Postmortem of “C”

Author:
Talha Zubaer
Dept. of CSE.
Daffodil International University

C programming:

C is a general purpose , structured programming language. Its instruction consist of terms that resemble Algebric expressions.Argumented by certain English keywords such as, if ,else, do ,do while etc.
Anybody can start learning programming with any programming language, but C is always a better language to start with.

Some Information -

----- C is developed originally in 1970's.
----- Was developed by Denis Ritchie.
----- And it was developed at Bell Telephone Lab(Now part of the AT&T).

Structure of a C program
Every c program contains one or more modules called functions. One of the functions must be called "main". The program will always begin by executing the main function.

Must Haves:

A function called "heading" which consists of the function name, followed by the optional list of arguments, enclosed in parentheses.
 A list of argument declarations.
A compound statement which compiles the remainder of the function.

Compilers:

*A **compiler** is a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code.*

A program that translates from a low level language to a higher level one is a decompiler. A program that translates between high-level languages is usually called a source-to-source compiler or transpiler. A language rewriter is usually a program that translates the form of expressions without a change of language. The term compiler-compiler is sometimes used to refer to a parser generator, a tool often used to help create the lexer and parser.
A compiler is likely to perform many or all of the following operations: lexical analysis, preprocessing, parsing, semantic analysis (syntax-directed translation), code generation, and

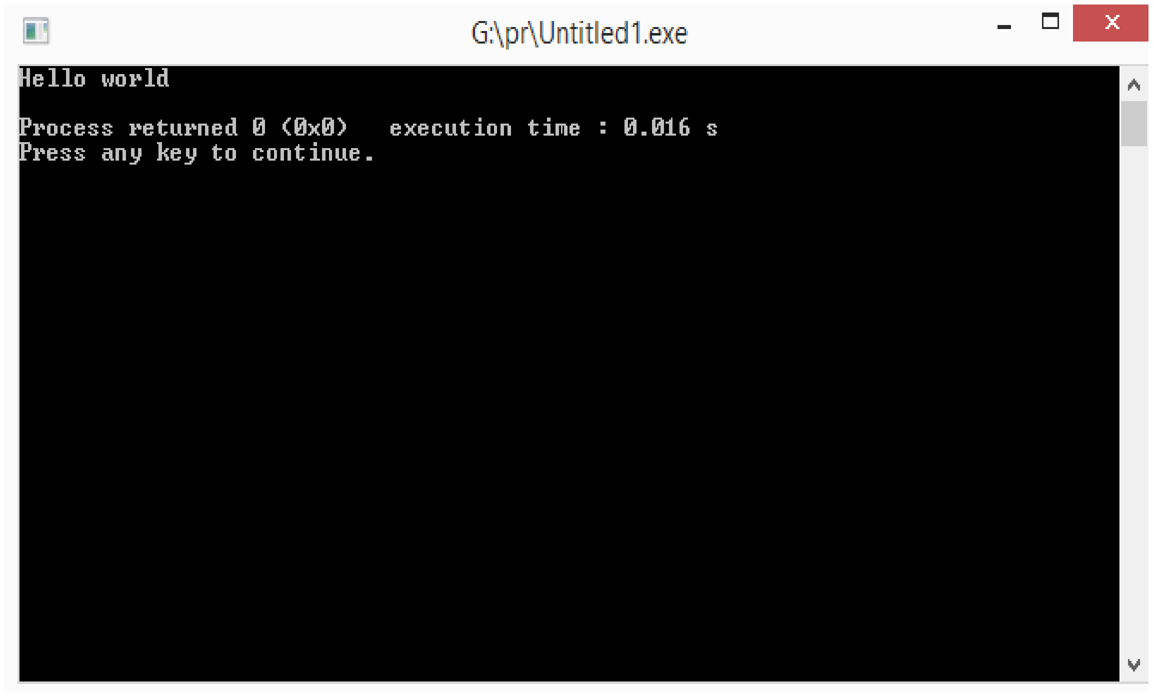
code optimization. Program faults caused by incorrect compiler behavior can be very difficult to track down and work around.

1. First Program:

```
#include <stdio.h>

int main()

{

  printf("Hello world\n");

  return 0;

}
```

Input: Hello World

Output:



```
Hello world

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

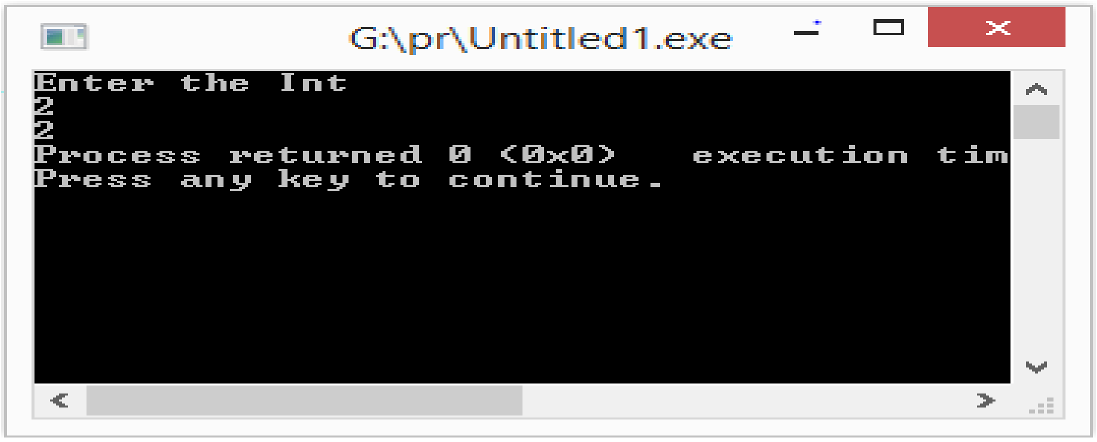
2-Printing An Integer

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

```
int main()

{

   int a;

   printf("Enter the Int\n");

   scanf("%d",&a);

   printf("%d",a);

   return 0;

}
```

Input: 2

Output:



G:\pr\Untitled1.exe

```
Enter the Int
2
2
Process returned 0 (0x0)   execution tim
Press any key to continue.
```

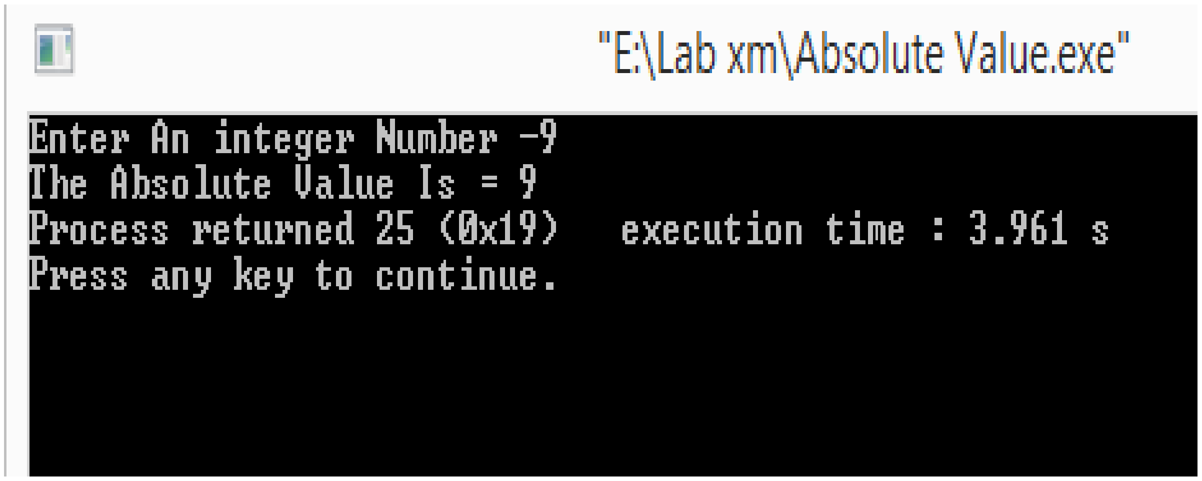
3- Absolute Value.

```
#include<stdio.h>

#include<math.h>

void main()

{

   int c,n;

   printf("Enter An integer Number ");
```

```
    scanf("%d", &n);

    c=abs(n);

    printf("The Absolute Value Is = ""%d",c);

}
```

Input: -9

Output:

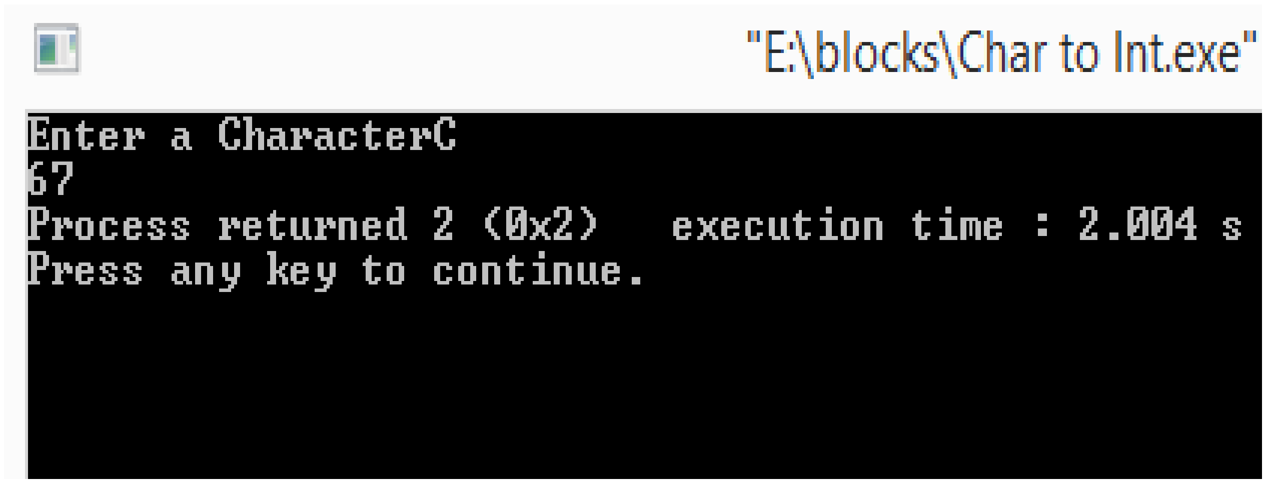


"E:\Lab xm\Absolute Value.exe"

```
Enter An integer Number -9
The Absolute Value Is = 9
Process returned 25 (0x19)   execution time : 3.961 s
Press any key to continue.
```

4-Character to Integer

```
#include <stdio.h>

  void main()

  {

    int n;

    printf("Enter a Character");

    scanf("%c", & n);

    printf("%d", n);

  }
```



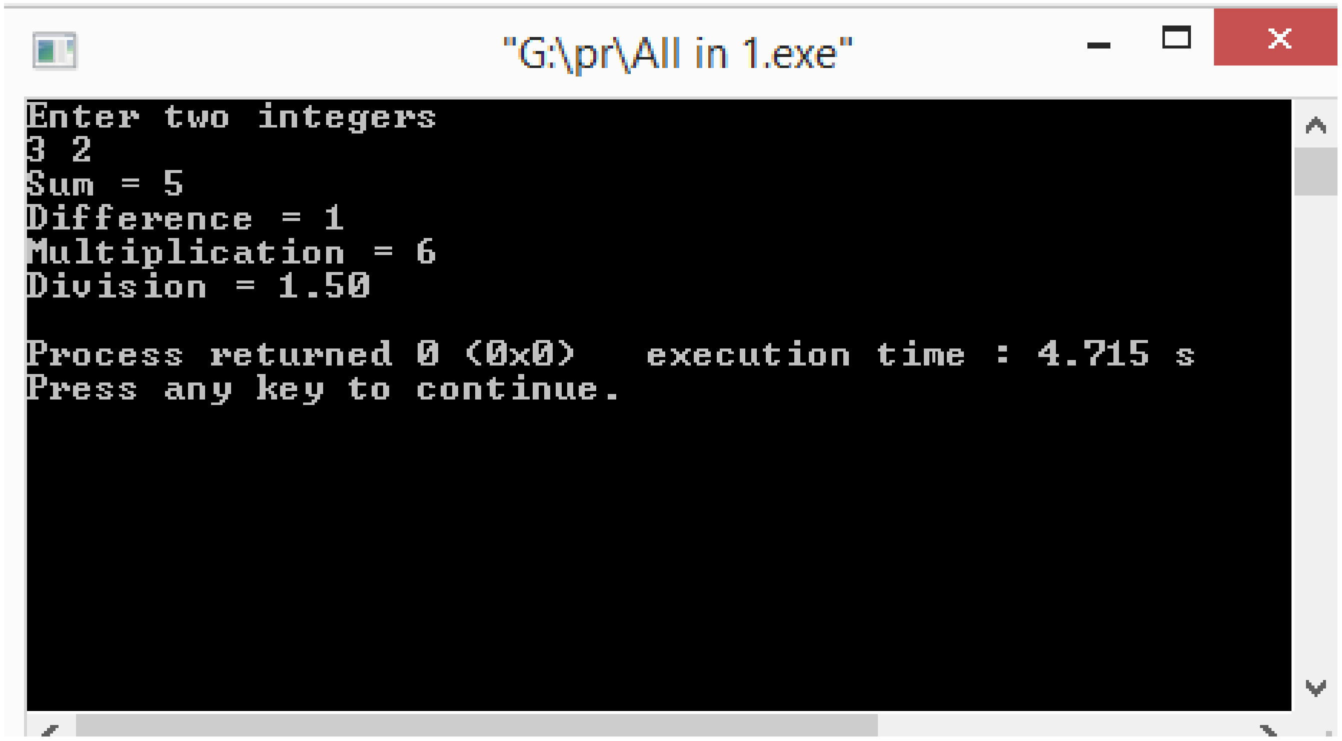
```
Enter a CharacterC
67
Process returned 2 (0x2)   execution time : 2.004 s
Press any key to continue.
```

5-Mathematical Op:

```
#include <stdio.h>

int main()

{

  int first, second, add, subtract, multiply;

  float divide;


  printf("Enter two integers\n");

  scanf("%d%d", &first, &second);


  add      = first + second;

  subtract = first - second;

  multiply = first * second;

  divide   = first / (float)second;   //typecasting
  printf("Sum = %d\n",add);

  printf("Difference = %d\n",subtract);

  printf("Multiplication = %d\n",multiply);

  printf("Division = %.2f\n",divide);


  return 0;

}
```

Input: 3 2

Output:


"G:\pr\All in 1.exe"

```
Enter two integers
3 2
Sum = 5
Difference = 1
Multiplication = 6
Division = 1.50

Process returned 0 (0x0)   execution time : 4.715 s
Press any key to continue.
```

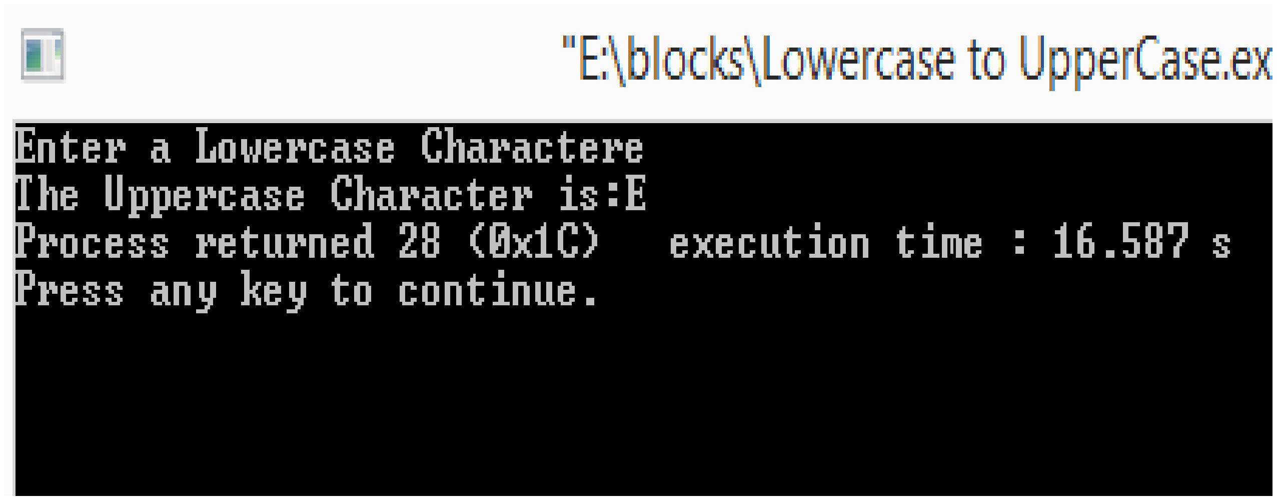
5—

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

```
void main()

{

   char c;

   printf("Enter a Lowercase Character");

   scanf("%c", &c);

   c=c-32;

   printf("The Uppercase Character is:-""%c", c);

}
```

Input: E

Output:

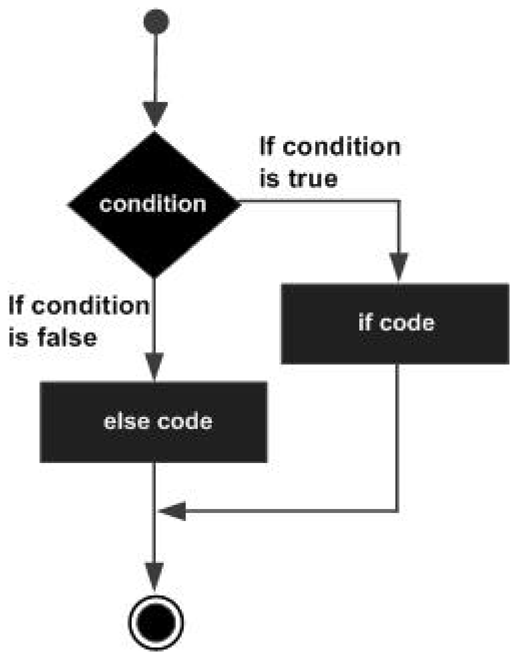


"E:\blocks\Lowercase to UpperCase.ex

