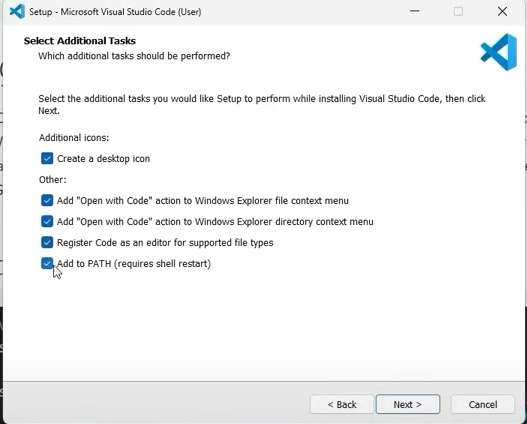
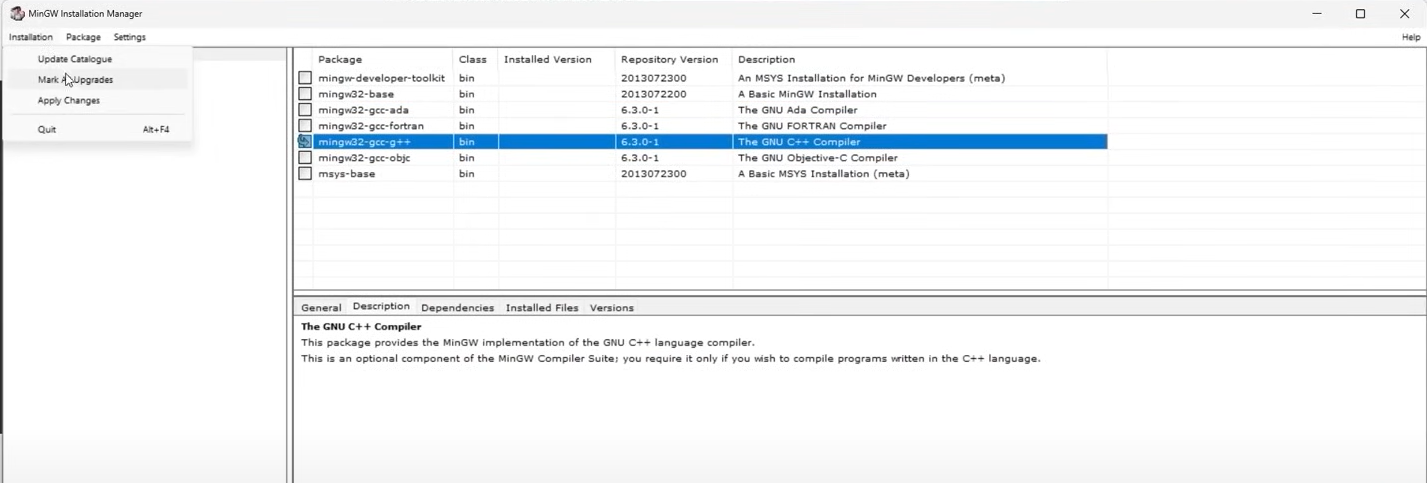
Learning C programming from beginning..

Here we are going to learn C programming in VS code and install mingw installer . In VS code installation process tick all these option :





Tick all option of Vs code and tick the fifth option telling you to install gcc-g++ compiler.

And important thing you have to set new variable environment , to run code and later you can do same thing using .RUN extension in VS code.

# C language boilerplate code.

#include <stdio.h>

Int main(){

printf(“ Hello world “);

return 0;

}

# Variables

A variable is a container which stores value.

A variable is a entity whose value can be changed.

For example :

int a; // variable decleration

int A; // variable decleration

a = 34 ; // a is assigned value ‘3’

b = 45.6 ;

c = ‘A’ ;

For printing variable :

int a;

a = 3;

printf(“ %d “, a);

Rules :

Variables can have underscore (\_)

Variable must be alphabet.

Variable is case sensitive.

Note : C program is case sensitive..

Note : An entity whose value does not change is called as a constant.

## Data types :

int, float, char , etc are data types which help to address the value of variable. For example

int is for numeric value

float is for decimal numeric value

char is for single character value

That how it works

Syntax :

int a;

a = 56;

or it could be

int a = 56;

