

INSTITUTE OF ENGINEERING

Pulchowk Campus, Lalitpur



Subject: C Programming

Lab Report 1 and 2

Title: **Formatted I/O and Branching**

Submitted by:

Susheel Thapa 077BCT090

Submitted to:

Department of Electronics
and

Computer Engineering

Checked by

Content of Lab Report:

Background Information

C Programming

Editor Used

Compiler

Formatted I/O

Branching

Code and Output

Algorithm

Flowchart

Source Code

Output

Analysis

Conclusion

Background Information

What is C Programming?

C programming is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. C is the most widely used computer language.

Why to Learn C Programming?

- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

Editor

Here, I have used Visual Studio Code as my editor. You can download the editor from [Download Visual Studio Code - Mac, Linux, Windows](#) . Select your operating system and download it.

Compiler

Here, I have used **gcc** as my compiler provided by MinGWw64. You can download it via [Download MinGW-w64 - for 32 and 64 bit Windows from SourceForge.net](#). Your download will start automatically. Run the downloaded .exe file. After, you have installed MinGW-w64, you need to configure it.

1. In the Windows search bar, type 'settings' to open your Windows Settings.
2. Search for Edit environment variables for your account.
3. Choose the Path variable and then select Edit.
4. Select New and add the Mingw-w64 destination folder path to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If you used the settings above to install Mingw-w64, then add this to the path: **C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin.**
5. Select OK to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available.

Check your installation

Open command prompt or power shell and type:

```
C:\Users\user>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
C:\Users\user>gcc
gcc: fatal error: no input files
compilation terminated.

C:\Users\user>_
```

If you get similar result, you are good to go.

Formatted Input

Integer

%wd --> w == Width of the int variable

Syntax:

```
scanf("%2d%3d", &a, &b);
```

Here the width of a is 2 and that of b is 3.

Float/Double

%wf --> w is width of float variable

Syntax:

```
scanf("%3f%4f", &a, &b);
```

This is also same as before the only difference is that previous one is integer and this one is float

Width of a is 3 and of b is 4.

Note that the '.' also occupy one width space

String

`%ws` --> `w` is the width of string

Syntax:

```
scanf("%12s", str);
```

This set the field of `str` to 12. And `str` cannot have more than 12 width space or

Character

Read string in C using `scanf()` with `%c`

```
scanf("%10c", str);
```

It is not a good idea to use this method. The above program read exactly 9 characters if we try to give less than 9 characters then it will not give output until at least 9-character size is reached.

Read string in C using scan set conversion code (`[...]`)

```
scanf("%[a-z]",str);
```

This reads lowercase characters whenever it encounters another character except for lowercase charcater then it stops reading

Read string in C using `scanf` with `[\n]` (single line)

```
scanf("%[\n]", str);
```

The circumflex (^) plays an important role while taking input. `[\n]` will read input until the user presses the enter key, otherwise, it will keep reading input from the screen.

Multiline input using `scanf`

```
scanf("%[^\n]", str);
```

The above program will read input until the user enters the star * symbol. After * symbol, it will stop reading. In this way, we can read multiple lines of characters. It will read white space also, only stop reading when a * symbol came. Otherwise, it will keep reading the input from the user.

Formatted Output

Integer Output

`%wd:`

Here ‘d’ is used in its standard meaning while ‘w’ specifies the integer number which represents the minimum field width of any integer output data.

Floating Point Output

`%w.nf:`

Here ‘f’ is used in its standard meaning to represent the floating-point number, while ‘w’ specifies the integer number which represents the minimum field width of any integer output data. Here ‘n’ specifies the number of digits after the decimal. By default, the six digits have to be printed after the decimal.

String Output

`%w.ns:`

In formatted string output the decimal point and the latter ‘n’ are used optionally. Here ‘n’ specifies the first ‘n’ characters of the string that will be displayed along with the ‘w-n’ leading blanks.

Branching

If...else statement

Syntax:

```
if(boolean_expression)  
{ /* statement(s) will execute if the Boolean expression is true */}  
else  
{ /* statement(s) will execute if the Boolean expression is false */}
```

If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed.

If...else if...else Statement

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

When using if...else if..else statements, there are few points to keep in mind –

- An if can have zero or one else's and it must come after any else ifs.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax:

```
if(boolean_expression 1)  
{ /* Executes when the boolean expression 1 is true */}  
else if( boolean_expression 2)  
{ /* Executes when the boolean expression 2 is true */}  
else if( boolean_expression 3)  
{ /* Executes when the boolean expression 3 is true */}  
else  
{ /* executes when the none of the above condition is true */}
```

For Loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for ( init; condition; increment)  
{ statement(s);}
```

Here is the flow of control in a 'for' loop –

- The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

While Loop

A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax

```
while(condition)  
{ statement(s);}
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any nonzero value. The loop iterates while the condition is true.

When the condition becomes false, the program control passes to the line immediately following the loop.

Do ...While Loop

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Syntax:

```
do { statement(s);  
while( condition );
```

The conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

CODE AND OUTPUT

Lab 3

1. Write a program to read a character using getch()/getchar() and display using putchar()/putchar()

Algorithm:

- Start
- Declare a variable
- Read the value of variable
- Display the variable
- End

Source code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main ()
{
    char a_getch, a_getchar;
    system("cls");
    printf("Enter another character: ");
    a_getchar = getchar();
    printf("Enter any character: ");
    a_getch = getch(); /*This getch wonot be display in termianl as you type*/
    printf("\nValue taken via getch is ");
    putchar(a_getchar);
    printf("\nValue taken via getch is ");
    putchar(a_getch);
    getch();
    return 0;
}
```

Output:

```
Enter another character: F
Enter any character:
Value taken via getchar is F
Value taken via getch is * _
```

2. Write a program to read a character and string using scanf() and display it using printf().

Algorithm:

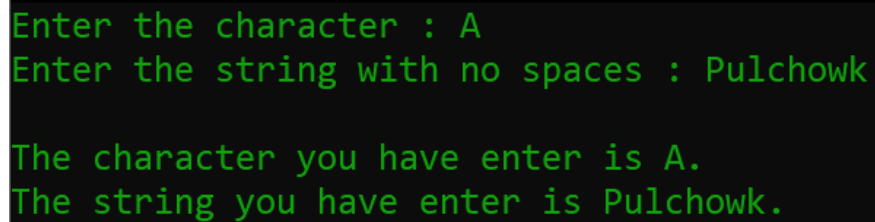
- Start
- Declare a variable
- Read the value of variable (scanf())
- Print the readed variable value
- End

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    int character, string[20];
    system("cls");
    printf("Enter the character : ");
    scanf("%c", &character);
    printf("Enter the string with no spaces : ");
    scanf("%s", string);
    printf("\nThe character you have enter is %c.", character);
```

```
printf("\nThe string you have enter is %s.", string);  
getch();  
return 0;  
}
```

Output:

A screenshot of a terminal window with a black background and green text. It shows the program's output: 'Enter the character : A', 'Enter the string with no spaces : Pulchowk', 'The character you have enter is A.', and 'The string you have enter is Pulchowk.'

```
Enter the character : A  
Enter the string with no spaces : Pulchowk  
  
The character you have enter is A.  
The string you have enter is Pulchowk.
```

3. Write a program to read a string using gets() and display using puts()

Algorithm:

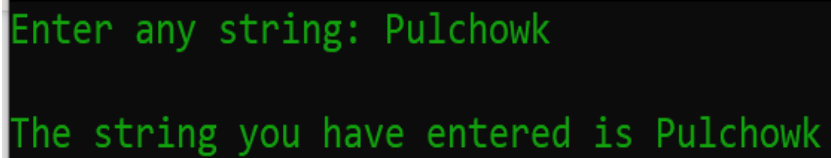
- Start
- Declare a string variable
- Read the value of string and store in the declared variable (gets ())
- Display the variable value
- End

Source Code:

```
#include <stdio.h>  
#include <conio.h>  
#include <stdlib.h>  
  
int main ()  
{  
    char string_gets[90];  
    system("cls");  
    printf("Enter any string: ");  
    gets(string_gets);  
    printf("\nThe string you have entered is ");  
    puts(string_gets);  
}
```

```
getch();  
return 0;  
}
```

Output:



```
Enter any string: Pulchowk  
The string you have entered is Pulchowk
```

4.This illustrates different format specifications for printing integer numbers.