```
Enter a Lowercase Charactere
The Uppercase Character is:E
Process returned 28 (0x1C)   execution time : 16.587 s
Press any key to continue.
```

If-Else Statement:

We can understand the whole procedure by this following flow program.

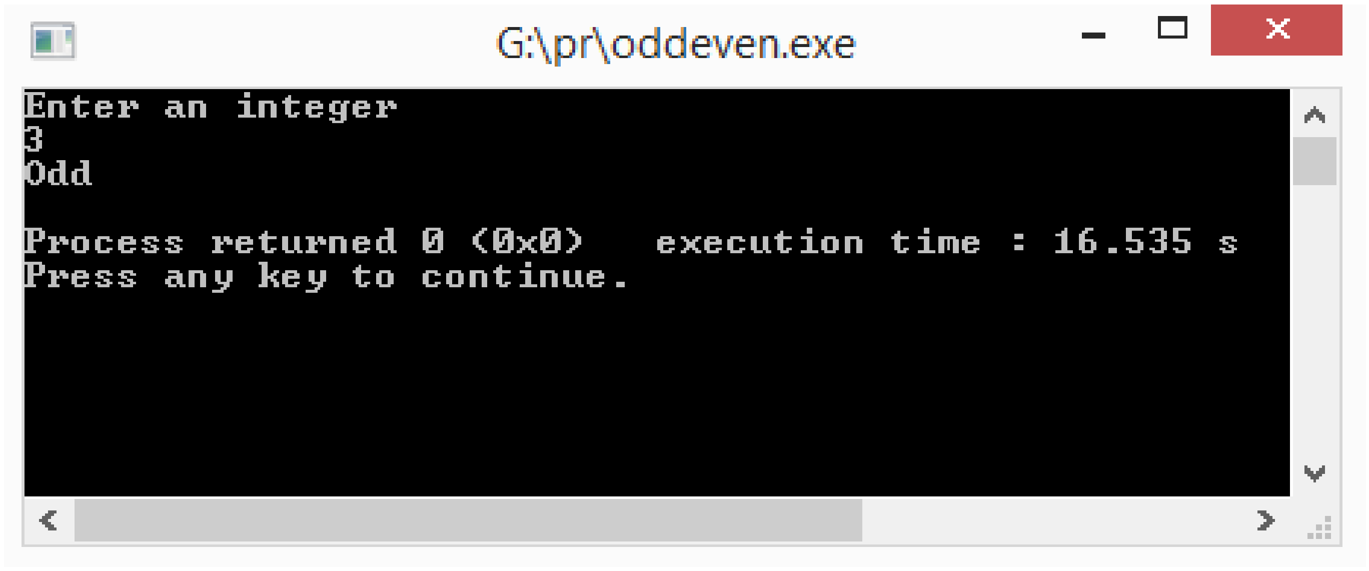


6- ODD or EVEN ?

```
#include<stdio.h>


main()

{

  int n;


  printf("Enter an integer\n");

  scanf("%d",&n);


  if ( n%2 == 0 )

    printf("Even\n");

  else

    printf("Odd\n");
  return 0;
}
```

Input: 3

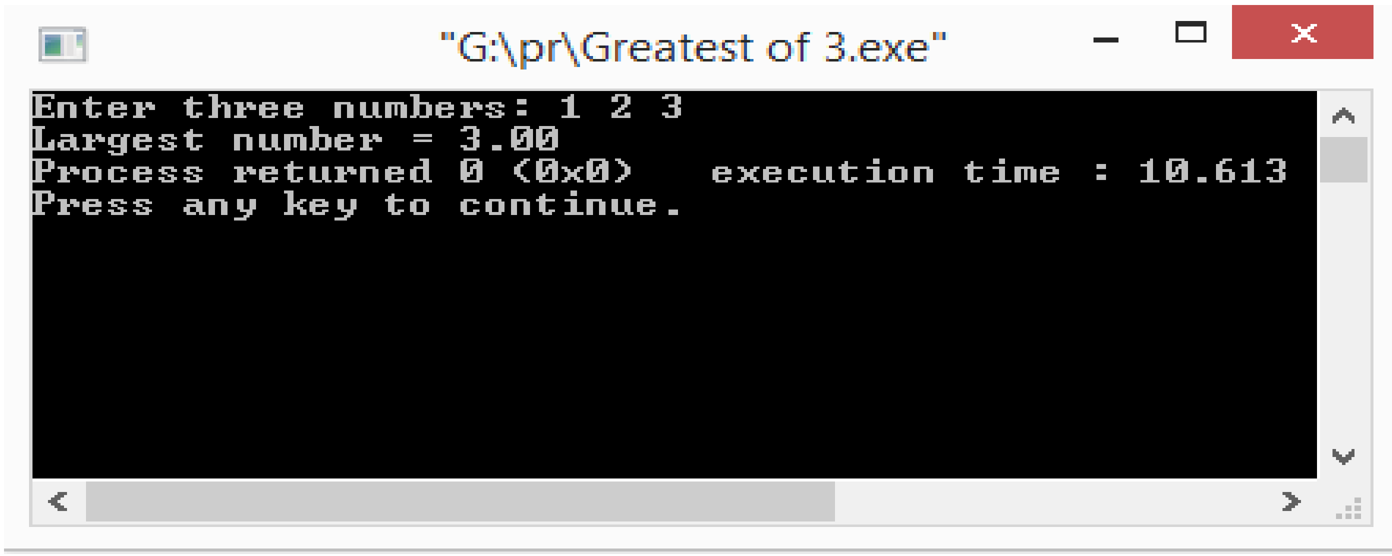
Output:



```
Enter an integer
3
Odd

Process returned 0 (0x0)   execution time : 16.535 s
Press any key to continue.
```

7- Finding The Greatest amongst 3 Numbers.

```
#include <stdio.h>
int main(){
    float a, b, c;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &a, &b, &c);
    if (a>=b)
    {
        if(a>=c)
          printf("Largest number = %.2f",a);
        else
        printf("Largest number = %.2f",c);
    }
    else
    {
      if(b>=c)
        printf("Largest number = %.2f",b);
      else
        printf("Largest number = %.2f",c);
    }
    return 0;
}
```



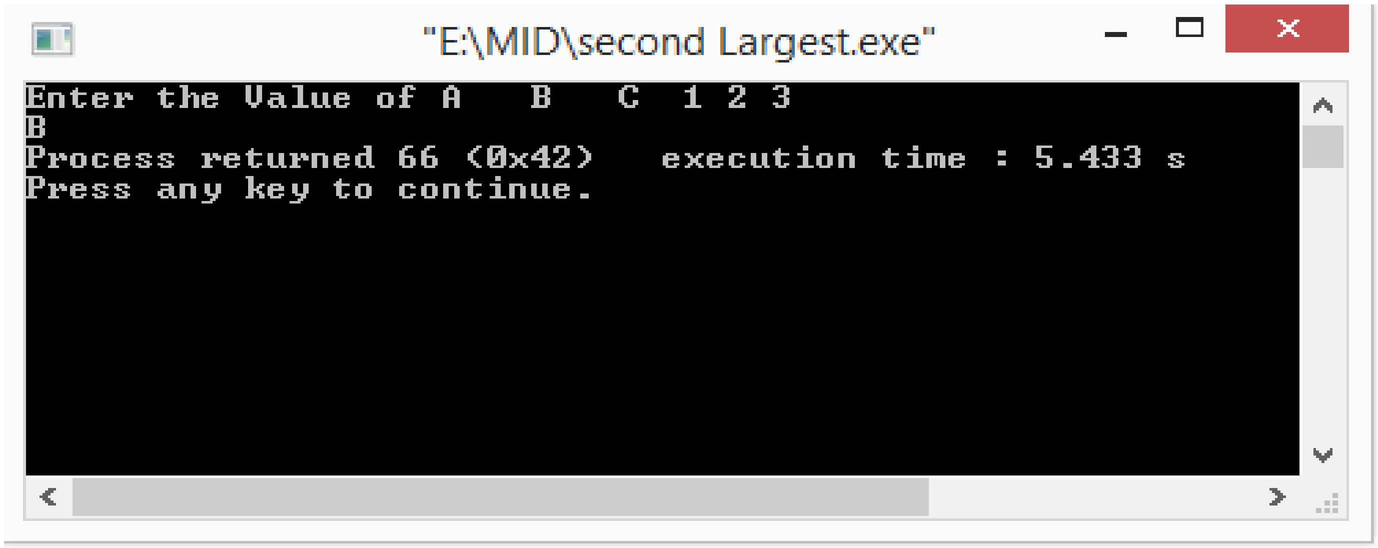
```
Enter three numbers: 1 2 3
Largest number = 3.00
Process returned 0 (0x0)   execution time : 10.613
Press any key to continue.
```

8- Finding the second Largest Number ,

```
 #include<stdio.h>
void main()
{
  int a,b,c;
```

```
printf("Enter the Value of A   B   C  ");
scanf("%d%d%d", &a,&b,&c);

if(a>b)
{
   if(a>c)
   {
      if(b>c)
         printf("B");
      else
         printf("C");
   }
   else
      printf("A");
 }

else
 {
    if(b>c)
    {
      if(a>c)
         printf("A");
      else
```

```
            printf("C");

        }

        else

          printf("B");



        }

}
```

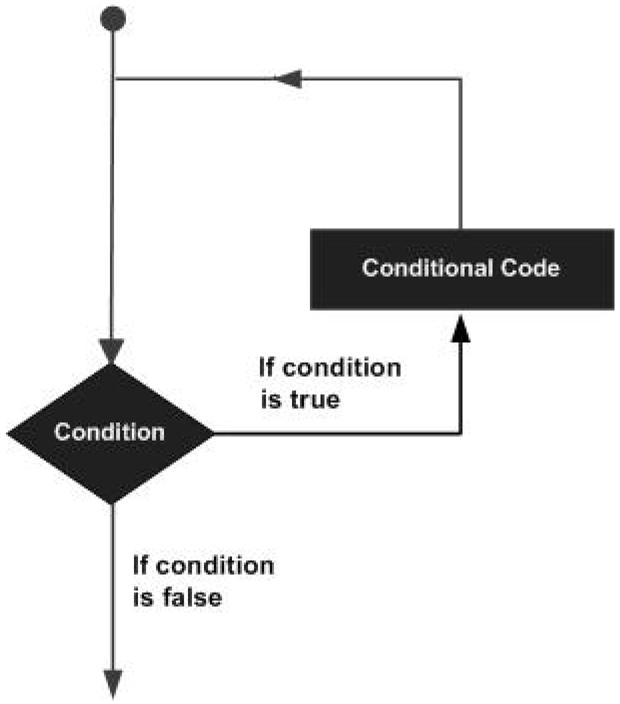


```
Enter the Value of A   B   C  1 2 3
B
Process returned 66 (0x42)   execution time : 5.433 s
Press any key to continue.
```

Loops:

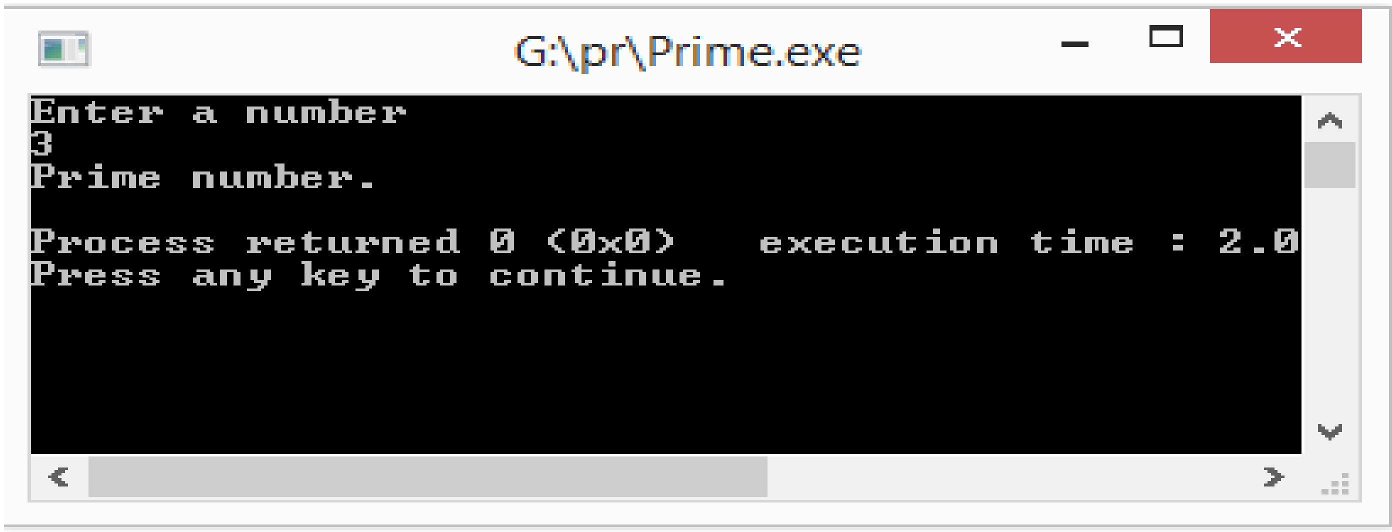
In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times. Given below is the general form of a loop statement in most of the programming languages –

The flow program for loops:



9- Finding the Prime Number

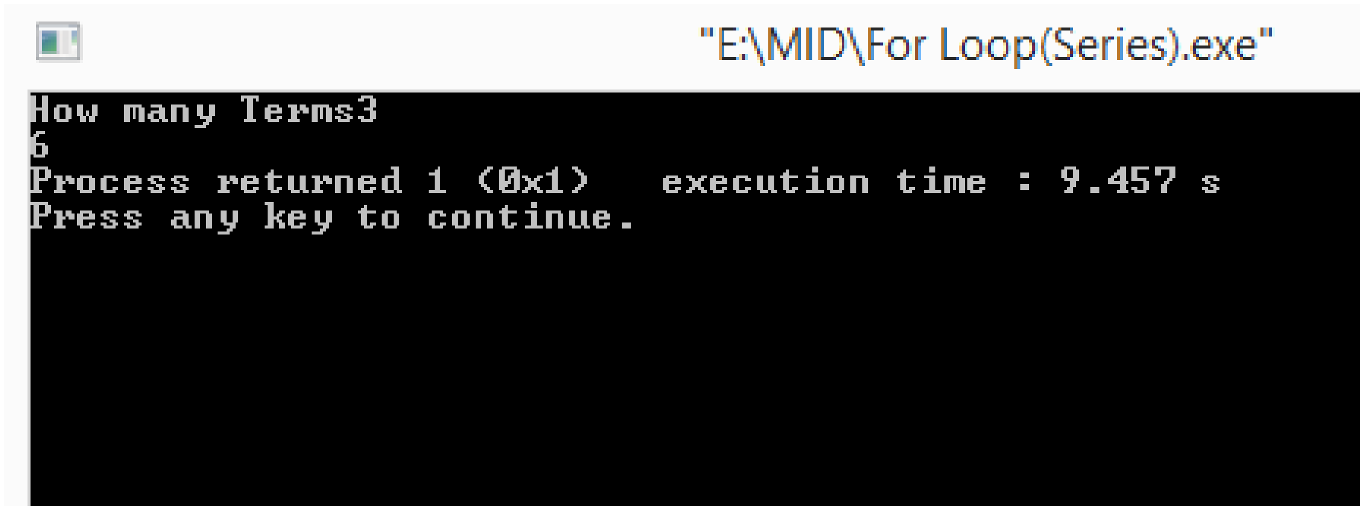
```
#include <stdio.h>

main()
{
   int n, c;

   printf("Enter a number\n");
   scanf("%d", &n);

   if ( n == 2 )
      printf("Prime number.\n");
   else
   {
       for ( c = 2 ; c <= n - 1 ; c++ )
       {
           if ( n % c == 0 )
              break;
       }
       if ( c != n )
          printf("Not prime.\n");
       else
          printf("Prime number.\n");
   }
   return 0;
}
```



```
Enter a number
3
Prime number.

