Slides #6

Conditionals

Chapter 4

CMPT 125/128

Topics

1) How can we work with true and false?
2) How can write if statements?
3) Is there an "easier" way to work with a single variable with multiple possible values?

Boolean Expressions

Boolean Expressions

- Boolean Expressions evaluate to...
  - Called a “condition.”
- Most often used with if or loop (while, do, for..)

```
long loops = 45;
// Check loop length
if (loops > 30) {
    cout << “This may take a while!”;
}
// Count down
while (loops != 0) {
    cout << “Counting down...” << loops << endl;
    loops --;
}
```

Truth?

- How C++ interprets values:
  - Things which are false:
    - the keyword false (which evaluates to 0)
    -
  - Things which are true:
    - the keyword true (which evaluates to 1)
    -

Equality

- Equality Operators:
  - Equal:
  - Not Equal:
- Examples:

```
cout << (0 == 0);              // 1
cout << ('b' != 'a');          // 1
cout << ("hi"=="bye");         // 0
cout << (true != false);       // 1
```

- Equality vs Assignment:
  - if the values are the same (==).
  - the values the same (=).

Relational Operators

- Compare two values to see which is less/greater:
  - < Less than
  - <= Less than or equal to
  - > Greater than
  - >= Greater than or equal to
- Examples:

```
cout << (age >= 65);  // 1 if senior
if (age >= 65) {...}      // Check for senior
if (value < 0) {...}      //
if (0 > value) {...}      // Also check for neg. value
if (value <= 0) {...}     // Check for non-positive value
```


Logical Operators

- Logical Operators accept Boolean arguments:
  - And (true && true) == true
  - Or (true || false) == true
  - Not !true == false, !false == true
- Examples:

```
int     a, b, timeLeft;
bool    havRetired, unemployed, doorIsOpen;

// if a equals 1 and b equals 2 then...
if ((a==1) && (b==2)) {...}

// If either (or both) haveRetired or unemployed then...
if (haveRetired || unemployed) {...}

// If (not doorIsOpen) and (timeLeft > 0) then...
if (!doorIsOpen && (timeLeft > 0)) {...}
```

Precedence

- Examples:

```
int      x = 5, y = 4;
bool   r;

r = x < y + 1;

r = x < y == 4

r = x + 1 || y;

r = x < y || ! y > x;
```

Prec. Level	Op.	Operation	Associates
1	+ - !	unary plus/minus not	R to L
2	* / %	mult, div, remainder	L to R
3	+ -	add subtract	L to R
4	< > <= >=	comparisons	L to R
5	== !=	equal, not equal	L to R
6	&&	AND	L to R
7	||	OR	L to R
8	= += -= *= ...	assignments	R to L

Order can be forced by parentheses.
See text appendix B for full table.

If

If Statement

- The if statement performs conditional execution:

```
cout << “Value is negative.” << endl;
```

- Style
  - Indent the “then” statements to make it easy to read. (Compiler does not care!)
  - Good style...
- Also valid (but bad style):

```
if (value < 0)
    cout << “Value is negative.” << endl;

if (value < 0) cout << “Value is neg.” << endl;
```


if, if-else

- if

```
if (age < 18) {
    cout << "Minor";
}
```

- if-else

```
if (age < 18) {
    cout << "Minor";
} else {
    cout << "Adult";
}
```

- if, else if, else

```
if (age < 18) {
    cout << "Minor";
} else if (age < 65) {
    cout << "Adult";
} else {
    cout << "Senior";
}
```

Nested if

if statements can be

other if statements

```
if (value == 'a') {
    if (value == 'b') {
        doB();
    } else {
        doElse();
    }
}
```

Without {...} the code above could be either options on the right.

```
if (value == 'a')
    if (value == 'b')
        doB();
    else
        doElse();
```

```
if (value == 'a')
    if (value == 'b')
        doB();
else
    doElse();
```


Multiple statements

- Use {...} to place multiple statements in the if or else:

```
int iqValue;
cin >> iqValue;
if (iqValue > 150)
    genius = true;
else
    genius = false;
    cout << “Not so smart” << endl;
```

- Above code will...
  - Must use block statement {...} with multiple statements.

