**Programming Fundamentals I**

**COSC1436— Fall 2015**

**Lab 5. Loops**

**Objectives:**

1. to develop algorithms and program that use repetition control structures
2. to explore how to construct and use counter-controlled, sentinel, flag-controlled and eof loops
3. to practice constructing and using for, while, and do-while loops
4. to learn how to form and use nested loops
5. to practice using flow charts in designing and verifying algorithms with repetition control structures statements

**Make sure to demo your work to your instructor for each task to get credit. All documents (flowchart and test data) that need to be submitted for this lab should be copy and pasted to this file before submitting the file to the D2L dropbox.**

**Task 1: Working with while loops**

1. The factorial of n, denoted n!, is defined as:

0! = 1

1! = 1

n! = 1\*2\*3\*….\*n for n > 1

So 4! = 1\*2\*3\*4 = 24 and 6! = 1\*2\*3\*4\*5\*6 = 720

Write an algorithm (pseudocode) to do the following:

* prompt user to input an integer number n ( >= 0)
* calculate the factorial of the number entered
* output the result to standard output device

1. Create a flowchart for your algorithm using Raptor and verify it for the example cases listed above.
2. Modify your flowchart of Exercise 2 to valid the input using a loop that repeatedly prompts the user to enter a valid input (i.e. a number greater or equal to 0).
3. Write a C++ program for the flowchart of Exercise 3.

**Demo your program to your instructor. Copy and paste your flowchart below for submission to the D2L dropbox upon completion of your lab.**

**Task 2: Working with counter-controlled while loops**

1. Write an algorithm (pseudocode) to read a set of sales data items from standard input and calculate and output their total and their average. Prompt user to enter number of data items.
2. Create a test data set to verify your algorithm. How many cases are needed? Explain.

**Write your test data set below for submission to the D2L dropbox upon completion of your lab.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of items | List data items | Expected output |
| Case 1: |  |  |  |
|  |  |  |  |

1. Create a flowchart for your algorithm on Raptor and verify it using your test data. **Copy and paste your flowchart below for submission to the D2L dropbox upon completion of your lab.**
2. Write a C++ program that implements your flowchart. Use a do-while loop for input validation and a for loop to calculate the total (You may skip this exercise until we cover for and do-while loops)

**Demo your program to your instructor.**

**Task 3: Working with sentinel-controlled loops**

1. In the program of the previous task, we assumed the number of data items is known beforehand. Sometimes that’s not the case. In those cases, instead of asking user for the number of data items, the program prompts the user to enter a sentinel as the last piece of data. Please refer to the textbook or class notes for the definition and use of sentinels.

Modify your program from the previous task so that the loop stops reading sales data when -1 is entered. Make sure to modify the prompt to inform the user to enter -1 when there are no more data items to enter.

**Demo your modified program for credit.**

**Task 4: Working with files and end-of-file controlled loops**

**Note:** You may skip this task until we cover the section about using files for input/output

1. Modify your program from Task 3 so that sales data is read from a text file instead of being entered in the standard input device. Assume the data items are stored in the file separated by blank spaces and -1 is used as a sentinel.
2. Modify your program from Exercise 1 above to read the sales data until the end of the file (i.e. no sentinel is used).

**Demo your modified program for credit.**

**Task 5: Working with flag-controlled loops, do-while loops, and nested loops**

1. The following program simulates the number guessing game that you’re all familiar with. Create a new C++ project in Visual Studio, add a new C++ file and copy the following code into it:

//PLACE YOUR NAME HERE

//Date:

// Program simulates the number guessing game

#include<iostream>

using namespace std;

const int MAX = 1000;

int main()

{

char ch;

cout << "Think of an integer number between 0 and " << MAX<<endl;

cout << "Write it down on a piece of paper then hit a key to continue"<<endl<<endl;

cin.get(ch);

cout << endl<<"I am going to guess the number you picked... " ;

cout << "You just need to give me some hints"<<endl<<endl;

cout <<"Hit any key when you're ready..."<<endl<<endl;

cin.get(ch);

int first = 0; // First element of list.

int last = MAX; // last element of the list

int middle; // variable containing the current middle value of the list

int count = 1;

while (count <= 10)

{

middle = first + (last - first) / 2;

cout <<" Is it: "<< middle <<"?" <<endl<<endl;

cout <<"Press y for Yes or N for No:";

cin >> ch;

ch = toupper(ch);

/\* ADD A LOOP TO VALIDATE INPUT IS EITHER 'Y' or 'N'\*/

if (ch == 'Y')

{

cout <<"Bingo...I did it!!!"<<"in "<< count <<" guesses!!" <<endl;

cout <<"Goodbye"<< endl;

cin.get(ch); cin.get(ch);

break;

}

else

{

cout <<"Umm! Is your number higher or lower than "<< middle<< "?";

cout <<endl << endl;

cout <<"Press H for higher and L for lower:";

cin >> ch;

ch = toupper(ch);

/\* ADD A LOOP TO VALIDATE INPUT IS EITHER 'H' or 'L'\*/

if (ch == 'H' ) // if number > guessed value then number

// is in [middle+1 ... last]

first = middle + 1;

else if (ch == 'L' ) // if number < guessed value then number is

// in [first ... middle-1]

last = middle -1;

}

count ++;

}

return 0;

}

1. The program uses a break statement to exit the loop when the number is guessed correctly. The break and continue statements however are considered to be a bad programming practice because they violate the one-entry point/one-exit point rule for proper loop construction which makes debugging much more difficult. Modify the program so it uses a **flag variable** that is set to true when the number is guessed correctly. The flag variable should also be used to control the loop.
2. Add loops to validate the user input where indicated with upper-case comments.
3. The program allows the user to play the game only once each time it’s executed. Modify the program so that it allows the user to play the game repeatedly in each run.
4. Place the outer-while loop inside a do-while loop.
5. The do-while loop should ask the user at the end of each while-loop iteration if they wish to play again. Based on the user’s response, the do-while loop would either exit or iterate one more time.

**Demo your modified program for credit.**