Process returned 0 (0x0)   execution time : 2.0
Press any key to continue.
```

10- series: 1+2+3+…….+n

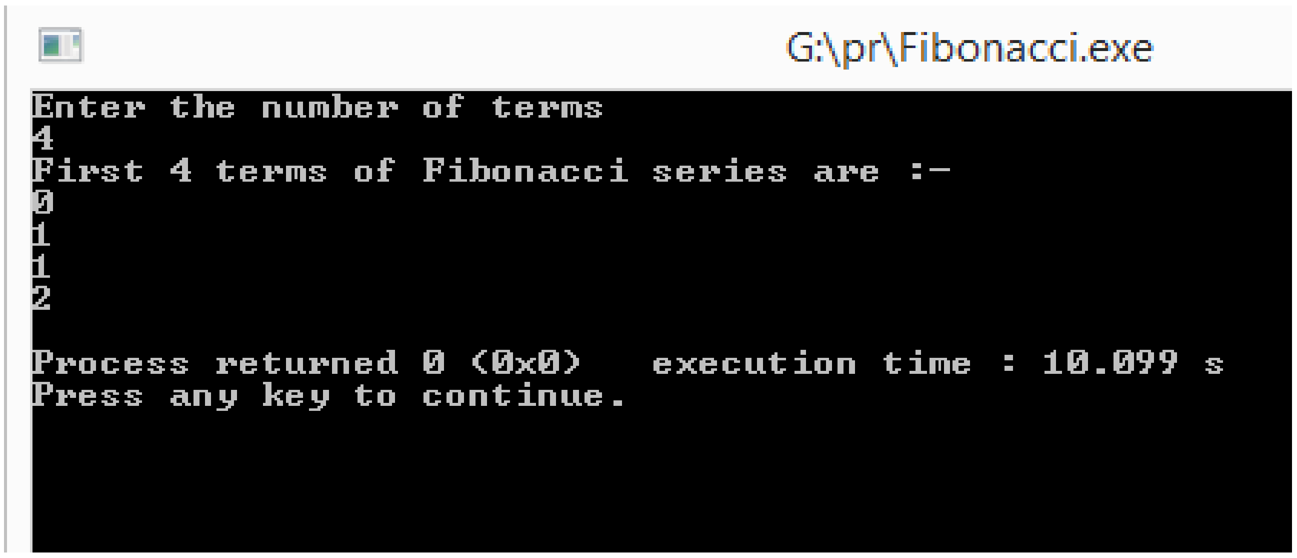
```
#include<stdio.h>

void main()

{

   int a,i,sum=0;



   printf("How many Terms");

   scanf("%d", &a);

   for(i=1; i<=a; i++)

      sum=sum+i;

   printf("%d", sum);



}
```



```
How many Terms3
6
Process returned 1 (0x1)   execution time : 9.457 s
Press any key to continue.
```

11- Fibonacci Series

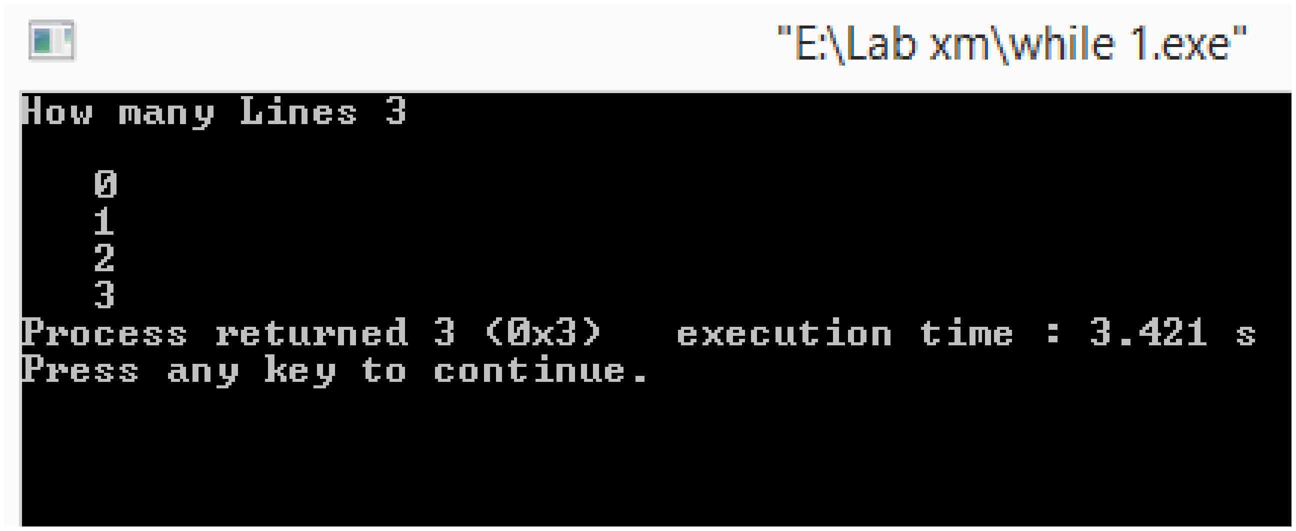
```
#include <stdio.h>

main()
{
   int n, first = 0, second = 1, next, c;

   printf("Enter the number of terms\n");
   scanf("%d",&n);

   printf("First %d terms of Fibonacci series are :-\n",n);

   for ( c = 0 ; c < n ; c++ )
   {
      if ( c <= 1 )
         next = c;
      else
      {
         next = first + second;
         first = second;
         second = next;
      }
      printf("%d\n",next);
   }

   return 0;
}
```



```
Enter the number of terms
4
First 4 terms of Fibonacci series are :-
0
1
1
2

Process returned 0 (0x0)   execution time : 10.099 s
Press any key to continue.
```

12-— Printing Numbers by While Loop.

```
 #include<stdio.h>

void main()

{

   int i,n;

   printf("How many Lines ");

   scanf("%d", &n);

   i=0;

   while (i<=n)

   { printf("\n %3d",i);

   i++;


   }
```


"E:\Lab xm\while 1.exe"

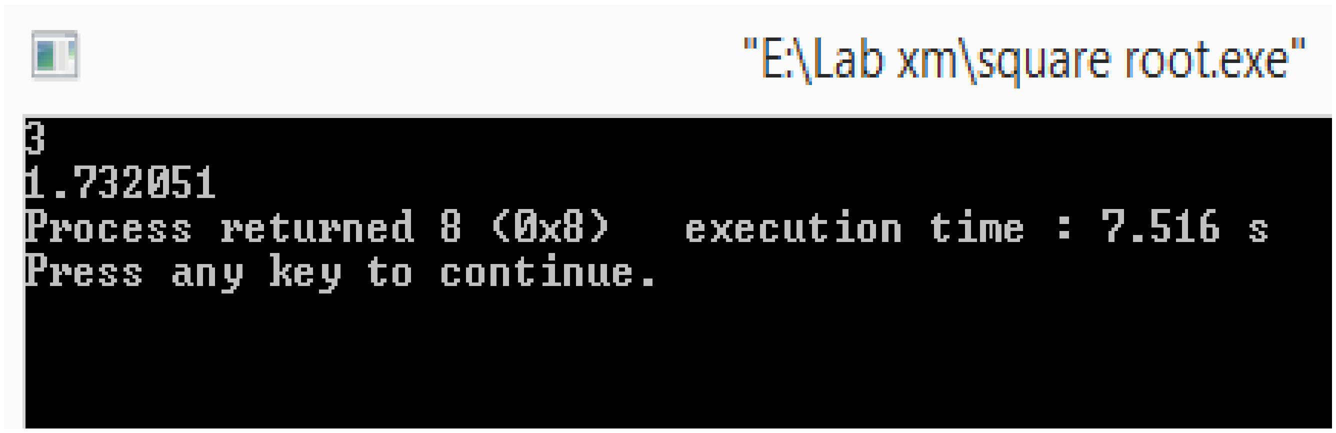
```
How many Lines 3

   0
   1
   2
   3
Process returned 3 (0x3)   execution time : 3.421 s
Press any key to continue.
```

13- Square Root Value finding

```
#include<stdio.h>

#include<math.h>

void main()

{

   float m,n;

   scanf("%f", &n);

   m=sqrt(n);

   printf("%f",m);

}
```

Input: 3

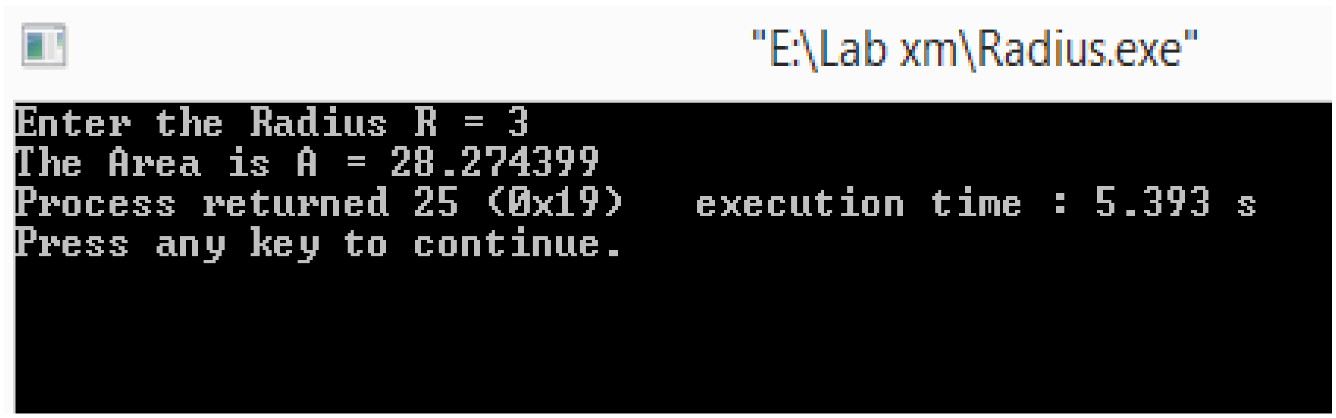
Output:



```
3
1.732051
Process returned 8 (0x8)   execution time : 7.516 s
Press any key to continue.
```

14- Finding the Radius

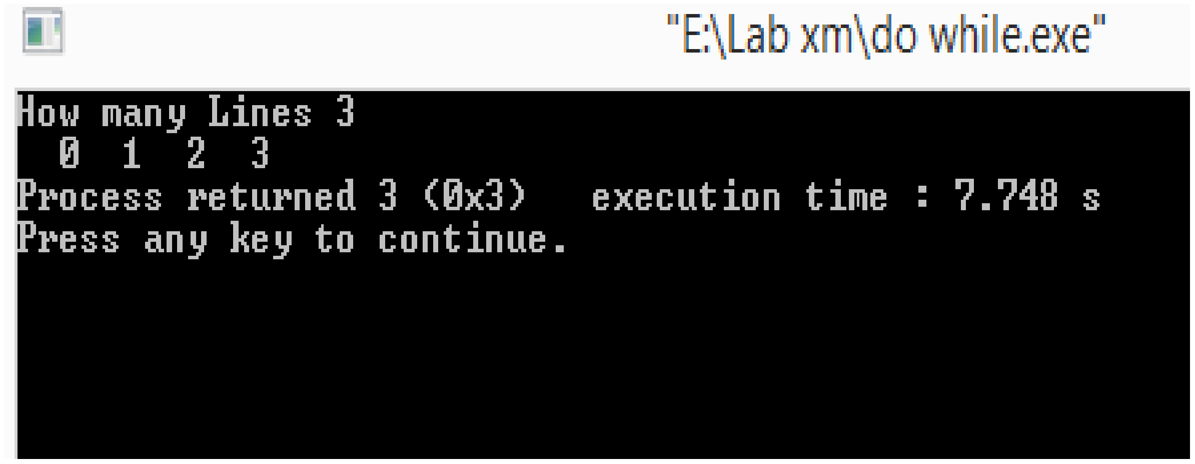
```
#include<stdio.h>

#include<math.h>

void main()

{

   float p=3.1416,r,a;



   printf("Enter the Radius R = ");

   scanf("%f", &r);

   a=p*(r*r);

   printf("The Area is A = ""%f",a);

}
```



```
Enter the Radius R = 3
The Area is A = 28.274399
Process returned 25 (0x19)   execution time : 5.393 s
Press any key to continue.
```

15- Printing Numbers by DO-while Loop

```
#include<stdio.h>

void main()

{

   int i,n;

   printf("How many Lines ");

   scanf("%d", &n);

   i=0;

   do

   {

      printf("%3d",i);

      i++;

   } while(i<=n);

}
```



"E:\Lab xm\do while.exe"

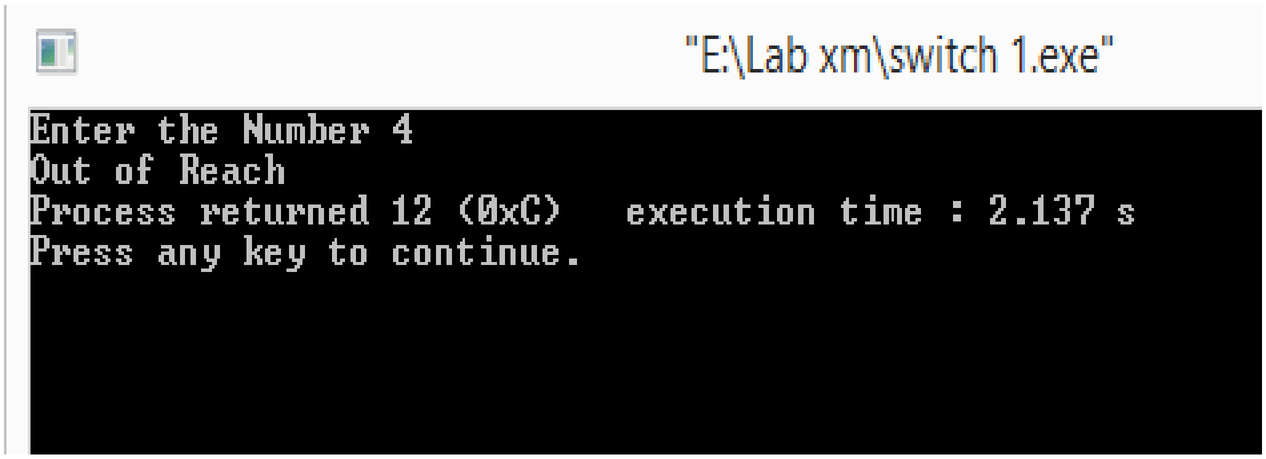
```
How many Lines 3
  0  1  2  3
Process returned 3 (0x3)   execution time : 7.748 s
Press any key to continue.
```

Switch Case

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

16-Switch Case:Finding Statements via switch case coding

```
#include<stdio.h>
void main()
{
   int i,j,n;
   printf("Enter the Number ");
   scanf("%d", &n);

   switch(n)
   {
      case 0:
         printf("zero");
      break;

      case 1:
         printf("One");
      break;

      default:
         printf("Out of Reach");

   }
}
```



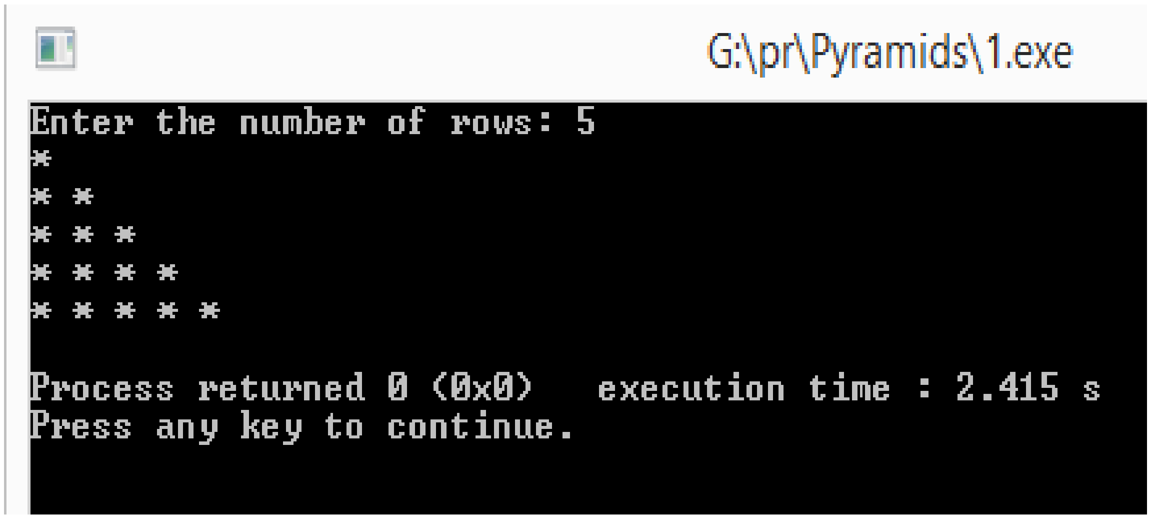
```
Enter the Number 4
Out of Reach
Process returned 12 (0xC)   execution time : 2.137 s
Press any key to continue.
```

PYRAMID

1- Triangle made with star;

```
#include <stdio.h>
int main()
{
   int i,j,rows;
   printf("Enter the number of rows: ");
   scanf("%d",&rows);
   for(i=1;i<=rows;++i)
   {
      for(j=1;j<=i;++j)
      {
        printf("* ");
      }
      printf("\n");
   }
```

```
    return 0;

  }
```



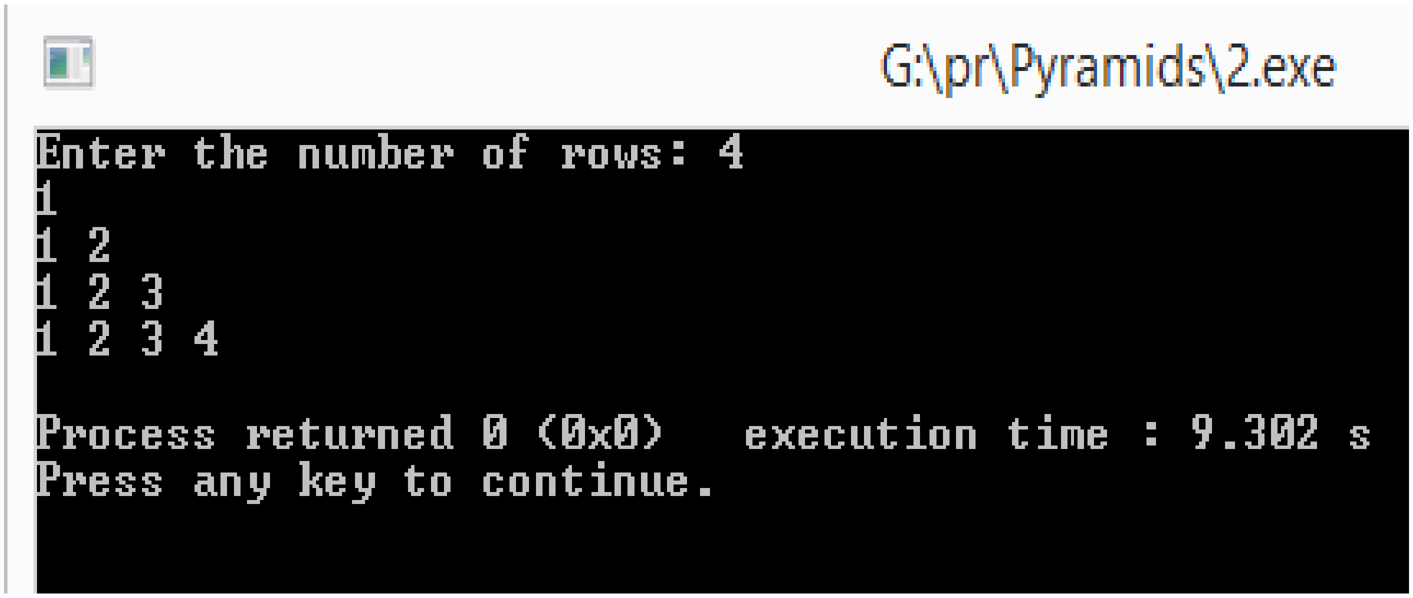
G:\pr\Pyramids\1.exe

```
Enter the number of rows: 5
*
* *
* * *
* * * *
* * * * *

Process returned 0 (0x0)   execution time : 2.415 s
Press any key to continue.
```

2- Pyramid 2