Common Errors

-
```
if (a < b);
    cout << "less";
```
Extra ';' ends the if statement (creates a). cout << "less"; will always execute.

-
```
if (a=b)
    cout << "Equal!";
```
a=b is the Should be a == b. VERY common bug!

-
```
if (a < b)
    a=b;
    c=d;
```
To selectively execute more than one statement, you need to create a

(And, it's just better style!)

Quick test for non-zero

- Use if to check for non-zero values:

```
cout << "Enter a number to invert: ";
double invMe = 1;
cin >> invMe;
if (invMe) {
    cout << "Inverted: " << (1/invMe);
}
```

Comparing floating point

- Floating point values are often not exact.

```
double a;
a   = 0.1;        // a = 0.1 * 1
a += 0.1;         // a = 0.1 * 2
a += 0.1;         // a = 0.1 * 3
a += 0.1;         // a = 0.1 * 4
a += 0.1;         // a = 0.1 * 5
a += 0.1;         // a = 0.1 * 6
a += 0.1;         // a = 0.1 * 7
a += 0.1;         // a = 0.1 * 8
a += 0.1;         // a = 0.1 * 9
a += 0.1;         // a = 0.1 * 10

cout << "a: " << (a) << endl;
cout << "a==1: " << (a == 1) << endl;
```

Output:

Formatted output

a: 0.99999999999999989000
a==1: 0

Scope

- Variables exist in a scope:
  - Variables defined in a block...

- What are the errors?

```
if (a > b) {
    int fromThen = 1;
} else {
    cout << fromThen;
    int fromElse = 2;
}
cout << fromThen;
cout << fromElse;
```


Comparing characters & strings

- Characters are numbers (see ASCII table):

```
cout << ( 'A' < 'B');          // 1
cout << ( 'Z' < 'a');          // 1
cout << ( '0' < '1');          // 1

if (val < '0' || val > '9') {
    cout << "Not a digit (0-9).";
}
```

- Strings:

```
cout << ("Mark" < "Mary");          // 1
cout << ("MARK" < "mark");          // 1
cout << ("Hello " != "world!");     // 1
cout << ("C++" == "awesome");       // ?!?!?
```


Switch

Switch

- switch can replace a large nested if's which

```
if (myChar == 'a') {
    countA ++;
} else if (myChar == 'b') {
    countB++;
} else if (myChar == 'c') {
    countC++;
} else {
    countNone++;
}
```

```
switch(myChar) {
    case 'a':
        countA++;
        break;
    case 'b':
        countB++;
        break;
    case 'c':
        countC++;
        break;
    default:
        countNone++;
}
```


switch basics

- The expression is evaluated and control...
- Each case must be for a single constant value:

```
case 'a':       // OK
case 50:        // OK
default:        // OK - when nothing else matches
case <10:       // BAD!  Compile error.
case someVar:   // BAD!  Compile error.
```

- Allowable switch Types:
  - Can only switch on: char, short, int, long, bool.
  - Cannot switch on: float, double, string.
  - Why would switch on a floating point be a bad idea?

break

-
- Without a break, execution will “fall through” to the next case.

```
switch(myChar) {
    case 'a':
        countA++;
        break;
    case 'b':
        countB++; // Fall through
    case 'c':
        countC++; // Fall through
    default:
        countNone++;
}
```

if myChar == 'b', then this executes: countB++, countC++, and countNone++!

Intentional Fall-through

- Compare these two structures:

```
if (myChar == 'a') {
    isVowel = true;
} else if (myChar == 'e') {
    isVowel = true;
} else if (myChar == 'i') {
    isVowel = true;
} else if (myChar == 'o') {
    isVowel = true;
} else if (myChar == 'u') {
    isVowel = true;
} else {
    isVowel = false;
}
```

```
switch(myChar) {
    case 'a':   // Fall through
    case 'e':   // Fall through
    case 'i':   // Fall through
    case 'o':   // Fall through
    case 'u':
        isVowel = true;
        break;
    default:
        isVowel = false;
}
```

Fall through intentionally used to implement feature. Note it's documented!

Unintentional Fall through

- A very common bug is an unintentional fall through.

```
switch(myChar) {
    case 'a':
        countA++;
        break;
    case 'b':
        countB++;
    case 'c':
        countC++;
        break;
    default:
        countNone++;
}
```


Summary

- Boolean expressions evaluate to true or false.
  - Comparison: ==, !=, >, <, >=, <=
  - Logical expressions: &&, ||, !
- if-else statement control program branching.
  - Use switch to check different values of a single variable or expression.