



C Programming

Lecture Six

Pointers in C

*This material is developed by IMTSchool for educational use only  
All copyrights are reserved*

## Introduction To Pointers

Pointer is a data type used to hold an address of another variable. Pointers are one of the most important advantages of C programming.

### Syntax:

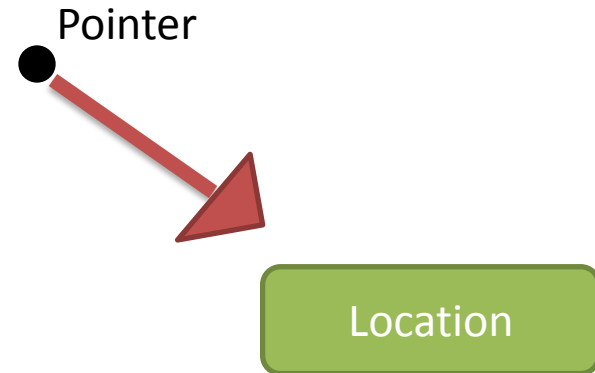
Pointee\_Type \*Pointer\_Name;

Or

Pointee\_Type\* Pointer\_Name;

Or

Pointee\_Type \* Pointer\_Name;



### Example

int \*ptr ;

int\* ptr ;

int \* ptr;

No difference between the 3 syntax. All of them define a pointer that points to *int* type

## The Address-Of Operator

The address-of operator is a unary operator represented by the symbol &. It gives the address of any variable if it is preceded the variable name.

### Example:

```
int x= 10;
int* ptr;
ptr = &x;
```

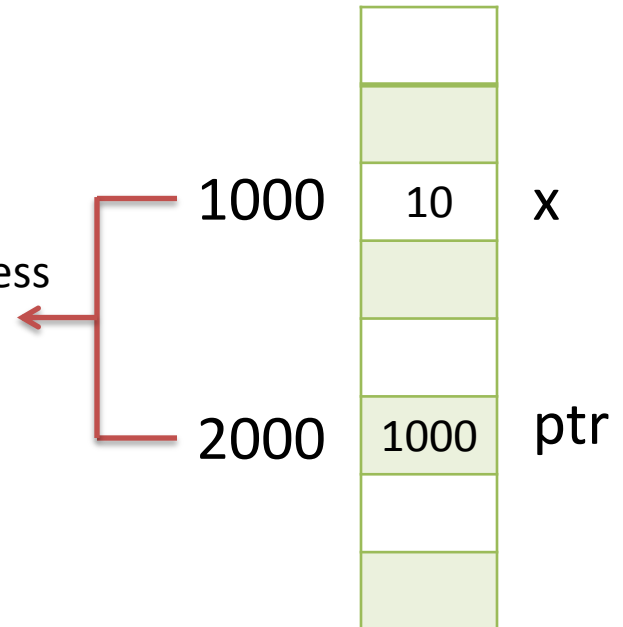
Define int named x and  
initialize it with value 10

Define pointer to int named ptr

Initialize ptr with the address of x

Memory Address

Memory



### Note

keep in mind that the pointer itself is a variable and has an address in the memory like any variable. The only difference is that the pointer holds an address of another variable

## The dereference operator

The dereference operator is a unary operator represented by the symbol \*. It operates on a pointer variable, and gives the ability to access the variable that the pointer points to.

### Example:

```
int x= 10;
int* ptr;
ptr = &x;
```

Define int named x and  
initialize it with value 10

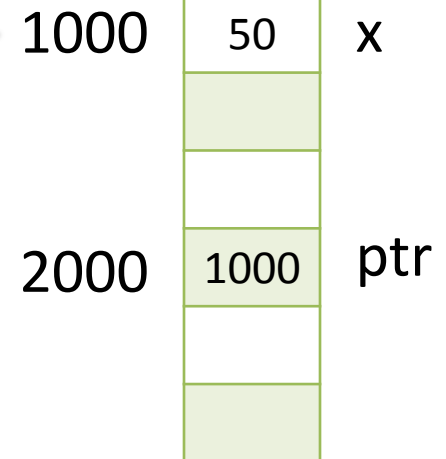
Define pointer to int named ptr

Initialize ptr with the address of x

```
*ptr = 50;
```

*This line means go to ptr, find out what is inside it and consider its address and jump to it, then assign 50*