```
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

```
#include <stdio.h>

int main()

{

   int i,j,rows;

   printf("Enter the number of rows: ");

   scanf("%d",&rows);

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

   {

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

      {
        printf("%d ",j);
      }
      printf("\n");
   }
   return 0;
}
```



G:\pr\Pyramids\2.exe

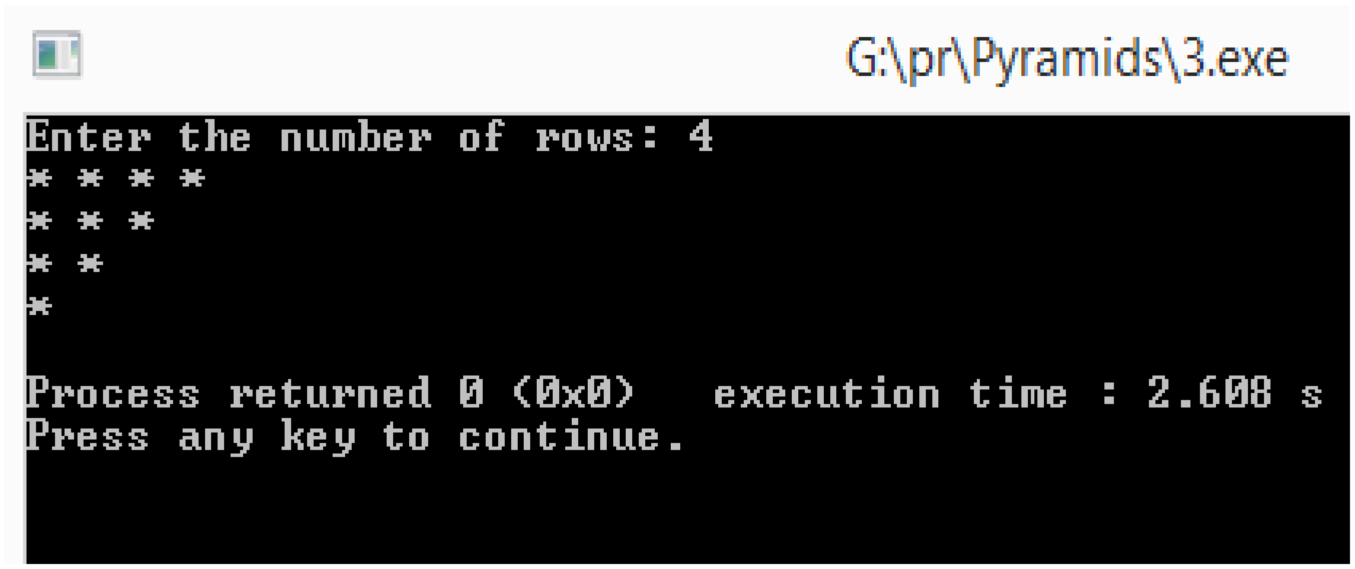
```
Enter the number of rows: 4
1
1 2
1 2 3
1 2 3 4

Process returned 0 (0x0)   execution time : 9.302 s
Press any key to continue.
```

3- Pyr 3:

```
* * * * *
* * * *
* * *
* *
*
```

```
#include <stdio.h>
int main()
{
   int i,j,rows;
```

```
    printf("Enter the number of rows: ");

    scanf("%d",&rows);

    for(i=rows;i>=1;--i)

    {

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

        {

           printf("* ");

        }

    printf("\n");

    }

    return 0;

}
```



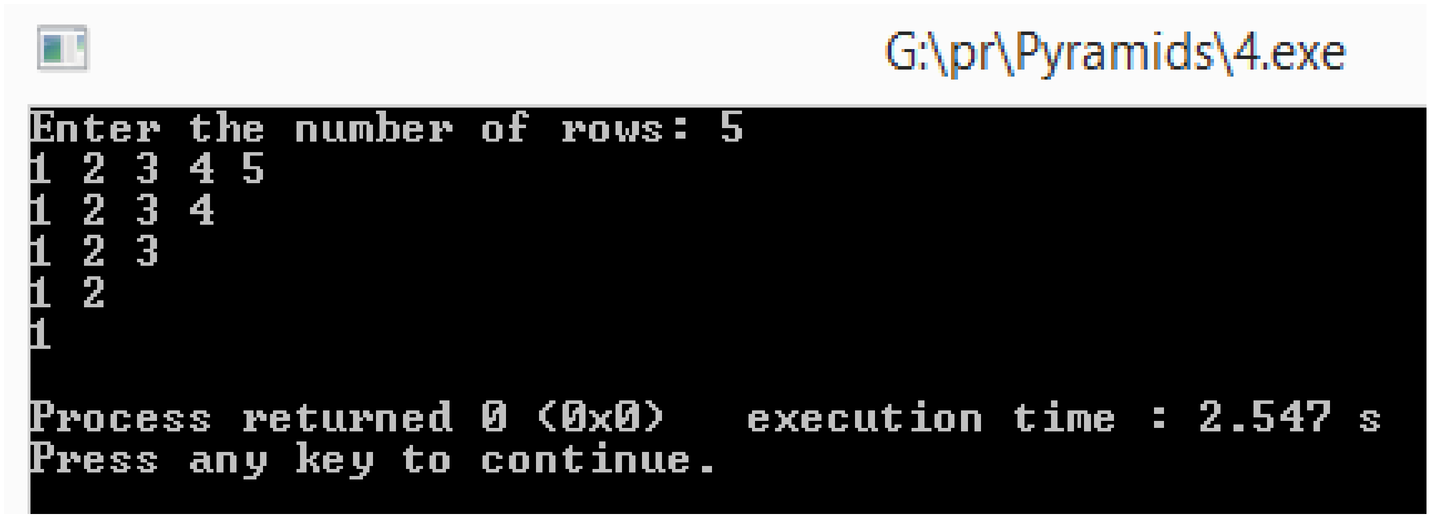
```
Enter the number of rows: 4
* * * *
* * *
* *
*

Process returned 0 (0x0)   execution time : 2.608 s
Press any key to continue.
```

Pyramid: 4

```
1  2  3  4  5
1  2  3  4
1  2  3
```

```
1  2
1
```

```
#include <stdio.h>

int main()

{

   int i,j,rows;

   printf("Enter the number of rows: ");

   scanf("%d",&rows);

   for(i=rows;i>=1;--i)

   {

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

      {

         printf("%d ",j);

      }

   printf("\n");

   }

   return 0;

}
```



```
Enter the number of rows: 5
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

Process returned 0 (0x0)   execution time : 2.547 s
Press any key to continue.
```

Pyramid 5:

```
*

      * * *

    * * * * *

  * * * * * * *

* * * * * * * * *
```

#include <stdio.h>

int main()

{

   int i,space,rows,k=0;

   printf("Enter the number of rows: ");

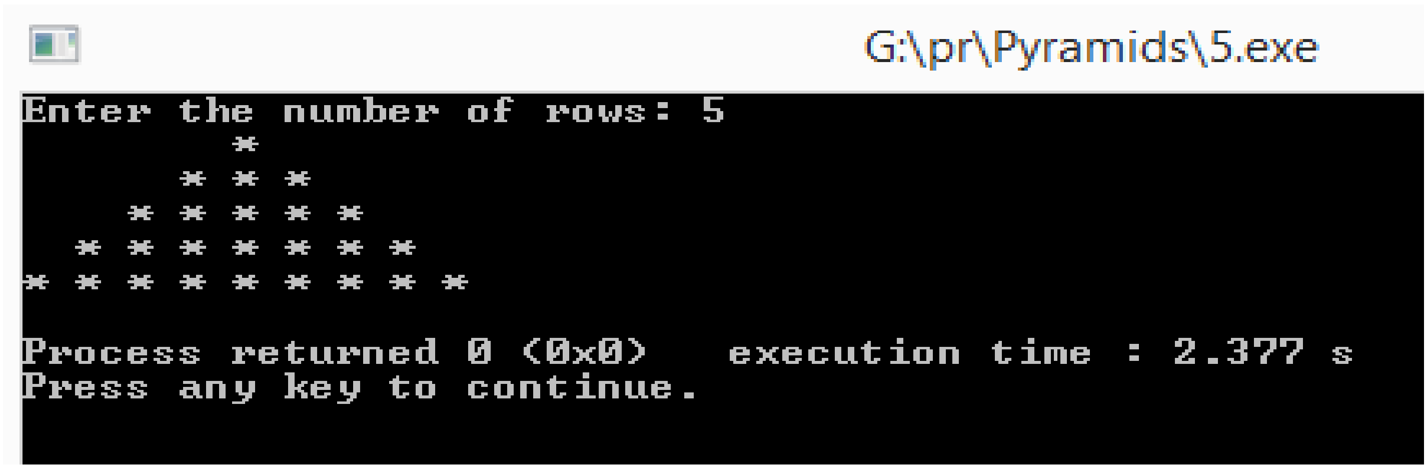
   scanf("%d",&rows);

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

   {

      for(space=1;space<=rows-i;++space)

```
        {
          printf("  ");
        }
        while(k!=2*i-1)
        {
          printf("* ");
          ++k;
        }
        k=0;
        printf("\n");
    }
    return 0;
}
```

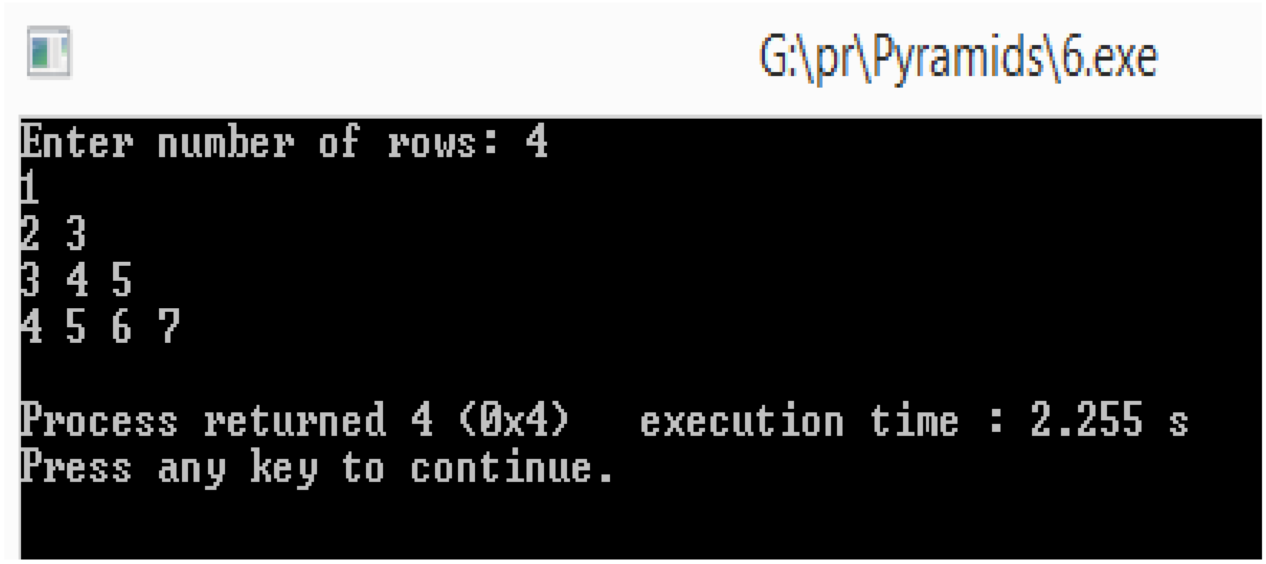


```
G:\pr\Pyramids\5.exe
Enter the number of rows: 5
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *

Process returned 0 (0x0)   execution time : 2.377 s
Press any key to continue.
```

Pyramid: Floyd’s

```
1
2 3
4 5 6
7 8 9 10
```

```
#include<stdio.h>

int main()

{

   int rows,i,j,k=0;

   printf("Enter number of rows: ");

   scanf("%d",&rows);

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

   {

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

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

      ++k;

      printf("\n");

   }

}
```



G:\pr\Pyramids\6.exe

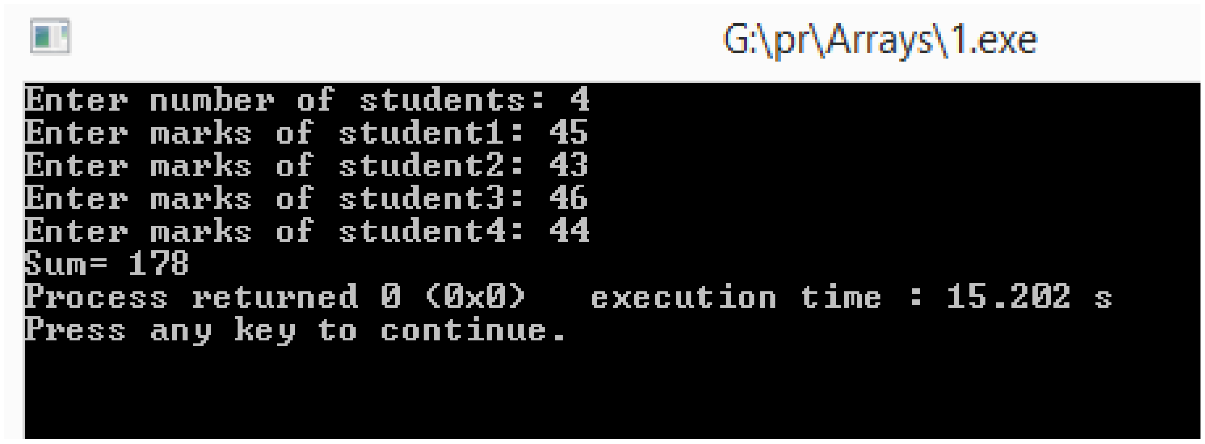
```
Enter number of rows: 4
1
2 3
3 4 5
4 5 6 7

Process returned 4 (0x4)   execution time : 2.255 s
Press any key to continue.
```

Array

Arrays are useful critters that often show up when it would be convenient to have one name for a group of variables of the same type that can be accessed by a numerical index. For example, a tic-tac-toe board can be held in an array and each element of the tic-tac-toe board can easily be accessed by its position (the upper left might be position 0 and the lower right position 8). At heart, arrays are essentially a way to store many values under the same name. You can make an array out of any data-type including structures and classes.

1-- program to find the sum marks of n students using arrays

```
#include <stdio.h>

int main(){

    int marks[10],i,n,sum=0;

    printf("Enter number of students: ");

    scanf("%d",&n);

    for(i=0;i<n;++i){

        printf("Enter marks of student%d: ",i+1);

        scanf("%d",&marks[i]);

        sum+=marks[i];
    }

    printf("Sum= %d",sum);

return 0;

}
```



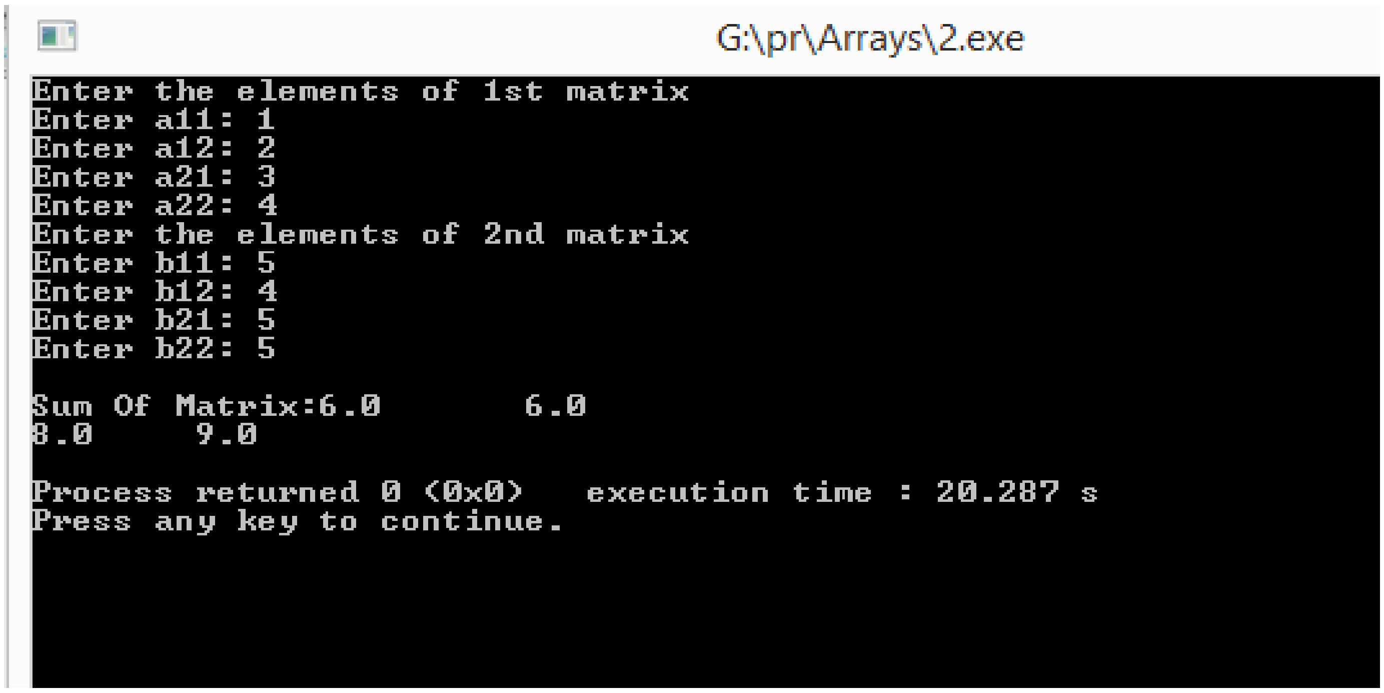
G:\pr\Arrays\1.exe

```
Enter number of students: 4
Enter marks of student1: 45
Enter marks of student2: 43
Enter marks of student3: 46
Enter marks of student4: 44
Sum= 178
Process returned 0 (0x0)   execution time : 15.202 s
Press any key to continue.
```

