Data Structure

Lecture 4

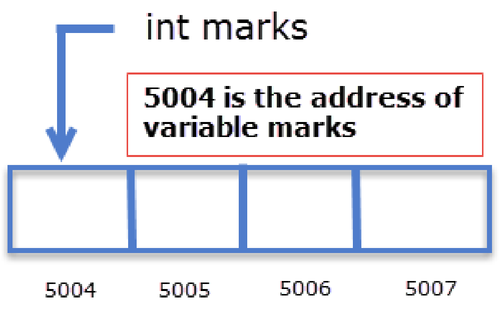
By : Dr.Nahla Bishri

Agenda

- Introduction to pointers
- Pointers and 1D arrays

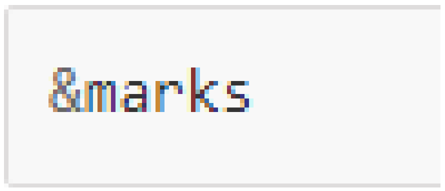
What is a Pointer?

- A pointer is a variable used to store a memory address.



Address Operator (&)

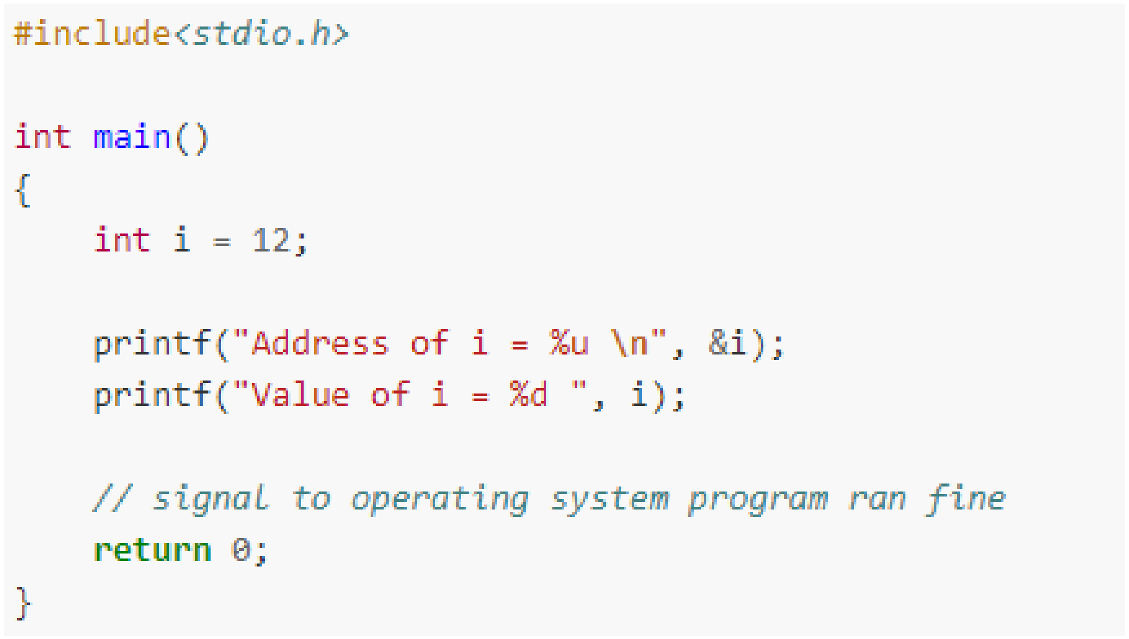
- To find the address of a variable, C provides an operator called address operator &.



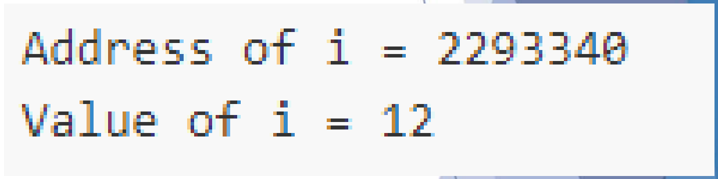
```
&marks
```

- Address of operator can't be used with constants or expression; it can only be used with a variable.

