Short-term Hands-on Supplementary Course on C Programming

SESSION 7: Functions

NIVEDHITHA D
KARTHIK D

Time: 6:30 - 8:00 PM
Date: June 15th, 2022
Location: Online

Agenda

1. Administrative Instructions
2. What are functions?
3. Why do we need functions?
4. Using Functions in C: Demo
   a. Before main()
   b. Prototype for after main()
   c. Macros
5. Functions and Arrays
6. const Function Parameters
7. Tutorial: Pass-by-Value and Pass-by-Reference
8. Next Session

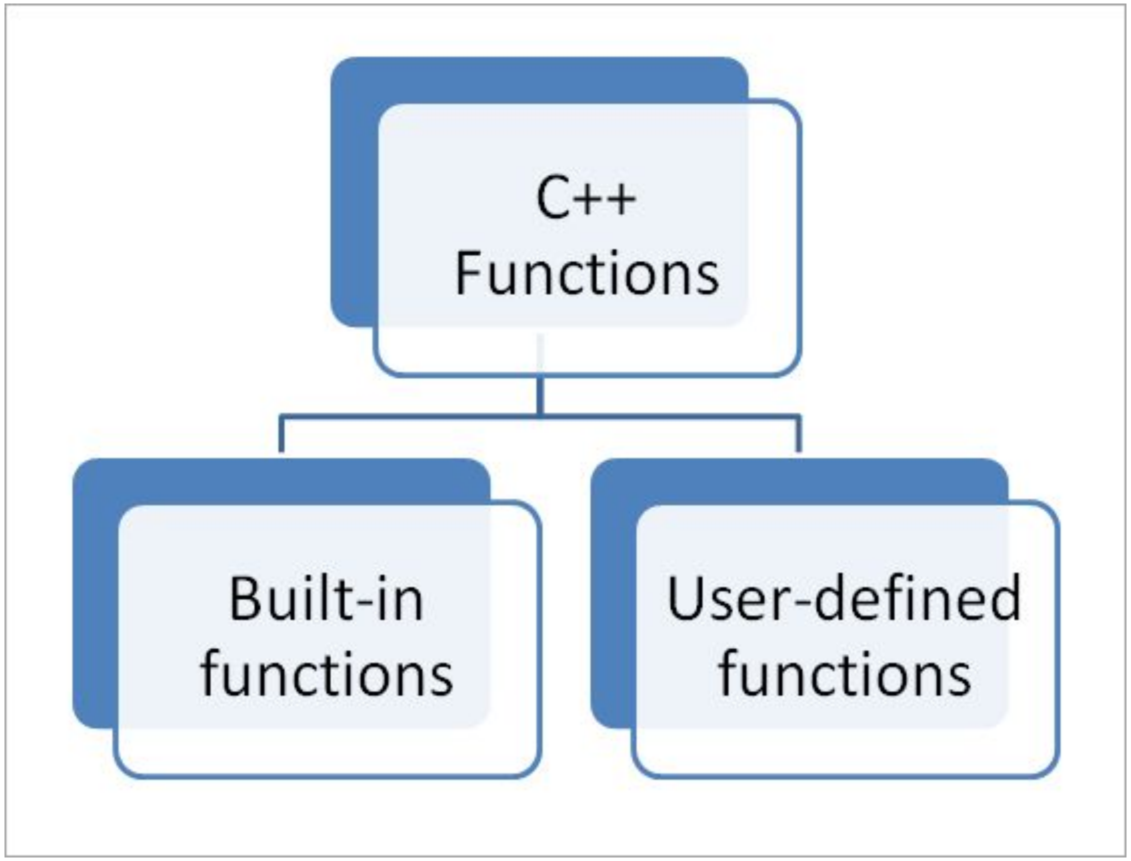
Administrative Instructions

- Please fill out the feedback form - will be shared in the chat
- Join us on Microsoft Teams,

