

Computer Programming

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

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

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
- Conditional execution using “if ... else ...” statement

Overview of This Lecture



- Conditional execution of statements in C++
 “switch ... case ...” statement
- Conditional expressions in C++

Recalling Some Useful Facts



- Program: sequence of **compiler directives**, **declarations**, **instructions**
- Normally, computer executes instructions
 - In same order in which they appear in program
- **“if ... else ...”** statement allows computer to execute instructions in non-linear order

A Programming Problem

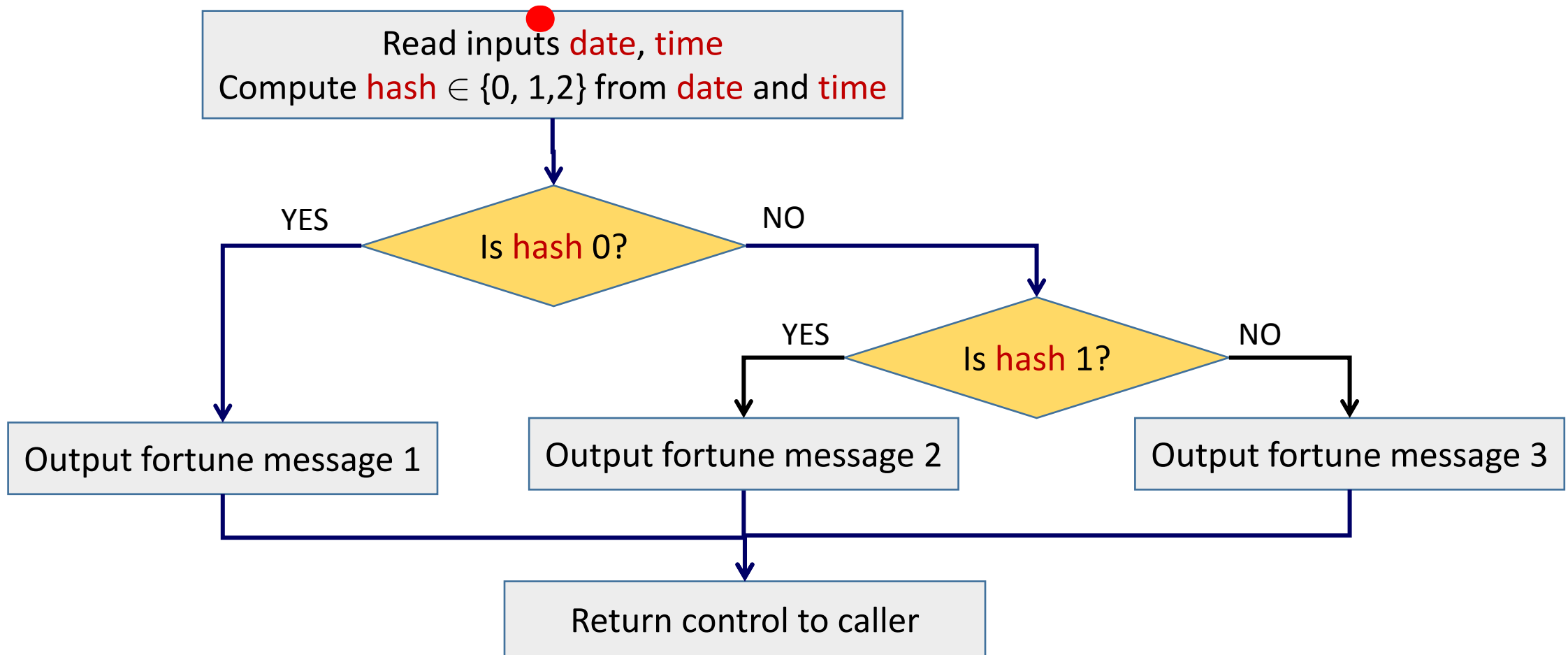


We want to implement a simple “fortune” program

Read date and time as integers

Output one of three pre-determined “fortune” messages depending on date and time

Flowchart for Simple “fortune” Program



An Example “fortune” Program

// A simple “fortune” program

```
int main() {  
    int date, time, hash; // Variable declarations  
    cout << “Give date (DDMMYYYY) and time (HHMM)” << endl;  
    cin >> date >> time;  
    hash = (date + time) % 3; // Compute a hash value in {0, 1, 2}  
    if (hash == 0) { cout << “Time and tide wait for none.” << endl; }  
    else {  
        if (hash == 1) { cout << “The pen is mightier than the sword.” << endl; }  
        else { cout << “Where there is a will, there is a way.” << endl; }  
    }  
    return 0;  
}
```

