|  |  |
| --- | --- |
| Computer Programming with C- EMCS 514 | Submitted to  **Dr. Md. Zakir Hossain Associate Professor Dept. Of Computer Science and Engineering. Comilla University.**  Submitted By  Md. Jahidul Islam (EMCS0708054) |

**Batch: 7th**

**Repository link:** https://github.com/Jahid-Riaad/EMCS.git

* **Calling all of the below mentioned functions sequentially from this main program.**

void main()

{

getSumAvg();

getSquareRoot();

getMax();

getEvenNumbers();

dayOfWeek();

firstLastDigit();

checkPrime();

pyramidOfNumbers() ;

matrixSum ();

infiniteLoop();

}

1. **Writing a C program two enter five numbers and find their sum and average (Not using array here).**

#include<stdio.h>

void getSumAvg()

{

int a,b,c,d,e,sum,avg;

printf("\n Enter the 5 numbers to find their sum and average.");

//Accepting numbers from user....

printf("\n Enter the 1st number: ");

scanf("%d",&a);

printf("\n Enter the 2nd number: ");

scanf("%d",&b);

printf("\n Enter the 3rd number: ");

scanf("%d",&c);

printf("\n Enter the 4th number: ");

scanf("%d",&d);

printf("\n Enter the 5th number: ");

scanf("%d",&e);

//Calculating SUM of the numbers.

sum=a+b+c+d+e;

//Calculating Average of the numbers.

avg=(a+b+c+d+e)/5;

//Printing results

printf("\n The Sum is: %d",sum);

printf("\n The Average is: %d",avg);

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

}

1. **Writing a C program to enter any number and calculate its square root.**

#include <math.h>

#include <stdio.h>

int getSquareRoot() {

double number, squareRoot;

printf("\n Enter a number to find the square root of it: ");

scanf("%lf", &number);

// Computing the square root using built in function

squareRoot = sqrt(number);

printf("\n Square root of %.2lf = %.2lf", number, squareRoot);

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

return 0;

}

1. **Writing a C program to find maximum between three numbers (Not using array here).**

#include <stdio.h>

void getMax()

{

int a, b, c, max;

printf("\n Enter 3 number to get Maximum number from them.");

//Getting input of numbers

printf("\n Enter the 1st number: ");

scanf("%d",&a);

printf("\n Enter the 2nd number: ");

scanf("%d",&b);

printf("\n Enter the 3rd number: ");

scanf("%d",&c);

//Comparing inputs between them.

if (a>b && a>c)

max = a;

else if (b>a && b>c)

max = b;

else

max = c;

printf("\n The Maximum number is %d", max);

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

return 0;

}

1. **Writing a C program to find all even numbers from 1 to n.**

#include <stdio.h>

int getEvenNumbers()

{

int lowest, highest ;

printf("\n Even numbers between 2 numbers (inclusive):");

printf("\n Enter the lowest number of the range:");

scanf("%d", &lowest );

printf("\n Enter the highest number of the range:");

scanf("%d", &highest );

printf("\n highest number: %d", highest);

printf("\n Even numbers between %d and %d are as follows: ", lowest, highest);

//iterating through the range.

for ( int i = lowest; i <= highest; i++)

{

//Checking Even numbers and printing.

if(i%2 == 0)

{

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

}

}

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

return 0;

}

1. **Writing a C program to print day of week using switch case (Taking Input: any integer number, Giving Output: Day name).**

In the below program I have assumed "Saturday" as the first day of week.

#include <stdio.h>

int dayOfWeek()

{

int day;

/\* Getting input the day number from user\*/

printf("\n Enter day number(1-7): ");

scanf("%d", &day);

switch(day)

{

case 1:

printf("The day of the week is \"Saturday\"");

break;

case 2:

printf("The day of the week is \"Sunday\"");

break;

case 3:

printf("The day of the week is \"Monday\"");

break;

case 4:

printf("The day of the week is \"Tuesday\"");

break;

case 5:

printf("The day of the week is \"Wednesday\"");

break;

case 6:

printf("The day of the week is \"Thursday\"");

break;

case 7:

printf("The day of the week is \"Friday\"");

break;

default:

printf("Invalid input! Please enter day number between 1-7.");

dayOfWeek();

}

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

return 0;

}

1. **Writing a C program to find first and last digit of a number using loop.**

#include <stdio.h>

int firstLastDigit()

{

int number, sum=0, firstDigit, lastDigit;

printf("Please enter a positive number: ");

if (scanf("%d", &number))

{

if (number > 0)

{

// Finding the last digit of the number

lastDigit = number % 10;

//Finding the first digit by dividing number by 10 until number is greater then or equal 10.

while(number >= 10)

{

number = number / 10;

}

firstDigit = number;

printf("\nThe first digit is %d and the last digit is %d\n\n", firstDigit,lastDigit);

}

else

firstLastDigit();

}

else

printf("\nSorry ! Your input is invalid, please try again.");

return 0;

}

1. **Writing a C program to check whether a number is prime or not using a user defined function.**

**Prime Number:** We know prime numbers can't be divided by other numbers than itself or 1 and any negative number, 1 is not a prime number.