Prob:2- Multidimensional Array In C

```
#include <stdio.h>

int main(){

  float a[2][2], b[2][2], c[2][2];

  int i,j;

  printf("Enter the elements of 1st matrix\n");

/* Reading two dimensional Array with the help of two for loop. If there was an array of 'n' dimension,
'n' numbers of loops are needed for inserting data to array.*/

  for(i=0;i<2;++i)

     for(j=0;j<2;++j){

     printf("Enter a%d%d: ",i+1,j+1);

     scanf("%f",&a[i][j]);
```

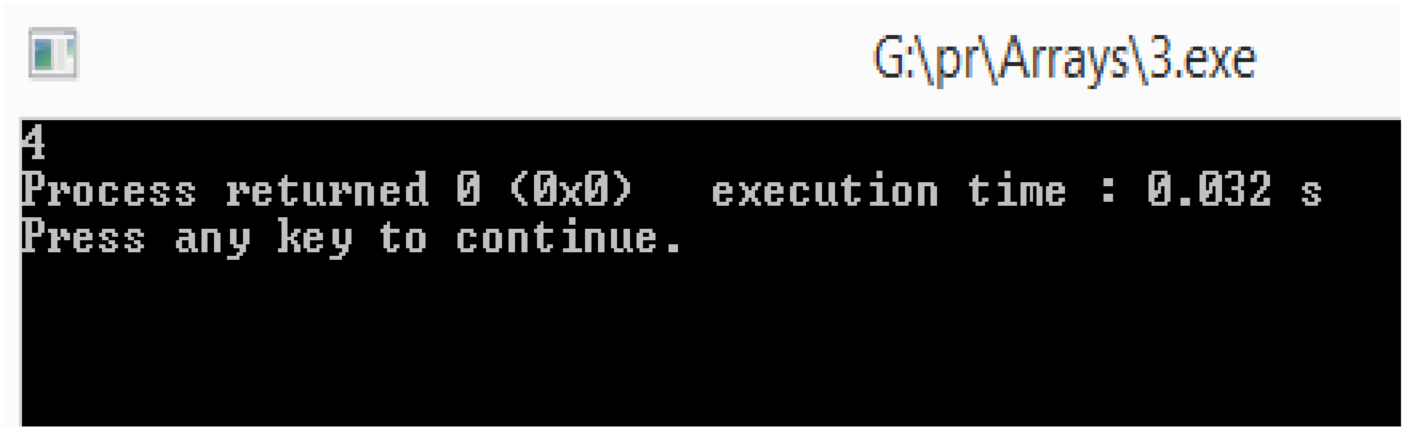
```
      }
  printf("Enter the elements of 2nd matrix\n");
  for(i=0;i<2;++i)
      for(j=0;j<2;++j){
      printf("Enter b%d%d: ",i+1,j+1);
      scanf("%f",&b[i][j]);
      }
  for(i=0;i<2;++i)
      for(j=0;j<2;++j){
/* Writing the elements of multidimensional array using loop. */
      c[i][j]=a[i][j]+b[i][j];  /* Sum of corresponding elements of two arrays. */
      }
  printf("\nSum Of Matrix:");
  for(i=0;i<2;++i)
      for(j=0;j<2;++j){
      printf("%.1f\t",c[i][j]);
          if(j==1)            /* To display matrix sum in order. */
             printf("\n");
     }
return 0;
}
```


G:\pr\Arrays\2.exe

```
Enter the elements of 1st matrix
Enter a11: 1
Enter a12: 2
Enter a21: 3
Enter a22: 4
Enter the elements of 2nd matrix
Enter b11: 5
Enter b12: 4
Enter b21: 5
Enter b22: 5

Sum Of Matrix:6.0       6.0
8.0     9.0

Process returned 0 (0x0)   execution time : 20.287 s
Press any key to continue.
```

3-Passing One-dimensional Array In Function

```
#include <stdio.h>

void display(int a)

  {

  printf("%d",a);

  }

int main(){

  int c[]={2,3,4};

  display(c[2]);  //Passing array element c[2] only.

  return 0;

}
```

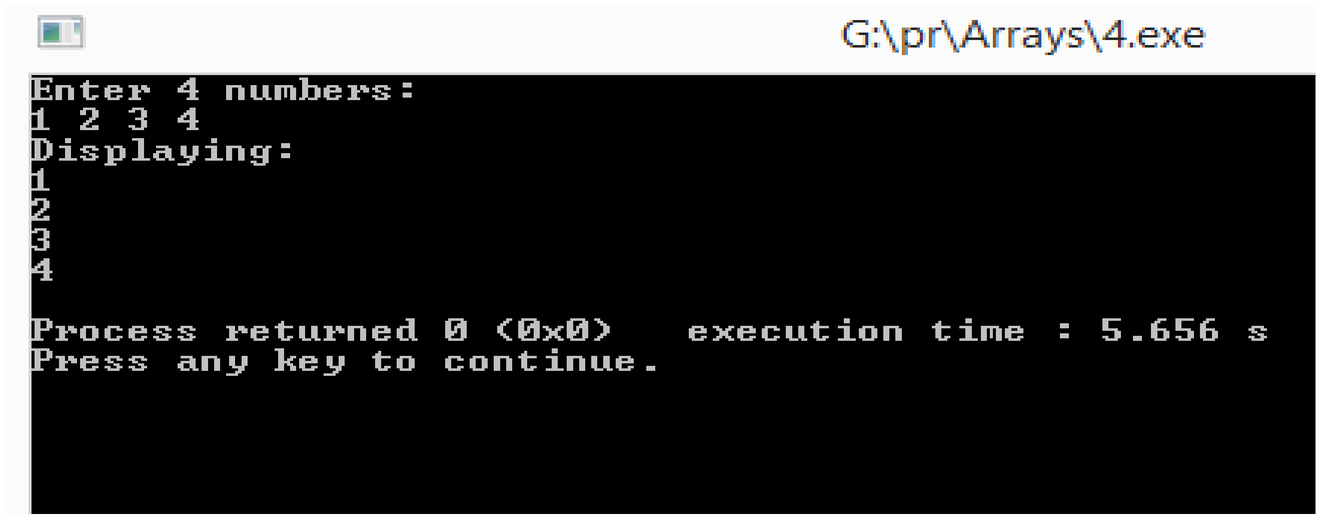


```
4
Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

4-Passing Multi-dimensional Arrays to Function

```
#include<stdio.h>

void Function(int c[2][2]);

int main(){

  int c[2][2],i,j;

  printf("Enter 4 numbers:\n");

  for(i=0;i<2;++i)

    for(j=0;j<2;++j){

        scanf("%d",&c[i][j]);

    }

  Function(c);   /* passing multi-dimensional array to function */

  return 0;

}

void Function(int c[2][2]){

/* Instead to above line, void Function(int c[][2]){ is also valid */

  int i,j;

  printf("Displaying:\n");

  for(i=0;i<2;++i)

    for(j=0;j<2;++j)
```

```
        printf("%d\n",c[i][j]);

}
```

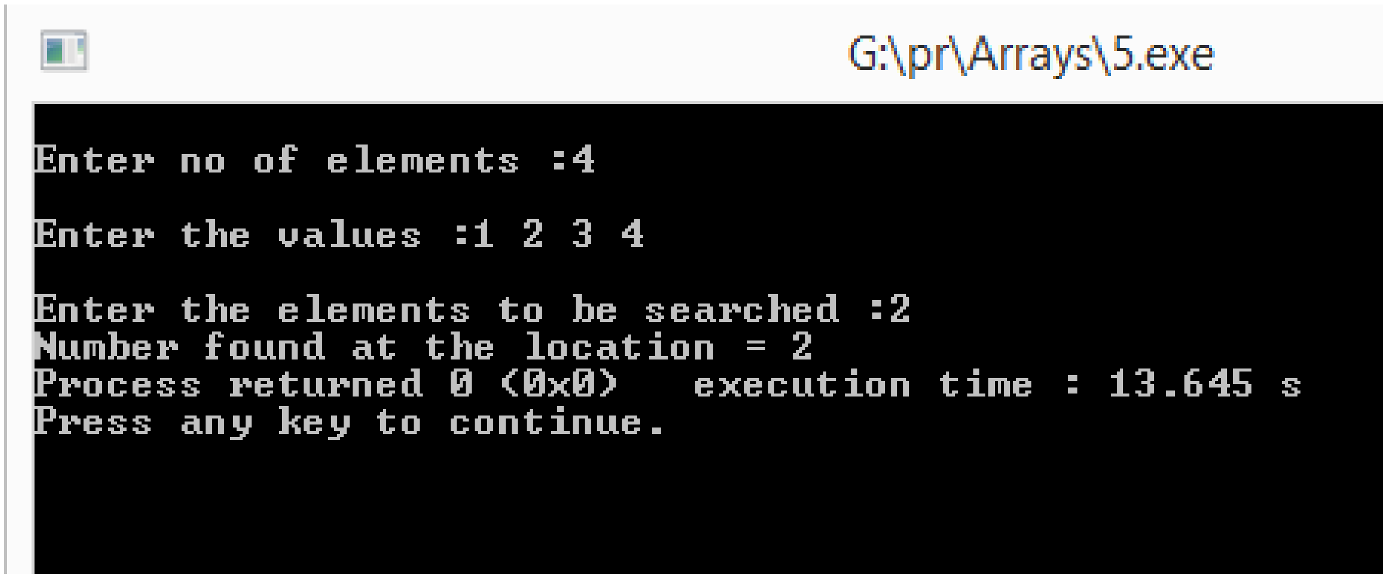


G:\pr\Arrays\4.exe

```
Enter 4 numbers:
1 2 3 4
Displaying:
1
2
3
4

Process returned 0 (0x0)   execution time : 5.656 s
Press any key to continue.
```

5- Array search

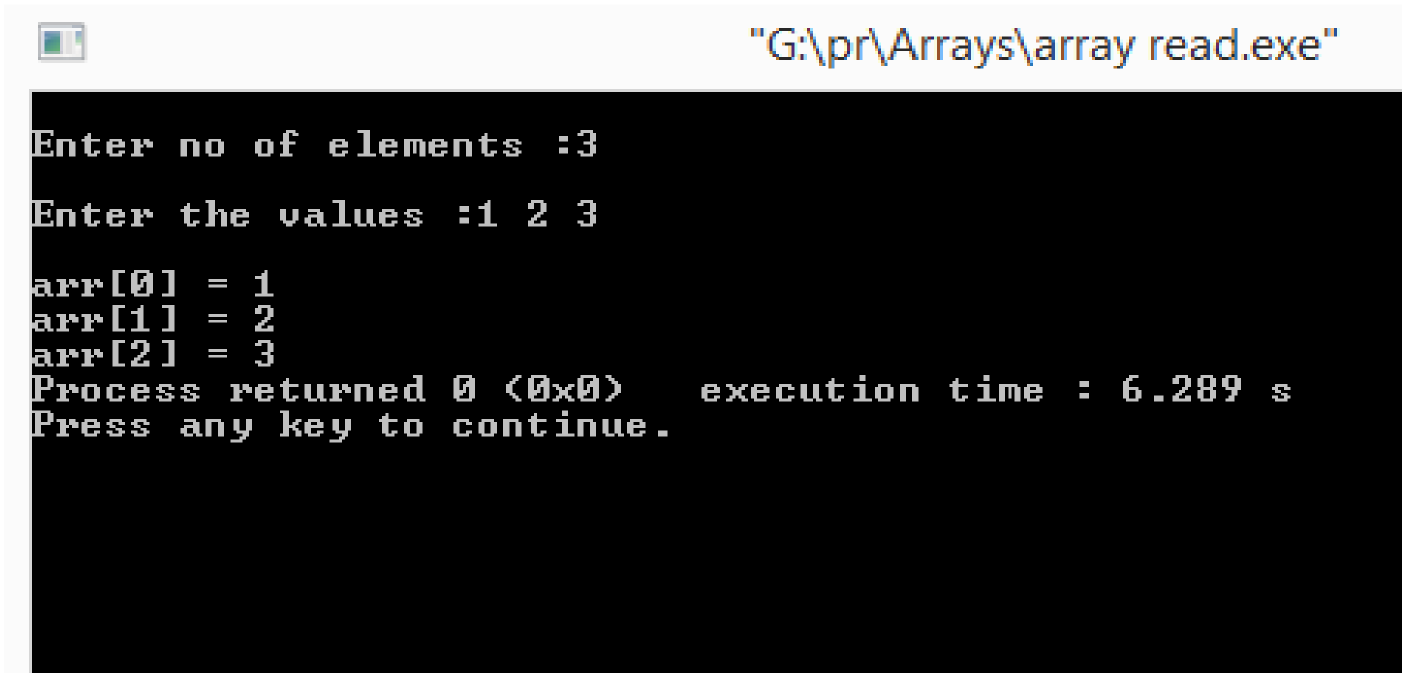
```
#include<stdio.h>


int main() {

  int a[30], ele, num, i;


  printf("\nEnter no of elements :");

  scanf("%d", &num);


  printf("\nEnter the values :");

  for (i = 0; i < num; i++) {

    scanf("%d", &a[i]);

  }
  //Read the element to be searched

  printf("\nEnter the elements to be searched :");

  scanf("%d", &ele);



  //Search starts from the zeroth location

  i = 0;

  while (i < num && ele != a[i]) {

    i++;

  }



  //If i < num then Match found

  if (i < num) {

    printf("Number found at the location = %d", i + 1);

  } else {

    printf("Number not found");

  }



  return (0);

}
```



```
Enter no of elements :4

Enter the values :1 2 3 4

Enter the elements to be searched :2
Number found at the location = 2
Process returned 0 (0x0)   execution time : 13.645 s
Press any key to continue.
```

6- Reading An Array

```
#include<stdio.h>


int main() {
  int i, arr[50], num;


  printf("\nEnter no of elements :");
  scanf("%d", &num);


  //Reading values into Array
  printf("\nEnter the values :");
  for (i = 0; i < num; i++) {
    scanf("%d", &arr[i]);
  }


  //Printing of all elements of array
  for (i = 0; i < num; i++) {
    printf("\narr[%d] = %d", i, arr[i]);

  }



  return (0);

}
```



```
"G:\pr\Arrays\array read.exe"

Enter no of elements :3

Enter the values :1 2 3

arr[0] = 1
arr[1] = 2
arr[2] = 3
Process returned 0 (0x0)   execution time : 6.289 s
Press any key to continue.
```

C Programming String:

In C programming, array of character are called strings. A string is terminated by null character /0. For example:

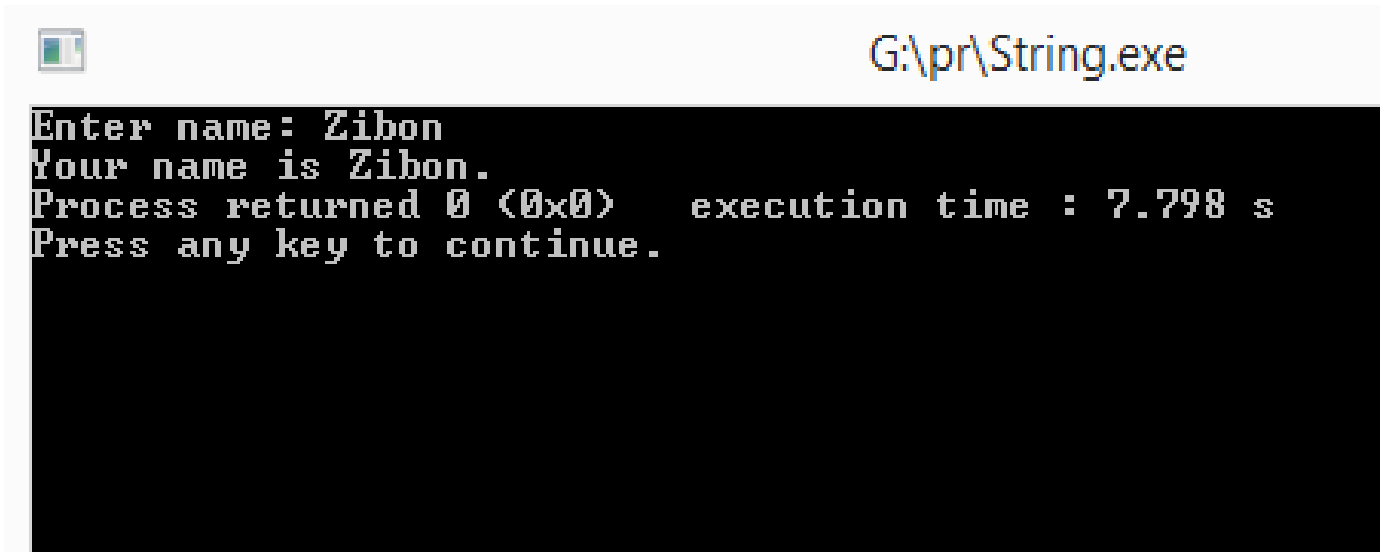
"c string" is a string.

1- **Writing a C program to illustrate how to read string from terminal.**

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