Nested
if ... else ...
statements

An Example “fortune” Program

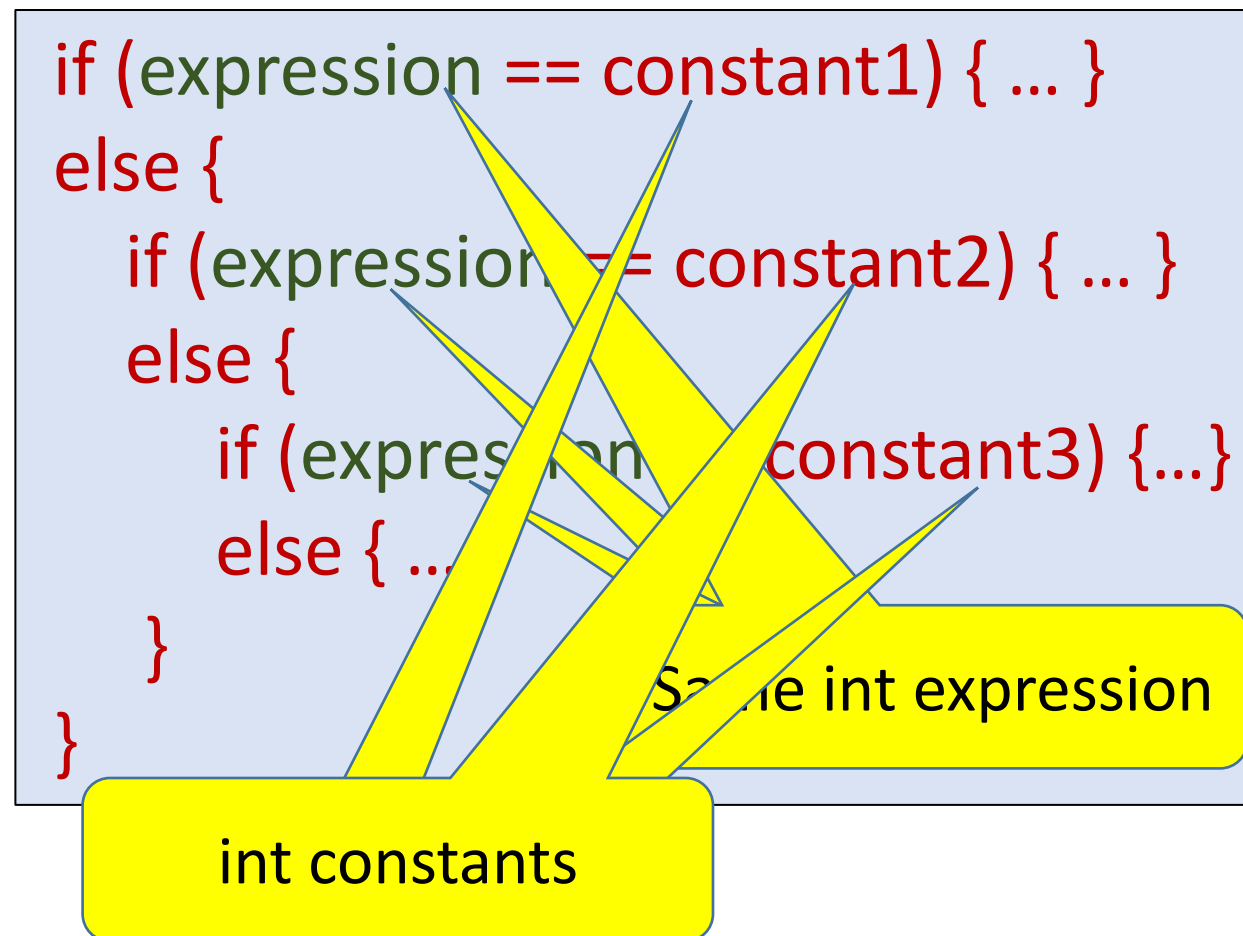
// A simple “fortune” program

```
int main() {  
    int date, time, hash; // Variable declarations  
    cout << “Give date (DDMMYYYY) and time (HHMM)” << endl;  
    cin >> date >> time;  
    hash = (date + time) % 3; // Compute a hash value in {0, 1, 2}  
    if (hash == 0) { cout << “Time and tide wait for none.” << endl; }  
    else {  
        if (hash == 1) { cout << “The pen is mightier than the sword.” << endl; }  
        else { cout << “Where there is a will, there is a way.” << endl; }  
    }  
    return 0;  
}
```

What if we
had 10
“fortune”
messages?

