**Lesson5 - Lab –Counter Controlled Loops**

# AIMS

To consolidate fundamentals of the C language, with emphasis on decision and repetition control structures

# Objectives

When you have completed these exercises, you should be able to:

• Demonstrate the use of program control using decision and repetition structures • Write counter controlled loop using **for loop**.

**Background:**

The basic repetition constructs in C are the:

|  |  |  |
| --- | --- | --- |
| **while** (condition)  {  statement;  } | **do{**  statement;  **}while**(condition); | **for**(initialisation; condition; increment)  {  statement;  } |
|  |  | is  condition  true ?  initialisation;  yes  no  statement;  increment; |

These allow a program block to be repeatedly executed and where the amount of repetition is controlled either by "counting" a known number of repetitions, or by performing a logical test (evaluating a relational condition) to determine exit from the "loop".

# Exercises

Complete the following exercises, making sure that each program is saved on completion of each exercise logged and uploaded to Brightspace.

## Exercises 1 – Counter Controlled loop using while loop and for loop

Start the Integrated Development Environment and create a new C project **Lab5ex1** Write a main function that:

1. firstly, uses the **while loop** to display the odd integers from 1 to 19;
2. secondly, uses a **for loop** to display the odd integers from 1 to 19;

1 3 5 7 9 11 13 15 17 19

Compile, link and execute the program.

Code: Log your code and run attempt here >>

/\* Program: lab5ex1

Name: Nneoma Onwe

Date:

\*/

# include <stdio.h>

int main()

{

int count = 0;

printf("using while loops\n");

count = 1;

while (count <= 19)

{

printf("%d ",count);

count = count + 2;

}

printf("\n");

printf("using while loops\n");

for (count = 1; count <= 19 ; count = count + 2)

{

printf("%d ", count);

}

printf("\n");

return 0;

}

### Test Results >> (screen capture)

|  |
| --- |
|  |

## Exercises 2 –Running Sum using Counter Controlled loop

Start the Integrated Development Environment and create a new C project **Lab5ex2** Write a main function that uses a **for loop** to compute the sum of the numbers 1 to 10.

|  |  |
| --- | --- |
| 10  𝑖𝑖 | = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55 |

i=1

Compile, link and execute the program.

Code: Log your code and run attempt here >>

/\* Program: lab5ex2

Name:

Date:

\*/

#include <stdio.h>

int main()

{

int i = 0;

int sum= 0;

for (int i=1; i <= 10; i++)

{

sum = sum + i;

printf("i= %d\tsum = %d\n",i, sum);

}

return 0;

}

### Test Results >> (screen capture)

|  |
| --- |
|  |

## Exercises 3

Create a new C project **Lab5ex3** and write a main function that displays a table of values (1 .. 10) along with their square and a running sum e.g.

1 1 1 2 4 3

1. 9 6
2. 16 10
3. 25 15
4. 36 21
5. 49 28
6. 64 36
7. 81 45
8. 100 55

Hint: the first column which contains the values of x, for x = 1 to x =10, the second column contains the values of X2 where x = 1 to x =10 and the third column contain the value of the running sum.

Compile, link and execute the program.

Code: Log your code and run attempt here >>

/\* Program: lab5ex3

Name:

Date:

\*/

#include <stdio.h>

int main()

{

int runningSum = 0;

printf("x\tX^2\trunning sum\n");

printf("----------------------\n");

for (int x = 1; x <= 10; x++) {

int square = x \* x;

runningSum += x;

printf("%d\t%d\t%d\n", x, square, runningSum);

}

return 0;

}

### Test Results >> (screen capture)

|  |
| --- |
|  |

## Exercise 4 - Nested For Loop to Draw Rectangle

Write a program using a nested for loop that draws a rectangle that has a **horizontal side double the size of the vertical side** where the size is supplied by the user.

*Example-*for a side of size 5 your program would display the following shape:

* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*
* \* \* \* \* \* \* \* \* \*
* \* \* \* \* \* \* \* \* \*
* \* \* \* \* \* \* \* \* \*

The program should

1. ask the user to enter a size for the side of a rectangle. Then the program should check that the size is an integer between 1 and 20 inclusive, if the user enters an invalid size, then repeatedly ask the user for the size until a proper value is entered.