Memory



## Expected Output

Write a C code defines an int initialized with 10, then print it, then define a pointer that points to that int and change the value of the int through the pointer to be 20, then print it again.

```
X before change is: 10  
X after change is: 20
```

# Time To Code



# Pass by value vs. pass by address

```
int z;
```

## Pass by Value

means you are making a copy in memory of the actual parameter's value that is passed in, a copy of the contents of the actual parameter.

```
void ByValValue_func(int x);
```

← Function takes an argument by value

```
ByValue_func(z);
```

← Pass the variable z by value

## Pass by Address

Sometimes called pass by reference, means you are making a copy in memory of the address of the actual parameter.

```
void ByAddress_func(int *ptr);
```

← Function takes an address as argument

```
ByAddress_func(&z);
```

← Pass the variable z by address

### Expected Output

```
Please Enter Value 1: 10  
Please Enter Value 2: 30  
The result is: 40
```

# Time To Code



### Expected Output

```
Please Enter Value 1: 15  
Please Enter Value 2: 5  
The result of Summation is: 20  
The result of Subtraction is: 10
```

Write a C code that ask the user to enter 2 values and save them in two variables, then the program sends these variables to function that calculates the summation and subtraction of them, the function returns the 2 results and return them in two pointers received as input arguments . The program then print the 2 results.

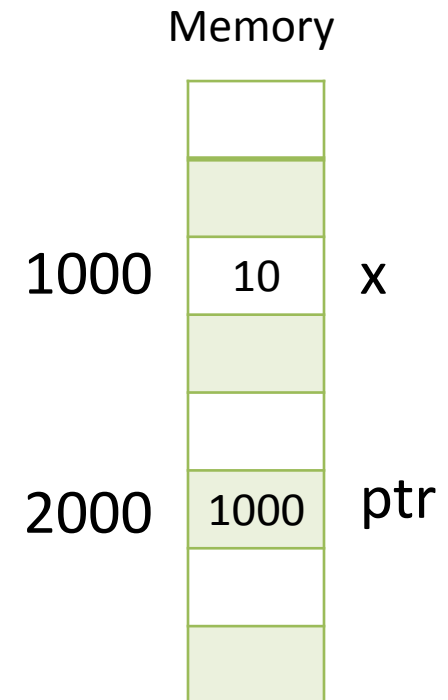
# Time To Code





What will be the output of the following lines:

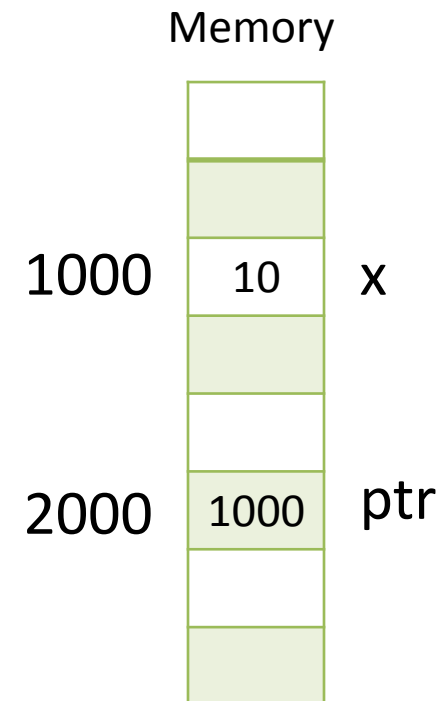
- `printf ("%d", x) ;`
- `printf ("%p", &x) ;`
- `printf ("%p", ptr) ;`
- `printf ("%d", *ptr) ;`
- `printf ("%p", &ptr) ;`



Note: %p used to print an address

What will be the output of the following lines:

- a. `printf ("%d", x);`  $\xrightarrow{10}$
- b. `printf ("%p", &x);`  $\xrightarrow{1000}$
- c. `printf ("%p", ptr);`  $\xrightarrow{1000}$
- d. `printf ("%d", *ptr);`  $\xrightarrow{10}$
- e. `printf ("%p", &ptr);`  $\xrightarrow{2000}$



## Operations on Pointers

### 1- Increment

Increments the pointer value by one step ( The step is the size of the pointee)

#### Example

```
prt ++; /* Assuming int is 4 bytes */
        /* prt now = 1004 */
```

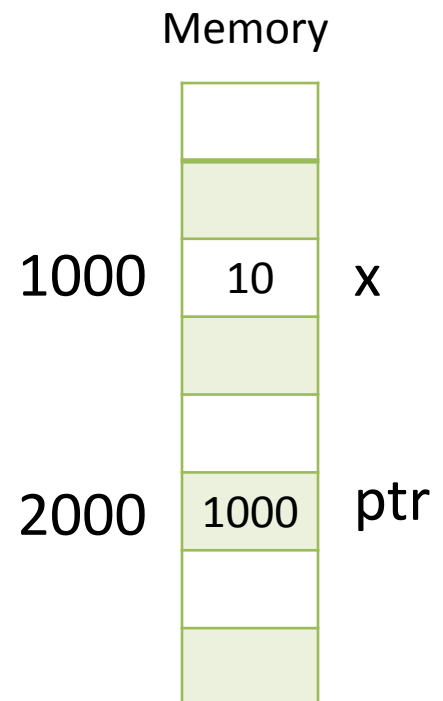
### 2- Decrement

Decrements the pointer value by one step

#### Example

```
prt --; /* Assuming int is 4 bytes */
        /* prt now = 996 */
```

```
int x= 10;
int* ptr;
ptr = &x;
```



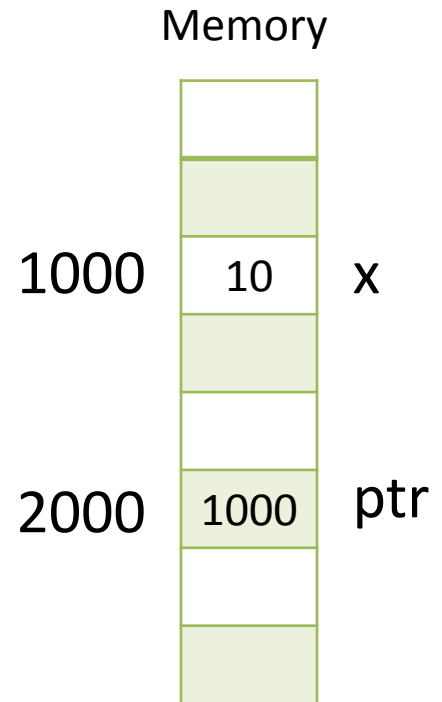
# Operations on Pointers

## 3- Subtraction

a- Subtract value from the pointer

*Example*

```
ptr = ptr - 4; /* Assuming int is 4 byte */
              /* ptr now = .984          */
```



b- Subtract pointer from pointer

*Example*

```
int x = ptr1 - ptr2; /* Assuming ptr1 = 2000 and ptr2 = 1000 */
                    /* Both of them are pointers to int which is 4 byte */
                    /* ptr1 - ptr2 returns number of steps between them */
                    /* difference between them is a value = 250          */
```