  Team Code: **rzlaicv**

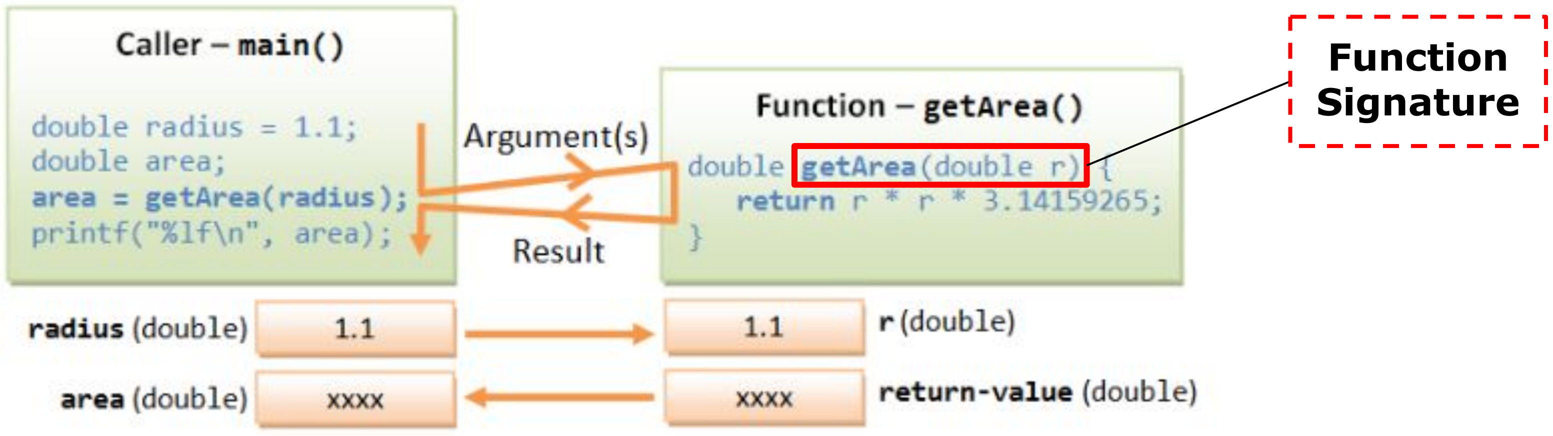
GITHUB REPOSITORY! ⭐🌟

What are Functions?



At times, a certain portion of code has to be used many times. Instead of re-writing the codes many times, it is better to put them into a "**subroutine**", and "call" this "subroutine" many time - for ease of **maintenance** and **understanding**. This subroutine is called a function (in C/C++).

Why do we need Functions?