```
int main(){

    char name[20];

    printf("Enter name: ");

    scanf("%s",name);

    printf("Your name is %s.",name);

    return 0;

}
```

Input: Zibon



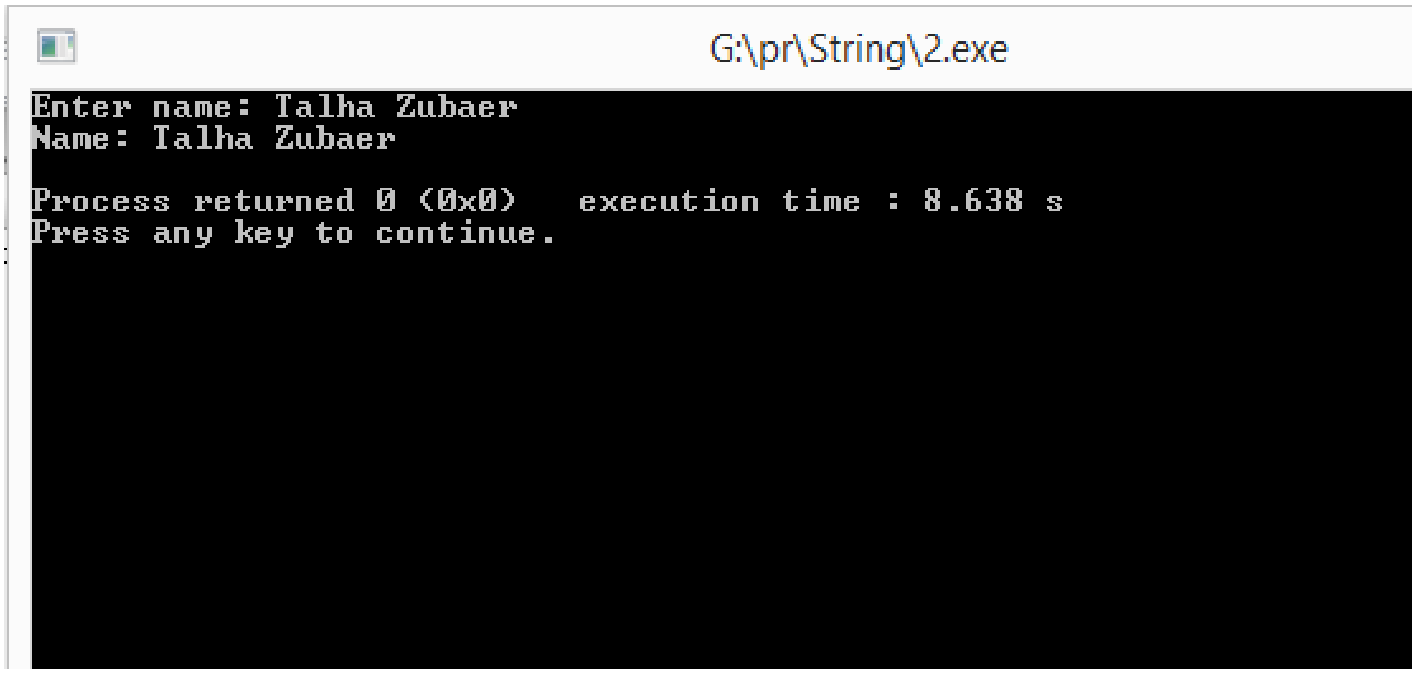
```
Enter name: Zibon
Your name is Zibon.
Process returned 0 (0x0)   execution time : 7.798 s
Press any key to continue.
```

2- **C program to read line of text manually.**

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

```
int main(){

    char name[30],ch;

    int i=0;

    printf("Enter name: ");

    while(ch!='\n')    // terminates if user hit enter

    {

        ch=getchar();

        name[i]=ch;

        i++;

    }

    name[i]='\0';       // inserting null character at end

    printf("Name: %s",name);

    return 0;

}
```

Input: Talha Zubaer



```
Enter name: Talha Zubaer
Name: Talha Zubaer

Process returned 0 (0x0)   execution time : 8.638 s
Press any key to continue.
```

POINTER

Pointers in C are easy and fun to learn. Some C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect C programmer. Let's start learning them in simple and easy steps.

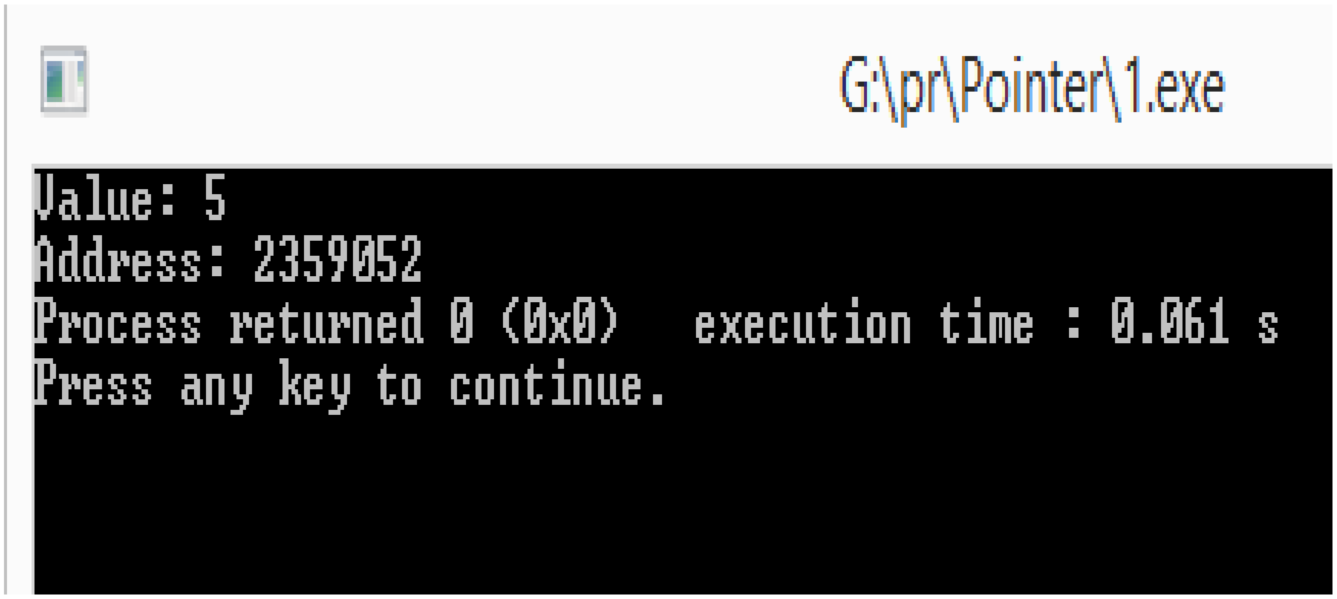
As you know, every variable is a memory location and every memory location has its address defined which can be accessed using ampersand (&) operator, which denotes an address in memory. Consider the following example, which prints the address of the variables defined −

1- Structure of a pointer or, a simple program that shows the syntax of pointer:

```
#include <stdio.h>

int main(){

  int var=5;

  printf("Value: %d\n",var);

  printf("Address: %d",&var);  //Notice, the ampersand(&) before var.
  return 0;
}
```

Input: 5

Output:



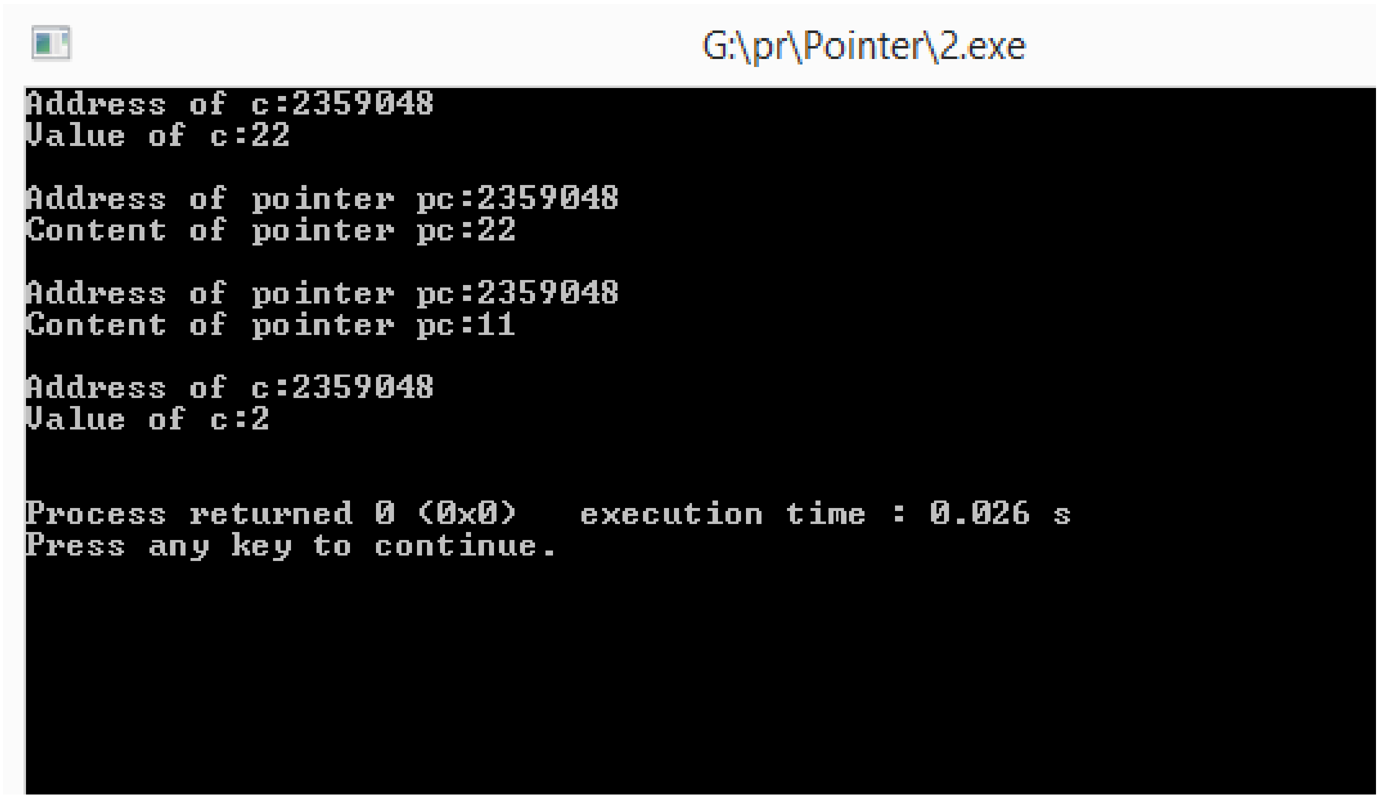
```
G:\pr\Pointer\1.exe
Value: 5
Address: 2359052
Process returned 0 (0x0)   execution time : 0.061 s
Press any key to continue.
```

NB: it can obtain different value of address while using this code.

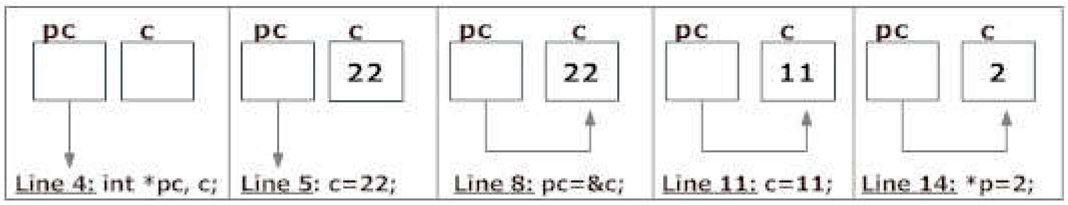
2- Example To Demonstrate Working of Pointers

```
#include <stdio.h>
int main(){
   int* pc;
   int c;
   c=22;
   printf("Address of c:%d\n",&c);
   printf("Value of c:%d\n\n",c);
   pc=&c;
   printf("Address of pointer pc:%d\n",pc);
   printf("Content of pointer pc:%d\n\n",*pc);
```

```
  c=11;

  printf("Address of pointer pc:%d\n",pc);

  printf("Content of pointer pc:%d\n\n",*pc);

  *pc=2;

  printf("Address of c:%d\n",&c);

  printf("Value of c:%d\n\n",c);

  return 0;

}
```

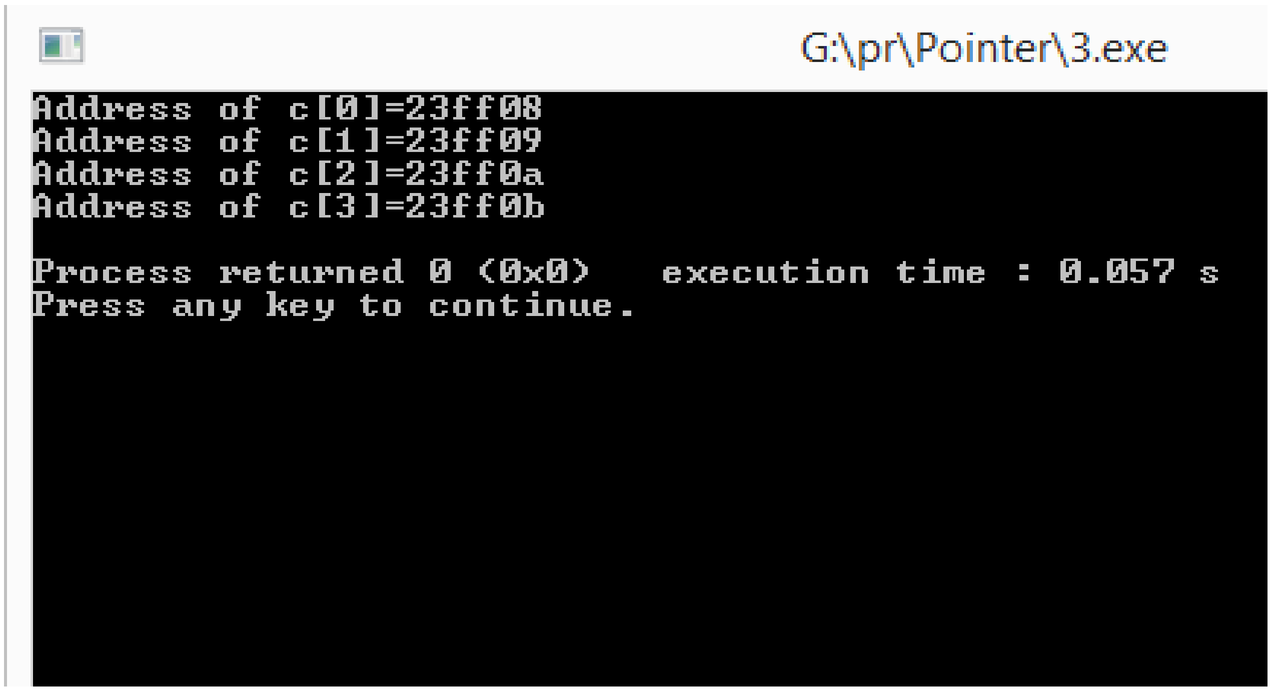


```
Address of c:2359048
Value of c:22

Address of pointer pc:2359048
Content of pointer pc:22

Address of pointer pc:2359048
Content of pointer pc:11

Address of c:2359048
Value of c:2


Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```



3- Using Pointers with Array

```
#include <stdio.h>

int main(){

  char c[4];

  int i;

  for(i=0;i<4;++i){

    printf("Address of c[%d]=%x\n",i,&c[i]);

  }

  return 0;

}
```



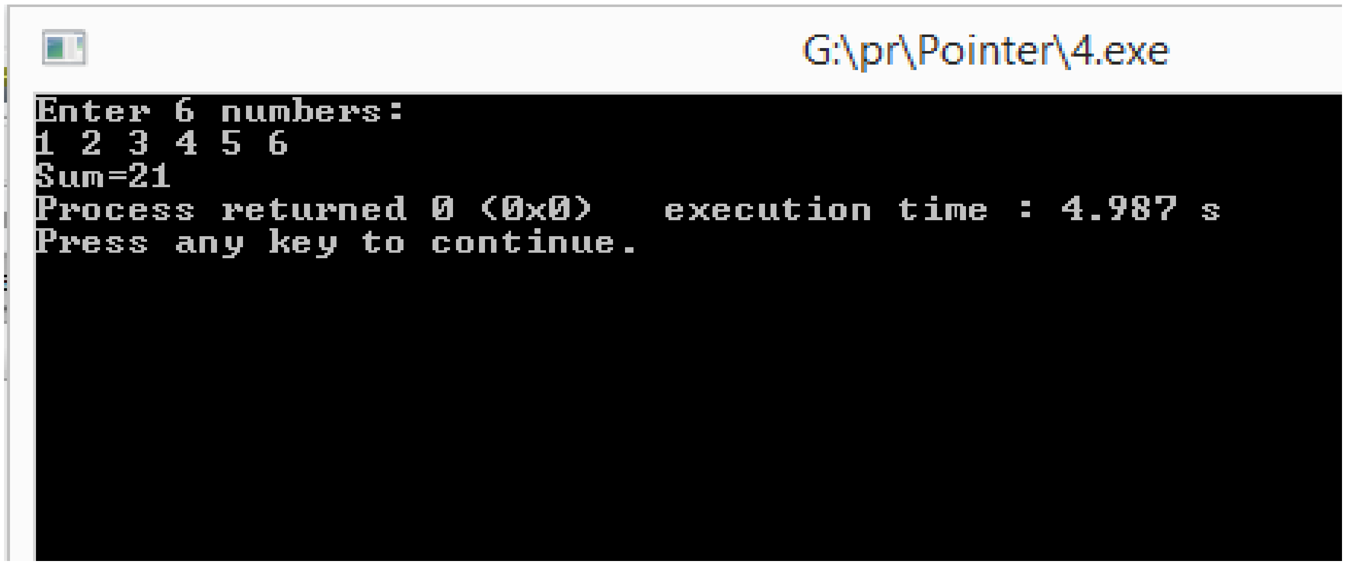
```
Address of c[0]=23ff08
Address of c[1]=23ff09
Address of c[2]=23ff0a
Address of c[3]=23ff0b

Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
```