1. draw the rectangular shape using the \* symbol.

Note that a single space separates the horizontal \*s.

Compile, link and execute the program.

Code: Log your code and run attempt here >>

/\* Program: lab5ex4

Name:

Date:

\*/

#include <stdio.h>

int main()

{

int size;

int validInput = 0;

while (!validInput) {

// Ask the user to enter the size and validate it

printf("Enter the size of the rectangle (1-20): ");

if (scanf("%d", &size) == 1 && size >= 1 && size <= 20) {

validInput = 1; // Set the flag to exit the loop

} else {

while (getchar() != '\n'); // Clear input buffer

printf("Invalid input. Please enter a valid size between 1 and 20.\n");

}

}

// Draw the rectangular shape

for (int i = 1; i <= size; i++) {

for (int j = 1; j <= size \* 2; j++) {

printf("\* ");

}

printf("\n");

}

return 0;

}

### Test Results >>

|  |
| --- |
|  |

# Optional Homework Exercises

## Exercise 5: Draw Triangle

Write a program a draw a triangle. The program should:

1. ask the user to enter a size for the size of the triangle. Then the program should check that the size is an integer between 1 and 20 inclusive, if the user enters an invalid size, then repeatedly ask the user for the size until a proper value is entered.
2. draw the right angled triangle shape using the \* symbol.

Note: the triangle size indicates the number of rows of \*s. A single space separates the horizontal \*s. The number of \*’s in each row is the same as the row number, i.e. row 1 has 1 \*, row 2 has 2 \*s etc.

*Example-* for a size 6 your program would display the following shape:

\*

* + \*
  + \* \*
  + \* \* \*
  + \* \* \* \*
  + \* \* \* \* \*

In this version of the code, we use the **validSize** variable to control whether the loop should continue asking for input. It will keep asking for input until a valid size is entered without using the **break** statement.

1. Compile, link, and execute the program to see it in action.
2. Top of Form

#include <stdio.h>

int main() {

int size;

// Ask the user to enter the size and validate it

int validSize = 0;

while (!validSize) {

printf("Enter the size of the triangle (1-20): ");

if (scanf("%d", &size) == 1 && size >= 1 && size <= 20) {

validSize = 1; // Valid input, exit the loop

} else {

while (getchar() != '\n'); // Clear input buffer

printf("Invalid input. Please enter a valid size between 1 and 20.\n");

}

}

// Draw the right-angled triangle

for (int i = 1; i <= size; i++) {

for (int j = 1; j <= i; j++) {

printf("\* ");

}

printf("\n");

}

return 0;

}

A black screen with white text

Description automatically generated

# Exercise 6: Draw a Tree

Create a program using loops to display a Christmas tree composed of stars, for example

\*

* \*
* \* \*
* \* \* \*
* \* \* \* \*
* \* \* \* \* \*
* \*
* \*

Allow the user to input different sizes for the tree. Make sure to include error checking on the input size value.

#include <stdio.h>

int main() {

int size;

// Ask the user to enter the size and validate it

int validSize = 0;

while (!validSize) {

printf("Enter the size of the Christmas tree: ");

if (scanf("%d", &size) == 1 && size >= 3 && size <= 20) {

validSize = 1; // Valid input, exit the loop

} else {

while (getchar() != '\n'); // Clear input buffer

printf("Invalid input. Please enter a size between 3 and 20.\n");

}

}

// Draw the Christmas tree

for (int i = 1; i <= size; i++) {

for (int j = 1; j <= size - i; j++) {

printf(" ");

}

for (int k = 1; k <= i \* 2 - 1; k++) {

printf("\*");

}

printf("\n");

}

// Draw the tree trunk

for (int i = 1; i <= size / 3; i++) {

for (int j = 1; j <= size - 2; j++) {

printf(" ");

}

printf("\*\n");

}

return 0;

}

A screenshot of a computer

Description automatically generated