### 

### Use of comment in c program /

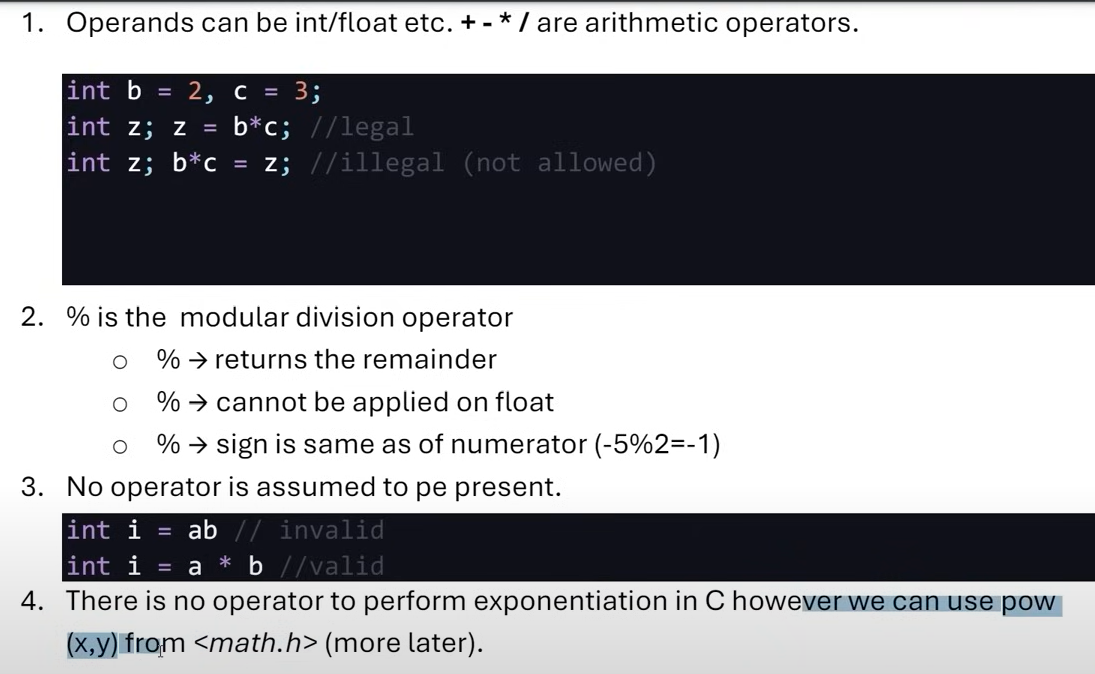
// : Single line comment

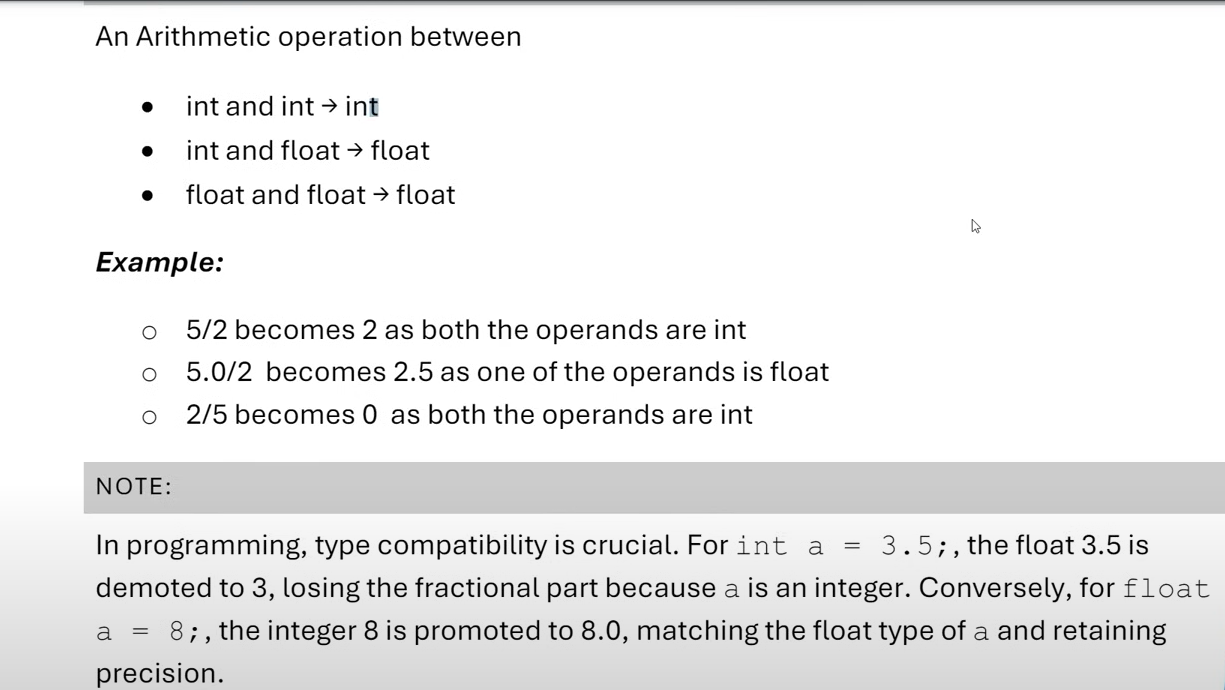
/\* ……. \*/ : Multiple line comment

IT will help to made clarification in code as we can describe some notes in plain text.

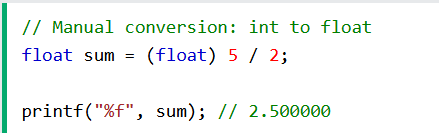
### Use of scanf() function for taking input from user :

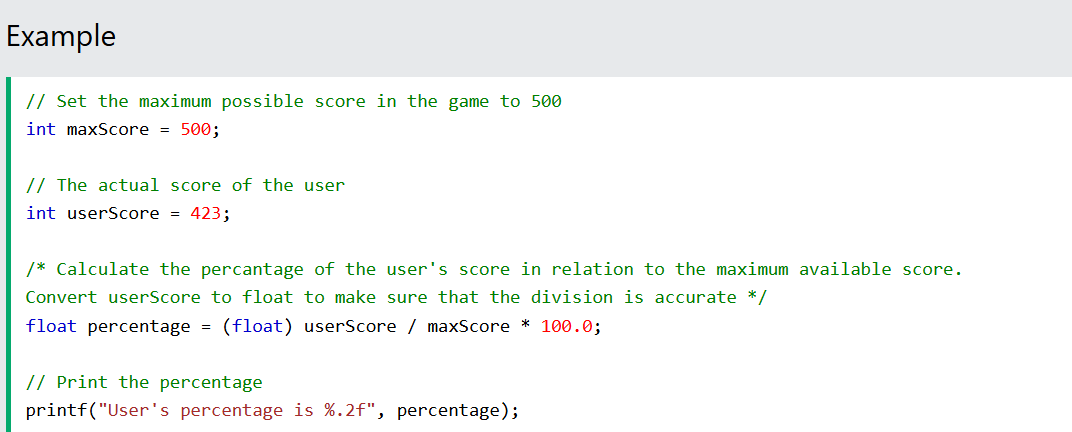
# Now the concept is I have some basics but the question is am I capable of solving the problems in real world, it could be from finance , school, commerce, bank, communication, transportation, health, art, etc…. but how am I able to solve it through program, through automation,. I think it’s not about solving math problem only..

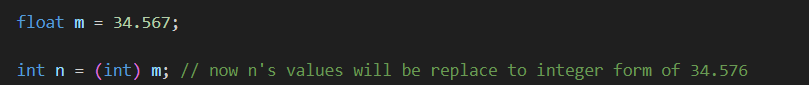
Note : Variable is a name of memory location.



## Typecasting :







CONDITIONAL INSTRUCTIONS

Decision making instruction in C :

If – else statement

Syntax :

If ( condition to be true ) {

then only run the program inside this braces !

}

Another amazing part what if the if condition is not meet or condition is false ..!

