PF LAB 09
Pointers to array
Void and double pointers

Pointers Arrays in C

Passing Array Elements to a Function

Array elements can be passed to a function by calling the function by value, or by reference.
In call by reference we pass addresses of array elements to the function.
/* Demonstration of call by value */

```
main( )
{
int i ;
int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
for ( i = 0 ; i <= 6 ; i++ )
display ( marks[i] ) ;
}
display ( int m )
{
printf ( "%d ", m ) ;
}
```

And here's the output...

```
55 65 75 56 78 78 90
```

/* Demonstration of call by reference */

```
main( )
{

 int i ;
 int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
for ( i = 0 ; i <= 6 ; i++ )
disp ( &marks[i] ) ;


}
 disp ( int *n )
```

```
{
printf ( "%d ", *n ) ;

 }
```

And here’s the output... 55 65 75 56 78 78 90

Pointers and Arrays

To be able to see what pointers have got to do with arrays, let us first learn some pointer arithmetic. Consider the following example:

```
main( )
{
 int i = 3, *x ;
 float j = 1.5, *y ;
 char k = 'c', *z ;
 printf ( "\nValue of i = %d", i ) ;
 printf ( "\nValue of j = %f", j ) ;
 printf ( "\nValue of k = %c", k ) ;
 x = &i ;
 y = &j ;
 z = &k ;
 printf ( "\nOriginal address in x = %u", x ) ;
 printf ( "\nOriginal address in y = %u", y ) ;
 printf ( "\nOriginal address in z = %u", z ) ;

 x++ ;
 y++ ;
 z++ ;
 printf ( "\nNew address in x = %u", x ) ;
 printf ( "\nNew address in y = %u", y ) ;
 printf ( "\nNew address in z = %u", z ) ;
}
```

What do you think what will be output if we increment pointer which point to value of integer, float and character?

Here is the output

```
Value of i = 3
Value of j = 1.500000
Value of k = c
Original address in x = 65524
Original address in y = 65520
Original address in z = 65519
New address in x = 65526
New address in y = 65524
 New address in z = 65520
```

Now let start with pointer to array, suppose we have an array.

num [] = { 24, 34, 12, 44, 56, 17 }.

24	34	12	44	56	17
65512	65514	65516	65518	65520	65522

The following figure shows how this array is located in memory.

Here is a program that prints out the memory locations in which the elements of this array are stored.

```
main( )
{
int num[ ] = { 24, 34, 12, 44, 56, 17 } ;
int i ;
for ( i = 0 ; i <= 5 ; i++ )
 {
printf ( "\nelement no. %d ", i ) ;
 printf ( "address = %u", &num[i] ) ;
 }
}
```

The output of this program would look like this:

```
  element no. 0 address = 65512
  element no. 1 address = 65514
  element no. 2 address = 65516
  element no. 3 address = 65518
 element no. 4 address = 65520
  element no. 5 address = 65522
```

Note that the array elements are stored in contiguous memory locations, each element occupying two bytes, since it is an integer.

Accessing array elements Without using subscripted variables

LET SEE AN EXAMPLE

```
main( )
{
 int num[ ] = { 24, 34, 12, 44, 56, 17 } ;
 int i, *j ;
 j = &num[0] ; /* assign address of zeroth element */
 for ( i = 0 ; i <= 5 ; i++ )
 {
 printf ( "\naddress = %u ", j ) ;
 printf ( "element = %d", *j ) ;
 j++ ; /* increment pointer to point to next location */
 }
  }
```

First we assign address of zeroth element in j On incrementing j it points to the next memory location of its type (that is location no. 65514). But location no. 65514 contains the second element of the array, therefore when the printf( ) statements are executed for the second time they print out the second element of the array and its address (i.e. 34 and 65514)... and so on till the last element of the array has been printed.

