Lab **7**

Do While Loops

Learning Objectives

Want to repeat some instructions multiple times, you need to use repetition. C gives you several constructs suitable for such scenarios. Last week we used the while loops, today we will repeatedly execute a set of instructions until the conditions is false using do-while loop.

After completing this lab you will be able to write repetition statements in C, as well as, differentiate between while and do-while loops, and which to use for different problems.

Basic Lab Instructions!

- ❖ Talk to your classmates for help.
- ❖ You may want to bring your textbook to future labs to look up syntax and examples.
- ❖ Stuck? Confused? Have a question? Ask a TA/Lab Engineer for help, or look at the book or past lecture slides.
- ❖ Complete as many problems as you can within the allotted time. You don't need to keep working on these exercises after you leave the lab.
- ❖ Before you leave today, make sure to check in with one of the Lab Engineers/TAs in the lab to get credit for your work.

Lab Tasks

Creating a New Project

After completing this section you will have created a project for today's lab.

1. Open the solution CS110Labs you created in the lab last week.
2. Add a new project called Lab06 within the solution CS110Labs. Set the project as startup project (right-click the project in Solution Explorer). STOP HERE! start with the lab tasks, and refer to the following steps accordingly.
3. Click File in the top menu bar of Visual Studio, and then select the Add New Item option (Add Existing Item option depending on the lab task).
4. Select C++ File
5. Type in the code listed.
6. Compile (Build Lab06) the program and execute it (Execute without Debugging).

Task #1: do-while Swimming

So far you have only worked with one type of loop: the while loop. But there is another type: the "do-while" loop.

The do-while loop works almost exactly like a while loop. In fact, most of the time they are equivalent. Examine the program (DoWhileVacation.c) below to see if you can figure out the tiny difference.

```c
#include <stdio.h>
#include <stdlib.h>

int main( )
{
  char guestName1[] = "HAMZA";
  char guestName2[] = "ILSA";

  int totalNights;
  int nightsSpent;
  double totalCost;

  printf( "How many nights do you plan to stay? " );
  scanf( "%d", &totalNights );

  printf( "\nTotal planned stay: %d nights.\n", totalNights );
  printf( "%s travels to Chitral....\n\n", guestName1 );

  nightsSpent = 0;
  totalCost = 0.0;
  while ( nightsSpent < totalNights )
  {
    printf( "%s spends a night.\n", guestName1 );
    nightsSpent++;
    totalCost += 1500.0;
    printf( "\tNights spent: %d\n", nightsSpent );
    printf( "\tNights left:  %d\n", totalNights - nightsSpent );
  }

  printf( "\nTotal cost of stay: Rs. %.2f\n", totalCost );

  printf( "\nTotal planned stay: %d nights.\n", totalNights );
  printf( "%s travels to Chitral....\n\n", guestName2 );

  nightsSpent = 0;
  totalCost = 0.0;
  do
  {
    printf( "%s spends a night.\n", guestName2 );
    nightsSpent++;
    totalCost += 1500.0;
    printf( "\tNights spent: %d\n", nightsSpent );
    printf( "\tNights left:  %d\n", totalNights - nightsSpent );
  }
  while ( nightsSpent < totalNights );

  printf( "\nTotal cost of stay: Rs. %.2f\n", totalCost );

  return EXIT_SUCCESS;
}
```

What You Should See

Hamza and Ilsa are both planning a vacation. At the end of their stay they calculate the total cost.

Run the program, and type in 3 for the total planned vacations.

```
How many nights do you plan to stay? 3

Total planned stay: 3 nights. HAMZA travels to Chitral....
HAMZA spends a night.
        Nights spent: 1
        Nights left:  2
HAMZA spends a night.
        Nights spent: 2
        Nights left:  1
HAMZA spends a night.
        Nights spent: 3
        Nights left:  0
Total cost of stay: Rs. 4500.00

Total planned stay: 3 nights. ILSA travels to Chitral....
ILSA spends a night.
        Nights spent: 1
        Nights left:  2
ILSA spends a night.
        Nights spent: 2
        Nights left:  1
ILSA spends a night.
        Nights spent: 3
        Nights left:  0
Total cost of stay: Rs. 4500.00
```

What You Should Do on Your Own

Assignments turned in without these things will receive no credit.