If ( condition ){

Run this code ;

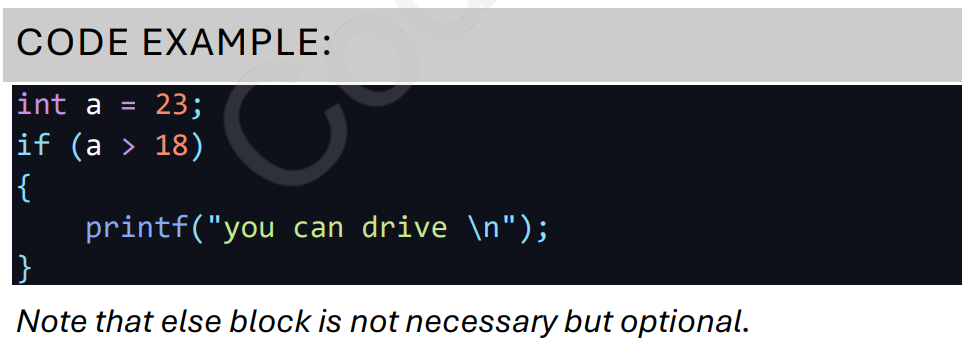
}

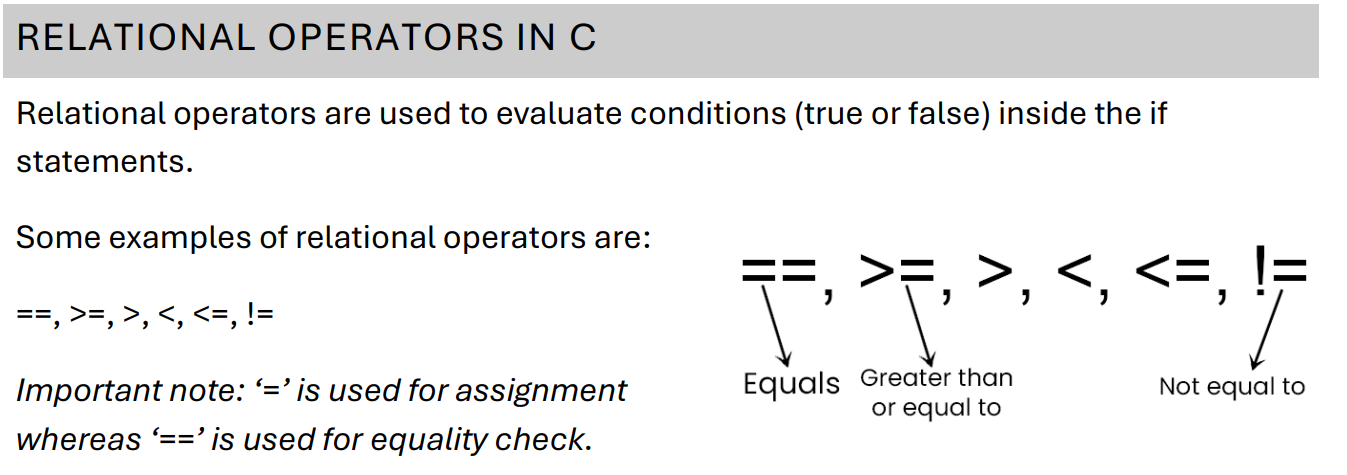
else {

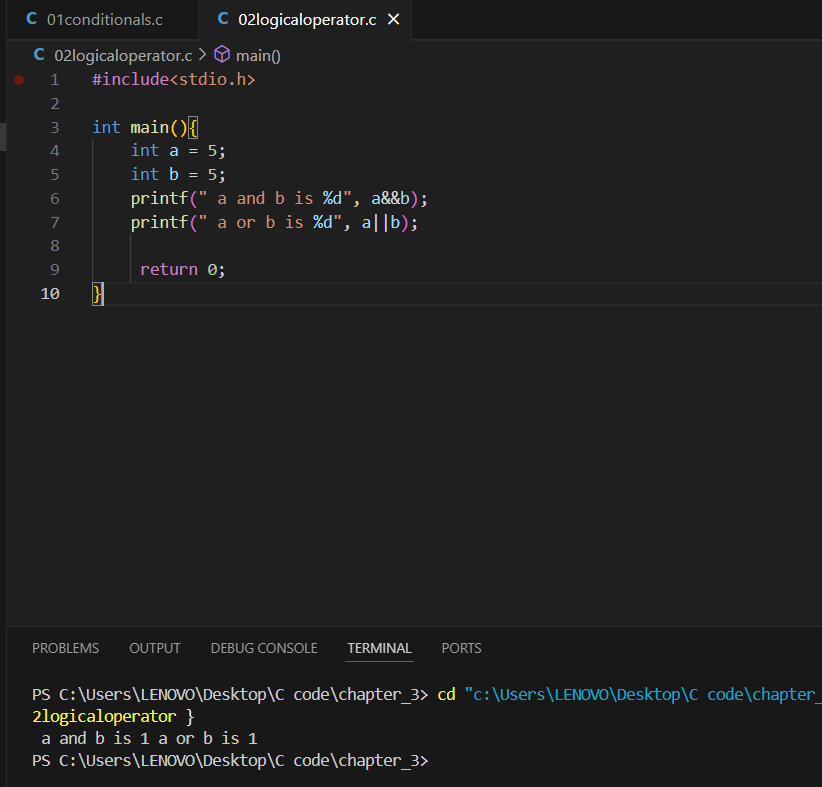
run this code;

}

In this case we have alternative or say filter for the condition where it does not meet to be true or it is false.. !







**Explanation:**

1. **Logical AND (&&)**:
   * a && b evaluates to true (1) if both a and b are non-zero values.
   * Since both a = 5 and b = 5 are non-zero, a && b evaluates to 1.
2. **Logical OR (||)**:
   * a || b evaluates to true (1) if either a or b is a non-zero value.
   * Again, since both a = 5 and b = 5 are non-zero, a || b evaluates to 1.

**Key Notes:**

* In C, non-zero values are treated as true in logical expressions, while zero is treated as false.
* Logical operators && (AND) and || (OR) operate on the truthiness of the values.