Source Code:

```
#include <stdio.h>  
#include <conio.h>  
#include <stdlib.h>  
int main()  
{  
    int a = 12345;  
    system("cls");  
    printf("\nCase 1: %d", a);  
    printf("\nCase 2: %i", a);  
    printf("\nCase 3: %15d", a);  
    printf("\nCase 4: %-15d", a);  
    printf("\nCase 5: %015d", a);  
    printf("\nCase 6: %+15d", a);  
    printf("\nCase 7: %3d", a);  
    getch();  
    return 0;  
}
```

Output:

```
Case 1: 12345
Case 2: 12345
Case 3:          12345
Case 4: 12345
Case 5: 00000000012345
Case 6: +12345
Case 7: 12345
```

5.This example illustrates different format specifications for printing real numbers.

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
float n = 123.9876;
system("cls");
printf("\nCase 1: %f", n);
printf("\nCase 2: %e", n);
printf("\nCase 3: %g", n);
printf("\nCase 4: %15.4d", n);
printf("\nCase 5: %-15.3d", n);
printf("\nCase 6: %015.4ed", n);
printf("\nCase 7: %.8d", n);
printf("\nCase 8: %2.2d", n);
getch();
return 0;
```

```
}
```

Output:

```
Case 1: 123.987602
Case 2: 1.239876e+002
Case 3: 123.988
Case 4:      -536870912
Case 5: -536870912
Case 6: 00001.2399e+002d
Case 7: -536870912
Case 8: -536870912
```

6.This example illustrate different format specifier for printing character

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    char ch = 'a';
    system("cls");
    printf("\nCase 1: %c", ch);
    printf("\nCase 2: %10c", ch);
    printf("\nCase 3: %-10c", ch);
    getch();
    return 0;
}
```

Output:

```
Case 1: a
Case 2:      a
Case 3: a
```

7.This example illustrate different format specifier for printing string

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main ()
{
    char str[20] = "I love Balgung.";
    system("cls");
    printf("\nCase 1: %s", str);
    printf("\nCase 2: %18s", str);
    printf("\nCase 3: %-18s", str);
    printf("\nCase 4: %18.8s", str);
    printf("\nCase 5: %-18.9s", str);
    printf("\nCase 6: %5s", str);
    printf("\nCase 7: %10s", str);
    getch();
    return 0;
}
```

Output:


```
Case 1: I love Balgung.
Case 2:      I love Balgung.
Case 3: I love Balgung.
Case 4:              I love B
Case 5: I love Ba
Case 6: I love Balgung.
Case 7: I love Balgung.
```

8.This example illustrates the concept of printing mixed data.

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    int n = 12345;
    float m = 123.9876;
    char ch = 'a';
    char str[20] = "I love Baglung.";
    system("cls");
    printf("n=%7d\nm=%12.5f\nch=%-2c\nstr=%16s\n", n, m, ch, str);
    getch();
    return 0;
}
```

Output:

```
n= 12345
m= 123.98760
ch=a
str= I love Baglung.
```

9.This example illustrates different format specifications for reading integers numbers

Source Code:

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

int main()
{
    int a, b;
    system("cls");
    printf("Enter the number: ");
    scanf("%d", &a);
    printf("The read and stored value of a is %d\n\n", a);
    printf("Enter another integers number: ");
    scanf("%3d", &b);
    printf("The read and stored value of b is %d\n", b);
    getch();
    return 0;
}
```

Output:

```
Enter the number: 123456789
The read and stored value of a is 123456789

Enter another integers number: 1234567
The read and stored value of b is 123
```

10.This example illustrates the concept of reading string using %wc format specification

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    char str[50];
    system("cls");
    printf("Enter the string: ");
    scanf("%10c", &str);
    printf("Read string : %s", str);
    getch();
    return 0;
}
```

Output:

```
Enter the string: iAmVeryGoodBoyLivesInChitwan
Read string : iAmVeryGoo
```

11.This example shows the concept of defining search set to read string.

Source Code:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main()
{
    char str[70];
    system("cls");
    printf("Enter the string: ");
    scanf("%[a-zA-Z0-9]",str);
    printf("Read string: %s",str);
    getch();
    return 0;
}

```

Output:

```

Enter the string: noiwjeq$
Read string: noiwjeq

```

```

Enter the string: ak.heo9nwekUW3*KAF
Read string: ak

```

12.This example show the concept of defining search set to read string

Source Code:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main()
{

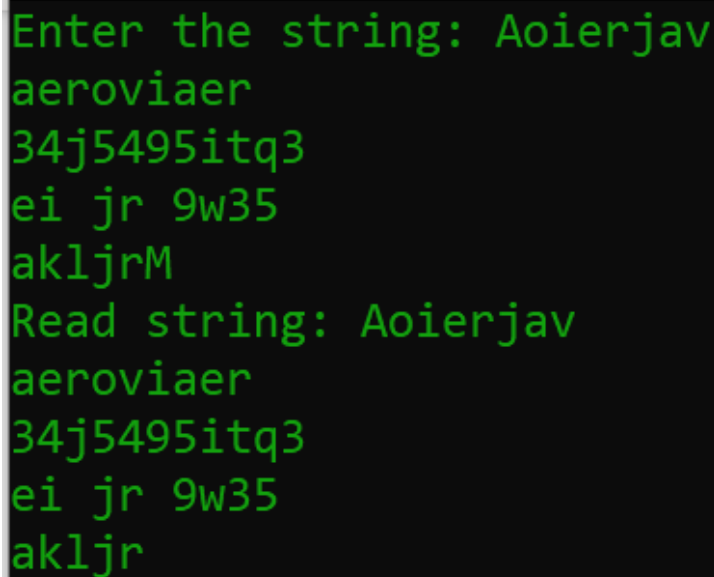
```

```

char str[70];
system("cls");
printf("Enter the string: ");
scanf("%[^M]",str);
printf("Read string: %s",str);
getch();
return 0;
}

```

Output:



```

Enter the string: Aoierjav
aeroviaer
34j5495itq3
ei jr 9w35
akljrM
Read string: Aoierjav
aeroviaer
34j5495itq3
ei jr 9w35
akljr

```

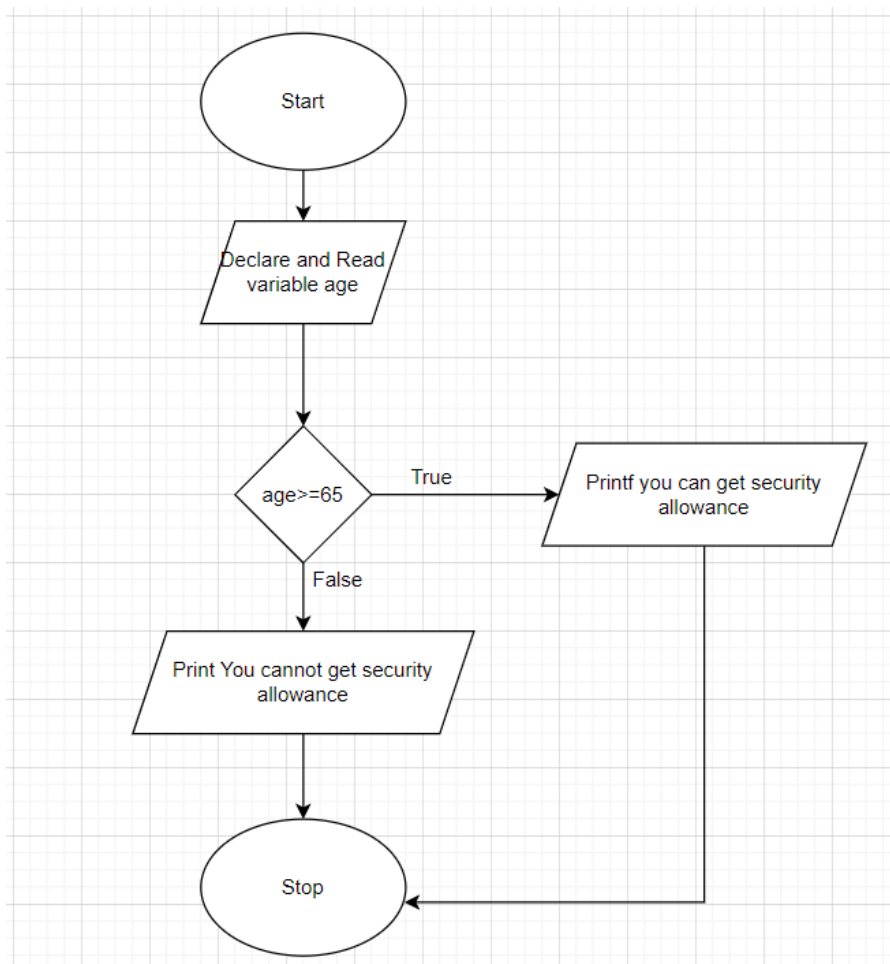
LAB 4

1.If a person age is greater than 65, he gets security allowance. Write a program to read the age of a person and display the appropriate message.

