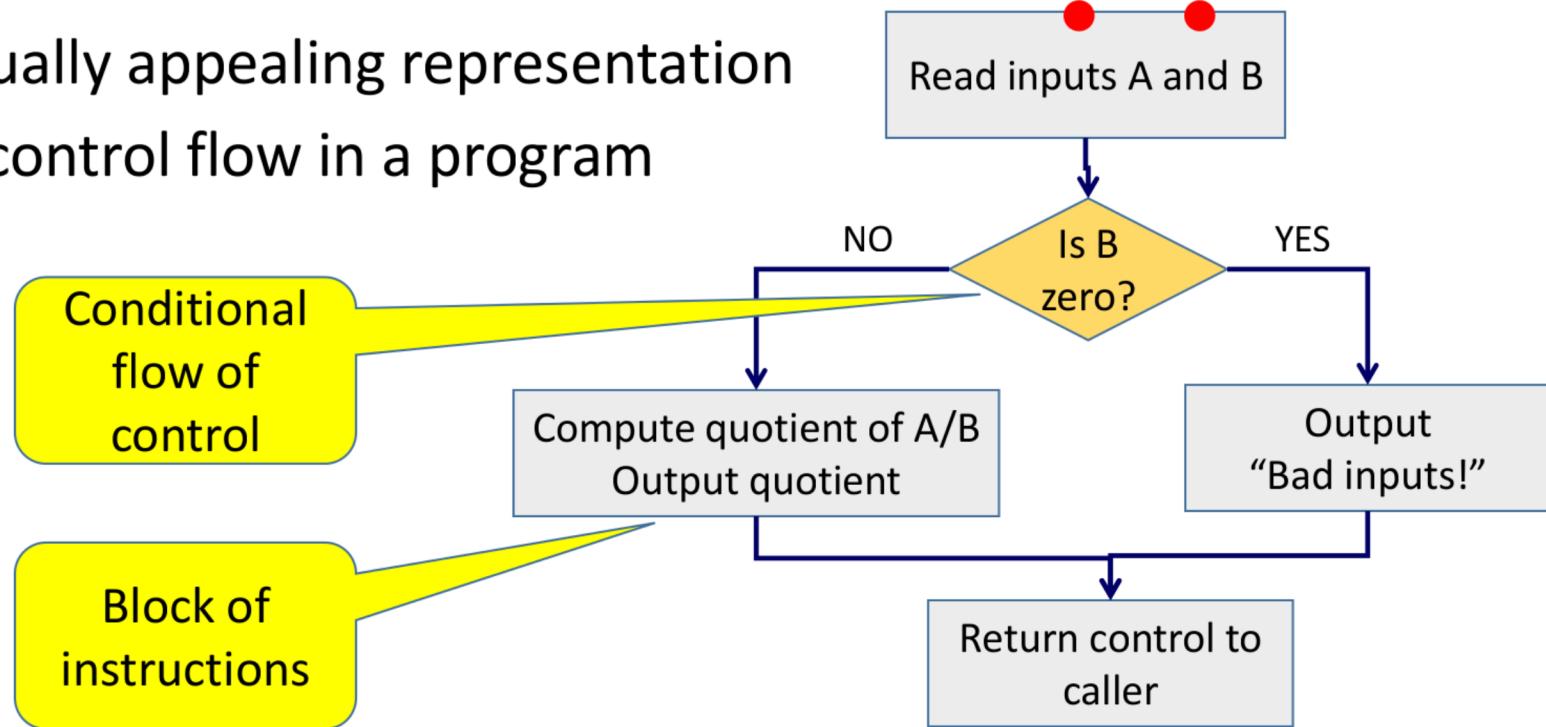
• Read inputs A and B	Unconditionally execute first
-----------------------	-------------------------------

• Output the string "Bad inputs!"	Execute next since B is 0
-----------------------------------	---------------------------

• Return control to caller/OS	Unconditionally execute last
-------------------------------	------------------------------

