POINTERS

The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer.

e.g., int n = 10;

int* p = &n; // Variable p of type pointer is pointing to the address of the variable n of type integer.

Defining Pointer variables

& Address-of0operator

***** dereferencing operator/ value at address operator/ indirection operator

To print an address of a variable using c programme we use %u format specifier which is an unsigned integer. We use *, called value at address to print the value stored at a particular address.

```
Main()

{

Int num=5;

Printf("Address =%u",&num);

Printf("value = %d",num);

}
```

Pointer Declaration

Syntax: - Data_type *Pointer_name

Pointer_name gives the name of the pointer.

*tells that the variable is a pointer variable

Data_type is used to tell the compiler the data_type of the variable used.

J=&num gets the address of the num variable and store it in the j and j is another variable with another address we need to declare j and j is declared using pointer *j gives the value of the address variable

NULL POINTER

- To initialize a pointer variable when it is not assigned any valid address
- To pass null pointer to a function when we don't want any valid memory address
- To check for null pointers before accessing any pointer
- The value of NULL is 0

ARRAY

An array name are known as pointer constants in an array the value of the arr[] is stored in for this in array it is referd by the subscript notation

e.g.,printf("first %d",arr[0]) //prints the value of the first array similarly printf("first %d",*val); and printf("first %d",*(val+0));

characters and pointers

in this case it is simple if we are using pointer variable to point to the character string

e.g., char name[]="hello"; here name point to s

char *pname = "hello"; here pname points to s

in both these case the function is same and the pname is a pointer variable. The character pointer will only point to the first character in the string. The address of variable declared id stored in the pname

Array of integers

Int *ipara[25]; // 25 pointers to intregers

In here the address is called by the function argument %p and using &variablename[pointer];

Pointers passing function

While passing the value we use & before the variable name and while operating it in the function we use * operator with the variable.

DYNAMIC MEMORY ALLOCATION

In c if the allocated memory is insufficient for an array we use dynamic memory allocation. The library functions used are malloc(),calloc(),realloc() and free() it falls under the header file stdlib.h

malloc() memory allocation

It allocates emmory and leaves unintialized. it reverses a block of memory of special numbers of bytes, it returns a pointer of void which can be casted into pinters

Syntax: - ptr = (castype*)malloc(Size);

E.g.,ptr = (int*)malloc(100*sizeof(float));

The above statement allocate 400 bytes of memory. It's because the size of float is 4 bytes. The expression result in an NULL pointer if the memory cannot be allocated

calloc() contiguous allocation

It allocates memory and initialize all bit to zero

Syntax: - ptr = (castyope*)calloc(n,size);

e.g., ptr = (float*)calloc(25,sizeof(float));

this allocate contiguous space in memory for 25 elements of type float.

Free()

It allocates to release free space

e.g., free(ptr);

the statement free spaces for ptr

realloc()

if the previously allocated memory is insufficient or more than requires it can be changed

e.g., ptr =reralloc(ptr,c);

here ptr is reallocated with a new file size x

Important questions

1. **Define a pointer variable.**
   The pointer in C language is a variable which stores the address of another variable.
2. **Which are the operators used for defining pointer variable**?
   This variable can be of type int, char, array
3. **Advantages of using pointer variable.**
   - It reduces the complicity of a programme.
   - Increases the execution speed.
   - Useful in the case of passing multiple elements in functions.
   - Efficient in handling datas
4. **Describe the syntax for variable declaration, initialization and accessing of pointer variables.**

```
int a, *p;  // declaring the variable and pointer
  a = 10;
  p = &a;     // initializing the pointer

  printf("%d", *p);    //this will print the value of 'a'

  printf("%d", *&a);   //this will also print the value of 'a'

  printf("%u", &a);    //this will print the address of 'a'

  printf("%u", p);     //this will also print the address of 'a'

  printf("%u", &p);    //this will print the address of 'p
```

5. **Describe the syntax for variable declaration, initialization and accessing of pointer to an array.**

```
Int arr[]={2,3,4,5,6,};//declaration
Int *a,I;
Ptr =a;//initializes
For(i=0;i<5;i++)
        printf("arr[%d]: %d\n",loop,*(ptr+loop));
```

6. **Describe the syntax for variable declaration, initialization and accessing of array of pointers.**

```
  int  var[] = {10, 100, 200};//declaration
  int i, *ptr[MAX];//intialization

  for ( i = 0; i < MAX; i++) {
    ptr[i] = &var[i]; /* assign the address of integer. */
  }

  for ( i = 0; i < MAX; i++) {
    printf("Value of var[%d] = %d\n", i, *ptr[i] );
  }
```

7. **Differentiate between pointer to an array and array of pointers.**
   Pointer to an array is also known as array pointer. We are using the pointer to access the components of the array.