Example

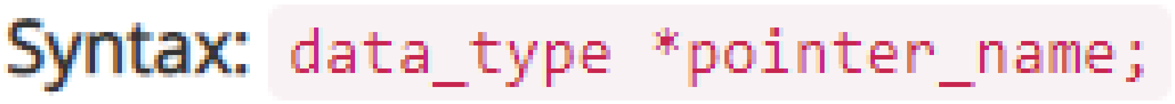


```c
#include<stdio.h>

int main()
{
    int i = 12;

    printf("Address of i = %u \n", &i);
    printf("Value of i = %d ", i);

    // signal to operating system program ran fine
    return 0;
}
```

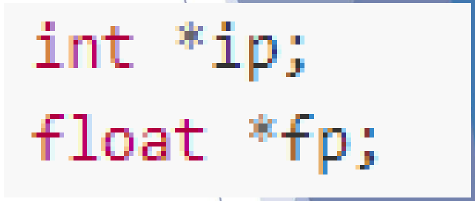


```
Address of i = 2293340
Value of i = 12
```

Declaring Pointer Variables



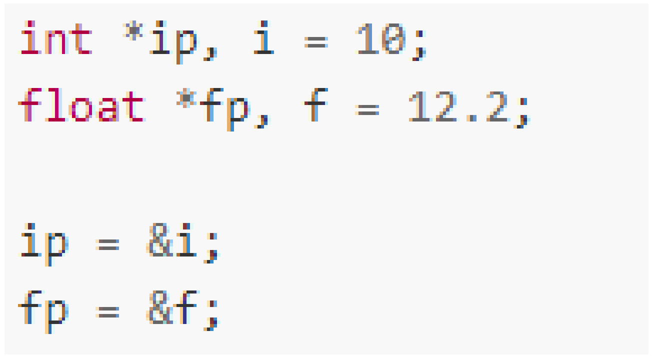
Syntax:
```
data_type *pointer_name;
```



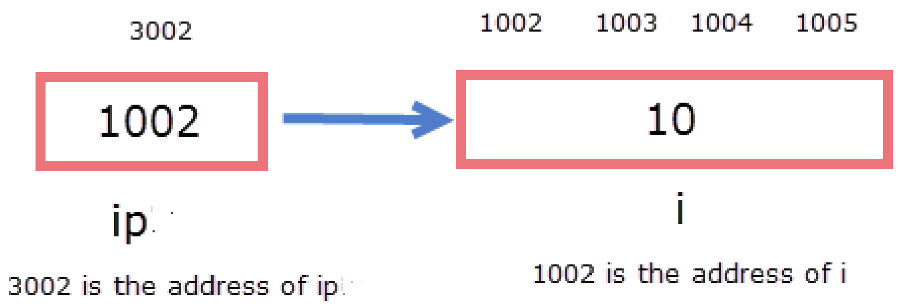
```
int *ip;
float *fp;
```

- A pointer variable ip can store the address of variables of type int .
- The pointer variable fp can only store the address of variable of type float.
- The type of variable (also known as base type) ip is a pointer to int and type of fp is a pointer to float.

Assigning Address to Pointer Variable



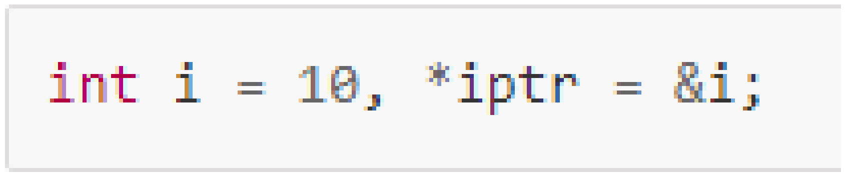
```
int *ip, i = 10;
float *fp, f = 12.2;

ip = &i;
fp = &f;
```



ip pointing to address of i

Dr.Nahla Bishri

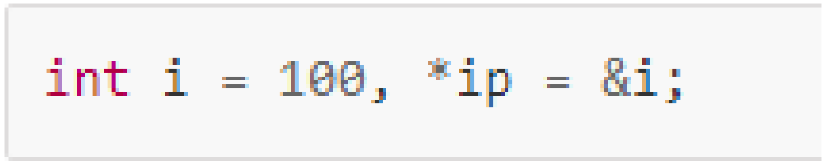
- We can initialize the pointer variable at the time of declaration, but in this case, the variable must be declared and initialized before the pointer variable.



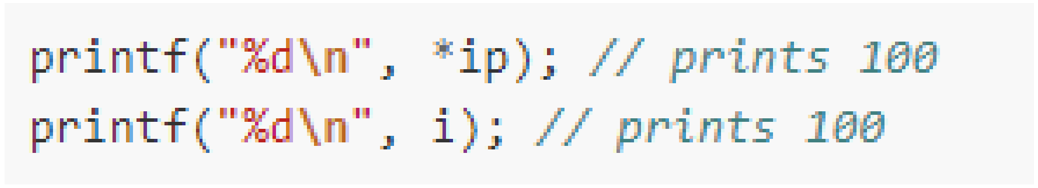
```
int i = 10, *iptr = &i;
```

Dereferencing Pointer Variable

- Dereferencing a pointer variable simply means accessing data at the address stored in the pointer variable.