Algorithm:

- ❖ Start
- ❖ Declare a variable age
- ❖ Read the value of age and store the value in age
- ❖ if age>65
 - True: Print you will get security allowance
 - False: Print you won't get security allowance.
- ❖ Stop

Flowchart:



Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int main(){
    int age;
    system("cls");
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("\nYour age is %d.",age);
```

```
if(age >=65){  
    printf("\nYou can get security allowance.");  
}  
else{  
    printf("\nYou can't get security allowance.");  
}  
getch();  
return 0;  
}
```

Output:

```
Enter your age: 65  
  
Your age is 65.  
You can get security allowance.
```

```
Enter your age: 15  
  
Your age is 15.  
You can't get security allowance.
```

2. Write a program to read an integer from the user and check whether it is positive, zero or negative and display the appropriate message.

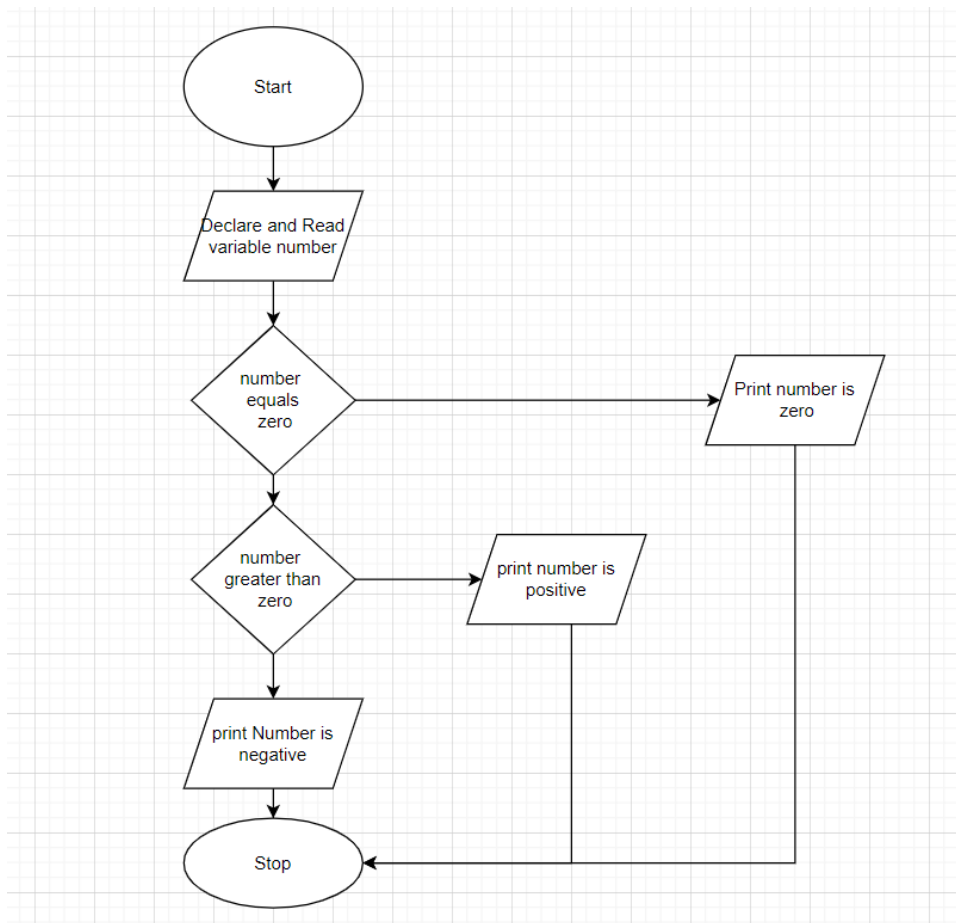
Algorithm:

- ❖ Start
- ❖ Declare a variable number
- ❖ Read the value of number
- ❖ Check number == 0
 - True: Print Number is zero
 - False:
 - Check number > 0
 - True: Print number is greater than zero

- False: Print number is less than zero.

❖ Stop

Flow Chart:



Source Code:

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
int main(){
    int number;
    system("cls");
    printf("Enter any number: ");
```



```
scanf("%d", &number);
printf("\nNumber = %d", number);
if (number == 0){
printf("\nIt is zero.");
}
else if(number>0){
printf("\nNumber is positive.");
}
else{
printf("\nNumber is negative.");
}
getch();
return 0;
}
```

Output:

```
Enter any number: 9
```

```
Number = 9
```

```
Number is positive.
```

```
Enter any number: -8
```

```
Number = -8
```

```
Number is negative. _
```

```
Enter any number: 0
```

```
Number = 0
```

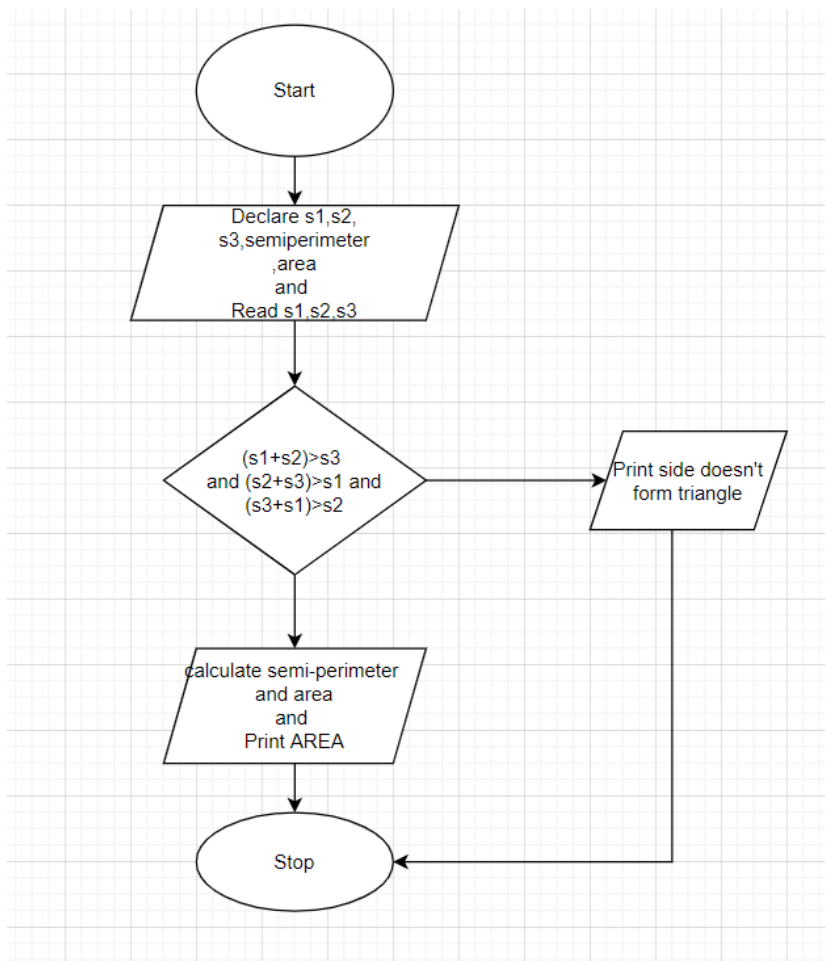
```
It is zero.
```

3. Write a program to read the three sides of triangle from the user and calculate the area of the triangle. Be sure to check the condition if triangle is formed or not.

Algorithm:

- ❖ Start
- ❖ Declare variable side one, side two, side three, area, semiperimeter
- ❖ Read the value of sides from user
- ❖ Check sum of any two sides > third side
 - True:
 - calculate semi-perimeter and then area
 - print the value of area
 - False:
 - Print error value don't form triangle
- ❖ Stop

Flow chart:



Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>

int main (){
float side_one, side_two, side_three, semi_perimeter, area_square, area;
system("cls");
printf("Side one: ");
scanf("%f",&side_one);
printf("Side two: ");
scanf("%f",&side_two);
printf("Side three: ");
scanf("%f",&side_three);
semi_perimeter = (side_one+ side_three+side_two)/2.0;
area_square = semi_perimeter*(semi_perimeter-side_one)*(semi_perimeter-
side_two)*(semi_perimeter-side_three);
area = pow(area_square, 0.5);
if((side_one+side_three)>side_two && (side_two+side_three)>side_one &&
(side_one+side_two)>side_three)
{
printf("\nThe area of the triangle is %.2f.", area);
}
else
{
printf("\nError: Side given were not of triangle. Check your value once.");
}
getch();
```

```
return 0;  
}
```

Output:

