**LAB # 8**

**OBJECT**

***While Loop Structure and Do While Loop Structure***

**THEORY**

## While loop structure

‘While’ loop structure looks simpler than the ‘for’ loop. Although it uses the same element as ‘for’ loop structure but these elements are distributed throughout the program. It has the general form as:

**while ( Test expression ) statements;**

The value of the testing variable is checked at the top but all the variation like increment, decrement, comparison etc are taking place within the statements in body of the loop.

### Elements of ‘While’ loop

Certainly the expression in the parentheses is simpler than the three part expression of the ‘for’ loop. It dispenses with the initialization and increment expression retaining only the test expression. The initialization step is included in the variable declaration usually while the increment or decrement is present in the body of the loop.

### Usage of appropriate loop

One should use the most appropriate loop in the program. The ‘While’ loop shine in the condition where a loop may be terminated unexpectedly by condition developing within the loop. ‘While’ loop are more appropriate than ‘For’ loops when the condition that terminates the loop occur unexpectedly; for example, condition when the testing variable ( say a character variable ) is input from the keyboard.

**Example-1**

This program generates number from 0 to 9 and prints their sum.

#include “stdio.h”

void main (void)

{

int count = 0;

int total = 0 ;

while (count <10)

{

total = total + count ;

printf (“count = %d, total = %d\n”, count++, total);

}

}

**Output**

The result of this program will be as:

count = 0 , total = 0

count = 1 , total = 1

count = 2 , total = 3

count = 3 , total = 6

count = 4 , total = 10

count = 5 , total = 15

count = 6 , total = 21

count = 7 , total = 28

count = 8 , total = 36

count = 9 , total = 4

## Do while loop

The last of the three loop structure in C language is the ‘do while’ loop. This loop is very similar to the ‘while’ loop, the difference is that in the ‘do’ loop the test condition is evaluated after the loop is executed rather than before.

### General form

The general form of the ‘do while’ loop structure is as:

do

{

Statements;

}

while(expression);

If only one statement is repeated, the curly braces are not necessary. Most programmers include them, however, so that they can easily recognize that the ‘while’ that ends the ‘do’ loop, not the beginning o the ‘while’ loop.

## Do while operation

The ‘do’ loop execute the statement or statements while the expression is true. It stops when the expression becomes false. The ‘do’ loop is unique because it will always execute the code at least once, since the expression controlling the loop is tested at the bottom off the loop.

The operation of the loop is sort of an upside down version of the ‘while’ loop. The body of the loop is first executed then the test condition is checked therefore the body of the loop is always executed at least once.

### Use of do while loop

‘Do while’ loop is always utilized in conditions where the body of the loop is executed at least once. These are the situations when your program is waiting for some event to occur.

## Example-1

This program prints numbers and their running total.

#include “stdio.h”

void main(void)

{

int count = 0;

int total = 0;

do

{

total = total + count;

printf(“count = %d, total = %d \n”, count++,total);

}

while (count < 10);

}

**Output**

The result of the program will be as:

count = 0 , total = 0

count = 1 , total = 1

count = 2 , total = 3

count = 3 , total = 6

count = 4 , total = 10

count = 5 , total = 15

count = 6 , total = 21

count = 7 , total = 28

count = 8 , total = 36

count = 9 , total = 45

**Example-2**

This program is a game, which let you guess a letter.

#include “stdio.h”

void main (void)

{

char ch;

do

{

printf(“\n\nType in a digit from ‘a’ to ‘e’:”);

while ((ch=getche( ))!= ‘C’)

{

printf(“\nSorry,%c is incorrect.\n”, ch);

printf(“Try again”);

}

printf(“\nThat’s it!\n”);

printf(“\nPlay again? (Type ‘y’ or ‘n’):”);

}

while (getche( )== ‘y’);

printf(“\n Thanks for playing!”);

}

**Output**

The result of the program will be as:

Type in a digit from ‘a’ to ‘e’:

b

Sorry, b is incorrect.

Try again.

c

That’s it!

Play again? (Type ‘y’ or ‘n’)

Type in a digit from ‘a’ to ‘e’

d

Sorry, d is incorrect.

Try again.

c

That’s it

Play again? (Type ‘y’ or ‘n’)

Thanks for playing!

**TASKS TO BE PERFORMED**

1. Write a program that prints the first 50 even numbers by using while loop.

# PROGRAM

#include<stdio.h>

int loop();

void main(){

int n;

scanf("%d",&n);

loop(n);

getch();

}

int loop(int x){

while(x<=50){

if(x%2==0){

printf("\n%d",x);

}else{

}

x++;

}

}

1. Write a program that checks whether a number is prime or not by using if else structure and while loop.

# PROGRAM

#include<stdio.h>

int prime(int);

int main(){

int x;

printf("Enter any number: ");

scanf("%d",&x);

prime(x);

return 0;

}

int prime(int p){

int n, c=1;

for(n=2; n<p; n++){

if(p%n==0){

c++;

break;

}

}

if(c==1){

printf("Prime number");

}else{

printf("Not Prime number");

}

}

1. Write a program to calculate the factorial of any number using do while loop.

# PROGRAM

# #include <stdio.h>

# int fact();

# int main(){

# int x;

# printf("Enter any number to find factorial ");

# scanf("%d",&x);

# fact(x);

# }

# int fact(int n){

# long int c=1, fact=1;

# do{

# fact=fact\*c;

# printf("\nFactorial of number %d = %ld",c,fact);

# c++;

# }while(c<=n);

# getch();

# }