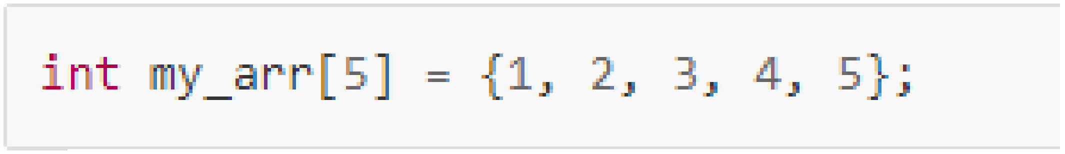
```c
int i = 100, *ip = &i;
```



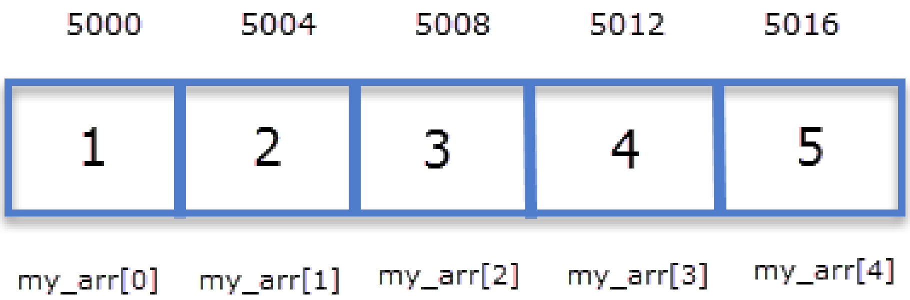
```c
printf("%d\n", *ip); // prints 100
printf("%d\n", i); // prints 100
```

Pointers and 1-D arrays

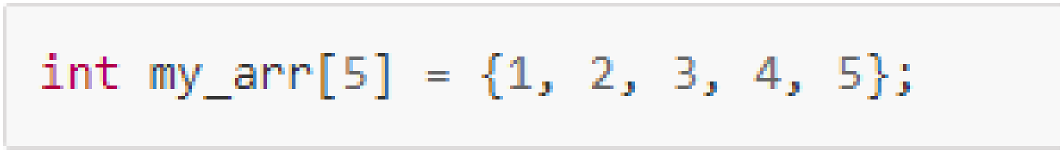
- The elements of an array are stored in contiguous memory locations.



```
int my_arr[5] = {1, 2, 3, 4, 5};
```



- The name of the array is a constant pointer that points to the address of the first element of the array or the base address of the array.
- We can use subscript notation (i.e using square brackets) to find the address of the elements of the array.



```
int my_arr[5] = {1, 2, 3, 4, 5};
```

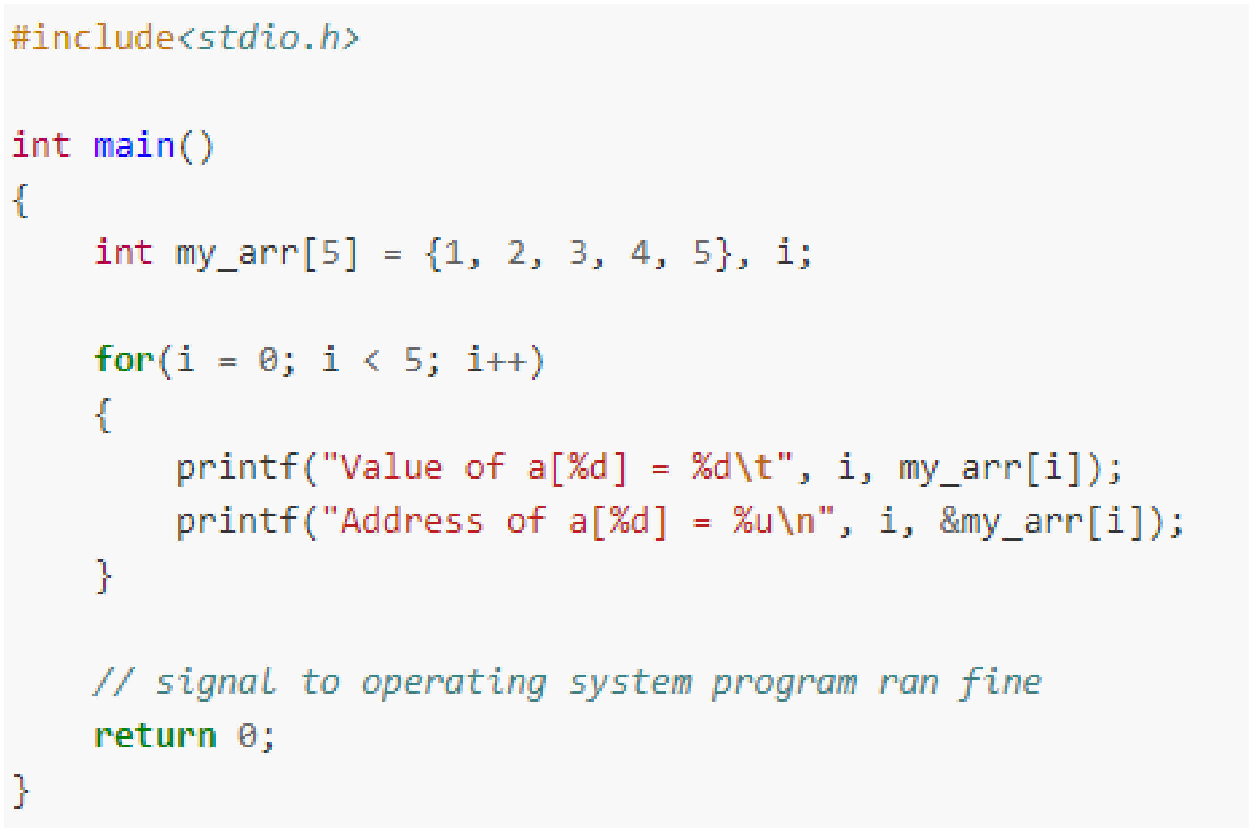
&my_arr[0] points to the address of the first element of the array .