## If – else statement

Syntax :

If ( expression ) {

if true then only execute this portion

}

else {

if first one is false then this portion of code will be executed.  
 }

syntax : if else if

if ( expression ) {

if it is true

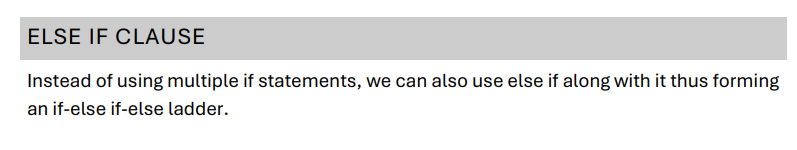
if ( condition ) {

if this second if condition is true

}

else {

if first one is false ,it is here

 }

Syntax :

if{ // Statements }

else if{ // Statements }

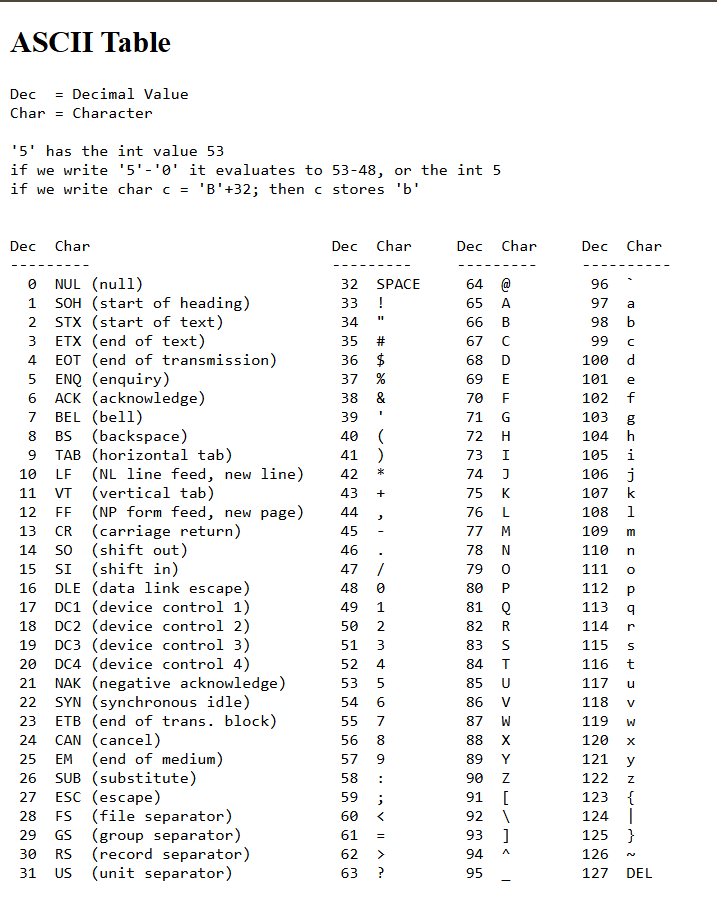
else{ // Statements }

Same concept but different syntax :

If ( percentage >= 80 )

If ( percentage >=60 && percentage < 80 )

If ( percentage >=60 && percentage <= 79 )



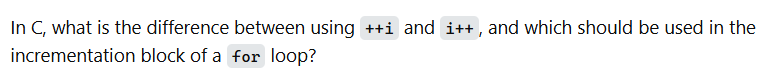
Concept of Loop :

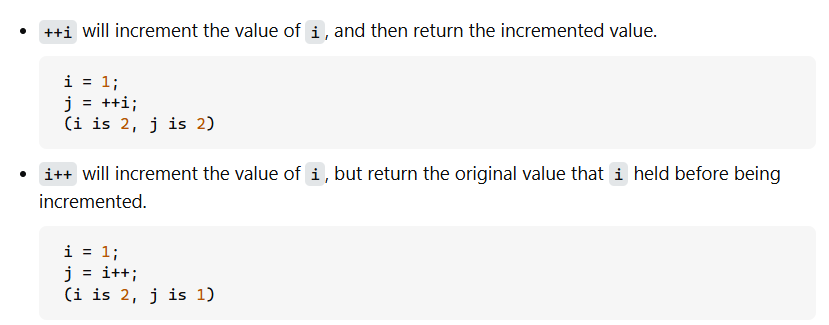
Increment and decrement operator.

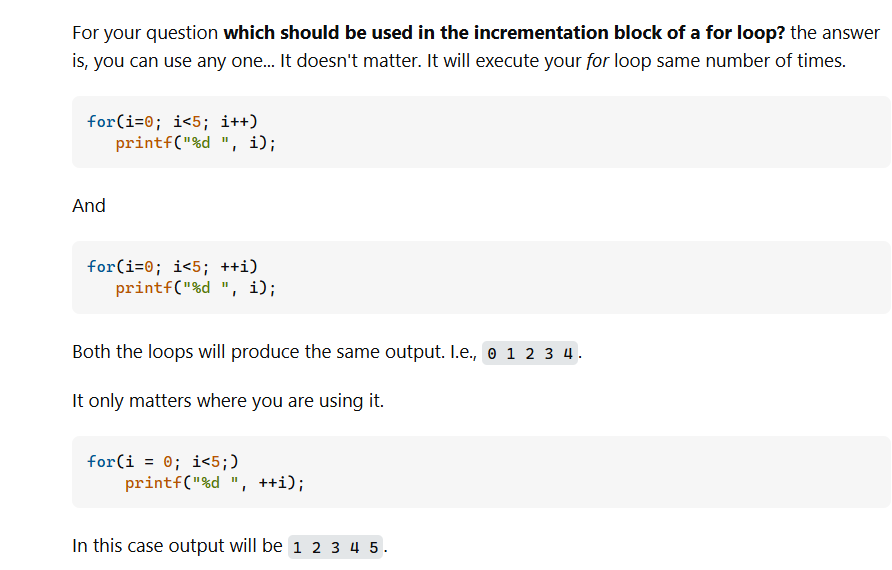
i++, i - -

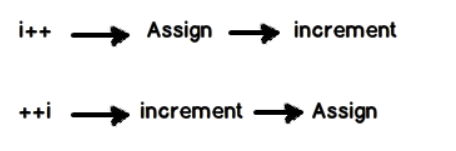
i++ : prints first and then increments ( post increment operator )

++i : increments first and then prints ( pre increment operator )