1. Run the program, and type in 3 for the number of nights. Do Hamza and Ilsa stay for the same number of nights? Put your answer in a comment.
2. Run the program again, but this time enter 0 for the total vacation. What changes?
3. Does Hamza check the total vacations, or does he just drive to Chitral?
4. What about Ilsa? Does she check the total vacations first or just drive to Chitral?
5. What is the difference between a while loop and a "do-while" loop?
6. One of these loops is sometimes called a "pre-test loop", and the other is called a "post-test loop". Which one is which?

You have to turn-in DoWhileSwimming.c.

Task #2: Flip Again?

Open FlipAgain.c in your editor, you'll see how using a do-while loop might be better than a while loop.

What You Should See

The code I have provided does not compile. Once you fix it, it will look roughly like this.

```
You flip a coin and it is... TAILS
Would you like to flip again (y/n)? y
You flip a coin and it is... HEADS
Would you like to flip again (y/n)? y
You flip a coin and it is... HEADS
Would you like to flip again (y/n)? n
```

What You Should Do on Your Own

Assignments turned in without these things will receive no credit.

1. The code as given does compile, but the program seems not to work. Notice that the while loop tests if again == 'y', but the variable again doesn't have a value at first. Give it a value so that the loop will run at least once.
2. Now that program is working, change the loop from a while loop to a do-while loop. Make sure it still works.
3. What happens if you delete what you added in step 1? Change the line back to just char again; Does the program still work? Why or why not? (Answer in a comment.)

You have to hand-in the modified FlipAgain.c.

Task #3: Shorter Double Dice

Redo the Dice Doubles task (the dice program with a loop) so that it uses a do-while loop instead of a while loop. Otherwise it should behave exactly the same.

If you do this correctly, there should be less code in this version ShorterDoubleDice.c.

```
HERE COME THE DICE!

Roll #1: 3
Roll #2: 5
The total is 8!

Roll #1: 6
Roll #2: 1
The total is 7!

Roll #1: 2
Roll #2: 5
The total is 7!

Roll #1: 1
Roll #2: 1
The total is 2!
```

You have to turn-in ShorterDoubleDice.c.

Task #4: Again with the Number-Guessing

Redo the Number-Guessing with a Counter assignment using a do-while loop instead of a while loop. Otherwise it should do exactly the same things (including the counter). Name it AgainWithTheNumberGuessing.c.

Make sure that it doesn't mess up if you guess it on the first try.

```
I have chosen a number between 1 and 10. Try to guess it.
Your guess: 5
That is incorrect.  Guess again.
Your guess: 4
That is incorrect.  Guess again.
Your guess: 8
That is incorrect.  Guess again.
Your guess: 6
That's right! You guessed it.
It only took you 4 tries.
```

You have to turn-in AgainWithTheNumberGuessing.c.

Task #5: Square Root

Write a program SquareRoot.c to take the square root of a number typed in by the user. Your program should use a loop to ensure that the number they typed in is positive. If the number is negative, you should print out some sort of warning and make them type it in again.

Note that it is possible to do this program with either a while loop or a do-while loop. (Though personally, I think this one is easier with a while loop.)

You can get the square root of a number n with using sqrt from math.h.

```
double sqrt(double);
```

Make sure you don't do this until the loop is done and you know for sure you've got a positive number.

```
SQUARE ROOT!
Enter a number: 9
The square root of 9 is 3.0000000000000000.

SQUARE ROOT!
Enter a number: 2
The square root of 2 is 1.4142135623730951.

SQUARE ROOT!
Enter a number: -9
You can't take the square root of a negative number.
Try again: -10
You can't take the square root of a negative number.
Try again: 10
The square root of 10 is 3.1622776601683795.
```

You have to turn-in SquareRoot.c.

Task #6: Right Triangle Checker

Write a program RightTriangleChecker.c to allow the user to enter three integers. You must use do-while or while loops to enforce that these integers are in ascending order, though duplicate numbers are allowed.

Tell the user whether or not these integers would represent the sides of a right triangle.

```
Enter three integers:
Side 1: 4
Side 2: 3
3 is smaller than 4.  Try again.
Side 2: -9
-9 is smaller than 4.  Try again.
Side 2: 5
Side 3: 1
1 is smaller than 5.  Try again.
Side 3: 5

Your three sides are 4 5 5
NO! These sides do not make a right triangle!

Enter three integers:
Side 1: 6
Side 2: 8
Side 3: 10

Your three sides are 6 8 10
These sides *do* make a right triangle.
```

You have to turn-in RightTriangleChecker.c.