my_arr and &my_arr[0] represent the same address

Example

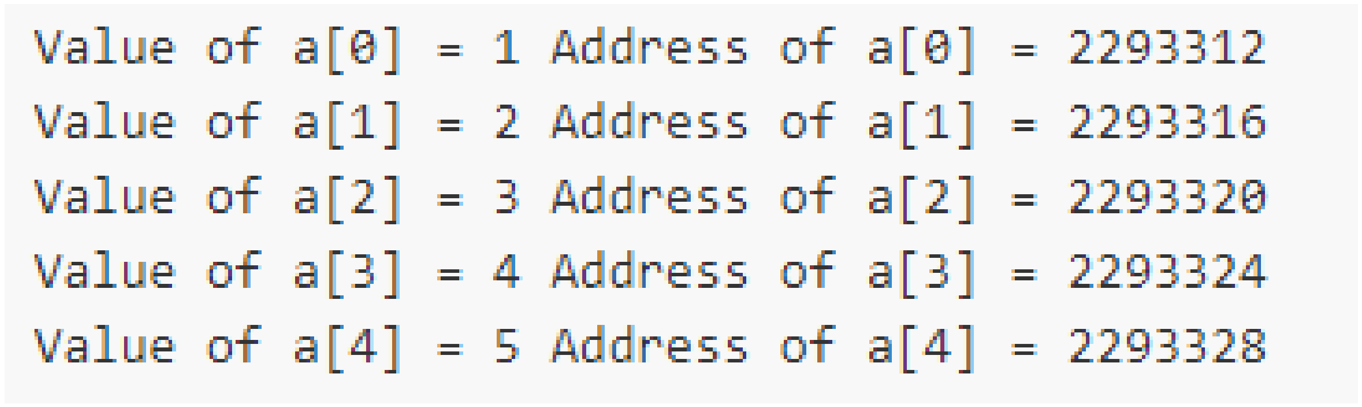
Write a program demonstrates that the elements of an array are stored in contiguous memory locations.

Example

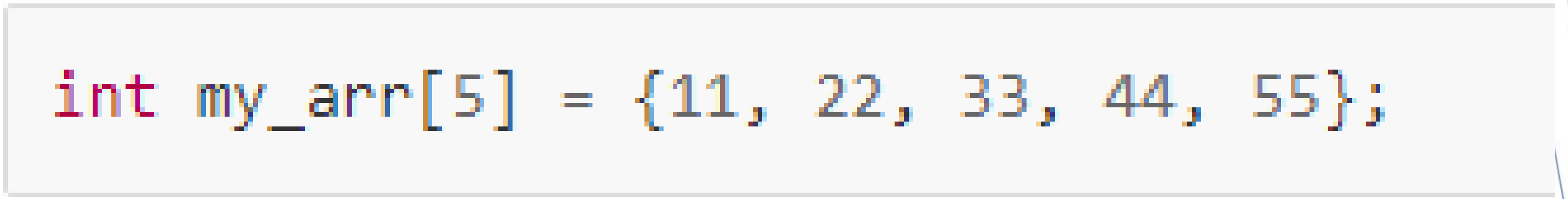


```
#include<stdio.h>

int main()
{
    int my_arr[5] = {1, 2, 3, 4, 5}, i;

    for(i = 0; i < 5; i++)
    {
        printf("Value of a[%d] = %d\t", i, my_arr[i]);
        printf("Address of a[%d] = %u\n", i, &my_arr[i]);
    }

    // signal to operating system program ran fine
    return 0;
}
```

