CISC 1110 (CIS 1.5)
Introduction to Programming Using C++

Spring 2012
Instructor : K. Auyeung

Email Address: kenny@sci.brooklyn.cuny.edu
Course Page: http://www.sci.brooklyn.cuny.edu/~kenny/cisc1110

Class Hours: MW 2:40 – 4:45PM 4428N

Agenda

- Relational Operators Revisited
- More control structures
- Break and Continue

Comparing Strings

the inequality operators (<, <=, >, >=) perform a lexical comparison between two strings
a “lexical comparison” is like checking if two strings are in alphabetical order: one is less than the other if it comes before the other alphabetically

• EXCEPT, the lexical comparison is case sensitive and uses the ASCII table, which means that all the upper case letters (A..Z) come before (are less than) all the lower case letters (a..z), e.g.:

```
string s1, s2, s3;
bool a1, a2; s1 = "ABC";
s2 = "DEF";
s3 = "abc ";
a1 = ( s1 < s2 );
a2 = ( s3 < s2 );
```

After the above code fragment:
the value of a1 will be true because "ABC" < "DEF" and
the value of a2 will be false because "abc" > "DEF"

Agenda

- Control structures continued…
  - `while` loop
  - `switch` statement
- Escape characters
- Random Numbers

Control Structures

- Allows programs to bifurcate, repeat code or make decisions
- Compound-statement, sometimes called a block, is a group of statements separated by '`;`' and bounded by braces '`{}`'
- The `while loop` - repeats a block of code **while** a condition is true
- The `switch` statement - functions like an if-else statement

While Loop - Syntax

```
while (condition)   {

    // body of loop
}


do  {

    // body of loop

}  while (condition);
```

Example #1

```
int balance = 500;
int expence = 0;
int count = 0;

while (balance > 0)  {

    cout << "Enter expense ";
    cin >> expence;

    balance -= expense;
    count++;
}

cout << "You've purchased " << count << "items\n";
```

Example #2

```
do {
   cout << "age ";
   cin >> age;
   if (age == 21)
     q = false;

} while( q == true);
```

Switch - Syntax

- Compares an expression to a set of constants. If it finds a match it will begin executing statements until it reaches a break at which point the switch is exited.

```
switch (expression)  {

    case constant1:
             // code here
    break;

    case constant2:
             // code here
    break;
    .
    .
    .
    default:
             // more code
}
```

Example

Switch

```
switch (x) {
        case 'a':
        cout << "A is for Apple";
        break;

        case 'b':
        cout << "B is for Bat";
        break;

        case 'c':
        cout << "C is for Cat";
        break;

        default:
        cout << "Huh!?";
}
```

If-else

```
if (x == 'a')
  cout << "A is for Apple"

else if (x == 'b')
  cout << "B is for Apple"

else if (x == 'c')
  cout << "C is for Apple'

else
  cout << "Huh!?";
```

Sample Code – switch.cpp

```
#include <iostream>

using namespace std;

int main()  {

    bool keepGoing = true;
    char x;

    while (keepGoing)  {

        cout << endl << "Enter a char ";
        cin >> x;

        switch (x) {
            case 'a':
                cout << endl << "A is for Apple" << endl;
            break;

            case 'b':
                cout << endl << "B is for Bat" << endl;
            break;

            case 'c':
                cout << endl << "C is for Cat" << endl;
            break;

            default:
                cout << endl << "Huh!?" << endl;
                keepGoing = false;
        }
    }

    cout << endl << endl;
    return 0;
}
```

Break Statement

• the break statement can also be used inside a loop (either for or while loops)

• the break statement causes the loop to stop executing and shifts program control to the end (outside) of the loop

• for example:

```
for ( int I = 1; i < 11; i++ ) {
    cout << "I am " << i << endl;
    if ( i % 3 == 0 ) {
        break;
    }
    cout << "and I am NOT divisible by 3!\n";
}
cout << "goodbye" << endl;
```

and the output will be:

I am 1 and I am NOT divisible by 3!
I am 2 and I am NOT divisible by 3!
I am 3
goodbye.

• The program will stop the loop as soon as the condition (i % 3 == 0) is true. The next line executed is outside of the loop, where “goodbye” is displayed.

Continue Statement

• the continue statement can also be used inside a loop (also for either for or while loops)

• the continue statement causes the loop to stop executing and shifts program control back to the beginning (inside) the loop

• for example:

```
for ( int i=1; i<11; i++ ) {
    cout << "I am " << i << endl;
    if ( i % 3 == 0 ) {
       continue;
    }
    cout << "and I am NOT divisible by 3!\n";
}
 cout << "goodbye" << endl;
```

and the output will be:

```
I am 1
and I am NOT divisible by 3!
I am 2
and I am NOT divisible by 3!
I am 3
I am 4
and I am NOT divisible by 3!
I am 5
and I am NOT divisible by 3!
I am 6
I am 7
and I am NOT divisible by 3!
I am 8
and I am NOT divisible by 3!
I am 9
I am 10
and I am NOT divisible by 3!
goodbye.
```

• the program will jump to the next iteration in the loop as soon as the condition (i % 3 == 0) is true.