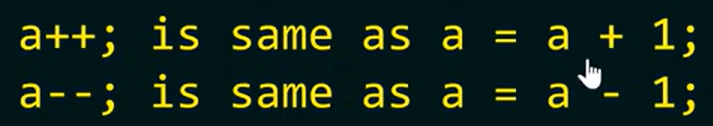
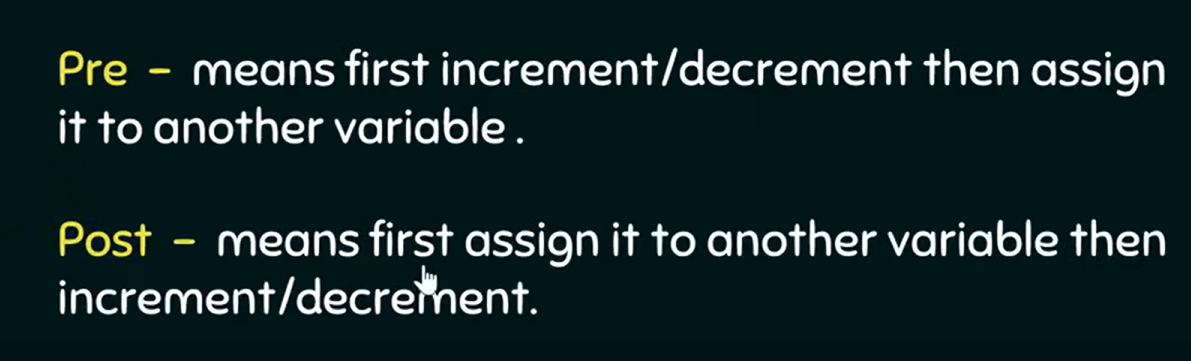


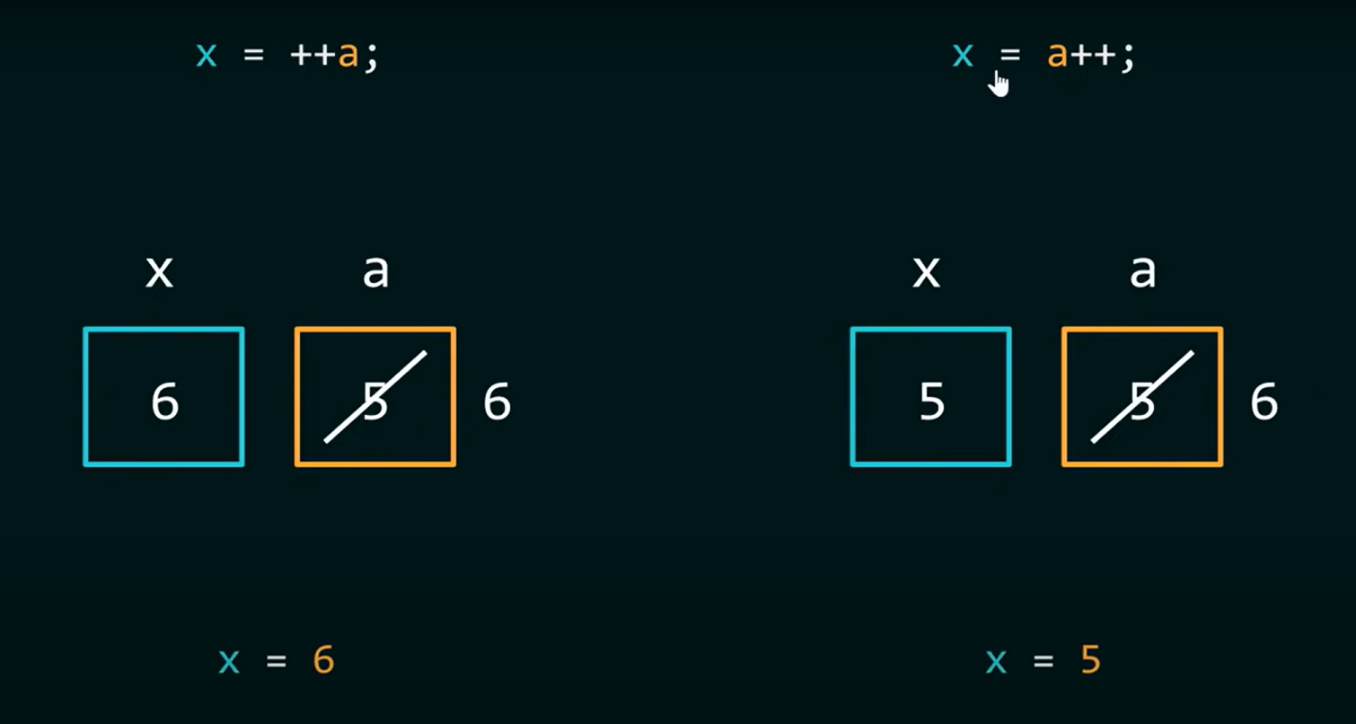






Simply :

Pre increment operator means ++ operator applied before the operand, I will be understanding this as 1 + operand, where post increment operator means ++ operator applied after the operand ie; operand + 1.



Compound Operator :

+=

i += 1; // same i = i + 1;

-=

i -= 2; // same as i = i – 2;

\*=

i \*= 3; //same as i = i \* 3;

Do-While loop :

Do-while loop : executes the code and then checks the condition

Do-while loop : While loop which executes at least once.

While loop : checks the condition and then executes the code, so there is no chance it would run at least once even when the condition meet unlike do-while loop.

Syntax of do-while loop :

do {

code;

}while (condition);

For loop :

Syntax

For ( initialize ; test/ check ; increment or decrement ){

//code;

}

Initialize : setting a loop counter to an initial value

Test : checking a condition

Increment : updating the loop counter

Note : In C Programming any non-zero value consider as “True”

Note : for ( i = 0; i < n; i++) process upto : 0 to n-1

Note : for ( i = 0; i<=n; i++) process upto : 0 to n

### **Break :**

Exit the loop now !, we can use break given some conditions like if

where we use **Continue** to skip the current iteration or process in any other case except loop !