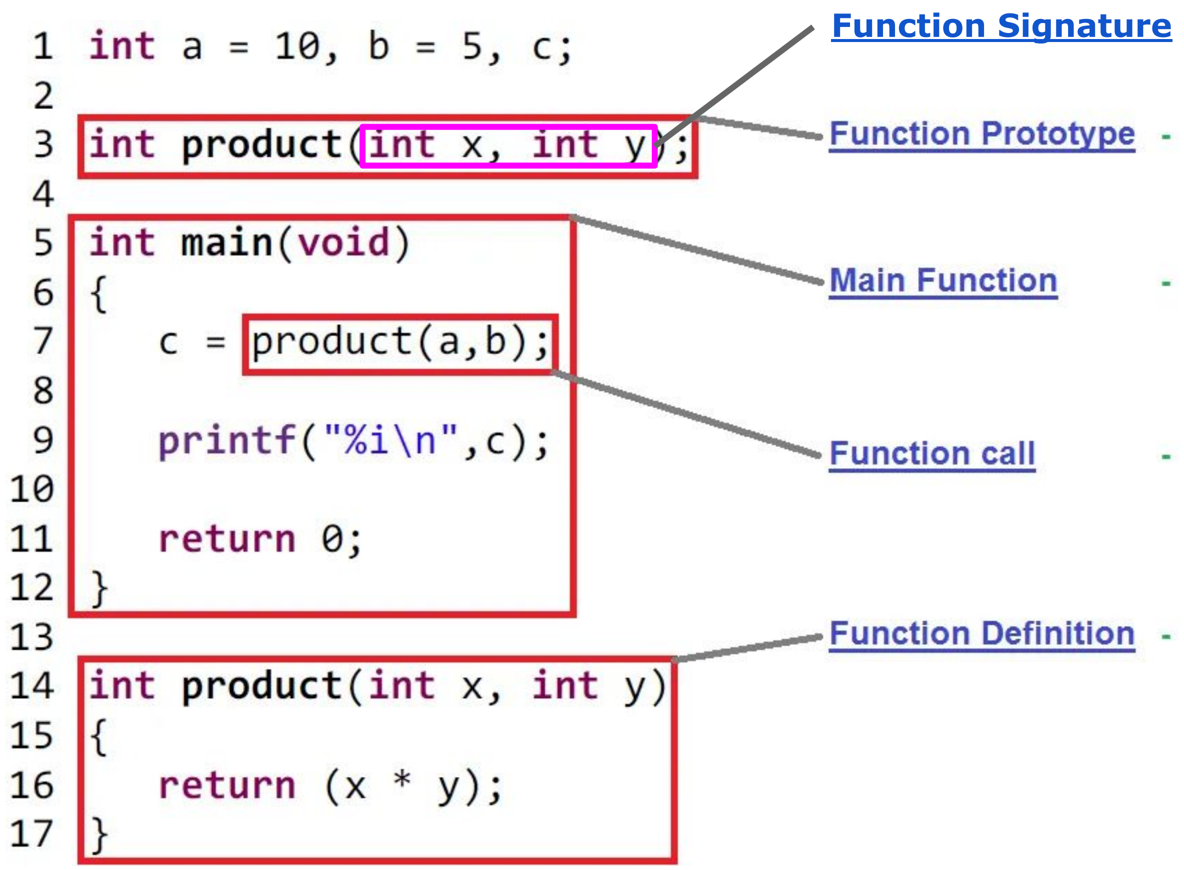
```
double radius = 1.1;
double area;
area = getArea(radius);
printf("%lf\n", area);
```

```
double getArea(double r) {
   return r * r * 3.14159265;
}
```

The benefits of using functions are:

- **Divide and conquer**: construct the program from simple, small pieces or components. Modularize the program into self-contained tasks.
- **Avoid repeating codes**: It is easy to copy and paste, but hard to maintain and synchronize all the copies.
- **Software Reuse**: you can reuse the functions in other programs, by packaging them into library codes.

Declaration of Functions



```c
int a = 10, b = 5, c;

int product(int x, int y);

int main(void)
{
    c = product(a,b);

    printf("%i\n",c);

    return 0;
}

int product(int x, int y)
{
    return (x * y);
}
```

Function Signature

Function Prototype - int is the return type and int x and int y are the function arguments

Main Function - int is always the return type and there are no arguments, hence the (void). Curly braces { } mark the start and end of the main function

Function call - product(a,b); a and b are global variables the function is passed. Here the values returned by the function are assigned to the variable c

Function Definition - contains the function statement return(x * y); the function returns x times y to the main function where it was called. Curly braces { } mark the start and end of the function