4- Declaring an array which can use pointer to alter the data of an array.

```
#include <stdio.h>

int main(){

 int i,class[6],sum=0;

 printf("Enter 6 numbers:\n");

 for(i=0;i<6;++i){
   scanf("%d",(class+i)); // (class+i) is equivalent to &class[i]
   sum += *(class+i); // *(class+i) is equivalent to class[i]
 }
 printf("Sum=%d",sum);
 return 0;
}
```

Input: 1 2 3 4 5 6

Output:



G:\pr\Pointer\4.exe

```
Enter 6 numbers:
1 2 3 4 5 6
Sum=21
Process returned 0 (0x0)   execution time : 4.987 s
Press any key to continue.
```

5- Pointer With Function

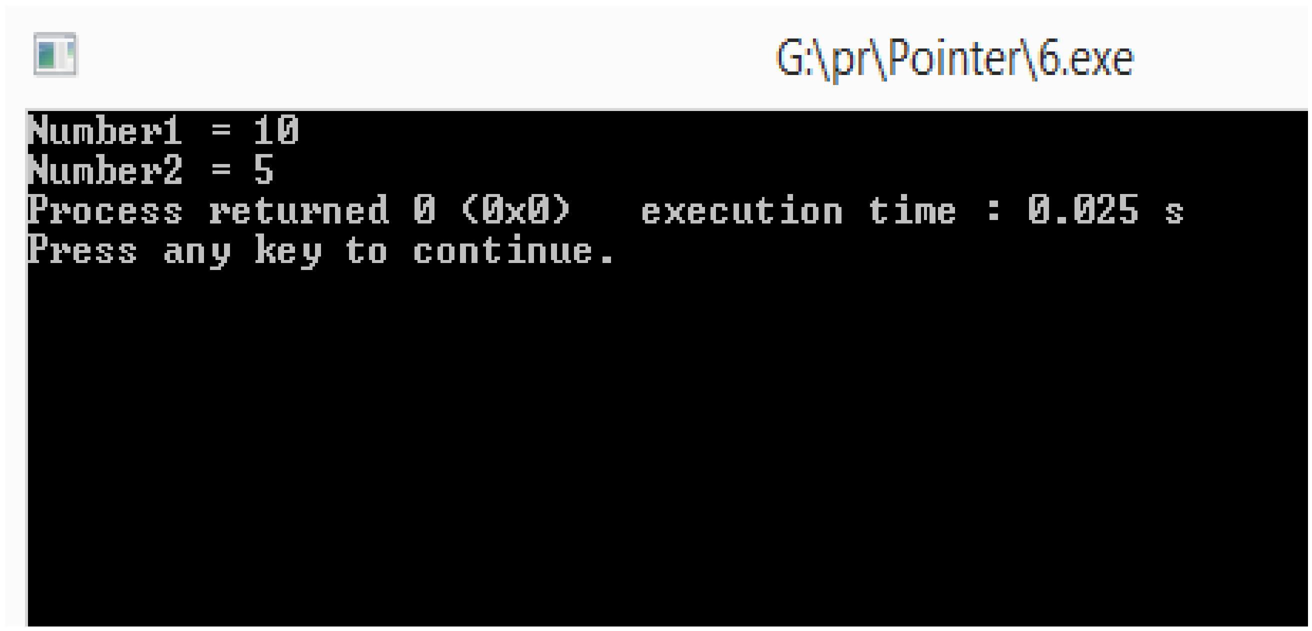
```
#include <stdio.h>
void swap(int *a,int *b);
int main(){
  int num1=5,num2=10;
  swap(&num1,&num2);  /* address of num1 and num2 is passed to swap function */
  printf("Number1 = %d\n",num1);
  printf("Number2 = %d",num2);
  return 0;
}
void swap(int *a,int *b){ /* pointer a and b points to address of num1 and num2 respectively */
  int temp;
  temp=*a;
  *a=*b;
  *b=temp;
}
```

Input—

Number1=10;

Number2=5.

Output:



G:\pr\Pointer\6.exe

```
Number1 = 10
Number2 = 5
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```

Explanation

The address of memory location *num1* and *num2* are passed to function and the pointers `*a` and `*b`accept those values. So, the pointer *a* and *b* points to address of *num1* and *num2* respectively. When, the value of pointer are changed, the value in memory location also changed correspondingly. Hence, change made to `*a` and `*b` was reflected in *`num1`* and *`num2`* in main function.

This technique is known as call by reference in C programming.

C Programming Functions

In programming, a function is a segment that groups code to perform a specific task.

A C program has at least one function `main( )`. Without `main()` function, there is technically no C program.

Types of C functions

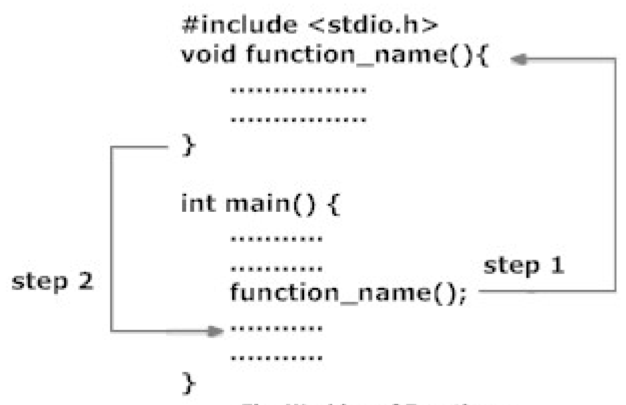
There are two types of functions in C programming:

- Library function
- User defined function

Syntax:

How user-defined function works in C Programming?

```
#include <stdio.h>

void function_name(){

................

................

}

int main(){

...........

...........
function_name();

...........

...........

}
```



```
#include <stdio.h>
void function_name(){
    .................
    .................
}

int main() {
    ............
    ............
    function_name();
    ............
    ............
}
```

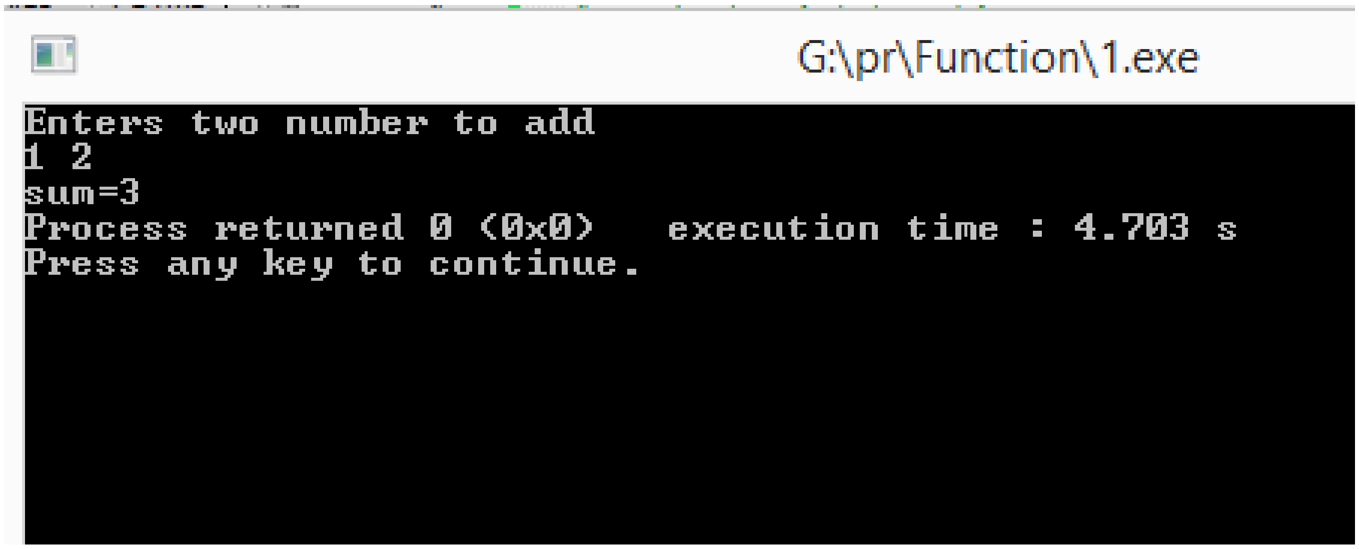
Fig: Working of Functions

This is how function works.

1- Write a C program to add two integers. Make a function add to add integers and display sum in main() function.

```
#include <stdio.h>

int add(int a, int b);          //function prototype(declaration)

int main(){

    int num1,num2,sum;

    printf("Enters two number to add\n");

    scanf("%d %d",&num1,&num2);

    sum=add(num1,num2);        //function call

    printf("sum=%d",sum);

    return 0;

}
```

Input : 1   2



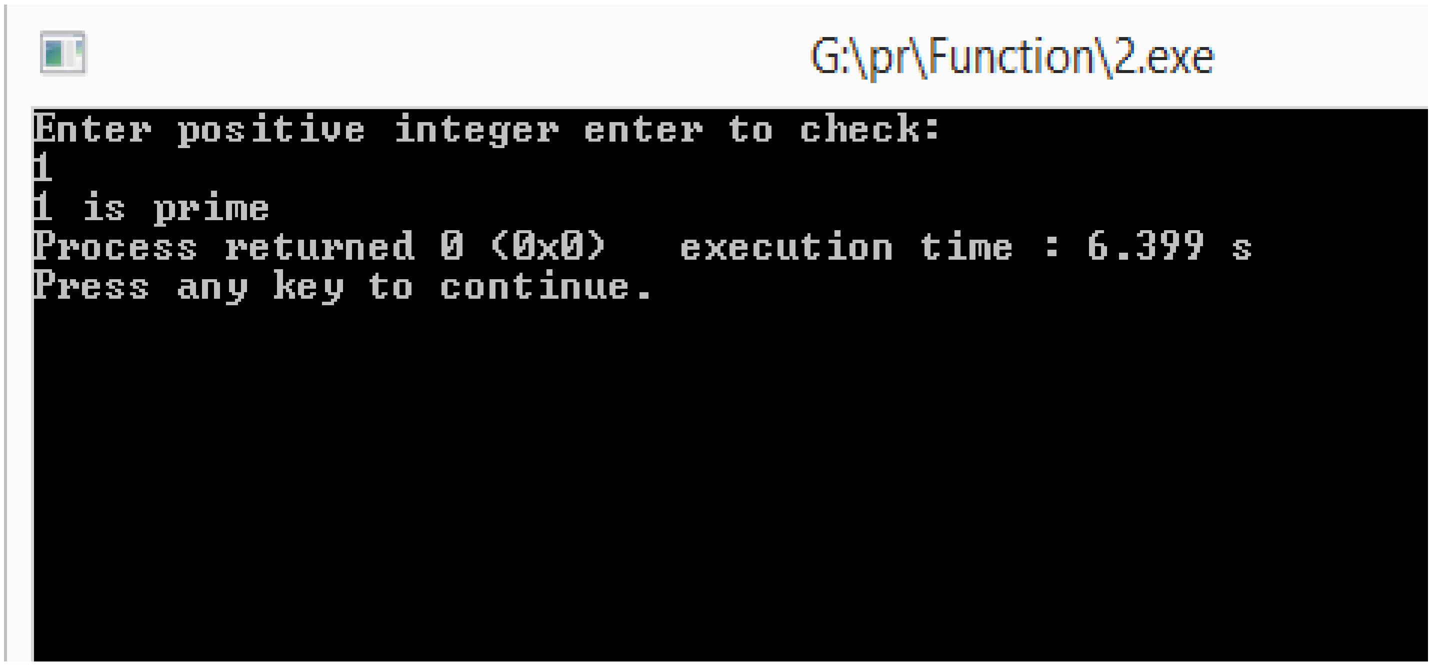
```
Enters two number to add
1 2
sum=3
Process returned 0 (0x0)   execution time : 4.703 s
Press any key to continue.
```

Function with no arguments and no return value.

```
#include <stdio.h>

void prime();

int main(){

  prime();     //No argument is passed to prime().

  return 0;

}

void prime(){

/* There is no return value to calling function main(). Hence, return type of prime() is void */

  int num,i,flag=0;

  printf("Enter positive integer enter to check:\n");

  scanf("%d",&num);

  for(i=2;i<=num/2;++i){

    if(num%i==0){

       flag=1;

    }
```

```
   }
   if (flag==1)
       printf("%d is not prime",num);
   else
      printf("%d is prime",num);
   }
```

INPUT : 1

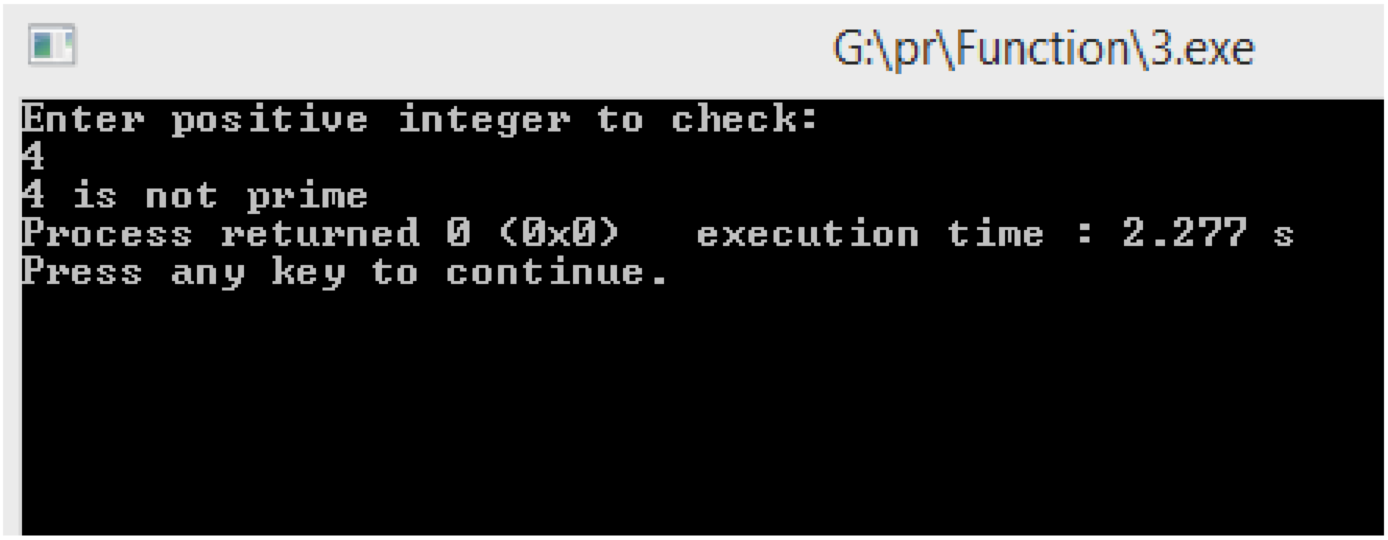


G:\pr\Function\2.exe

```
Enter positive integer enter to check:
1
1 is prime
Process returned 0 (0x0)   execution time : 6.399 s
Press any key to continue.
```

Function with no arguments but return value

```
#include <stdio.h>
int input();
int main(){
   int num,i,flag = 0;
   num=input();     /* No argument is passed to input() */
   for(i=2; i<=num/2; ++i){
   if(num%i==0){
       flag = 1;
       break;
   }
   }
   if(flag == 1)
       printf("%d is not prime",num);
   else
       printf("%d is prime", num);
   return 0;
}
int input(){   /* Integer value is returned from input() to calling function */
   int n;
   printf("Enter positive integer to check:\n");
   scanf("%d",&n);
   return n;
}
```



G:\pr\Function\3.exe

```
Enter positive integer to check:
4
4 is not prime
Process returned 0 (0x0)   execution time : 2.277 s
Press any key to continue.
```

Data Structure:

Structure is the collection of variables of different types under a single name for better handling. For example: You want to store the information about person about his/her name, citizenship number and salary. You can create these information separately but, better approach will be collection of these information under single name because all these information are related to person

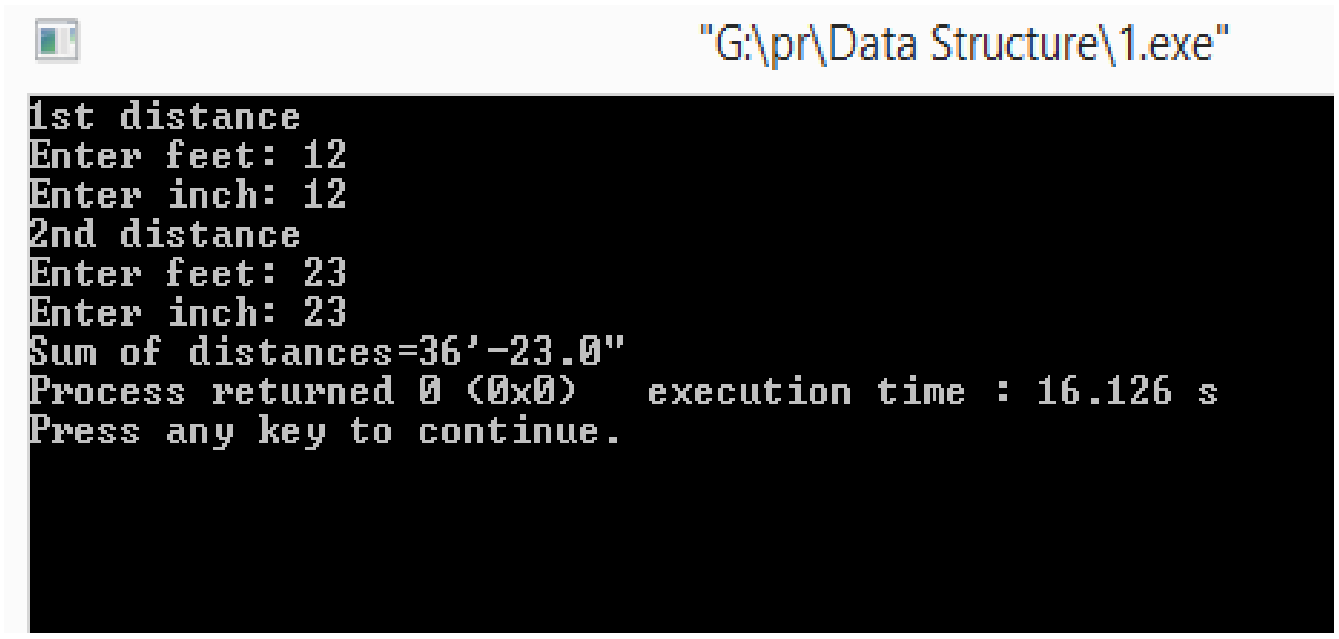
Syntax of structure:

```
struct structure_name

{

   data_type member1;

   data_type member2;

   .