Passing an Entire Array to a Function using subscript variable

Let see example

```
main( )
{
 int num[ ] = { 24, 34, 12, 44, 56, 17 } ;
 dislpay ( &num[0], 6 ) ;
}
display ( int *j, int n )
{
        int i ;
for ( i = 0 ; i <= n - 1 ; i++ )
 {
printf ( "\nelement = %d", *j ) ;
j++ ; /* increment pointer to point to next element */
 }
}
```

void pointer in C

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.
Let us consider an example

```
int a = 10;
char b = 'x';

void *p = &a;  // void pointer holds address of int 'a'
p = &b;
```

Double Pointer (pointer to pointer)

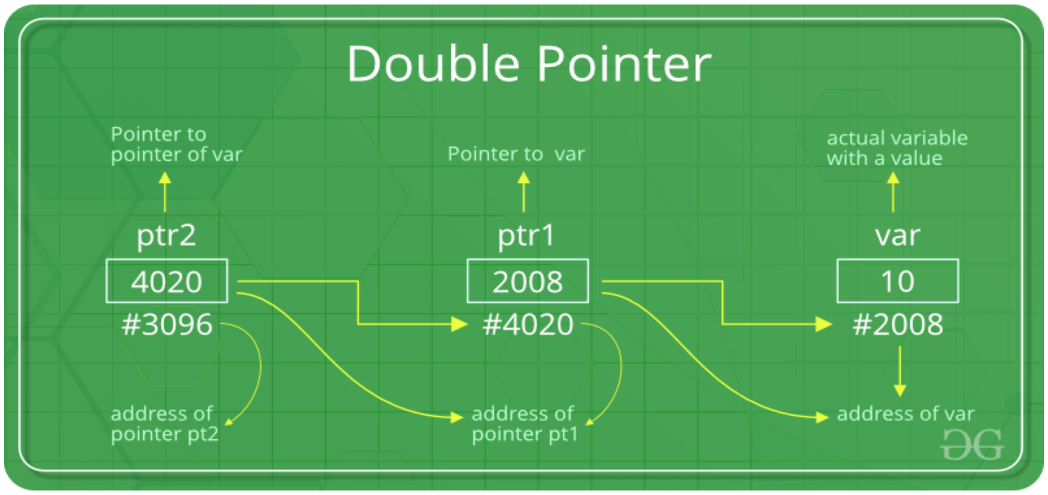
We already know that a pointer points to a location in memory and thus used to store the address of variables. So, when we define a pointer to pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as double pointers.

How to declare a pointer to pointer in C?

Declaring Pointer to Pointer is similar to declaring pointer in C. The difference is we have to place an additional '*' before the name of pointer.

Syntax:

int **ptr;   // declaring double pointers



Let us understand this more clearly with the help of the below program:

```
int main()
{
   int var = 789;

   // pointer for var
   int *ptr2;

   // double pointer for ptr2
   int **ptr1;

   // storing address of var in ptr2
   ptr2 = &var;
```

```
    // Storing address of ptr2 in ptr1
    ptr1 = &ptr2;

    // Displaying value of var using
    // both single and double pointers
    printf("Value of var = %d\n", var );
    printf("Value of var using single pointer = %d\n", *ptr2 );
    printf("Value of var using double pointer = %d\n", **ptr1);

  return 0;
}
```

Lab Task

Question # 01:
Write a function that print int, float and char values using void pointers but you have to declare a single void pointer variable. And its up to user which type of value it enters.

Question # 02:
Write a program to store n elements in an array and print the reverse elements of an array using pointer.

Question # 03:

Write a program to find separately the sum of the even and odd indexed elements of an array of size 10. Pass the even and odd elements to separate functions eg: sumEvenIndex(int*), sumOddIndex(int*) to carry out its sum. Also pass this array to a function called sortArray(int[]) and display the array elements into ascending order using pointer.

Question # 04:
Find the smallest number in an array using pointers.

Question # 05:
Write a single function that receives an array of 5 integers and returns the sum, average and standard deviation of these numbers without using return statement.

Call this function from main ( ) and print the results in main ( ).

$$\sigma = \sqrt{\frac{\Sigma(x - \mu)^2}{N}}$$

Where x represents each value in the population, µ is the mean value of the population, Σ is the summation (or total), and N is the number of values in the population.

Note: Use function type (call by reference)