Using Functions in C: DEMO

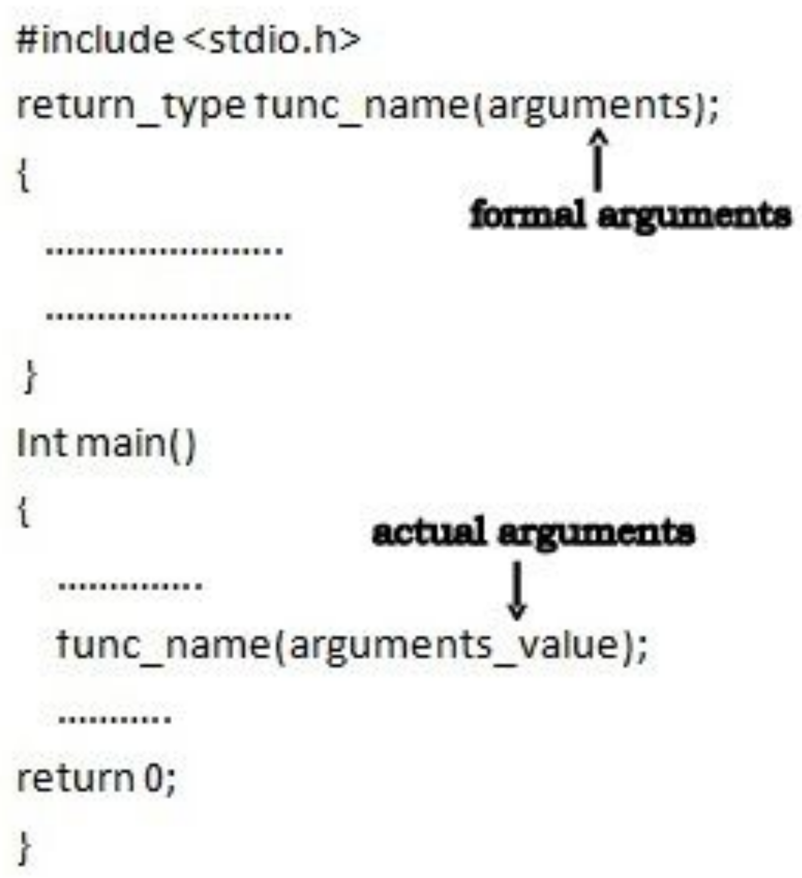
1
```c
int square(int n)
```

2
```c
int square(int n);
```
```c
int square(int n)
```

3
```c
#define SQUARE(x) (x * x)
```

4
```c
#include <math.h>
```
```c
pow(num, 2)
```

Pass-by-Value
vs.
Pass-by-Reference



```
#include <stdio.h>
return_type func_name(arguments);
{
   .........................
   .........................
}
Int main()
{
    ...............
    func_name(arguments_value);
    ............
return 0;
}
```