output


```
Value of a[0] = 1 Address of a[0] = 2293312
Value of a[1] = 2 Address of a[1] = 2293316
Value of a[2] = 3 Address of a[2] = 2293320
Value of a[3] = 4 Address of a[3] = 2293324
Value of a[4] = 5 Address of a[4] = 2293328
```

Using pointers to access elements and address of elements in an array



```c
int my_arr[5] = {11, 22, 33, 44, 55};
```

my_arr is same as &my_arr[0]
my_arr + 1 is same as & my_arr[1]
my_arr + 2 is same as & my_arr[2]
my_arr + 3 is same as & my_arr[3]
my_arr + 4 is same as & my_arr[4]

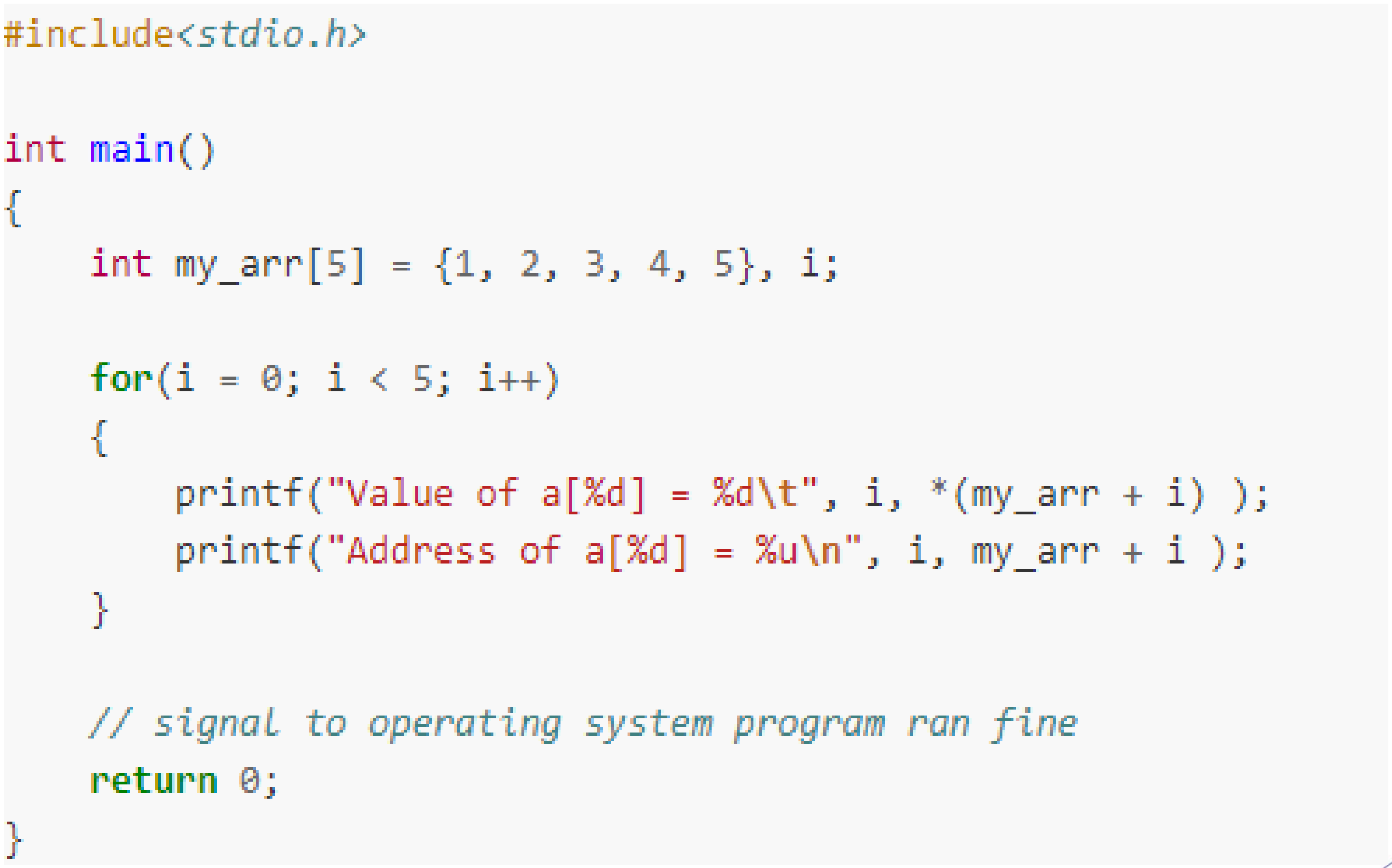
In general (my_arr + i) is same as writing & my_arr[i].

```
*(my_arr) is same as my_arr[0]
*(my_arr + 1) is same as my_arr[1]
*(my_arr + 2) is same as my_arr[2]
*(my_arr + 3) is same as my_arr[3]
*(my_arr + 4) is same as my_arr[4]
```