```
Side one: 3  
Side two: 4  
Side three: 5  
  
The area of the triangle is 6.00.
```

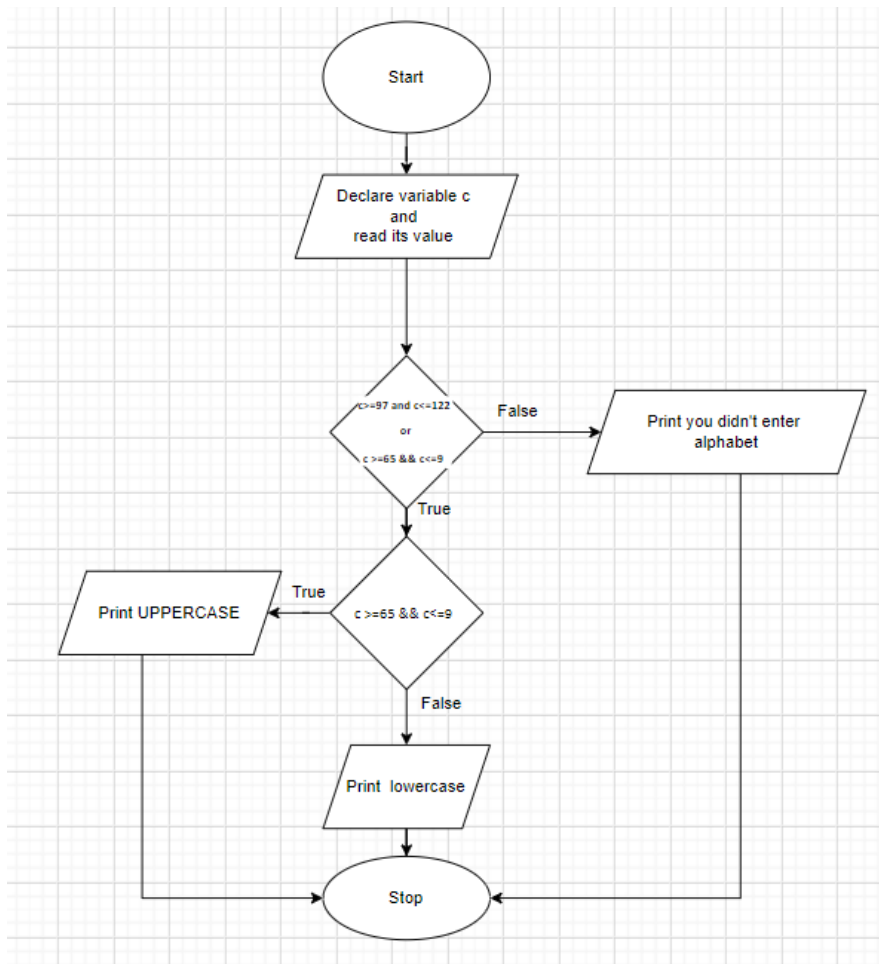
```
Side one: 1  
Side two: 9  
Side three: 12  
  
Error: Side given were not of triangle. Check your value once.
```

4. Write a program to read a character and check whether the character is upper or lower case

Algorithm:

- ❖ Start
- ❖ Declare the variable character
- ❖ Read the value of character
- ❖ $\text{character} \geq 97$ and $\text{character} \leq 122$ or $\text{character} \geq 65$ && $\text{character} \leq 90$
 - True:
 - $\text{character} \geq 65$ && $\text{character} \leq 90$
 - True:
 - Print it is upper case
 - False:
 - Print it is lower case
 - False:
 - Error!! You did not enter correct input
- ❖ Stop

Flow chart:



Source Code:

```

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int main(){
    char character;
    system("cls");
    printf("Enter an alphabet: ");
    scanf("%c", &character);
    printf("Alphabet: %c", character);
    if((character >= 97 && character <= 122) || (character >= 65 && character <= 90)){
        if (character >= 65 && character <= 90){

```

```

printf("\nIt is UPPER CASE");
}
else{
printf("\nIt is lower case.");
}
}
else{
printf("\nERROR: You didn't input alphabet.");
}
getch();
return 0;
}
/*

```

The ASCII value of the lowercase alphabet is from 97 to 122. And, the ASCII value of the uppercase alphabet is from 65 to 90. If the ASCII value of the character entered by the user lies in the range of 97 to 122 or from 65 to 90, that number is an alphabet.

**/*

Output:

```

Enter an alphabet: R
Alphabet: R
It is UPPER CASE_

```

```

Enter an alphabet: R
Alphabet: R
It is UPPER CASE_

```

```

Enter an alphabet: a
Alphabet: a
It is lower case._

```

```

Enter an alphabet: &
Alphabet: &
ERROR: You didn't input alphabet.

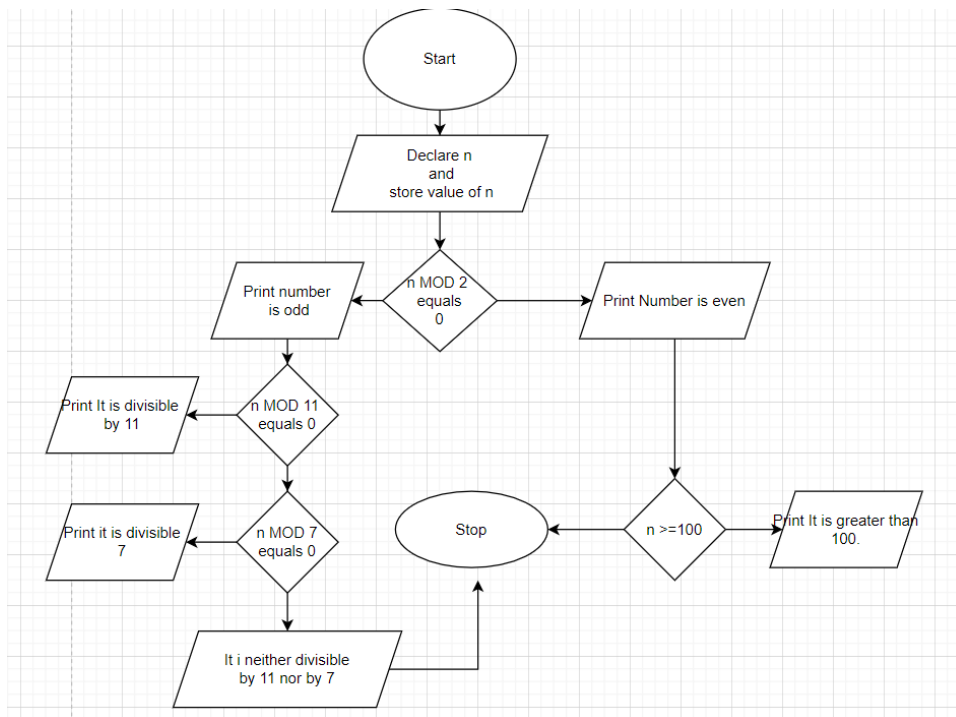
```

5. Write a program to read a unsigned integer and check whether the number is odd or even. If it is even , check whether it is greater than 100 or not and Display the message. If the number is odd, check whether it is divisible by 11 but by 7 and display the appropriate message

Algorithm:

- Start
- Declare variable number as store value given by user.
- Remainder when number / 2 equals to 0.
 - True :
 - print It is even number
 - check number > 100
 - True ;
 - print number is greater than 100.
 - False
 - print It is odd number.
 - remainder when number divided by 11 equals to 0
 - True :
 - Print number is divisible by 11
 - False :
 - remainder when number divided by 7 equals to 0
 - True :
 - number is divided by 7
 - False :
 - print Number is neither divisible by 11 not by 7
- Stop

Flow Chart:



Source Code:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main()
{
    unsigned int number;
    system("cls");
    printf("Enter any number: ");
    scanf("%u", &number);
    printf("\nNumber : %u", number);
    if (number % 2 == 0)
    {
        printf("\nNumber is even number.");
    }
    if (number > 100)
    {

```



```
printf("\nNumber is greater than 100.");
}
}
else
{
printf("\nNumber is odd number.");
if (number % 11 == 0)
{
printf("\nNumber is divisible by 11.");
}
else if (number % 7 == 0)
{
printf("\nNumber is divisible by 7.");
}
else
{
printf("\nNumber isn't divisible by 11 or 7.");
}
}
getch();
return 0;
}
```

Output:

```
Enter any number: 7

Number : 7
Number is odd number.
Number is divisible by 7.
```

```
Enter any number: 11

Number : 11
Number is odd number.
Number is divisible by 11.
```

```
Enter any number: 13

Number : 13
Number is odd number.
Number isn't divisible by 11 or 7.
```

```
Enter any number: 122

Number : 122
Number is even number.
Number is greater than 100.
```