# Chapter 5

Functions are block of code statements in programs where it is a container to save logic work or job which provides facility to perform the same task repeatedly more than once !

return\_type function\_name(parameter\_list) {

// Function body (logic)

return value; // (if return\_type is not void)

}

Lesson : Your code contains an error because you are **defining the function inside main()**, which is not allowed in C. Functions **must be defined outside** main().

First declaration of function, which is obviously we should declare outside of main() separately :

int sum( int, int );

secondly building function

int sum ( int variable , int variable ){

printf(“ The sum is %d \n”,variable+variable);

return variable+variable;

}

We can call function at anyplace in program.

sum ( 24, 50 );

sum ( a, b );

// here we can pass directly integer values or variable for any foreign function

int variable = sum ( a, b);

now we can store the return value from function into another variable

void function\_name() // this type of function does not any value and arguments either

No, it is **not mandatory** to declare a **function prototype** if you define the function **before** calling it in main().

Parameters = variable values

Arguments = real, raw values

**Passing by Value:**

When you call a function and pass a variable like sum(b), you are passing **the value of b** at the time of the function call, not the actual b variable.

Types of function :

* User defined functions
* Library functions

Recursive function call itself to solve program....

Recalling the concept :

#include <stdio.h>

int main() {

int a = 5;

// Understanding the Operators:

// a → just the current value.

// ++a → pre-increment: increases a by 1 before using its value.

// a++ → post-increment: uses the current value of a, then increases it by 1.

printf("a = %d\n", a); // prints 5

printf("++a = %d\n", ++a); // pre-increment: a becomes 6, then prints 6

printf("a++ = %d\n", a++); // post-increment: prints 6, then a becomes 7

printf("a = %d\n", a); // prints final value of a, which is now 7

return 0;

}

### ***Pointer***

Pointer is a variable which stores the address of the another variable.

#include<stdio.h>

int main(){

int a = 73;

printf(" The value of variable a is %d\n", a);

printf(" The address of the variable a where the value %d is stored is %p\n", a, &a);

// &a refers to the address or memory location of variable a

// %p is the format specifier to print a pointer (memory address)

int \*b = &a; // b is a pointer storing the address of variable a

printf(" The address stored in pointer b is %p\n", b);

// The value of pointer b is equal to the address of a (&a)

printf(" The value stored at pointer \*b is %d\n", \*b);

// Dereferencing: \*b gives the value stored at the memory location (73)

return 0;

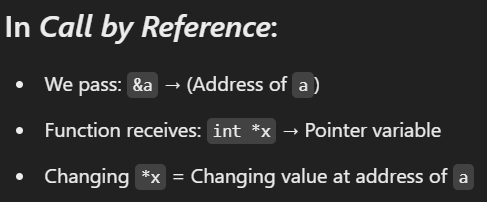
}

| **Operation** | **Description** | **Example** |
| --- | --- | --- |
| int \*p; | Declaring a pointer | p is a pointer to an integer |
| p = &a; | Assigning address | p stores the address of a |
| \*p = 20; | Dereferencing | Changes value at stored address (a = 20) |

***Function Call***

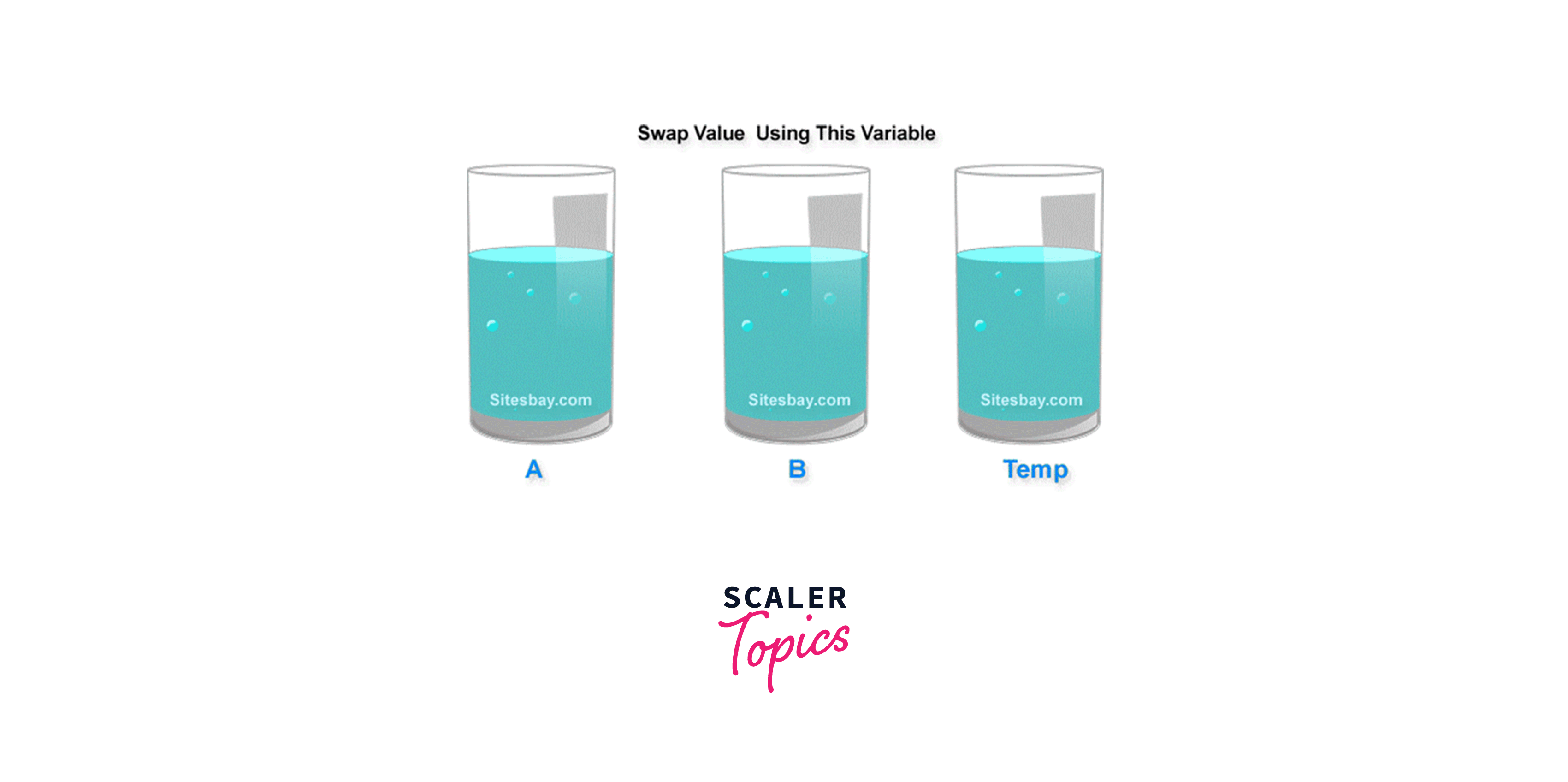
Function call by value , where the passed arguments is the copied value of original arguments, and the function can’t alter the value of original variables value.