In general, *(my_arr+i) is same as writing my_arr[i].

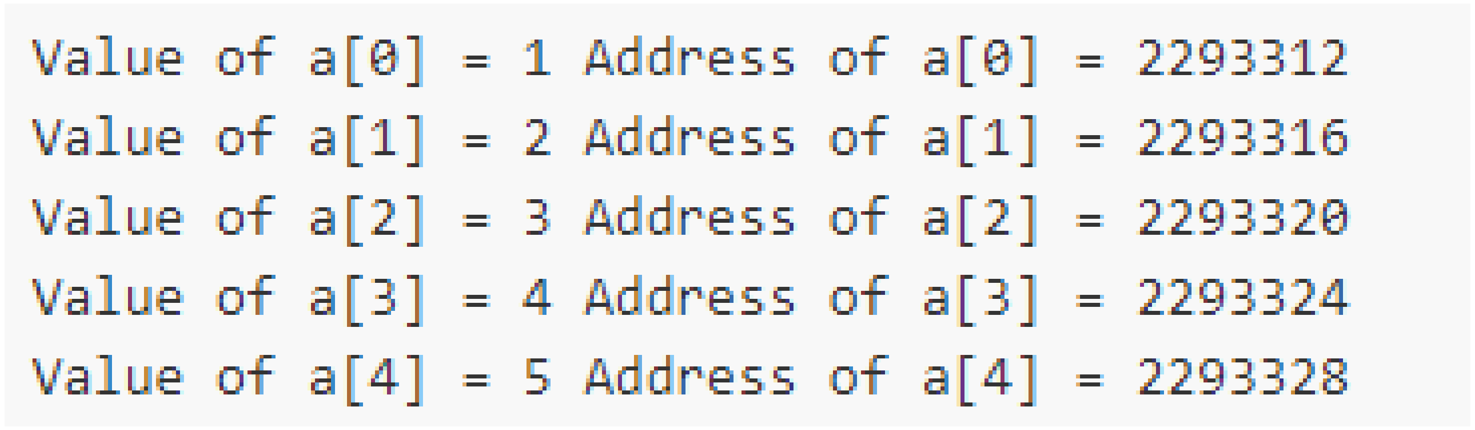
Example:

Zulwh#d#surjudp#wkdw#sulqwv#ydoxh#dqg#
dgguhvv#ri#duud|#hohphqw#xvlqj#srlqwhu1



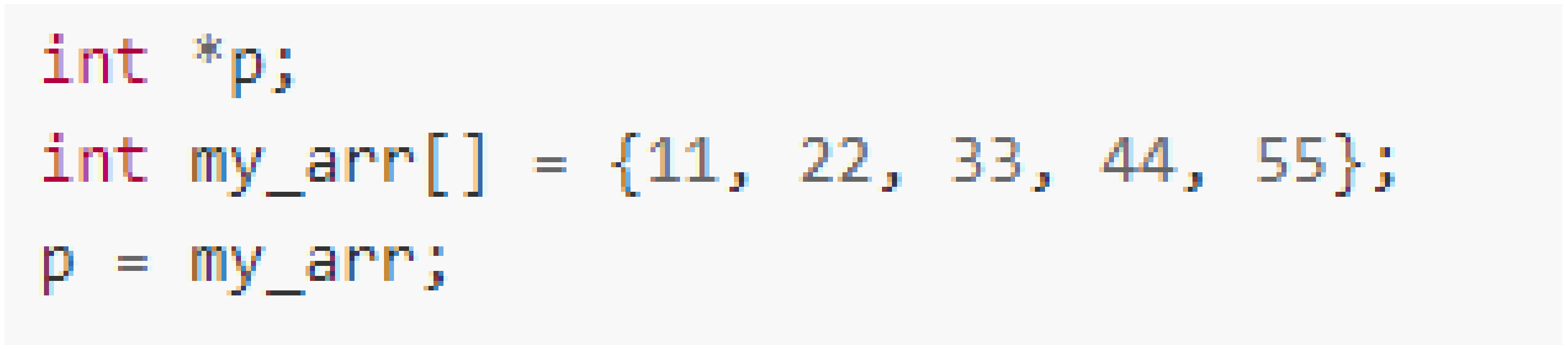
```c
#include<stdio.h>

int main()
{
    int my_arr[5] = {1, 2, 3, 4, 5}, i;

    for(i = 0; i < 5; i++)
    {
        printf("Value of a[%d] = %d\t", i, *(my_arr + i) );
        printf("Address of a[%d] = %u\n", i, my_arr + i );
    }

    // signal to operating system program ran fine
    return 0;
}
```