formal arguments
actual arguments

Pass-by-Value

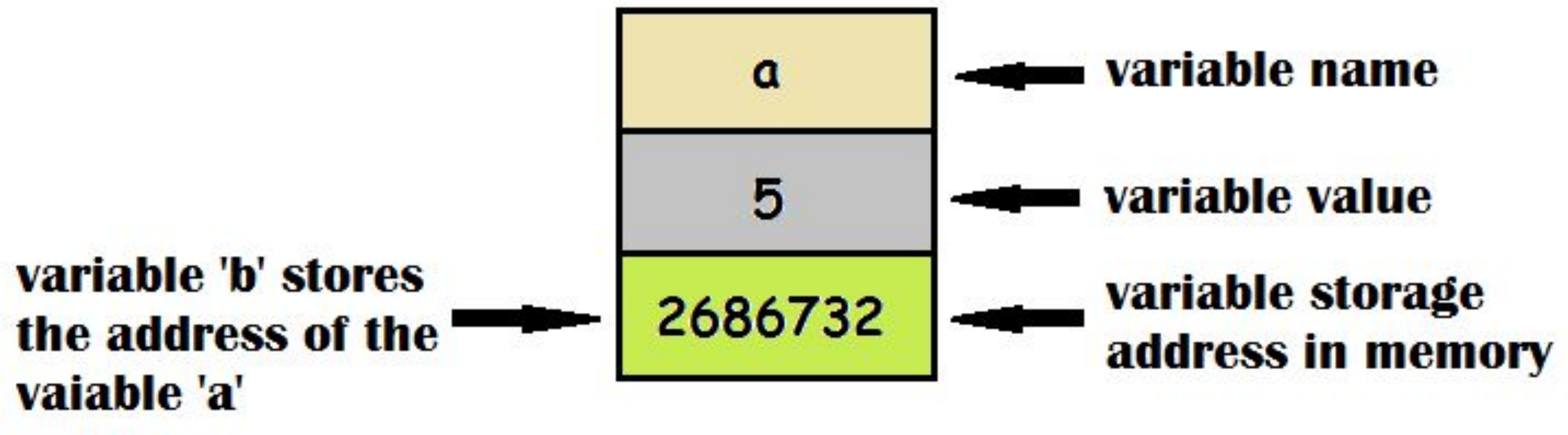
```
void swap(int a, int b)
```

Pass-by-Reference

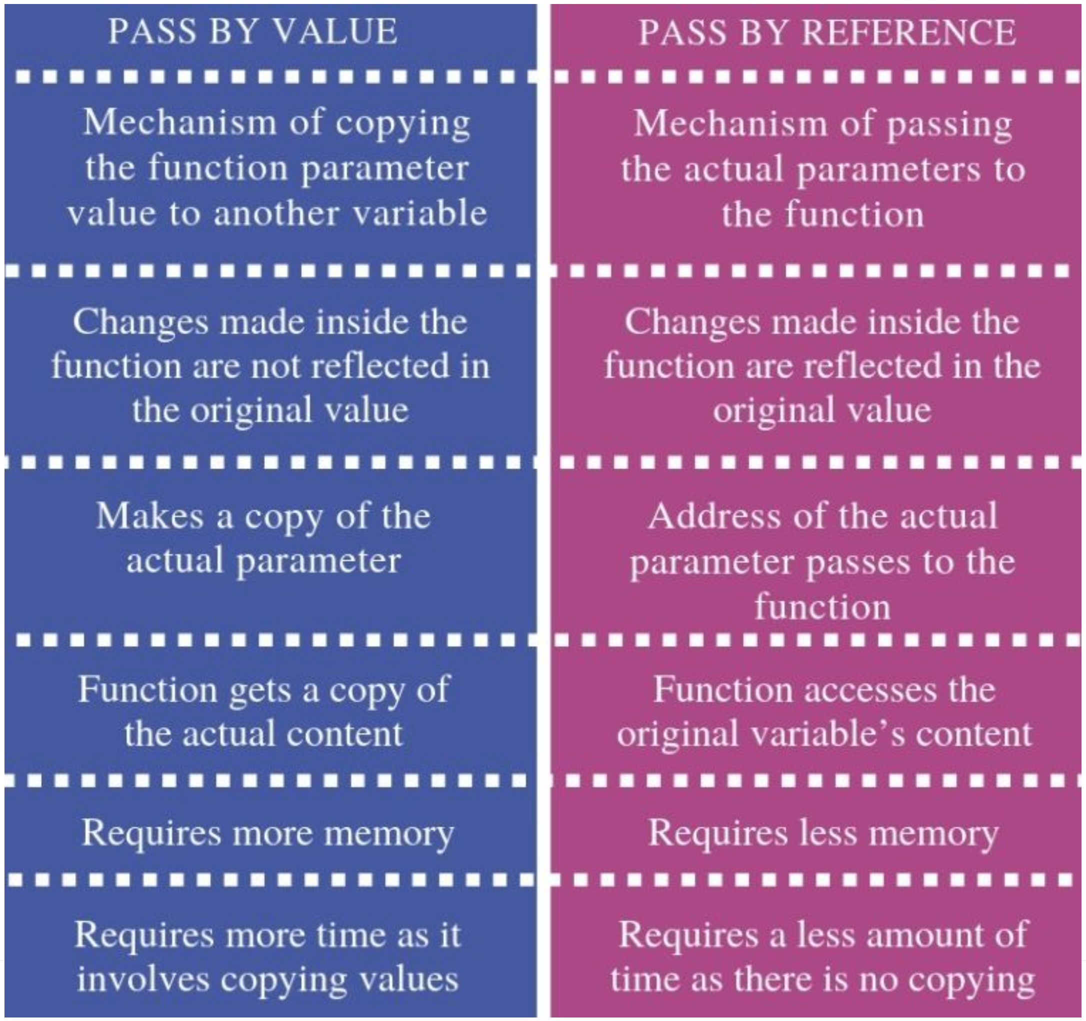
```
void swap(int& a, int& b)
```

Variable Storage in C

```
int a = 5;
int *b;
b = &a;
```

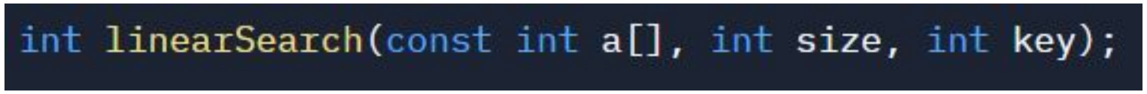

a
5
2686732
variable name
variable value
variable storage address in memory
variable 'b' stores the address of the vaiable 'a'