Function call by reference , where the memory address of variable are passed as arguments. I this case we can change the original value of variables using pointer.



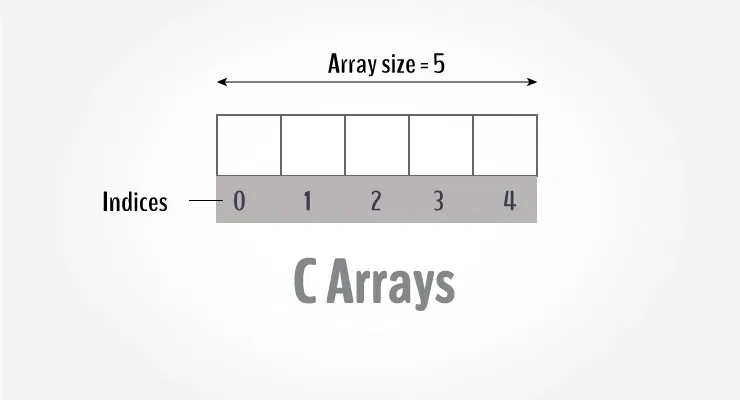
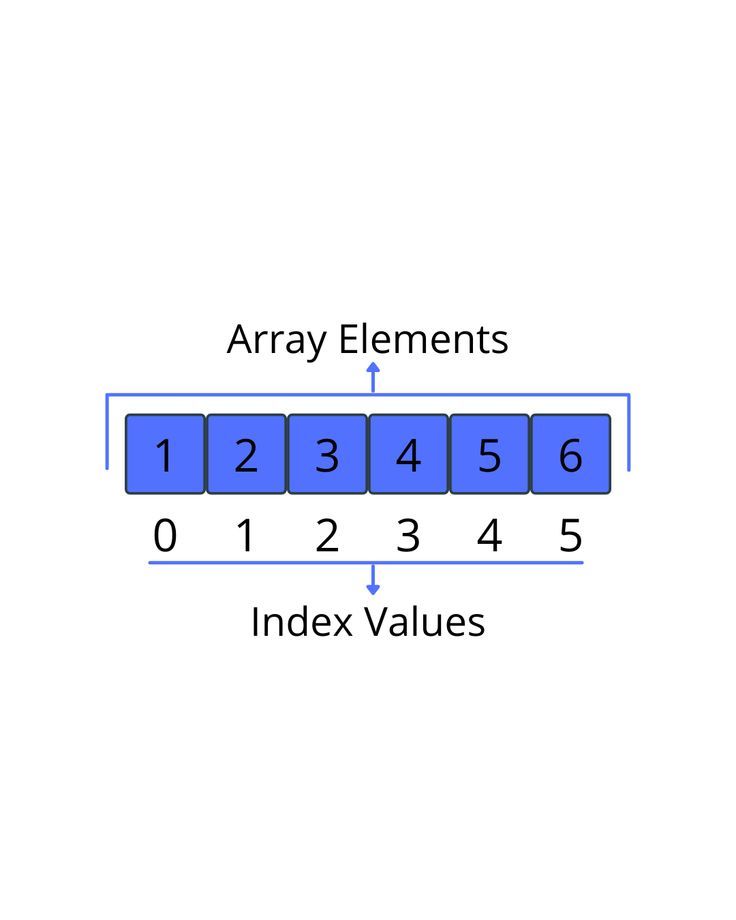
Concept of swapping :

<https://www.scaler.com/topics/images/swapping-using-temporary-variable.gif>



# **Array :**

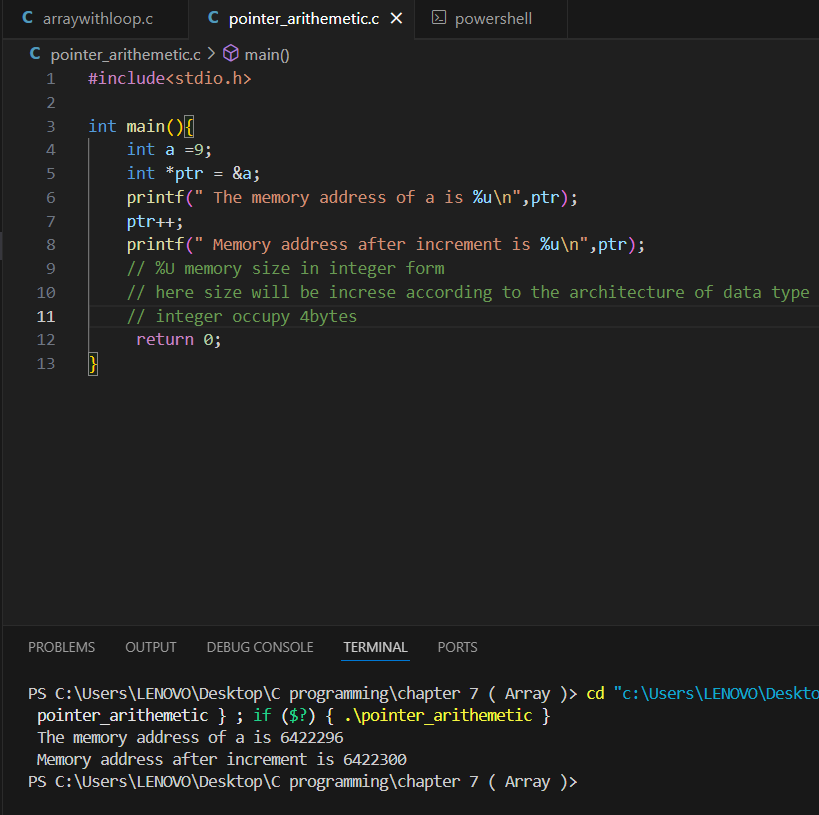
Array is a collection of similar elements. Array allows a single variable to store multiple values.