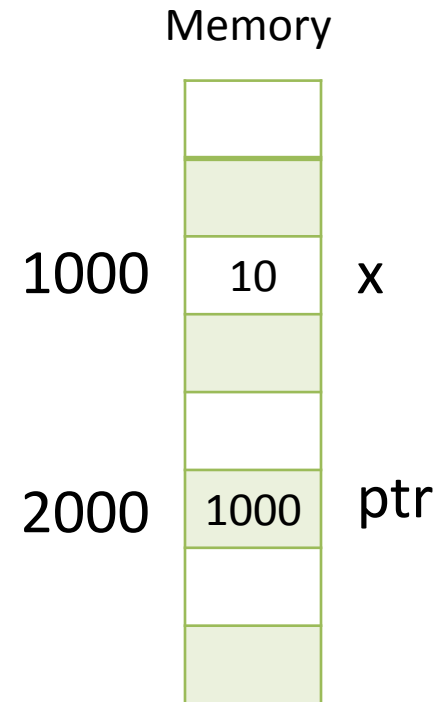
## Operations on Pointers

### 4- Addition

a- Add value to the pointer

#### *Example*

```
ptr = ptr + 4; /* Assuming int is 4 byte */
              /* ptr now = 1016          */
```



b- Add pointer to pointer

*Not Allowed, compilation Error*

Write a C code that calculates the summation of an array using a pointer on its first element.

**Notes:**

1- Name of the array is the address of the first element in it.

i.e. `int arr[10];`      `/* arr is the same as &arr[0] */`

2- All elements of the array are saved sequentially in the memory.

i.e. if you have a pointer to the first element in the array, incrementing the pointer makes the pointer point to the second element.

# Time To Code



## Passing array to function

**Remember that** *name of the array is the address of its first element.*

***i.e.*** if we have an array:

```
int arr[10];
```

Then `arr` is the same as `&arr[0]`.

**Which means** if you passed the array name `arr` to function called `func` for example, then you are passing address to int. So, the function `func` prototype should be declared as:

```
void func ( int *ptr);
```

Now, inside the function `func`, `*ptr` means that you are accessing the first element of the array, `*(ptr+1)` means that you are accessing the second element of the array, and so on ...

## Subscriptor Vs Dereference Operators

The subscriptor operator ( [ ] ) which used before with the array can be used also with pointer !


if we have a pointer

`int *ptr;`

**Then** ptr[0] is the same as \*ptr

**Also,** ptr[1] is the same as \*(ptr+1)

**Also,** ptr[10] is the same as \*(ptr+10).

What is the output of  
this code ... ? 

```
#include <stdio.h>

void func(int * ptr);

int arr[10] = {1,2,3,4,5,6,7,8,9,10};

void main(void)
{
    func(arr);
}

void func(int * ptr)
{
    for (i=0;i<10;i++)
    {
        printf("%d\n",ptr[i]);
    }
}
```



## Do You Like Math ... ?



Write a C code that define 2 arrays, and send them to a function that apply scalar multiplication between the two arrays and return the result, the main function then will print the result.

$$(a_1 \quad a_2 \quad \cdots \quad a_n) \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

# Time To Code



## Dangling Pointer

Dangling pointer is a pointer that points to deleted or de-allocated object.

Can You see the dangling  
pointer here ... ?



```
#include <stdio.h>

int* func(void);

void main(void)
{
    int *ptr = func();

    printf("%d", *ptr);
}

int* func(void)
{
    int x = 10;
    return &x;
}
```

## Wild Pointer

Wild pointer is any pointer that is used before initialization.

Can You see the wild pointer  
here ... ?



```
#include <stdio.h>

void main(void)
{
    int *ptr;

    printf("%d", *ptr);
}
```

The End ...



## Assignment 1

Write a C code that defines a function which takes an array as input argument and apply the bubble sorting algorithm on it, then print the result



## Assignment 2

Write a C code that define 3 int variables x, y and z and 3 pointers to int p, q and r. Set x, y, z to three distinct values. Set p, q, r to the addresses of x, y, z respectively.

- a- Print with labels the values of x, y, z, p, q, r, \*p, \*q, \*r.
- b- Print the message: Swapping pointers.
- c- Execute the swap code: r = p; p = q; q = r;
- d- Print with labels the values of x, y, z, p, q, r, \*p, \*q, \*r.

***Can you expect the output before you run the program ... ?***





[www.imtschool.com](http://www.imtschool.com)



[www.facebook.com/imaketechologyschool/](http://www.facebook.com/imaketechologyschool/)

*This material is developed by IMTSchool for educational use only  
All copyrights are reserved*