Syntax:- data type (*var name)[size of array];
“Array of pointers” is an array of the pointer variables. It is also known as pointer arrays.
Syntax:- int *var_name[array_size];

8. **What are null pointers?**
   - To initialize a pointer variable when it is not assigned any valid address
   - To pass null pointer to a function when we don’t want any valid memory address
   - To check for null pointers before accessing any pointer
   - The value of NULL is 0
9. **How do you interpret the following function declaration? int *p(char a[])**
   In this p is a pointer variable pointing to an character array a
10. **What is the difference between a character pointer and character array.**

Char a[10] = "geek";	Char *p = "geek";
1) a is an array	1) p is a pointer variable
2) sizeof(a) = 10 bytes	2) sizeof(p) = 4 bytes
3) a and &a are same	3) p and &p aren't same
4) geek is stored in stack section of memory	4) p is stored at stack but geek is stored at code section of memory

5) char a[10] = "geek";
a = "hello"; //invalid
> a, itself being an address and string constant is also an address, so not possible.
	5) char *p = "geek";
	p = "india"; //valid

6) a++ is invalid	6) p++ is valid

7) char a[10] = "geek";
a[0] = 'b'; //valid
	7) char *p = "geek";
	p[0] = 'k'; //invalid
	> Code section is r- only.

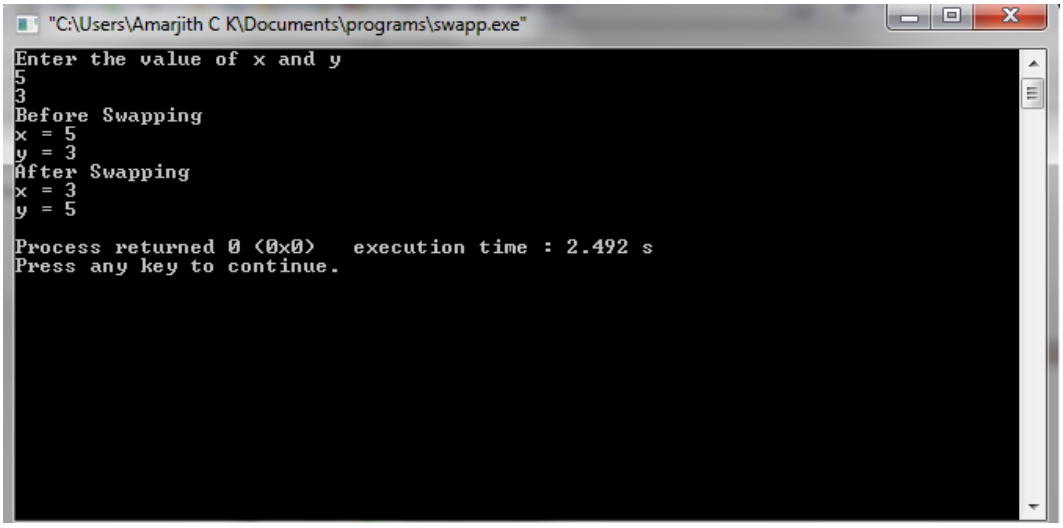
Github :- https://github.com/Amarjith-c-k/pointers

11. **Write a C program to swap the values of two variables using pointer.**

```
#include <stdio.h>

int main()
{
  int x, y, *a, *b, temp;

  printf("Enter the value of x and y\n");
  scanf("%d%d", &x, &y);

  printf("Before Swapping\nx = %d\ny = %d\n", x, y);

  a = &x;
  b = &y;

  temp = *b;
  *b = *a;
  *a = temp;
```

```
  printf("After Swapping\nx = %d\ny = %d\n", x, y);

  return 0;
}
```