Task #7: Collatz Sequence

Take any natural number n.

- If n is even, divide it by 2 to get n / 2.
- If n is odd, multiply it by 3 and add 1 to get 3n + 1.
- Repeat the process indefinitely.

In 1937, Lothar Collatz proposed that no matter what number you begin with, the sequence eventually reaches 1. This is widely believed to be true, but has never been formally proved.

Write a program CollatzSequence.c that inputs a number from the user, and then displays the Collatz Sequence starting from that number. Stop when you reach 1.

Sample Output

Here's an example of the expected output, assuming I start with 6 and print tabs between each number.

```
Starting number: 6
6       3       10      5       16      8       4       2       1
```

Or, starting with a different number:

```
Starting number: 11
11      34      17      52      26      13      40      20      10      5       16      8       4       2       1
```

Some numbers take quite a while to reach 1:

```
Starting number: 27
27      82      41      124     62      31      94      47      142     71      214     107     322     161
484     242     121     364     182     91      274     137     412     206     103     310     155     466
233     700     350     175     526     263     790     395     1186    593     1780    890     445     1336
668     334     167     502     251     754     377     1132    566     283     850     425     1276    638
319     958     479     1438    719     2158    1079    3238    1619    4858    2429    7288    3644    1822
911     2734    1367    4102    2051    6154    3077    9232    4616    2308    1154    577     1732    866
433     1300    650     325     976     488     244     122     61      184     92      46      23      70
35      106     53      160     80      40      20      10      5       16      8       4       2       1
```

Bonus #1. Count Steps

For +10 bonus points, also display the total number of steps in the sequence.

```
Starting number: 11
11      34      17      52      26      13      40      20      10      5       16      8       4       2       1

Terminated after 14 steps.
```

```
Starting number: 27
27      82      41      124     62      31      94      47      142     71      214     107     322     161
484     242     121     364     182     91      274     137     412     206     103     310     155     466
233     700     350     175     526     263     790     395     1186    593     1780    890     445     1336
668     334     167     502     251     754     377     1132    566     283     850     425     1276    638
319     958     479     1438    719     2158    1079    3238    1619    4858    2429    7288    3644    1822
911     2734    1367    4102    2051    6154    3077    9232    4616    2308    1154    577     1732    866
433     1300    650     325     976     488     244     122     61      184     92      46      23      70
35      106     53      160     80      40      20      10      5       16      8       4       2       1

Terminated after 111 steps.
```

Bonus #2. Largest Value

For +20 bonus points, display the largest value encountered in the sequence.

```
Starting number: 11
11    34    17    52    26    13    40    20    10    5     16    8     4     2     1

The largest value was 52.
```

```
Starting number: 27
27    82    41    124   62    31    94    47    142   71    214   107   322   161
484   242   121   364   182   91    274   137   412   206   103   310   155   466
233   700   350   175   526   263   790   395   1186  593   1780  890   445   1336
668   334   167   502   251   754   377   1132  566   283   850   425   1276  638
319   958   479   1438  719   2158  1079  3238  1619  4858  2429  7288  3644  1822
911   2734  1367  4102  2051  6154  3077  9232  4616  2308  1154  577   1732  866
433   1300  650   325   976   488   244   122   61    184   92    46    23    70
35    106   53    160   80    40    20    10    5     16    8     4     2     1

The largest value was 9232.
```

Bonus #3

For +30 bonus points, do both.

```
Starting number: 11
11    34    17    52    26    13    40    20    10    5     16    8     4     2     1

Terminated after 14 steps. The largest value was 52.
```

You have to turn-in CollatzSequence.c.

Hand in

Hand in the word file for this lab at the appropriate location on the blackboard system at LMS. You should hand in a single file named Lab_7_<your reg. No. XXX without angle brackets>.doc(x) that contains the following.

1. All completed C source files representing the work accomplished for this lab: DoWhileSwimming.c; FlipAgain.c; ShorterDoubleDice.c; AgainWithTheNumberGuessing.c; SquareRoot.c; RightTriangleChecker.c; and CollatzSequence.c.
2. A paragraph at the end that includes a) a brief explanation of the lab, and b) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. In-class lab time is not intended as free time for working on your program assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the program assignment.