CS150 Intro to CS I

Fall 2012

Chapter 4
Making Decisions

- Reading: Chapter 4 (4.4 pp. 166-168; 4.5 pp. 169-175; 4.6 pp.176-181; 4.8 pp. 182-189; 4.14 pp. 202-210
- Good Problems to Work: pp. 166-168 [4.14]; pp. 175 [4.15]; p. 180 [4.16]; p 190 [4.19, 4.20]; pp.209-210 [4.27, 4.29, 4.30]

Explicit Type Conversion

- A type cast expression lets you manually change the data type of a value
- The syntax for type casting is

```
static_cast<DataType>(Value)
```

  - Value is a variable or literal value
  - DataType is the data type that you are converting Value into

Example

```
double number = 3.7;

int val;

val = static_cast<int>(number);
```

What is saved into val?

if Statement

- We may want to execute some code if an expression is true, and execute some other code when the expression is false.
- This can be done with two if statements…

```
if (value >= LIMIT)

{
        // do something
}

if (value < LIMIT)
{
        // do something else
}
```

Double-Alternative if

- C++ provides a shortcut to combine 2 if statements

```
if (expression)
{
  // stmts if expression is true
}
else
{
  // stmts if expression is false
}
```

Problem

```
int number;
cout << "Enter a number, I'll tell you";
cout << " if it is odd or even: ";
cin >> number;
// write a double-alternative if here
```

Multiple-Alternative if

```
cout << "Enter two numbers: ";
cin >> num1 >> num2;

if(num1 > num2)
{
  cout << num1 << "is greater" << endl;
}
else if(num2 > num1)
{
  cout << num2 << "is greater" << endl;
}
else
{
  cout << "Numbers are equal" << endl;
}
```

Logical Operators

`&&`	**And**
`||`	**Or**
`!`	**Not**

Evaluating AND

`expr1 && expr2`

- For the complete expression to be true, both expr1 and expr2 must be true
- Example:

`(temp > HOT) && (humidity > STICKY)`

  - ➢ These are unbearable heat and humidity conditions
  - ➢ Both must be true for the entire expression to be true

Evaluating OR

```
expr1 || expr2
```

- The complete expression is true, if either expr1 or expr2 is true
- Example:

```
(salary < MIN_SALARY) || (MARRIED == status)
```

  - To qualify for financial aid, salary has to be less than some minimum salary OR you must be married
  - Only one condition has to be true

Evaluating NOT

`!expr`

- If expr is true, !expr is false
- If expr is false, !expr is true
- Example:

`!(salary < MIN_SALARY)`

  - What makes this true? False?

Operator Precedence (highest to lowest)

Unary plus & minus	+ - !	Left associative
Multiplication, division, and modulus	* / %	Left associative
Addition & subtraction	+ -	Left associative
Relational operators	< <= > >=	Left associative
Relational operators	== !=	Left associative
Logical AND	&&	Left associative
Logical OR	||	Left associative
Assignment	=	Right associative

Problem

- According to the operator precedence and associativity rules given on the previous slide, how will the following expressions be evaluated?

```
x < min + max

min <= x && x <= max

!x == y + 2

x = a + b % 7 * 2
```

Problem

- Are these two code snippets equivalent?

```
int x, y;
cin >> x >> y;
if(x > y)
{
  cout << x;
}
if(x < y)
{
  cout << y;
}
```

```
int x, y;
cin >> x >> y;
if(x > y)
{
  cout << x;
}
else
{
  cout << y;
}
```

Problem

- Write a C++ program segment that allows the user the ability to input an integer from the keyboard.
- If the integer is positive, increment a variable posCount by 1. If the integer is negative, increment a variable negCount by 1. If neither, increment zeroCount by 1

```
int posCount = 0,
    negCount = 0,
    zeroCount = 0;
```

switch statement

- Let’s look at the following program segment:

```
char choice;

cout << "E)dit  S)ave  Q)uit";
cin >> choice;

switch (choice)
{
  case 'E':  cout << "Time to edit " << endl;
             break;
  case 'S':  cout << "Time to save" << endl;
             break;
  default:   cout << "Illegal command" << endl;
}
```

switch format

```
switch(ordinaldatatype)
{
  case constantexpression:   // one or more stmts
                              break;
  case constantexpression:   // one or more stmts
                              break;
  …
  default :                   // one or more stmts
}
```

- What is an ordinal data type?
- `(ordinaldatatype)` can be a variable or expression
- `constantexpression` must be unique in each case
- `default` is optional
- `break;` resumes execution after the switch

Problem

1. Modify slide 17 to allow ‘E’, ‘e’, ‘S’, or ‘s’

2. Rewrite the logic for 1. as an if statement