OUTPUT

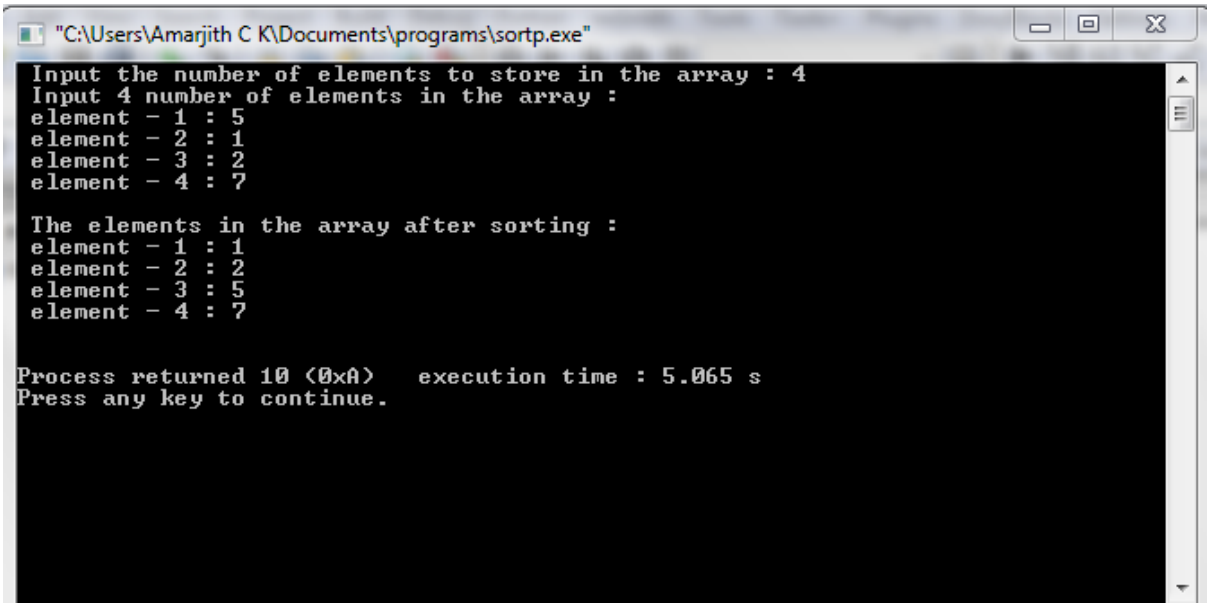


12. Write a C program using pointers to sort an array of integers by calling a function.

```
#include <stdio.h>
void main()
{
  int *a,i,j,tmp,n;
  printf(" Input the number of elements to store in the array : ");
  scanf("%d",&n);

  printf(" Input %d number of elements in the array : \n",n);
  for(i=0;i<n;i++)
    { printf(" element - %d : ",i+1);
      scanf("%d",a+i);                    }

  for(i=0;i<n;i++)
  {
   for(j=i+1;j<n;j++)
   {
     if( *(a+i) > *(a+j))
     {
    tmp = *(a+i);
    *(a+i) = *(a+j);
    *(a+j) = tmp;
     }
   }
  }
  printf("\n The elements in the array after sorting : \n");
```

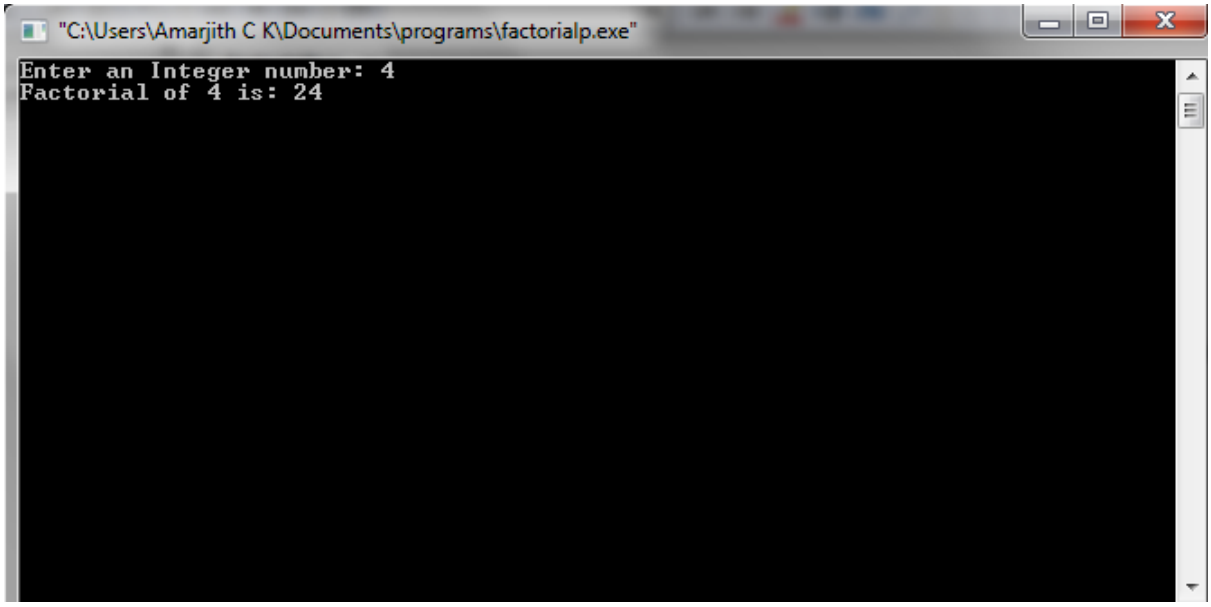
```
  for(i=0;i<n;i++)
    {
                                printf(" element - %d : %d \n",i+1,*(a+i));
                                                    }
printf("\n");
}
```

