Pointers and Arrays

```c
#include <stdio.h>
void display(int m){
	printf("\n%d",m);
}

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

//*OUTPUT*//
55
65
75
56
78
78
90
```

This is usually how we have been passing array elements to functions. We could also instead pass individual array element addresses to functions by replacing marks[i] with &marks[i] but will have to modify the display function to receive addresses instead of values.

```c
#include <stdio.h>
void display(int *m){
      printf("\n%d",*m);//value at address in the pointer m
}

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

/*OUTPUT*/

55

65

75

56

78

78

90
```

We now investigate what happens if we increment pointers of different types. It is recommended that you type this code yourself.

```c
#include <stdio.h>
int main( ) {
      int x=3,*X;
      long long y=4,*Y;
      char z='c',*Z;
      printf("\nValue of x=%d",x);
      printf("\nValue of y=%d",y);
      printf("\nValue of z=%c",z);
      X=&x;
      Y=&y;
      Z=&z;
      printf("\nOriginal Value in X=%u",X);
      printf("\nOriginal Value in Y=%u",Y);
      printf("\nOriginal Value in Z=%u",Z);
      X++;
      Y++;
      Z++;
      printf("\nNew Value in X=%u",X);
      printf("\nNew Value in Y=%u",Y);
      printf("\nNew value in Z=%u",Z);
      return 0;
}/*OUTPUT*/

Value of x=3

Value of y=4

Value of z=c

Original Value in X=6422020

Original Value in Y=6422008

Original Value in Z=6422007

New Value in X=6422024

New Value in Y=6422016

New value in Z=6422008
```

As we can see every time a pointer is incremented by 1,it points to the immediately next location of its type. So,if an integer pointer is incremented by 1,it will point to 4 addresses after the current location to point to next integer, since an integer is 4 bytes long .Similarly, if a char pointer is incremented by 1,it will point to 1 address after the current location to point to next char, since a char is 1 byte long. We can exploit "pointer incrementing" in arrays, since elements in arrays are always stored in contiguous memory locations.

```
#include <stdio.h>
int main( ) {
      int num[]={24,34,12,44,56,17};
      int i=0,*j;
      j=&num[0];
      while(i<=5){
            printf("address=%u of",j);
            printf(" element[%i]=%d\n",i,*j);
            i=i+1;
            j=j+1;
      }
      return 0;
}

/*OUTPUT*/

address=6422000 of element[0]=24

address=6422004 of element[1]=34

address=6422008 of element[2]=12

address=6422012 of element[3]=44

address=6422016 of element[4]=56

address=6422020 of element[5]=17
```

So we have to just get the address of the first element &array[0] and we can access the next element addresses and thus next array elements themselves by incrementing the address of first element. Also we can give address of first element of array to a function and the function can get the rest of the elements by incrementing the address of the first element of array.

```
#include <stdio.h>
int main( ) {
      int num[]={24,34,12,44,56,17};
      int i=0;
      while(i<=5){
            printf("\nAddress =%u ",&num[i]);
            printf("element=%d ",num[i]);
            printf("element=%d ",*(&num[0]+i));
            printf("element=%d\n",*(num+i));
            i++;
      }
      return 0;
}
/*OUTPUT*/

Address =6422016 element=24 element=24

Address =6422020 element=34 element=34

Address =6422024 element=12 element=12

Address =6422028 element=44 element=44

Address =6422032 element=56 element=56

Address =6422036 element=17 element=17
```

Notice the last two printf statements in while loop.Second last printf statement was expected but the last printf statement is unexpected.It turns out &num[0] =num.By just mentioning the array name we get the address of the first element of array.

Two dimensional array

We declare 2D array and access individual elements like this:

```
#include <stdio.h>
int main( ) {
      int stud[5][2]={
            {1234,56},
            {1212,33},
            {1434,80},
            {1312,78},
            {1203,75}
      };
      int i,j;
      for(i=0;i<=4;i++){
            printf("\n");
            for(j=0;j<=1;j++){
                  printf("%d ",stud[i][j]);
            }
      }
      return 0;
}
```

Now what can pointer incrementing do for us in 2D array?

```c
#include <stdio.h>
int main( ) {
      int stud[5][2]={
            {1234,56},
            {1212,33},
            {1434,80},
            {1312,78},
            {1203,75}
      };
      int i,j;
      for(i=0;i<=4;i++){
            printf("\nAddress of %dth 1-D array=%u ",i,stud+i);
            printf("has first element=%d\n",*(stud+i));
      }
      return 0;
}
/*OUTPUT*/


Address of 0th 1-D array=6422000 has first element=6422000


Address of 1th 1-D array=6422008 has first element=6422008


Address of 2th 1-D array=6422016 has first element=6422016


Address of 3th 1-D array=6422024 has first element=6422024


Address of 4th 1-D array=6422032 has first element=6422032
```

Observe that first printf statement of for loop.By incrementing the first element of 2D array(&stud[0]=stud) we get the next 1D array since a 2D array has 1D array as its elements. What about the second printf statement? Why are we getting addresses and not values?As 2D array has 1D array as its elements so *(stud+i) gives us the ith element.But ith element of 2D array is a 1D array so it will give the base address of the ith 1D array.We need another operator value at address(*) operator to get a value,which happens to be first element of ith 1D array.Observe next program:

```
#include <stdio.h>
int main( ) {
      int stud[5][2]={
            {1234,56},
            {1212,33},
            {1434,80},
            {1312,78},
            {1203,75}
      };
      int i,j;
      for(i=0;i<=4;i++){
            printf("\nAddress of %dth 1-D array=%u ",i,stud+i);
            printf("has first element=%d\n",*(*(stud+i)));
      }
      return 0;
}/*OUTPUT*/

Address of 0th 1-D array=6422000 has first element=1234
```

Notice the last printf statement of for loop in the above program.

```c
#include <stdio.h>
int main( ) {
      int stud[5][2]={
            {1234,56},
            {1212,33},
            {1434,80},
            {1312,78},
            {1203,75}
      };
      int i,j;
      printf("\nAddress of %d the 1-D array=%u",i,*(*(stud+2)+1));
      return 0;
}/*OUTPUT*/

Address of 0 the 1-D array=80
```

Question: print 75,1434 in the 2D array of above program using pointer notation. Modify the above program to find out.

We print the whole 2D array or matrix using pointer notation.

```c
#include <stdio.h>
int main( ) {
      int stud[5][2]={
            {1234,56},
            {1212,33},
            {1434,80},
            {1312,78},
            {1203,75}
      };
      int i,j;

      for(i=0;i<=4;i++){
            printf("\n");
            for(j=0;j<=1;j++){
                  printf("%d ",*(*(stud+i)+j));
            }
      }
      return 0;
}
```