Pass-by-Value vs. Pass-by-Reference

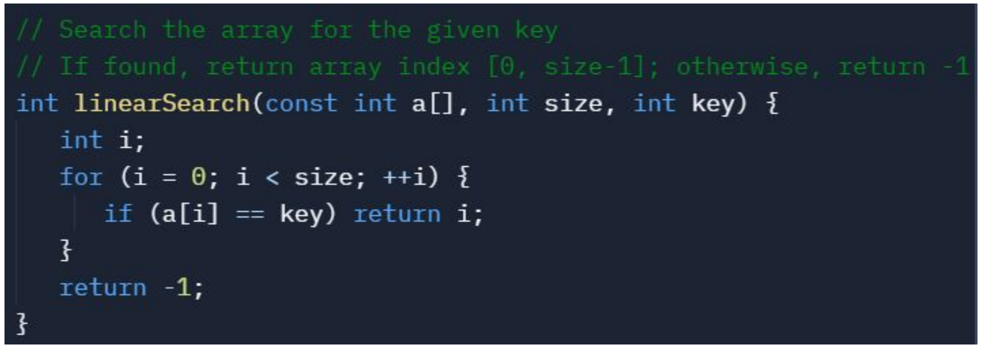


PASS BY VALUE	PASS BY REFERENCE
Mechanism of copying the function parameter value to another variable	Mechanism of passing the actual parameters to the function
Changes made inside the function are not reflected in the original value	Changes made inside the function are reflected in the original value
Makes a copy of the actual parameter	Address of the actual parameter passes to the function
Function gets a copy of the actual content	Function accesses the original variable's content
Requires more memory	Requires less memory
Requires more time as it involves copying values	Requires a less amount of time as there is no copying

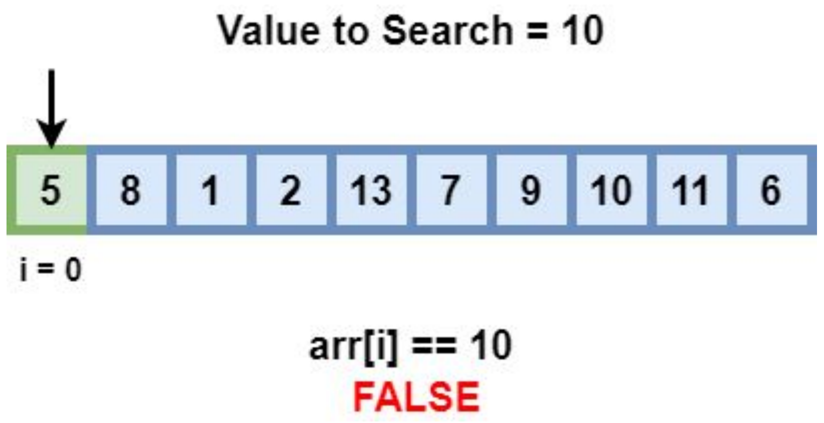
Functions and Arrays



```
int linearSearch(const int a[], int size, int key);
```



```
// Search the array for the given key
// If found, return array index [0, size-1]; otherwise, return -1
int linearSearch(const int a[], int size, int key) {
   int i;
   for (i = 0; i < size; ++i) {
      if (a[i] == key) return i;
   }
   return -1;
}
```