OUTPUT



13. Write a C program to find the factorial of a number using pointer.

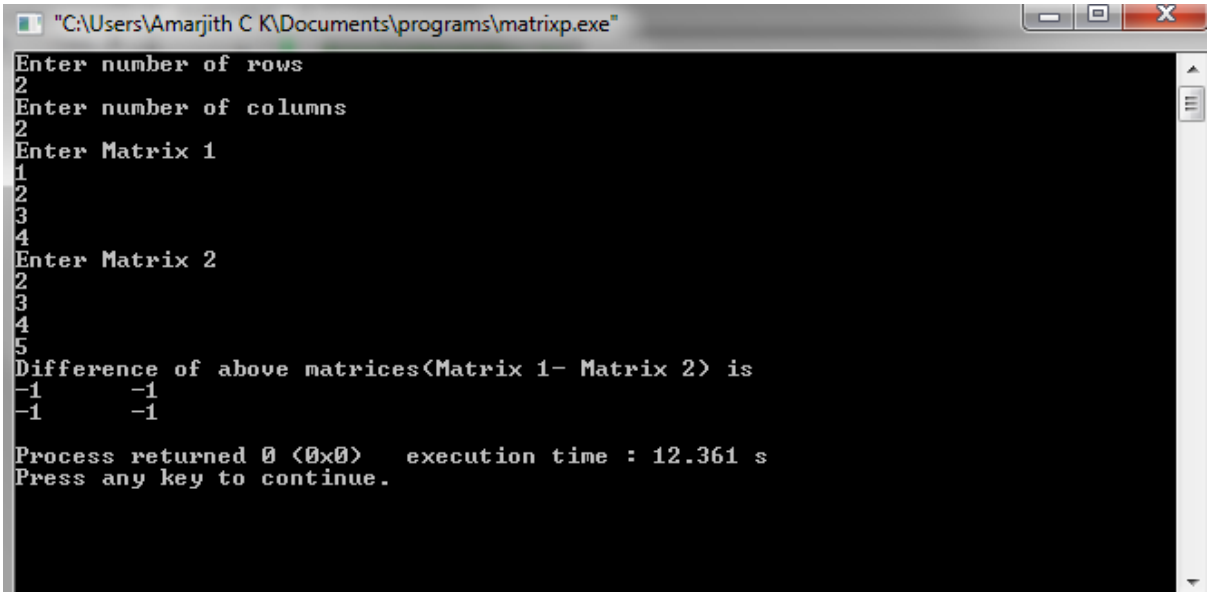
```
    #include<stdio.h>
void Factori(int,  long int *);
int main(){
       long int factorial;
       int numbr;
       printf("Enter an Integer number: ");
       scanf("%d",&numbr);
       Factori(numbr, &factorial);
       printf("Factorial of %d is: %ld", numbr, factorial);
       getch();

      return;
      }

void Factori(int n,  long int *factorial)
{
      int i;

     *factorial =1;
     for(i=1;i<=n;i++)
     {
     *factorial=*factorial*i;
     }
}
```



14. Write a C program using pointers to find the difference of two matrices A and B.

```
 #include<stdio.h>
int main()
{
 int i,j,rows,col;
 printf("Enter number of rows\n");
 scanf("%d",&rows);
  printf("Enter number of columns\n");
 scanf("%d",&col);

 int a1[rows][col],a2[rows][col],sub[rows][col];

 printf("Enter Matrix 1\n");
 for(i=0;i<rows;i++)
 {
  for(j=0;j<col;j++)
  {
   scanf("%d",(*(a1+i)+j));
  }
 }

  printf("Enter Matrix 2\n");
 for(i=0;i<rows;i++)
 {
  for(j=0;j<col;j++)
  {
   scanf("%d",(*(a2+i)+j));
  }
 }
 for(i=0;i<rows;i++)
```

```
{
 for(j=0;j<col;j++)
 {
  *(*(sub+i)+j)=*(*(a1+i)+j)-*(*(a2+i)+j);
 }
}

printf("Difference of above matrices(Matrix 1- Matrix 2) is\n");

 for(i=0;i<rows;i++)
{
 for(j=0;j<col;j++)
 {
  printf("%d\t",*(*(sub+i)+j));
 }

 printf("\n");
}

}
```



15. Distinguish between address stored in the pointer and value in that address. How does one pointer point to another pointer? Explain with examples.

The address stored in the pointer is called and used with dereferencing operator and the value is called by using both address of operator and dereferencing operators

Live Demo

```
#include <stdio.h>

int main () {
```

```
  int  var;
  int  *ptr;
  int  **pptr;

  var = 3000;

  /* take the address of var */
  ptr = &var;

  /* take the address of ptr using address of operator & */
  pptr = &ptr;

  /* take the value using pptr */
  printf("Value of var = %d\n", var );
  printf("Value available at *ptr = %d\n", *ptr );
  printf("Value available at **pptr = %d\n", **pptr);

  return 0;
}
```

OUTPUT

```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```