Practical -2
Strings, Array Pointers and function pointers

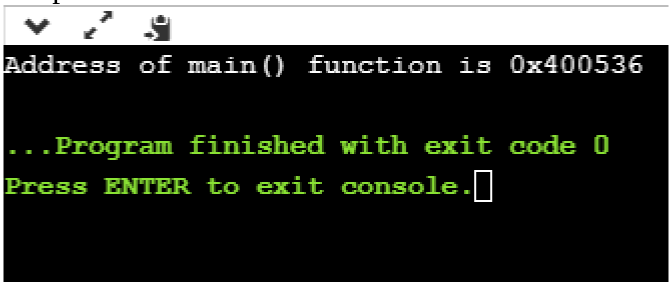
Function Pointer

As we know that we can create a pointer of any data type such as int, char, float, we can also create a pointer pointing to a function. The code of a function always resides in memory, which means that the function has some address. We can get the address of memory by using the function pointer.

ex.

```
#include <stdio.h>
int main()
{
    printf("Address of main() function is %p",main);
    return 0;
}
```

Output



```
Address of main() function is 0x400536

...Program finished with exit code 0
Press ENTER to exit console.
```

Declaration of a function pointer

Till now, we have seen that the functions have addresses, so we can create pointers that can contain these addresses, and hence can point them.

Syntax of function pointer

`1.return type (*ptr_name)(type1, type2…);`

For example:

`2.int (*ip) (int);`

In the above declaration, *ip is a pointer that points to a function which returns an int value and accepts an integer value as an argument.

`3.float (*fp) (float);`

Our next step is to assign the address of a function to the function pointer.

1. **`float (*fp) (int , int);`** // Declaration of a function pointer.
2. **`float func( int , int );`** // Declaration of function.
3. **`fp = func;`** // Assigning address of func to the fp pointer.

Calling an above function using a usual way is given below:

`result = func(a , b);` // Calling a function using usual ways.

Calling a function using a function pointer is given below:

`result = (*fp)( a , b);` // Calling a function using function pointer.

OR

`result = fp(a , b);`

Strings and Pointers
In C, a string can be referred to either using a character pointer or as a character array.
In the following code we are assigning the address of the string str to the pointer ptr.
char *ptr = str;

```
int main(void) {

  // string variable
  char str[6] = "Hello";

  // pointer variable
  char *ptr = str;

  // print the string
  while(*ptr != '\0') {
    printf("%c", *ptr);

    // move the ptr pointer to the next memory location
    ptr++;
  }

  return 0;
}
```

Array of pointers is an array whose elements are pointers to the base address of the string.It is declared and initialized as follows –

```
char *a[3 ] = {"one", "two", "three"};
//Here, a[0] is a ptr to the base add of the string "one"
//a[1] is a ptr to the base add of the string "two"
//a[2] is a ptr to the base add of the string "three"
```

Pointer to Arrays

The pointer can be used to access the array elements, accessing the whole array using pointer arithmetic, makes the accessing faster.

```
int arr[5] = {10, 20, 30, 40, 50};
int (*ptr)[5];
ptr = &arr;
```

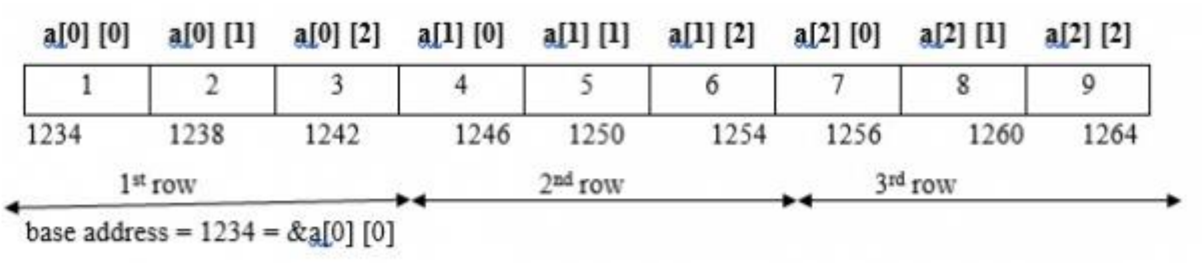
Once you store the address of the first element in 'p', you can access the array elements using *p, *(p+1), *(p+2) and so on

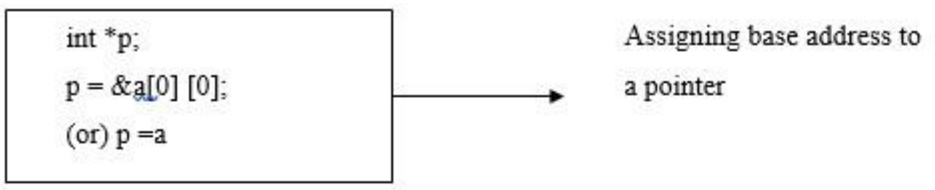
Pointer to 2D array
`(a + i)` points to ith 1-D array.

`*(a+i)` points to the base address of the ith 1-D array

`*(a + i)` points to the address of the 0th element of the 1-D array. So, `*(a + i) + 1` points to the address of the 1st element of the 1-D array `*(a + i) + 2` points to the address of the 2nd element of the 1-D array

*(a + i) + j points to the base address of jth element of ith 1-D array.
On dereferencing *(a + i) + j we will get the value of jth element of ith 1-D array.
*(*(a + i) + j)
By using this expression we can find the value of jth element of ith 1-D array.





❖ Pointer is used to access the elements of 2 – dimensional array as follows

a[i] [j] = *(p+i*columnsize+j)

```
#include<stdio.h>

int main()
{
    int arr[3][4] = {
                        {11,22,33,44},
                        {55,66,77,88},
                        {11,66,77,44}
                    };
    int i, j;
    for(i = 0; i < 3; i++)
    {
        printf("Address of %d th array %u \n",i , *(arr + i));
        for(j = 0; j < 4; j++)
        {
             printf("arr[%d][%d]=%d\n", i, j, *( *(arr + i) + j) );
        }
        printf("\n\n");
    }
    return 0;
}
```

Output

```
Address of 0 th array 2686736
arr[0][0]=11
arr[0][1]=22
arr[0][2]=33
arr[0][3]=44

Address of 1 th array 2686752
arr[1][0]=55
arr[1][1]=66
arr[1][2]=77
arr[1][3]=88

Address of 2 th array 2686768
arr[2][0]=11
arr[2][1]=66
arr[2][2]=77
arr[2][3]=44
```

Exercise

1) Write a C program to count vowels and consonants in a string using pointer
2) Write a C program to compare two strings using pointers.
3) Write a C program to copy a string to another without using strcpy function. Use pointers for both the strings.
4) Write a C Program to check whether the string entered by user is a palindrome string or not using pointer.
5) Write a C program to accept a string and a character from the user. Now using pointer delete all the occurrences of the character from the string and return the corrected string. Make it using user defined function.
6) Write a C program to copy one array to another using pointers. The size of both the arrays may be different.
7) Write a C program to create a function to accept a number and check whether it is even or odd. Call the function using function pointer
8) Create an array of 10 float numbers. Pass them to a function named average. Accept a number m from the user in the function. Calculate average of the first m numbers of the array and return to the caller.
9) Create a two dimensional array to store sales data of Quarter (Q1, Q2, Q3 and Q4) and for four companies (C1, C2, C3, C4). Accept the quarter and company from the user and display the sales figure. Use pointer to the 2D array and suitable pointer arithmetic.