`while` Loops

Lecture 6

Outline

1. Looping
2. `while` Loops
3. `do-while` Loops

Looping

Often you need to repeat the same computation, action, or sequence of steps many times

Loops

- Of course, you could use 100 **`cout`** statements to accomplish this, but that's a lot of copy-and-paste work

- Instead, programming languages have control flow mechanisms called loops that allow you to loop over (repeat) the same section of code as many times as you need

- Two of the most common types of loops are **`while`** loops and **`for`** loops

C++ `while` Loops

`while` loops are used to repeat a set of C++ statements *while* some condition is true

```
int count = 1;
while ( count <= 100 )
{
    cout << "I will not expose the ignorance
             of the faculty.” << endl;

    count = count + 1;
}
```

General Form of the `while` Loop

```
while ( BOOLEAN EXPRESSION )
{
   STATEMENT1;
   STATEMENT2;
   …
}
```

Loop Body

- The loop body executes over and over as long as the expression is **true**
  - Boolean expressions same as `if`/`else` if statements
- Each repetition is called an iteration of the loop

Behavior of the `while` Loop

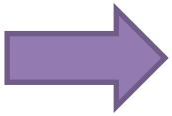
```
int input;
cout << "input: ";
cin >> input;

while ( input > 0 )
{
   cout << input << endl;
   input = input / 2;
}

cout << "Done." << endl;
```

Value of `input`

? 7 3 1 0

```
input: 7
7
3
1
Done.
```

Another Example



```
int i = 1;
int sum = 0;
while ( i <= 4 )
{
   sum = sum + i;
   i = i + 1;
}

cout << sum << endl;
```

```
10
```

Value of `i`

1 2 3 4 5

Value of `sum`

0 1 3 6 10

`i = i + 1`

- Always remember that `=` is NOT a statement of fact, it is a one time assignment of a value (RHS) to a variable (LHS)

- For example, if `i` currently has a value of 3, then it will plug that it to the right hand side of the equal sign, add one to get 4, then assign 4 back to the variable `i`

- The same is true for `sum = sum + i`

Notes on Execution

- When C++ reaches a **`while`** loop for the first time, it checks the condition
  - If it is **`true`** it begins running the statements inside the loop (in the loop body, between the curly braces)
  - If the condition is **`false`**, it skips past the while loop entirely

- When C++ executes the last statement inside a **`while`** loop and reaches the **`}`**, it goes back to the original **`while`** line and checks the condition again
  - The condition is only checked when the **`while`** line itself is executing, not after each statement inside the loop body

Exercise

Write a program that prints out all the numbers from 0 to N, where N is provided by the user. That is, ask the user for a number then print out all the numbers from 0 to that number.

Answer

```
#include <iostream>
using namespace std;

int main()
{
     int i = 0;
     int n;

     cout << "Enter N: ";
     cin >> n;

     while ( i <= n )
     {
          cout << i << endl;
          i = i + 1;
     }

     return 0;
}
```

Infinite Loops

- Always be careful to ensure that your loop conditions will be false eventually
- Loops that have conditions that are always true are infinite loops, and are usually a mistake
- You can halt a program stuck in an infinite loop by pressing Control-C in the console window

```
int iteration = 1;
while ( iteration > 0 )
{
    cout << "This is the song that doesn't end...";
    iteration = iteration + 1;
}
```

Increment/Decrement Operators

- C++ includes shorthand increment and decrement operators that are often useful with loops (and plenty of other times)

- **++** is the increment operator, used to increase a variable's value by one
  - `count++; // same as count = count + 1;`

- **--** is the decrement operator, used to decrease a variable's value by one
  - `i--;  // same as i = i - 1;`

do-while Loops

- A while loop body might be executed zero times if the expression is never true

- If you need to always execute the body at least once, use a **do-while** loop

```
char input;
do
{
    cout << "Enter y to print this message again\n";
    cin >> input;
} while ( input == 'y' );
```

General Form of the do-while Loop

Loop Body

```
do
{
   STATEMENT1;
   STATEMENT2;
   …
} while ( BOOLEAN EXPRESSION );
```

- Note: there must be a semicolon after the `while( BOOLEAN EXPRESSION )` in `do-while` loops, but NOT in `while` loops

Example: Sanitizing Inputs

```
double input;

do
{
   cout << "Enter a positive number: ";
   cin >> input;
} while ( input < 0 );

cout << "The square root is: " <<
        sqrt(input) << endl;
```

Exercise

Write a program that uses a **`do-while`** loop to read integer values from the user until a value between 0 and 100 is entered

Answer

```
#include <iostream>
using namespace std;

int main()
{
    int value;

    do
    {
        cout << "Enter a whole number [0, 100]: ";
        cin >> value;
    } while ( value < 0 || value > 100 );

    return 0;
}
```

Wrap Up

- Use `while` loops to repeat a series of statements so long as some condition is true
- Use `do-while` loops if you need to guarantee that the loop body executes at least once
- Be wary of infinite loops
- Use increment/decrement operators as shortcuts to add or subtract one from a variable