6. Write a program to determine the root of quadratic equation. Take the value of a, b and c from user.

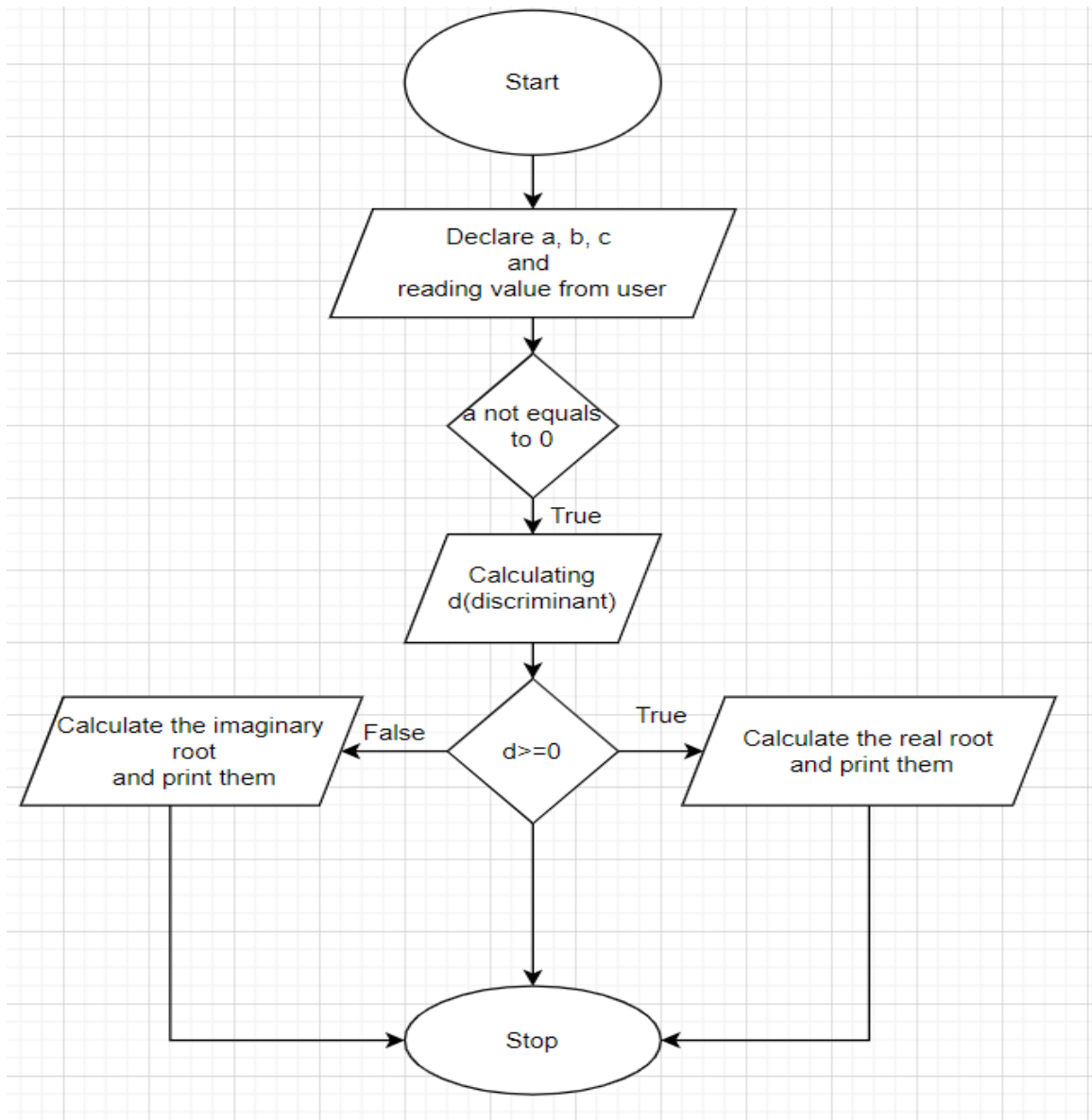
Algorithm:

- ❖ Start
- ❖ Declare a,b,c, and reading value of a,b,c
- ❖ Check a not equal to zero
 - True:
 - finding $d = \text{square root of } (b^2 - 4ab)$
 - Checking $d \geq 0$

- True:
 - ♦ calculating real root
- False:
 - ♦ Calculating imaginary root

❖ Stop

Flow Chart:



Source Code:

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

```

#include <conio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
float a, b, c, discriminant, root_one_real, root_two_real, root_imaginary;
system("cls");
printf("Enter the value of a,b,c:\n");
scanf("%f %f %f", &a, &b, &c);
if (a != (float)0)
{
discriminant = pow(b, 2) - 4 * a * c;
if (discriminant >= 0)
{
root_one_real = (-b + pow(discriminant, 0.5)) / (2 * a);
root_two_real = (-b - pow(discriminant, 0.5)) / (2 * a);
printf("\nRoot One : %.2f\nRoot Two : %.2f", root_one_real, root_two_real);
}
else if (discriminant < 0)
{
discriminant = -discriminant;
root_one_real = (-b) / (2 * a);
root_two_real = (-b) / (2 * a);
root_imaginary = (pow(discriminant, 0.5)) / (2 * a);
printf("\nRoot One : %.2f + %.2fi \nRoot Two : %.2f - %.2fi", root_one_real,
root_imaginary, root_two_real, root_imaginary);
}
}
}

```

```
else
{
printf("\nYou didnot give correct quadratic equation.");
}
return 0;
}
```

Output:

```
Enter the value of a,b,c:
1
2
1

Root One : -1.00
Root Two : -1.00

Enter the value of a,b,c:
1
2
9

Root One : -1.00 + 2.83i
Root Two : -1.00 - 2.83i
```

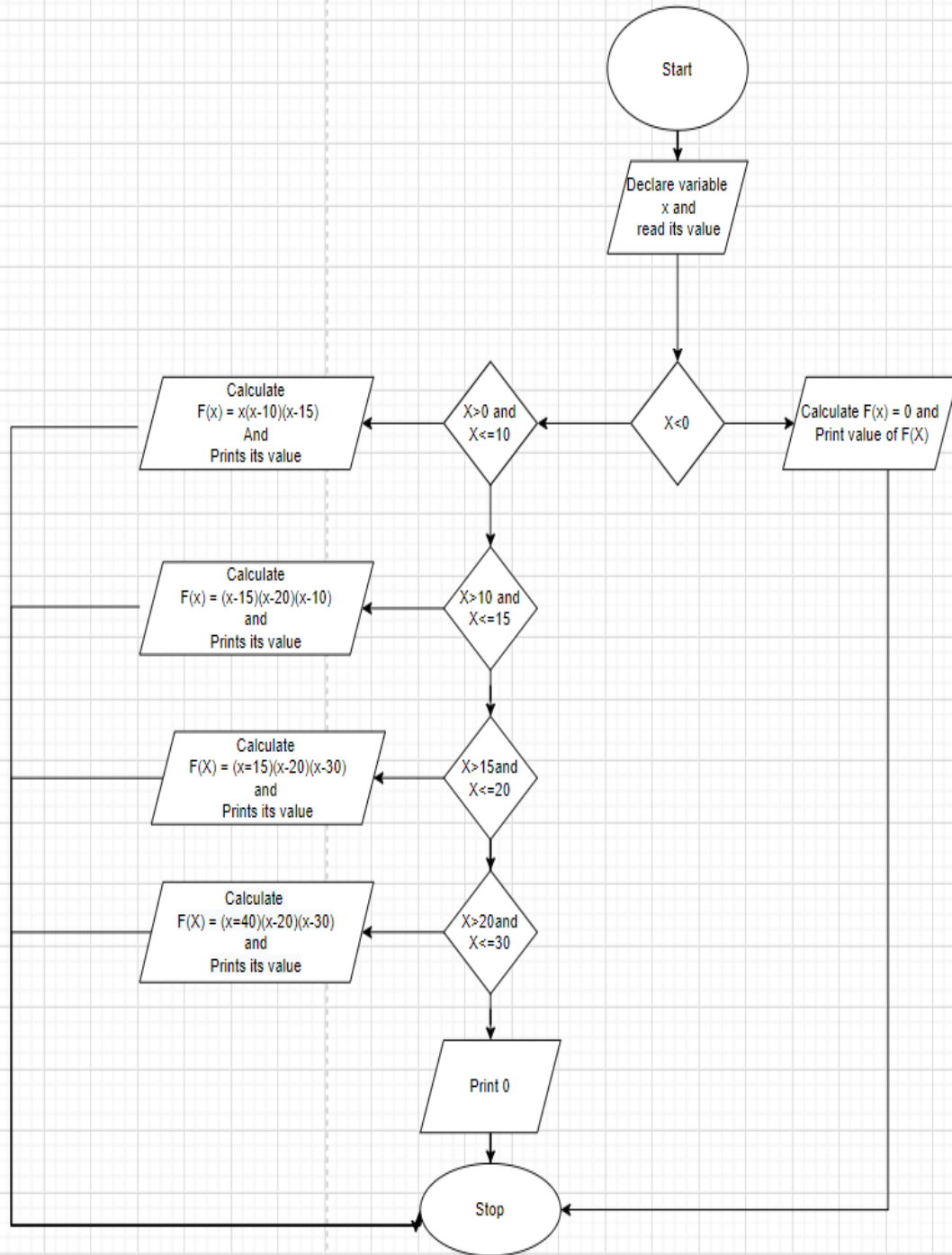
7. Write a program to evaluate the following function(F(x)) given by