#include<stdio.h>

int checkPrime()

{

int number, median=0,flag=0;

printf("Enter the positive number greater than 1 to check Prime: ");

scanf("%d",&number);

/\*We know prime numbers can't be divided by other numbers than itself or 1 and any negative number, 1 is not a prime number.\*/

if (number > 1)

{

median=number/2;

for(int i=2; i<=median; i++)

{

//Checking prime number or not

if(number%i==0)

{

printf("This is not a Prime Number.");

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

flag=1;

break;

}

}

if(flag==0)

{

printf("This is a Prime Number.");

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

}

}

else

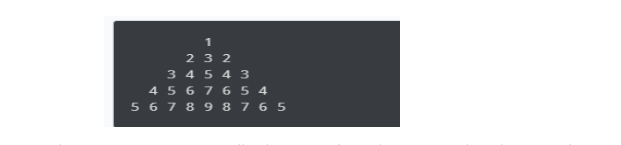
checkPrime();

return 0;

}

1. **Writing a C program to draw a pattern (Pyramid of Numbers).**

The output will be shown the same type as below.



#include <stdio.h>

int pyramidOfNumbers()

{

int indx, gap, rows, k = 0, count = 0, count1 = 0;

/\*Getting input for the number of row to be printed in pyramid.\*/

printf("Enter the number of rows for the \"Pyramid of Numbers\" : ");

scanf("%d", &rows);

/\*iterating through the number of rows until index is less than or equal of it.\*/

for (indx = 1; indx <= rows; ++indx)

{

/\*Making gaps to be printed though iteration, in first loop it will print 4 gaps fro the left side\*/

for (gap = 1; gap <= rows - indx; ++gap)

{

//Printing Gaps

printf(" ");

++count;

}

/\*measuring the middle point and iterating till k != 2 \* indx – 1.\*/

while (k != 2 \* indx - 1)

{

if (count <= rows - 1)

{

//printing numbers if

printf("%d ", indx + k);

++count;

}

else

{

//printing less numbers...

++count1;

printf("%d ", (indx + k - 2 \* count1));

}

//reserving flag to be continued...

++k;

}

count1 = count = k = 0;

printf("\n");

}

return 0;

}

1. **Writing a C program to find sum of each row and column of a matrix.**

/\* This program is written to accept a matrix of order rows x columns and find the sum of each row and each column of a matrix \*/

#include <stdio.h>

void matrixSum ()

{

static int array[10][10];

int indx1, indx2, rows, columns, sum = 0;

printf("Enter the order of the matrix\n");

scanf("%d %d", &rows, &columns);

printf("Enter the co-efficients of the matrix\n");

for (indx1 = 0; indx1 < rows; ++indx1)

{

for (indx2 = 0; indx2 < columns; ++indx2)

{

scanf("%d", &array[indx1][indx2]);

}

}

for (indx1 = 0; indx1 < rows; ++indx1)

{

for (indx2 = 0; indx2 < columns; ++indx2)

{

sum = sum + array[indx1][indx2] ;

}

printf("Sum of the %d row is = %d\n", indx1 + 1, sum);

sum = 0;

}

sum = 0;

for (indx2 = 0; indx2 < columns; ++indx2)

{

for (indx1 = 0; indx1 < rows; ++indx1)

{

sum = sum + array[indx1][indx2];

}

printf("Sum of the %d column is = %d\n", indx2 + 1, sum);

sum = 0;

}

}

1. **Infinite loop, explaining with examples. Writing some scenarios where we use infinite loop.**

**Infinite loop**: An infinite loop is a looping construct that does not terminate the loop and executes the loop forever. It is also called an indefinite loop or an endless loop. It either produces a continuous output or no output.

**Explanation with example:** We can create an infinite loop through various loop structures. The following are the loop structures through which we will define the infinite loop:

for loop

while loop

do-while loop

go to statement

C macros

For loop

Let's see the infinite 'for' loop. The following is the definition for the infinite for loop:

for(; ;)

{

// body of the for loop.

}

As we know that all the parts of the 'for' loop are optional, and in the above for loop, we have not mentioned any condition; so, this loop will execute infinite times.

Let's understand through an example.

#include <stdio.h>

int main()

{

for(;;)

{

printf("Hello, This is an infinite loop.");

}

return 0;

}

**Scenarios**: An infinite loop is useful for those applications that accept the user input and generate the output continuously until the user exits from the application manually.

In the following situations, this type of loop can be used:

1. All the operating systems run in an infinite loop as it does not exist after performing some task. It comes out of an infinite loop only when the user manually shuts down the system.
2. All the servers run in an infinite loop as the server responds to all the client requests. It comes out of an indefinite loop only when the administrator shuts down the server manually.
3. All the games also run in an infinite loop. The game will accept the user requests until the user exits from the game.