

Background

Repetition is an essential part of many computations. A sequence of instructions that the computer can repeat one or more times is called a *loop*. It is not unusual for a program to have many loops. Sometimes we even have programs with loops inside loops (which we call *nested loops*).

Almost every real-world application has at least one loop. Most graphical applications have something called the event-loop that checks to see whether the user has clicked a button or typed a key and then executes the appropriate code for handling that occurrence. Games usually have a loop which keeps track of whose turn it is.

Most loops have three parts:

1. A starting place
2. An ending place
3. A termination condition

In this lab, you will implement a simple guessing game. The computer will pick a random number. The user will type in a guess. If the guess is too low or too high, the program will tell the user that they have guessed incorrectly. This process will repeat (in a loop) until the user guesses the correct number.

Step 1. Setting Up

Create a directory named Project3. Change to that directory and create a file named project3.cc.

Initially, project3.cc should look like this:

```
#include <iostream>
#include <iomanip>
#include <stdlib.h>

using namespace std;

int main(int argc, char** argv){

}
```

Notice that in addition to our usual boilerplate, we have added another `#include` command. This new command pulls in the standard library. The standard library allows us to use something called the "random" function which is an easy way to obtain a random number.

Step 2. Initializing Variables

We are going to need two integer variables: one for the number and one for the guess.

In the main function, create an integer variable named "num". Initialize it to 0. Then create another integer variable named "guess" and initialize it to 0, too.

Now add the following two lines:

```
srandom(52);  
num = (int)(random()*100.0+1)/RAND_MAX;
```

This code asks the computer to give us a random number between 1 and 100. Don't worry about how it works for now. We will talk about functions later in the course.

Step 3. Writing a loop

Add the following lines of code to your main function:

```
while (guess != num){  
    cout << "Please enter a number: " << endl;  
    cin >> num;  
    cout << endl;  
  
    if (guess > num){  
        cout << "Too high." << endl;  
    }  
    else if (guess < num){  
        cout << "Too low." << endl;  
    }  
}  
cout << "You got it!" << endl;
```

The termination condition of this loop is "guess != num". In other words, while the user hasn't guessed the number, keep repeating the loop. The loop starts at the first curly brace after the while and includes everything until the last curly brace. The "You got it message" is not part of the loop. It will only be printed AFTER we terminate the loop.

Inside the loop, we use cout to print the message "Please enter a number" and then use cin to read a number into the variable named guess.

We print an end-of-line character so that the cursor will wrap to the next line.

Then we use an if-statement which will print the message "Too high" and an end-of-line character if guess is larger than num. Similarly, if guess is smaller than num, the program should print "Too low".

Step 4. Using a for loop

Playing one game of "guess a number" doesn't take very long. We would like to change the program so that the entire game repeats three times in a row. To do this, we can use a different type of loop called a "for" loop.

A for loop has four parts: an initialization statement, a termination condition, a loop increment, and a body. An example for loop is shown below:

```
for (int i=0; i < 10; i++){  
    cout << i << endl;  
}
```

This for loop counts from 0 to 9. The initialization, termination condition, and increment are grouped together between the parentheses in the for statement. They are separated by semi-colons. The body of the for loop lies between the curly braces.

The initialization statement is a command that is executed only once -- at the very beginning of the loop. In this case, it creates a new variable named "i" and sets its value to 0.

The termination condition is evaluated at the beginning of every repetition of the loop. If the condition is true, the loop is repeated. If it is not true, the loop is terminated and we skip down to the rest of the program. The increment is evaluated at the end of every repetition of the loop. In this case, it sets i to i + 1.

Add a for loop to your program. Everything that is currently in the main function should be in the body of the for loop. The for loop should use a variable named i to count from 0 to 2 so that the entire program will be repeated three times.

Step 5. Submitting

Make sure that you are in the directory about your Project3 folder. Then execute:

```
tar xzvpf Project3.tar.gz Project3/
```

This will create a file named Project3.tar.gz which you can submit in the usual manner at <https://narnia.homeunix.com/~robert/submit>