Algorithm:

- Start
- Declare variable x and read its value
- $X < 0$
 - True:

- Calculate $f(x) = 0$ and prints $f(x)$
- False:
 - $X > 0$ and $X \leq 10$
 - True:
 - Calculate $f(x) = x(x-10)(x-15)$ and prints $f(x)$
 - False:
 - $X > 10$ and $X \leq 15$
 - True:
 - Calculate $f(x) = (x-20)(x-10)(x-15)$ and prints $f(x)$
 - False:
 - $X > 20$ and $X \leq 30$
 - True:
 - Calculate $f(x) = (x-20)(x-30)(x-40)$ and prints $f(x)$
 - False:
 - Print 0
 - Stop

Flow Chart:



Source Code:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()

```

```

{
float number, function_value;
system("cls");
printf("To calculate:\n \tF(x)\n\n");
printf("X = ");
scanf("%f", &number);
if (number <= 0)
{
function_value = 0;
}
else if (number > 0 && number <= 10)
{
function_value = number * (number - 10) * (number - 15);
}
else if (number > 10 && number <= 15)
{
function_value = (number - 10) * (number - 15) * (number - 20);
}
else if (number > 15 && number <= 20)
{
function_value = (number - 30) * (number - 15) * (number - 20);
}
else if (number > 20 && number <= 30)
{
function_value = (number - 30) * (number - 20) * (number - 40);
}
else
{

```



```
function_value = 0;
}
printf("\nThe value of function is %f.", function_value);
getch();
return 0;
}
```

Output:

```
To calculate:
           F(x)

X = -6

The value of function is 0.000000.
```

```
To calculate:
           F(x)

X = 2

The value of function is 208.000000.
```

```
To calculate:
           F(x)

X = 12

The value of function is 48.000000.
```

```
To calculate:
```

```
F(x)
```

```
X = 18
```

```
The value of function is 72.000000.
```

```
To calculate:
```

```
F(x)
```

```
X = 25
```

```
The value of function is 375.000000.
```

```
To calculate:
```

```
F(x)
```

```
X = 46
```

```
The value of function is 0.000000.
```

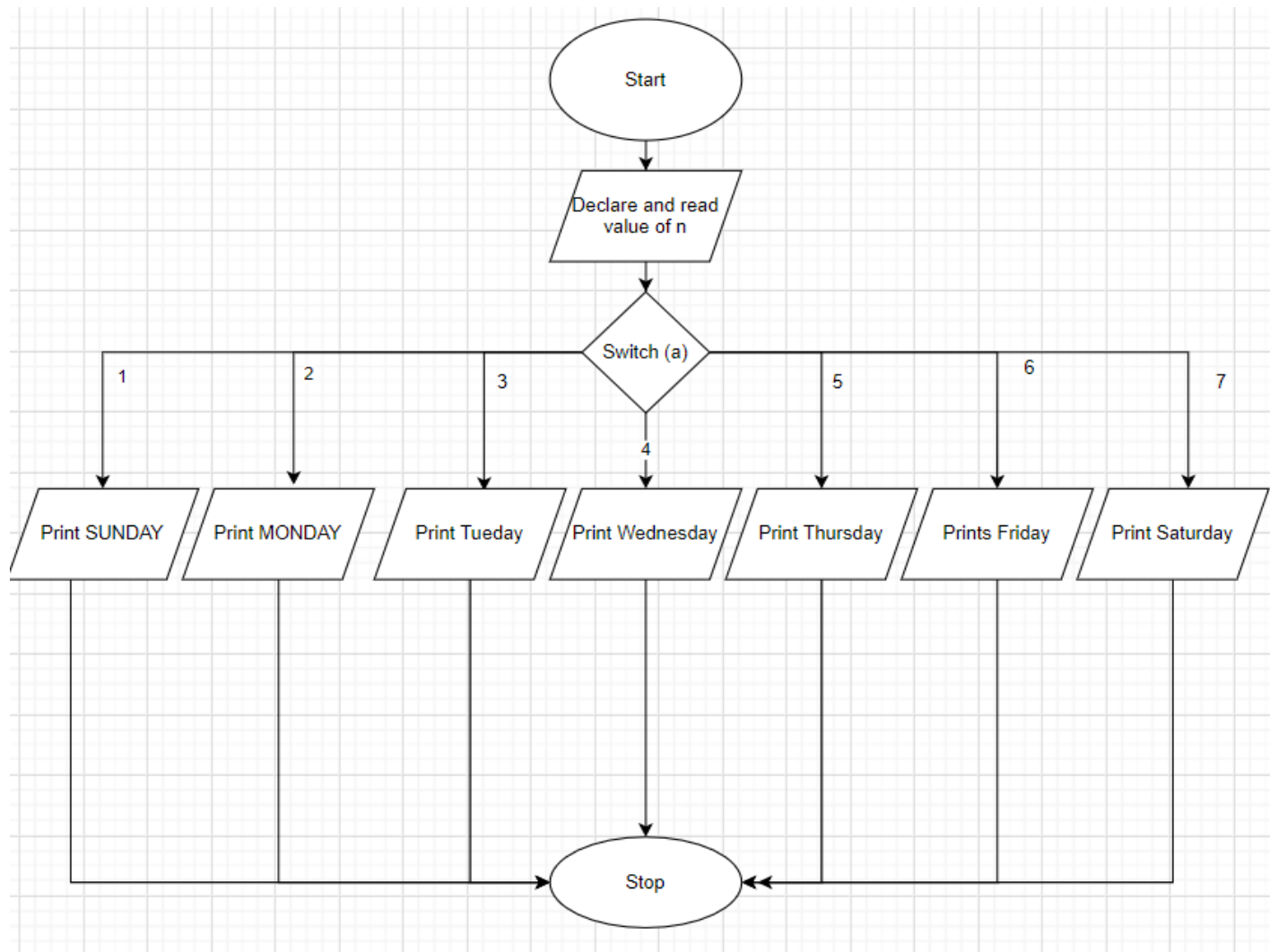
8. Write a program that prompts the user to input any integer from 1-7 and display the corresponding day in week.

Algorithm:

- Start
- Declare and read value of n
- Switch(n)
 - 1
 - Print SUNDAY
 - 2
 - Print MONDAY
 - 3
 - TUESDAY
 - 4

- WEDNESDAY
 - 5
 - THURSDAY
 - 6
 - FRIDAY
 - 7
 - SATURDAY
- Stop

Flow Chart:



Source Code:

```

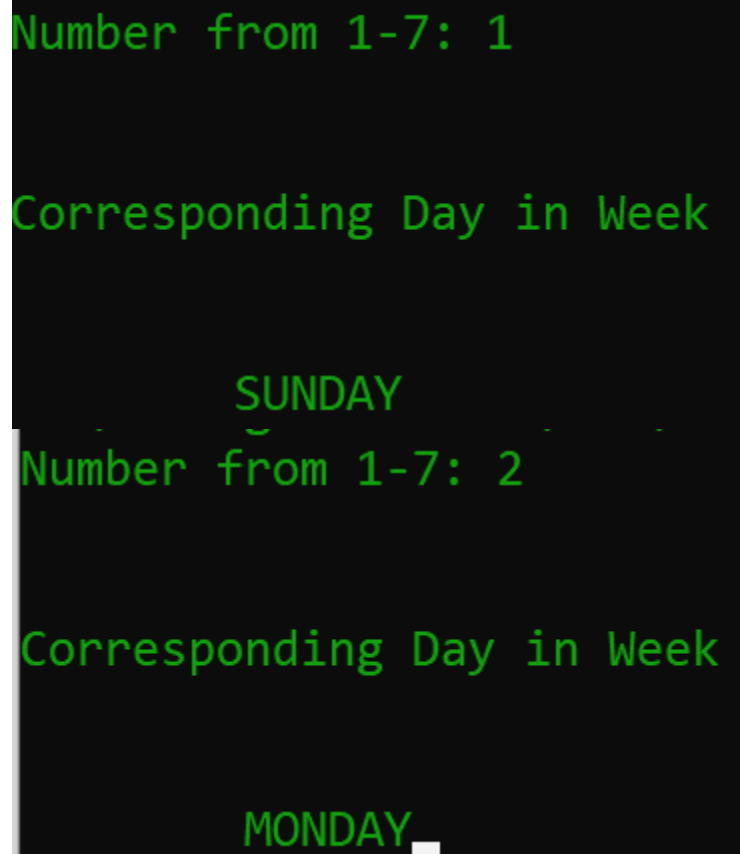
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

```