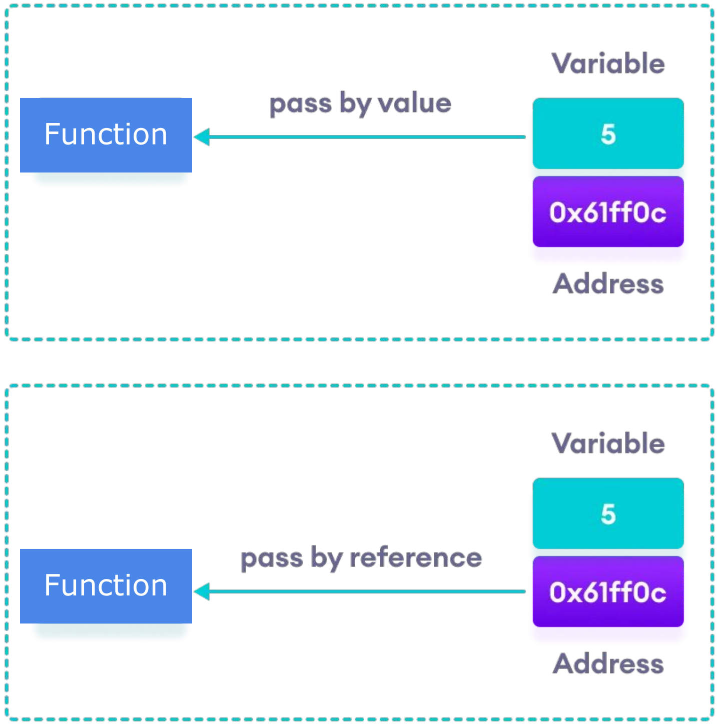
“const” Function Parameters

Pass-by-reference risks corrupting the original data. If you do not have the intention of modifying the arrays inside the function, you could use the const keyword in the function parameter. A const function argument cannot be modified inside the function.

Use const whenever possible for passing references as it prevents you from inadvertently modifying the parameters and protects you against many programming errors.

In a **linear search**, the search key is compared with each element of the array linearly. If there is a match, it returns the index of the array between [0, size-1]; otherwise, it returns -1 or the size of of the array (some implementations deal with only positive indexes). Linear search has complexity of O(n).

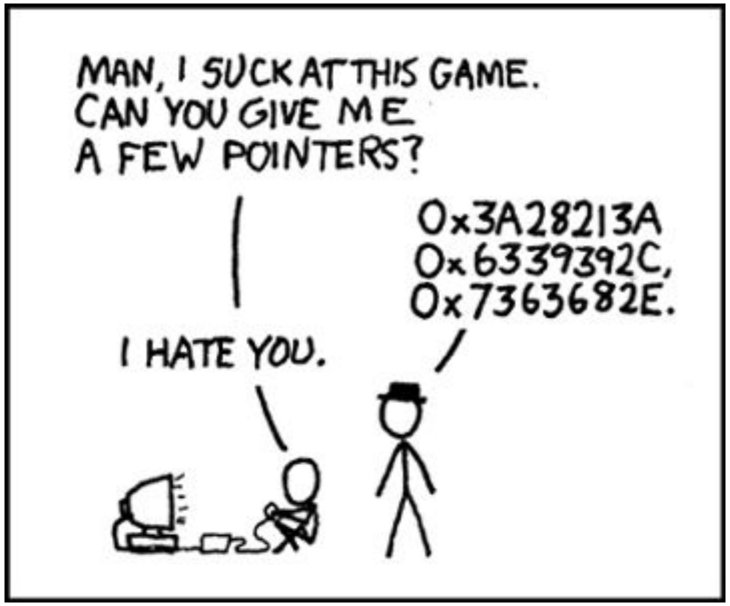
TUTORIAL

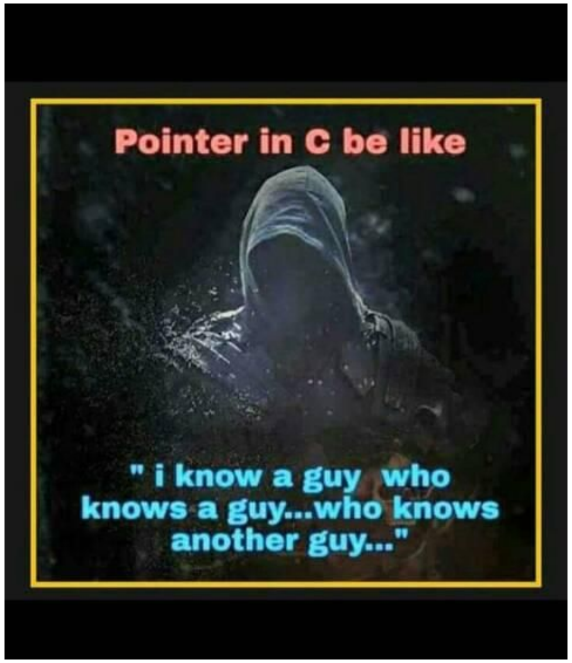


Pass-by-Value vs. Pass-by-Reference

Next Session

POINTERS!!!





Any Questions