   .

   data_type memeber;

}
```

1- Write a C program to add two distances entered by user. Measurement of distance should be in inch and feet.(Note: 12 inches = 1 foot)

```
#include <stdio.h>

struct Distance{

   int feet;

   float inch;

}d1,d2,sum;

int main(){

   printf("1st distance\n");

   printf("Enter feet: ");
   scanf("%d",&d1.feet);  /* input of feet for structure variable d1 */
   printf("Enter inch: ");
   scanf("%f",&d1.inch);  /* input of inch for structure variable d1 */
   printf("2nd distance\n");
   printf("Enter feet: ");
   scanf("%d",&d2.feet);  /* input of feet for structure variable d2 */
   printf("Enter inch: ");
   scanf("%f",&d2.inch);  /* input of inch for structure variable d2 */
   sum.feet=d1.feet+d2.feet;
   sum.inch=d1.inch+d2.inch;
   if (sum.inch>12){  //If inch is greater than 12, changing it to feet.
      ++sum.feet;
      sum.inch=sum.inch-12;
   }
   printf("Sum of distances=%d\'-%.1f\"",sum.feet,sum.inch);
/* printing sum of distance d1 and d2 */
   return 0;
}
```

Output:


"G:\pr\Data Structure\1.exe"

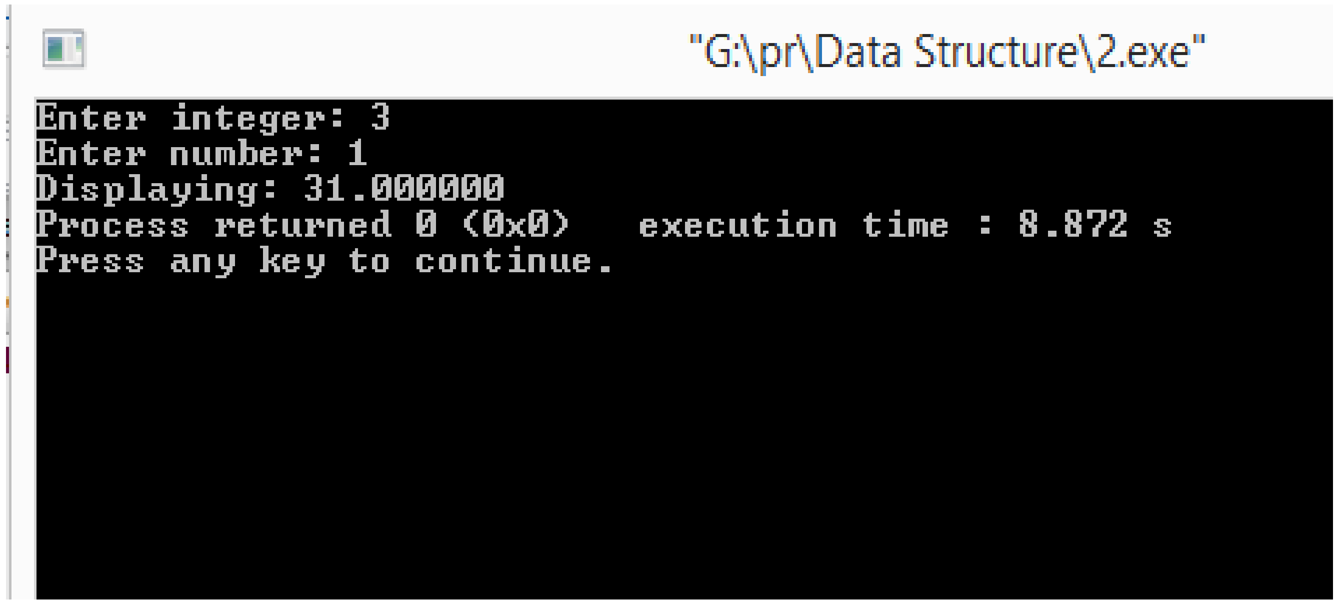
```
1st distance
Enter feet: 12
Enter inch: 12
2nd distance
Enter feet: 23
Enter inch: 23
Sum of distances=36'-23.0"
Process returned 0 (0x0)   execution time : 16.126 s
Press any key to continue.
```

C Programming Structure and Pointer

2- example to access structure's member through pointer.

```
#include <stdio.h>

struct name{

  int a;

  float b;

};

int main(){

    struct name *ptr,p;

    ptr=&p;           /* Referencing pointer to memory address of p */

    printf("Enter integer: ");

    scanf("%d",&(*ptr).a);
```

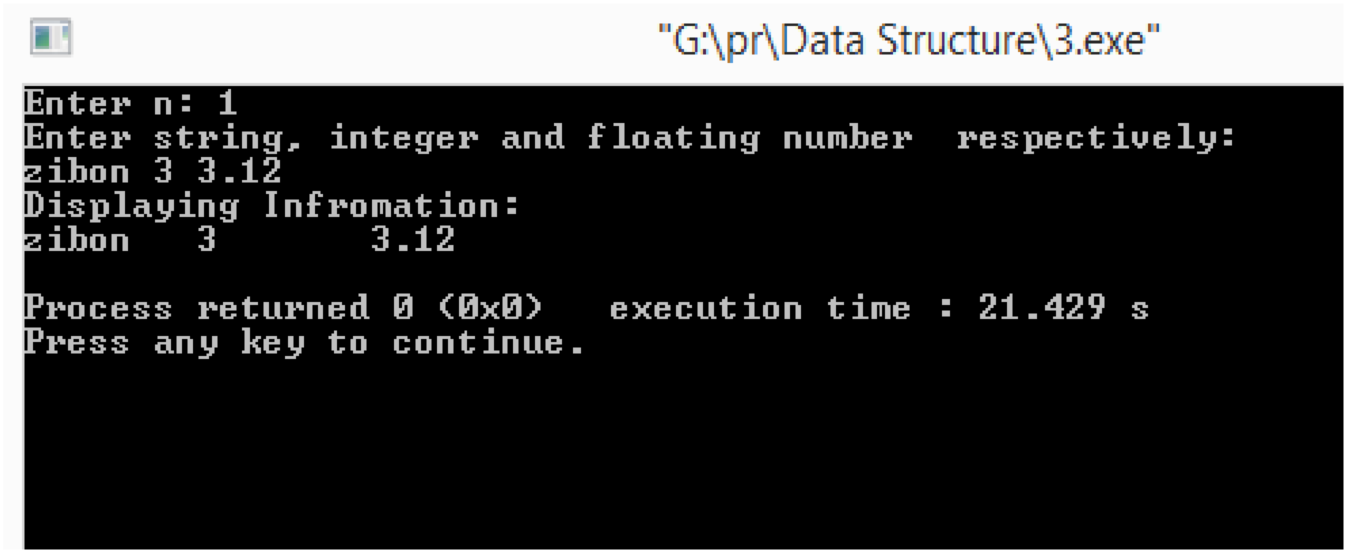
```
   printf("Enter number: ");
   scanf("%f",&(*ptr).b);
   printf("Displaying: ");
   printf("%d%f",(*ptr).a,(*ptr).b);
   return 0;
}
```



```
Enter integer: 3
Enter number: 1
Displaying: 31.000000
Process returned 0 (0x0)   execution time : 8.872 s
Press any key to continue.
```

3-Accessing structure member through pointer using dynamic memory allocation

```
#include <stdio.h>
#include<stdlib.h>
struct name {
  int a;
  float b;
  char c[30];
};
int main(){
  struct name *ptr;

  int i,n;

  printf("Enter n: ");

  scanf("%d",&n);

  ptr=(struct name*)malloc(n*sizeof(struct name));

/* Above statement allocates the memory for n structures with pointer ptr pointing to base
address */

  for(i=0;i<n;++i){

      printf("Enter string, integer and floating number  respectively:\n");

      scanf("%s%d%f",&(ptr+i)->c,&(ptr+i)->a,&(ptr+i)->b);

  }

  printf("Displaying Infromation:\n");

  for(i=0;i<n;++i)

      printf("%s\t%d\t%.2f\n",(ptr+i)->c,(ptr+i)->a,(ptr+i)->b);

  return 0;

}
```

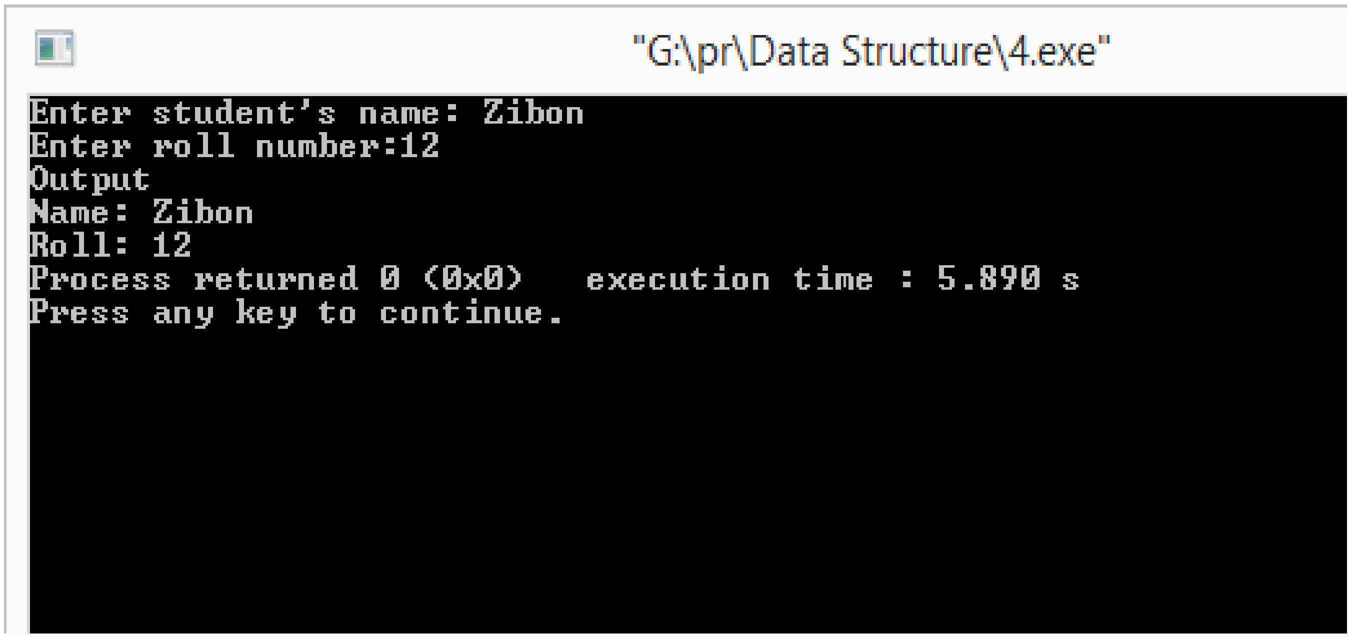


"G:\pr\Data Structure\3.exe"

```
Enter n: 1
Enter string, integer and floating number  respectively:
zibon 3 3.12
Displaying Infromation:
zibon   3       3.12

Process returned 0 (0x0)   execution time : 21.429 s
Press any key to continue.
```

4- Write a C program to create a structure student, containing name and roll. Ask user the name and roll of a student in main function. Pass this structure to a function and display the information in that function.

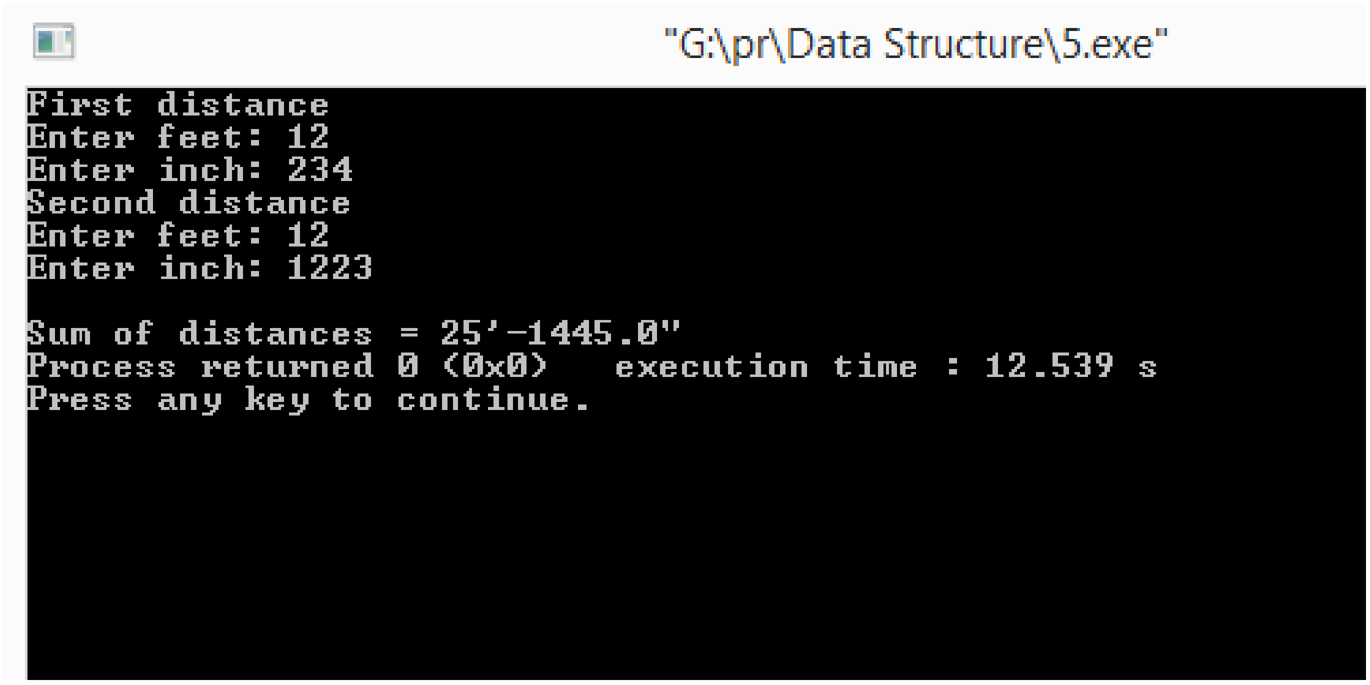
```
#include <stdio.h>

struct student{

   char name[50];

   int roll;

};

void Display(struct student stu);

/* function prototype should be below to the structure declaration otherwise compiler shows
error */

int main(){

   struct student s1;

   printf("Enter student's name: ");

   scanf("%s",&s1.name);

   printf("Enter roll number:");

   scanf("%d",&s1.roll);

   Display(s1);   // passing structure variable s1 as argument

   return 0;

}

void Display(struct student stu){

  printf("Output\nName: %s",stu.name);

  printf("\nRoll: %d",stu.roll);

}
```


"G:\pr\Data Structure\4.exe"

```
Enter student's name: Zibon
Enter roll number:12
Output
Name: Zibon
Roll: 12
Process returned 0 (0x0)   execution time : 5.890 s
Press any key to continue.
```

5- Writing a C program to add two distances(feet-inch system) entered by user. To solve this program, make a structure. Pass two structure variable (containing distance in feet and inch) to add function by reference and display the result in main function without returning it.

```
#include <stdio.h>

struct distance{

   int feet;

   float inch;

};

void Add(struct distance d1,struct distance d2, struct distance *d3);

int main()

{

   struct distance dist1, dist2, dist3;

   printf("First distance\n");

   printf("Enter feet: ");

   scanf("%d",&dist1.feet);

   printf("Enter inch: ");
   scanf("%f",&dist1.inch);

   printf("Second distance\n");

   printf("Enter feet: ");

   scanf("%d",&dist2.feet);

   printf("Enter inch: ");

   scanf("%f",&dist2.inch);

   Add(dist1, dist2, &dist3);


/*passing structure variables dist1 and dist2 by value whereas passing structure variable dist3
by reference */

   printf("\nSum of distances = %d\'-%.1f\"",dist3.feet, dist3.inch);

   return 0;

}

void Add(struct distance d1,struct distance d2, struct distance *d3)

{

/* Adding distances d1 and d2 and storing it in d3 */

    d3->feet=d1.feet+d2.feet;

    d3->inch=d1.inch+d2.inch;

    if (d3->inch>=12) {     /* if inch is greater or equal to 12, converting it to feet. */

        d3->inch-=12;

        ++d3->feet;

   }

}
```



"G:\pr\Data Structure\5.exe"

```
First distance
Enter feet: 12
Enter inch: 234
Second distance
Enter feet: 12
Enter inch: 1223

Sum of distances = 25'-1445.0"
Process returned 0 (0x0)   execution time : 12.539 s
Press any key to continue.
```