```
int main()
{
    int day_number;
    // system("cls");
    printf("Number from 1-7: ");
    scanf("%d", &day_number);
    printf("\n\nCorresponding Day in Week\n\n\n\t");
    switch (day_number)
    {
        case 1:
            printf("SUNDAY");
            break;
        case 2:
            printf("MONDAY");
            break;
        case 3:
            printf("TUESDAY");
            break;
        case 4:
            printf("WEDNESDAY");
            break;
        case 5:
            printf("THURSDAY");
            break;
        case 6:
            printf("FRIDAY");
            break;
        case 7:
```

```
printf("SATURDAY");  
break;  
default:  
printf("\t!!!ERROR!!!\n\n\tYou didn't give asked number.\n");  
break;  
}  
getch();  
return 0;  
}
```

Output:



```
Number from 1-7: 1  
  
Corresponding Day in Week  
  
SUNDAY  
Number from 1-7: 2  
  
Corresponding Day in Week  
  
MONDAY
```

Number from 1-7: 3

Corresponding Day in Week

TUESDAY

Number from 1-7: 4

Corresponding Day in Week

WEDNESDAY

Number from 1-7: 5

Corresponding Day in Week

THURSDAY

```
Number from 1-7: 6
```

```
Corresponding Day in Week
```

```
FRIDAY
```

```
Number from 1-7: 7
```

```
Corresponding Day in Week
```

```
SATURDAY
```

```
Number from 1-7: 78
```

```
Corresponding Day in Week
```

```
!!!ERROR!!!
```

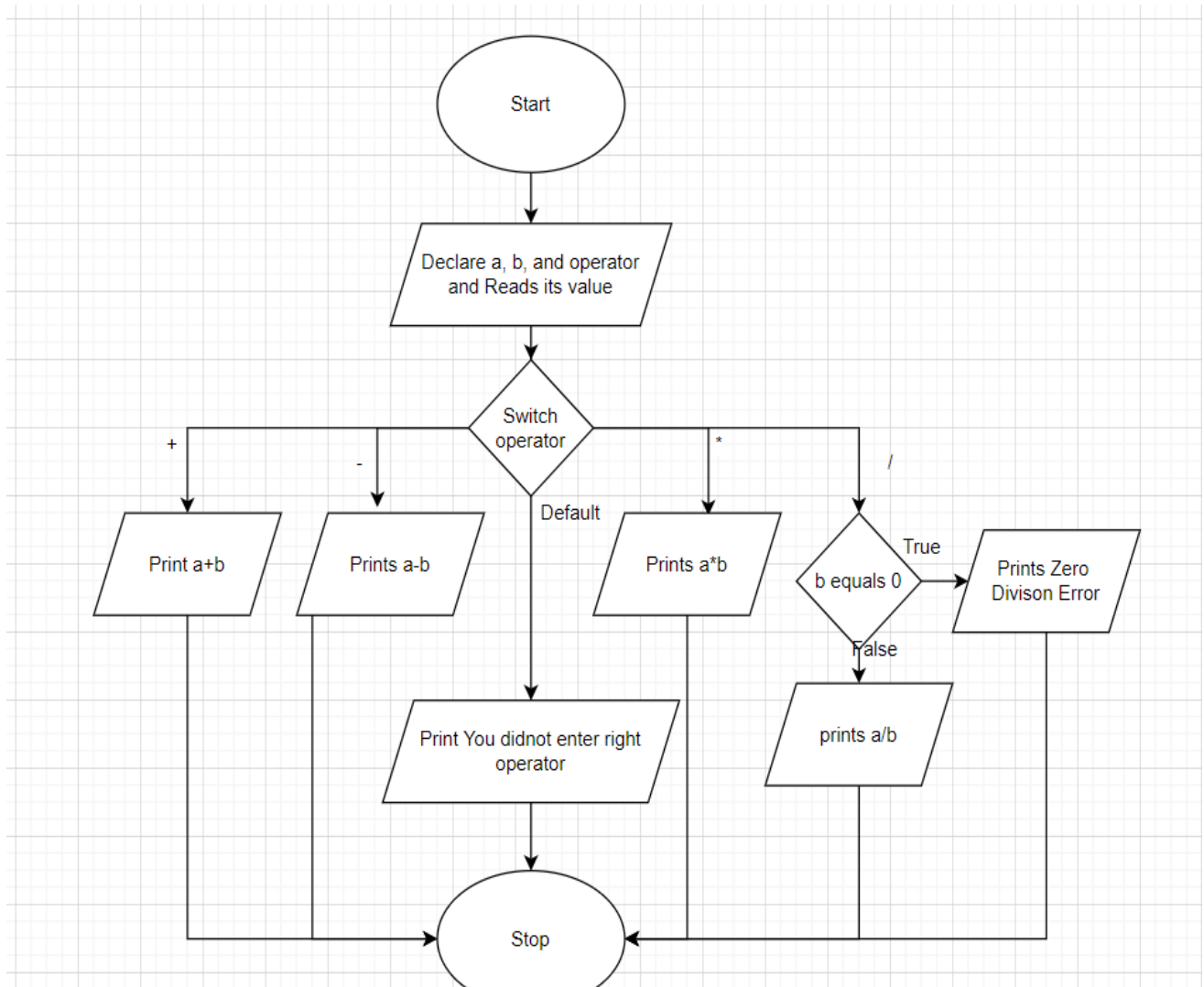
```
You didn't give asked number.
```

9. Write a program that ask the user a arithmetic operator and two operands and perform the corresponding operation

Algorithm:

- Start
- Declare a, b, and operator and read its value
- Switch(operator)
 - '+'
 - Print a+b
 - '-'
 - Print a-b
 - '*'
 - Print a*b
 - '/'
 - B equal to 0
 - True
 - Prints Zero division error
 - False:
 - Print a/b
 - Default
 - Print you didn't give correct operator
- Stop

Flow Chart:



Source Code:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    int operand_one, operand_two;
    char operator;
    float answer;
    system("cls");

```

```
printf("Operand one: ");
scanf("%d", &operand_one);
printf("Operand two: ");
scanf("%d", &operand_two);
printf("Operator: ");
scanf(" %c", &operator);
switch (operator)
{
case '+':
answer = operand_one + operand_two;
break;
case '-':
answer = operand_one - operand_two;
break;
case '*':
answer = operand_one * operand_two;
break;
case '/':
if (operand_two == 0)
{
printf("\nError! Zero Division error");
exit(0);
}
else
{
answer = (float)operand_one / operand_two;
break;
}
```

```
default:
printf("\nYou didn't give correct operator.");
exit(0);
}
printf("\nValue is %.2f.", answer);
getch();
return 0;
}
```

Output:

Operand one: 4 Operand two: 9 Operator: + Value is 13.00.	Operand one: 8 Operand two: 3 Operator: - Value is 5.00.
Operand one: 5 Operand two: 8 Operator: * Value is 40.00.	Operand one: 8 Operand two: 6 Operator: / Value is 1.33.
Operand one: 8 Operand two: 0 Operator: / Error! Zero Division error	