Output :



```
Value of a[0] = 1 Address of a[0] = 2293312
Value of a[1] = 2 Address of a[1] = 2293316
Value of a[2] = 3 Address of a[2] = 2293320
Value of a[3] = 4 Address of a[3] = 2293324
Value of a[4] = 5 Address of a[4] = 2293328
```

Assigning 1-D array to a Pointer variable



```
int *p;
int my_arr[] = {11, 22, 33, 44, 55};
p = my_arr;
```

- you can use pointer p to access address and value of each element in the array.

In general :
(p+i) denotes the address of the ith element and
*(p+i) denotes the value of the ith element.

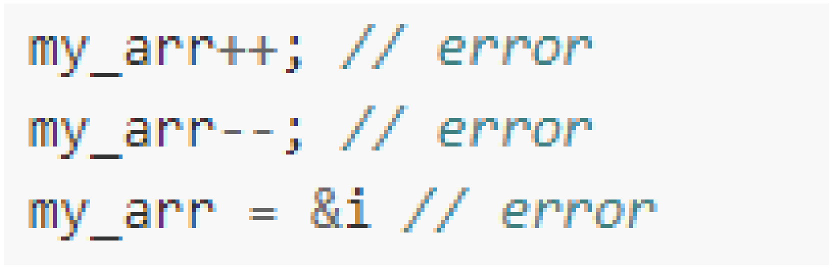
Differences between the name of the array and pointer variable:

- There are some differences between the name of the array (i.e my_arr) and pointer variable (i.e p):

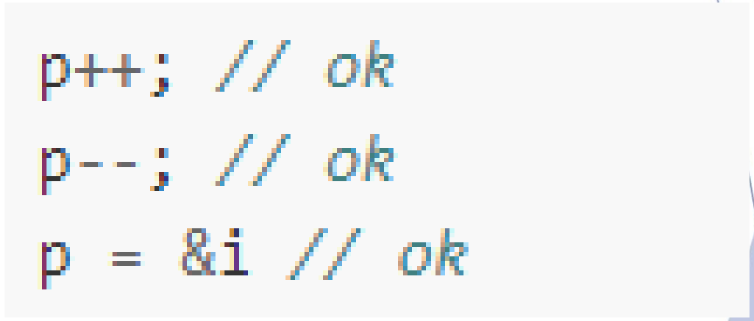
- The name of the array is a constant pointer hence you can't alter it to point to some other memory location.

- You can’t assign some other address to it nor you can apply increment/decrement operator like you do in a pointer variable.

- p is an ordinary pointer variable, so you can apply pointer arithmetic and even assign a new address to it.