“switch ... case ...” Statement in C++

- Alternative to (deeply) nested “if ... else ...”
- Applicable only when all conditions check equality of same integer expression with different constants



“switch ... case ...” Statement in C++

```
if (hash == 0) { cout << "Time and tide wait for none." << endl; }  
else {  
    if (hash == 1) { cout << "The pen is mightier than the sword." << endl; }  
    else { cout << "Where there is a will, there is a way." << endl; }  
}
```

```
switch (hash) {  
    case 0: cout << "Time and tide wa  
            break;  
    case 1: cout << "The pen is mighti  
            break;  
    default: cout << "Where there is a will, there is a way." << endl;  
}
```

switch, case, break, default
C++ keywords

“break” and Fall-Through

Suppose we did not put **any** “break” statements

Value of hash: 0

```
switch (hash) {  
    case 0: cout << "Time and tide wait for none." << endl;  
    case 1: cout << "The pen is mightier than the sword." << endl;  
    default: cout << "Where there is a will, there is a way." << endl;  
}  
return 0;
```

Fall-through of control

“break” and Fall-Through

Suppose we did not put **some** “break” statements





Value of hash: 0

```
switch (hash) {  
    case 0: cout << "Time and tide wait for none." << endl;  
  
    case 1: cout << "The pen is mightier than the sword." << endl;  
            break;  
    default: cout << "Where there is a will, there is a way." << endl;  
}  
return 0;
```

“break” and Fall-Through

Suppose we did not put some “break” statements

Value of hash: 1

```
switch (hash) {   
    case 0: cout << "Time and tide wait for none." << endl;  
  
    case 1: cout << "The pen is mightier than the sword." << endl;   
        break;   
    default: cout << "Where there is a will, there is a way." << endl;  
}  
return 0; 
```

Intentional Fall-Through

- Want all “fortune” messages to be printed if hash is 0, two of them to be printed if hash is 1, and only one otherwise.

```
switch (hash) {  
    case 0: cout << “Time and tide wait for none.” << endl;  
  
    case 1: cout << “The pen is mightier than the sword.” << endl;  
  
    default: cout << “Where there is a will, there is a way.” << endl;  
}  
return 0;
```

“default” in “switch ... case ...” Statement

Similar to final “else” branch in nested “if ... else ...” statements

If hash doesn't match any “case” values, “default” statements executed

What if hash is 2?

```
switch (hash) {  
    case 0: cout << "Time and tide wait for none." << endl;  
            break;  
    case 1: cout << "The pen is mightier than the sword." << endl;  
            break;  
    default: cout << "Where there is a will, there is a way." << endl;  
}  
return 0;
```

Conditional Expressions

- Standard arithmetic expressions
 - $(a + b) * c$, $(a + b) / c$, ...
- What if I want the same expression to take the value of $(a+b)*c$ if $(c \leq 0)$ and $(a+b)/c$ otherwise ?
- Easy solution:

```
if (c <= 0) { expr = (a + b) * c; }  
else {expr = (a + b) / c;}
```
- Alternative solution: $\text{expr} = (c \leq 0) ? (a + b) * c : (a + b) / c;$

Conditional Expressions



- General form
(logical expression) ? (if-expression) : (else-expression)
- if-expression and else-expression must be of same type
- Type of conditional expression is type of if-expression (or else-expression)
- Can be used for both arithmetic and logical expressions
 - if-expression and else-expression can be both arithmetic expressions
 - if-expression and else-expression can be both logical expressions

Summary



- Conditional execution of statements in C++ programs
 - “switch ... case ...” statement and its usage
 - “break” and fall-through
 - “default”
- Conditional expressions