Syntax :

dataType arrayName[arraySize];

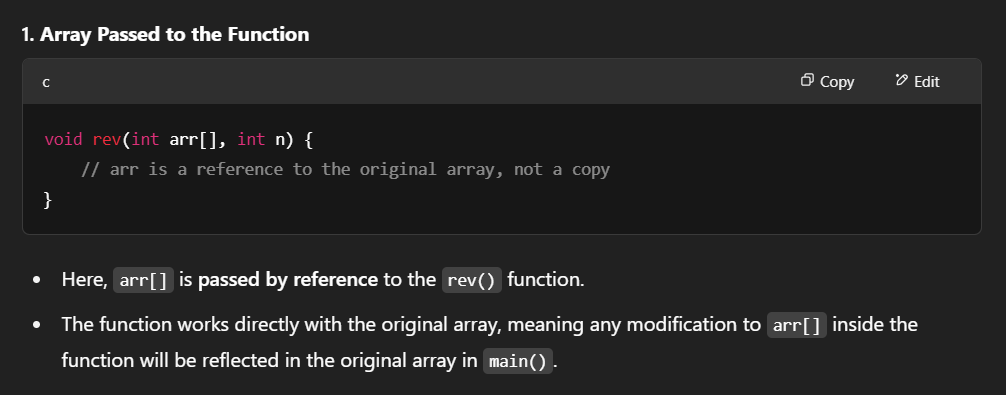




Integer occupy   
2 or 4 bytes

Float occupy 4 bytes

Char occupy 1 byte

The function rev() in your code is indeed capable of altering the original values at their positions because arrays in C are **passed by reference** to functions. This means the function works directly on the original array and can modify its contents, just like a pointer would. When you pass an array to a function in C, you are actually passing the **address** of the array, not a copy of it.

# **Array of string :**

    char s[] = {'A','B','C','\0'};

    char s[] = "ABC";

    char s[4] = "ABC";

way to declare string in C

notice : \0 is null character

Use of %c as printing character one by one but c provide us facility to print whole string at once using %s.

for (int i = 0; i < 4; i++)

    {

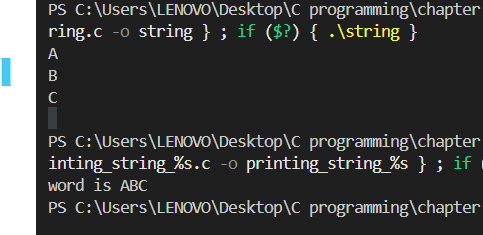
        printf("%c\n",s[i]); }

While using “ ie double quote no need to insert null character , it automatically set .

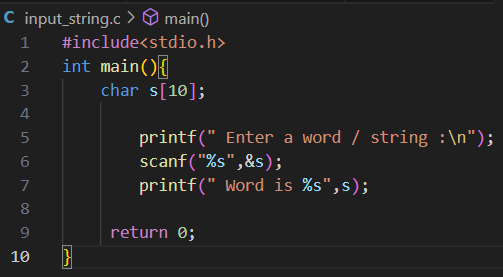
char s[] = "ABC";

        printf("word is %s",s);

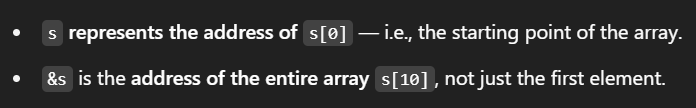
    printing character one by one vs printing whole string at once



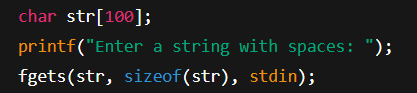
%s is more efficient to print word !



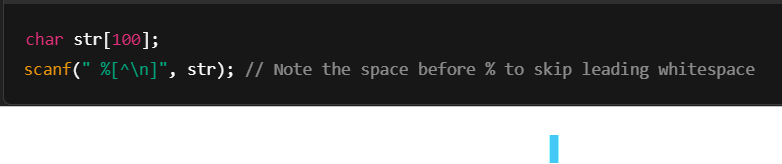
scanf("%s", s); // or scanf("%s", &s[0]);



Take multiple words along with space :



Alternative using scanf



We can use gets() and puts() too,…

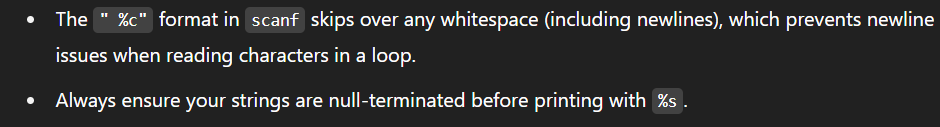
String function : #include<string.h> is necessary to include

Strlrn(“string\_name”)

strcpy(destination, source);

strcat( st1, st2);

strcmp(st1, st2);



### Structure :

structure is also a data type which can hold multiple data types inside it >>>

declaration :

struct structure\_name {

int id;

char name[50];

float salary; };

it should be declare outside of main function

inside main() or while declaring we can create variables :

struct structure\_name {

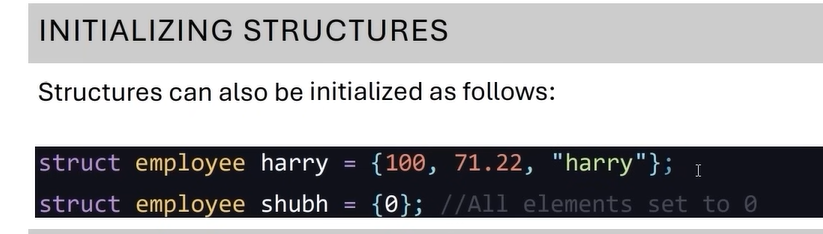
int id;

char name[50];

float salary; }e1; // notice that e1 is one structure variable

// alternative : struct structure\_name e1;

We can also create array of structure too

struct structure\_name e[100];