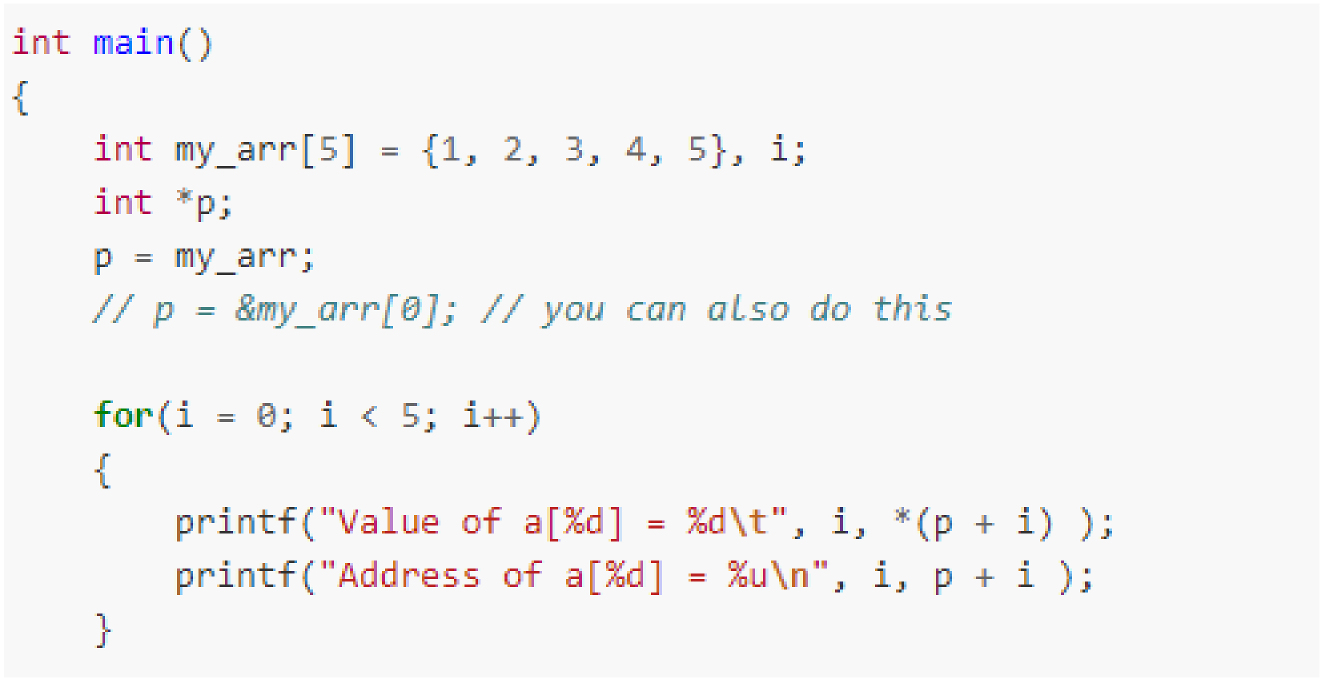
```
my_arr++; // error
my_arr--; // error
my_arr = &i // error
```



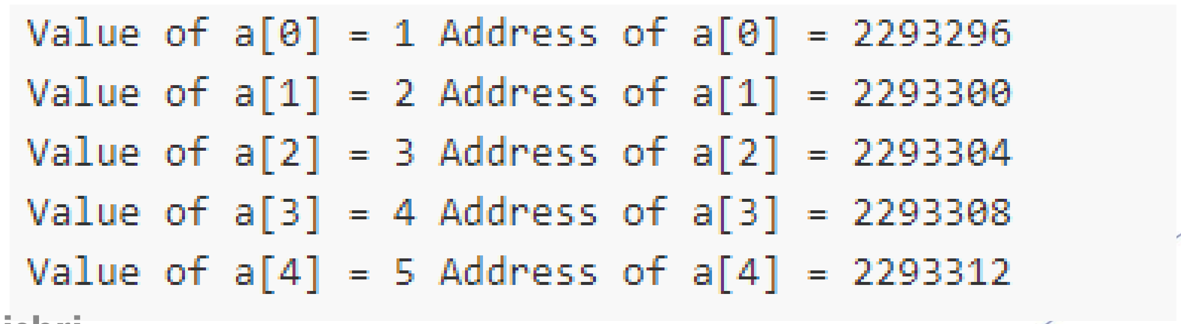
```
p++; // ok
p--; // ok
p = &i // ok
```

Example:

Zulwh#d#surjudp#wr#ghprqvwudwh#krz#|rx#
fdq#dffhvv#ydoxhv#dv#dgguhvv#ri#hohphqw#ri#
d#40G#duud|#e|#dvvljqlqj#lw#wr#d#srlqwhu#
ydduldeoh1



```c
int main()
{
    int my_arr[5] = {1, 2, 3, 4, 5}, i;
    int *p;
    p = my_arr;
    // p = &my_arr[0]; // you can also do this

    for(i = 0; i < 5; i++)
    {
        printf("Value of a[%d] = %d\t", i, *(p + i) );
        printf("Address of a[%d] = %u\n", i, p + i );
    }
```



```
Value of a[0] = 1 Address of a[0] = 2293296
Value of a[1] = 2 Address of a[1] = 2293300
Value of a[2] = 3 Address of a[2] = 2293304
Value of a[3] = 4 Address of a[3] = 2293308
Value of a[4] = 5 Address of a[4] = 2293312
```

Thank you