# Flowchart Representation

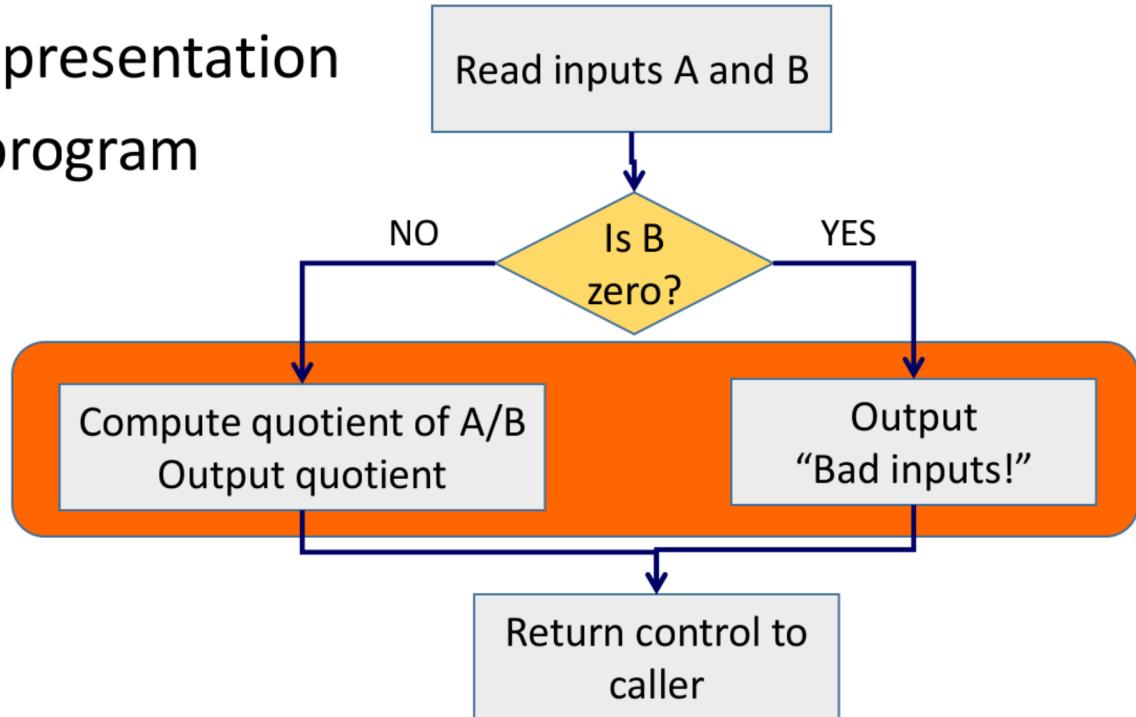
- Visually appealing representation of control flow in a program



# Flowchart Representation

- Visually appealing representation of control flow in a program

Control can only flow through one block, not both



# Conditional Execution in C++

- “**if ...else ...**” statement

```
if(B == 0) {  
    cout << "Bad inputs!" << endl;  
}  
  
else {  
    quotient = A/B;  
    cout << "Quotient is: " << quotient << endl;  
}
```

Keywords of C++

# Conditional Execution in C++

- “**if ...else ...**” statement

```
if (B == 0) {
```

Logical expression:  
Evaluates to **true/false**

```
    cout << "Bad inputs!" << endl;
```

```
}
```

```
else {
```

```
    quotient = A/B;
```

```
    cout << "Quotient is: " << quotient << endl;
```

```
}
```

# Conditional Execution in C++



- “**if ...else ...**” statement

```
if (B == 0) {
```

```
    cout << "Bad inputs!" << endl;
```

```
}
```

Block of statements:  
Grouped by { ... }

```
else {
```

```
    quotient = A/B;
```

```
    cout << "Quotient is: " << quotient << endl;
```

```
}
```

Another  
block

# Blocks in an “if ...else...” Statement



- Blocks can be any sequence of C++ statements
- Specifically, can be another “if ... else ...” statement

```
if (B == 0) {  
    cout << "Bad input!" << endl;  
}  
else {  
    if (B == 1) { cout << "Are you joking?" << endl; }  
    else { quotient = A/B; cout << "Quotient is: " << quotient << endl; }  
}
```

- Arbitrary nesting of “if ... else ...” statements allowed in C++

# “if ...” Without “else ...”



- “else ...” is optional in C++  
`if (B == 0) { ... }` equivalent to  
`if (B == 0) { ... } else { // Do Nothing }`
- Succinct way to write programs when nothing needs to be done in “else” branch

# “if ...else ...” Statements and {...}

- Calls for caution

```
if (B == 0)  
    cout << "Hello ";
```

```
cout << "world!!!" << endl;
```

Unconditionally executed

Output if B is 0:        “Hello world!!!”

Output if B is not 0:    “world!!!”

# “if ...else ...” Statements With Other Statements



- “if ... else ...” statement (or “if ... “ statement) can be sequenced with other statements
  - Similar to assignment and input/output statements for sequencing purposes
  - Can have “return” statements in “if ...” or “else ...” blocks

# A Program With Conditional Execution

---



```
#include <iostream>
using namespace std;
// Program to compute quotient
int main() {
    int A, B, Q; // Variable declarations
    cout << "Give A and B" << endl;
    cin >> A >> B;
    if (B == 0) { cout << "Bad inputs!!!" << endl; return -1;}
    else { Q = A/B; cout << "Quotient is: " << Q << endl; }
    cout << "Be happy!" << endl;
    return 0;
}
```

# Summary

---



- Conditional execution of statements in C++ programs
  - “**if ... else ...**” statement and its usage
- Nesting of “**if ... else ...**” statements
- An example program with conditional execution