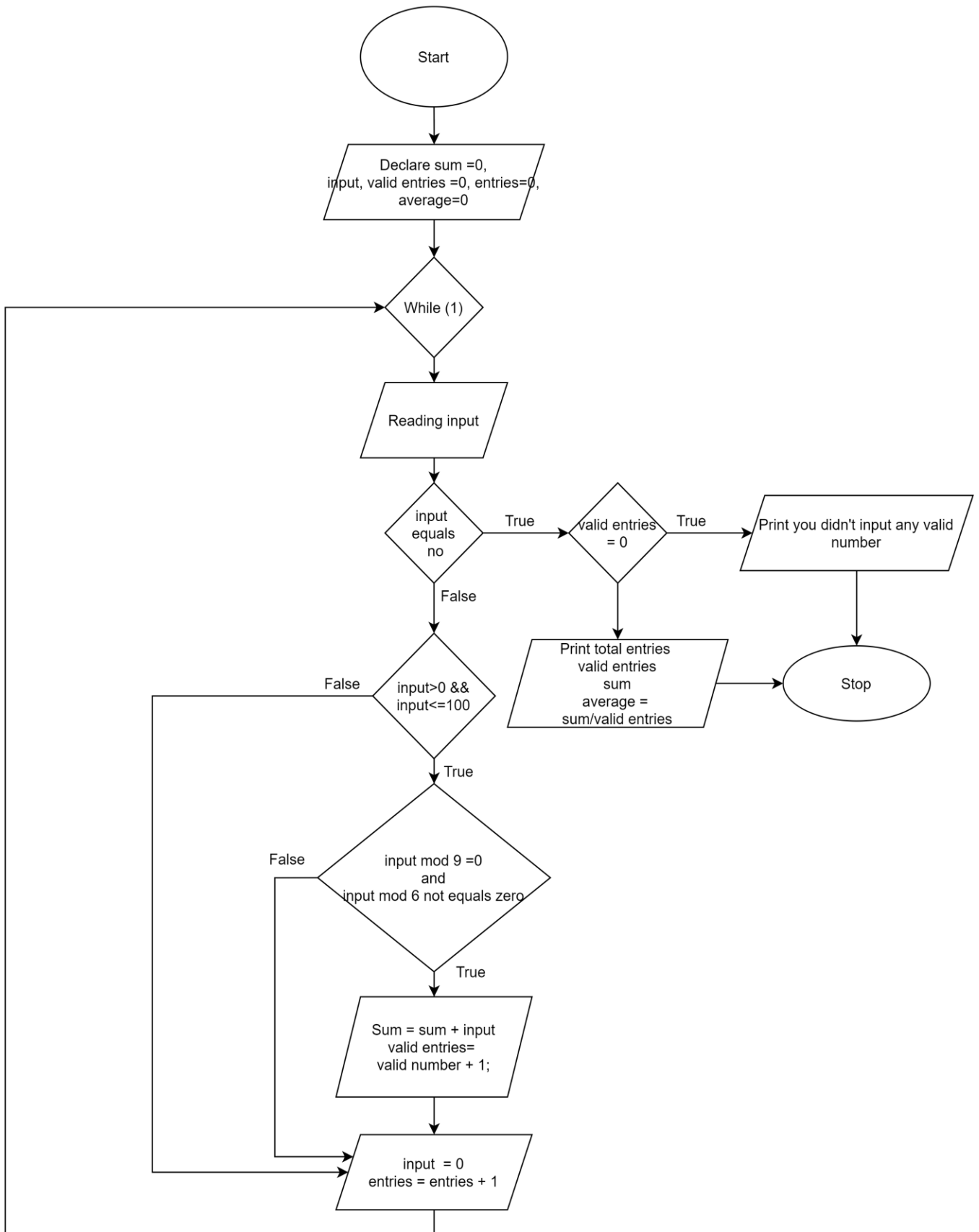
10."You are given a task to develop a system to read at least 100 integer numbers and continue until the user enters NO. Your system must have capacity to calculate the sum and average of those numbers which are exactly divisible by 9 but not by 6 and lies in between 1 to 100 and display a suitable message if no such number is read."

Write algorithm, flowchart and code to develop the system

Algorithm:

- Start
- Declare sum = 0, input, valid entries = 0, entries = 0, average = 0
- While(1)
 - Reads input
 - Input equals no
 - True
 - Valid equal zero
 - True
 - Print you did not enter any valid number.
 - Stop
 - False
 - Print Total entries, valid entries, sum, average =sum/valid entries
 - Stop
 - False
 - Input >0 and input <=100
 - True
 - Input mod 9 equals 0 and input mod 6 not equal 0
 - True
 - Sum = sum + input
 - Valid entries = valid entries +1
 - False:
 - Input = 0
 - Entries = entries +1

Flow Chart:



Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    int sum = 0, input_number = 0, no_of_valid_number = 0, valid_number[500], entries = 1;
    float average = 0;
    char input[100];
    system("cls");
    printf("\t\t <-----QUESTION 10 SYSTEM----->\n\n\n");
    printf("\t\t <-----TAKING INPUT----->\n\n\n");
    while (1)
    {
        printf("\nValue %d : ", entries);
        scanf("%s", input);
        if ((strcmp(input, "No") == 0) || (strcmp(input, "no") == 0) || (strcmp(input, "NO") == 0))
        {
            if (no_of_valid_number == 0)
            {
                system("cls");
                printf("\t\t <-----QUESTION 10 SYSTEM----->\n\n\n");
                printf("\t\t <-----SYSTEM ANALYSIS RESULT----->\n\n\n");
                printf("\t\t!!!!!!ERROR!!!!!!\n\n");
                printf(" You didn't enter the number with following characteristic:\n\n");
            }
        }
    }
}
```

```

printf("\t\t-->Divisible by 9\n");
printf("\t\t-->Not divisible by 6\n");
printf("\t\t-->Number should be between 1 to 100.\n\n");
printf("\n\n\t\t<---Thanks for using our sytem--->");
getch();
exit(0);
}
else
{
system("cls");
printf("\t\t<-----QUESTION 10 SYSTEM----->\n");
printf("\t\t<-----SYSTEM ANALYSIS RESULT----->\n\n\n");
printf("Total number of entries: %d\n", entries - 1);
printf("Valid Entries: %d\n\n", no_of_valid_number);
printf("\nValid Entries: ");
for (int i = 0; i < no_of_valid_number; i++)
{
printf(" %d", valid_number[i]);
}
average = (float)sum / no_of_valid_number;
printf("\n\nSum : %d", sum);
printf("\nAverage = %.2f", average);
printf("\n\n\n\t\t<---Thanks for using our sytem--->");
getch();
exit(0);
}
}
else

```

```
{  
for (int i = 0; i < strlen(input); i++)  
{  
    char c = input[i];  
    if (c == '1')  
    {  
        input_number = input_number * 10 + 1;  
    }  
    else if (c == '2')  
    {  
        input_number = input_number * 10 + 2;  
    }  
    else if (c == '3')  
    {  
        input_number = input_number * 10 + 3;  
    }  
    else if (c == '4')  
    {  
        input_number = input_number * 10 + 4;  
    }  
    else if (c == '5')  
    {  
        input_number = input_number * 10 + 5;  
    }  
    else if (c == '6')  
    {  
        input_number = input_number * 10 + 6;  
    }  
}
```



```
else if (c == '7')
{
input_number = input_number * 10 + 7;
}
else if (c == '8')
{
input_number = input_number * 10 + 8;
}
else if (c == '9')
{
input_number = input_number * 10 + 9;
}
else if (c == '0')
{
input_number = input_number * 10 + 0;
}
}
// printf("%d", input_number);
}
if (input_number > 0 && input_number <= 100)
{
if ((input_number % 9 == 0) && (input_number % 6 != 0))
{
sum = input_number + sum;
valid_number[no_of_valid_number] = input_number;
no_of_valid_number++;
}
}
```

```
input_number = 0;
```

```
entries++;
```

```
}
```

```
}
```

Output:

```
<-----QUESTION 10 SYSTEM----->
```

```
<-----TAKING INPUT----->
```

```
Value 1 : 45
```

```
Value 2 : 89
```

```
Value 3 : 15
```

```
Value 4 : 93
```

```
Value 5 : 4651
```

```
Value 6 : 9
```

```
Value 7 : 42
```

```
Value 8 : 72
```

```
Value 9 : 81
```

```
Value 10 : 56
```

```
Value 11 : 99
```

```
Value 12 : 6
```

```
Value 13 : no
```

```
<-----QUESTION 10 SYSTEM----->
<-----SYSTEM ANALYSIS RESULT----->

Total number of entries: 12
Valid Entries: 4

Valid Entries: 45 9 81 99

Sum : 234
Average = 58.50

<---Thanks for using our sytem--->_
```

For no any valid numbers,

```
<-----QUESTION 10 SYSTEM----->

<-----SYSTEM ANALYSIS RESULT----->

!!!!!!!ERROR!!!!!!!

You didn't enter the number with following characteristic:

-->Divisible by 9
-->Not divisible by 6
-->Number should be between 1 to 100.

<---Thanks for using our sytem--->_
```

Analysis

Here, we have learned above various ways to taking input from user like **search set**(In Lab 3 program 11 and 12) for string, **no of element to take** (Lab 3 Program 10), and many more. In similar ways, we learn how to format our output like **how many characters to display, how any number to display after decimal, indentation gap** in output, **fixing the width** when display of variables (Lab 3 Program 4 5 6 7 8).

Moreover, we learn how the **if else statement, if else if ladder, for loop, while loop, do... while loop** works.

Also, it helps to understand the workings of these statements and building logic on decision taking program.

Conclusion

In nut shell, we became acquaintance with Formatted **input/ output** and **Looping**. Also, we shape ourselves with these newly learnt statements.

