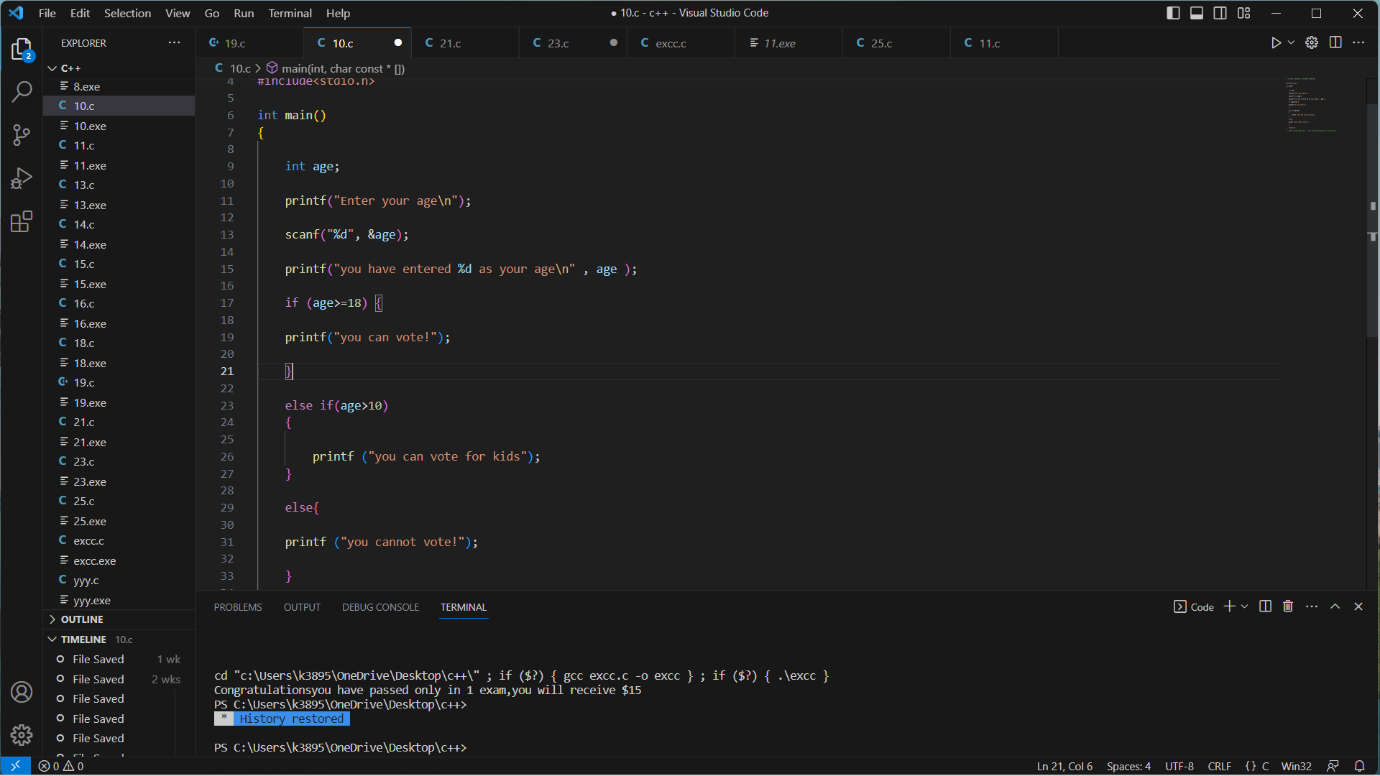
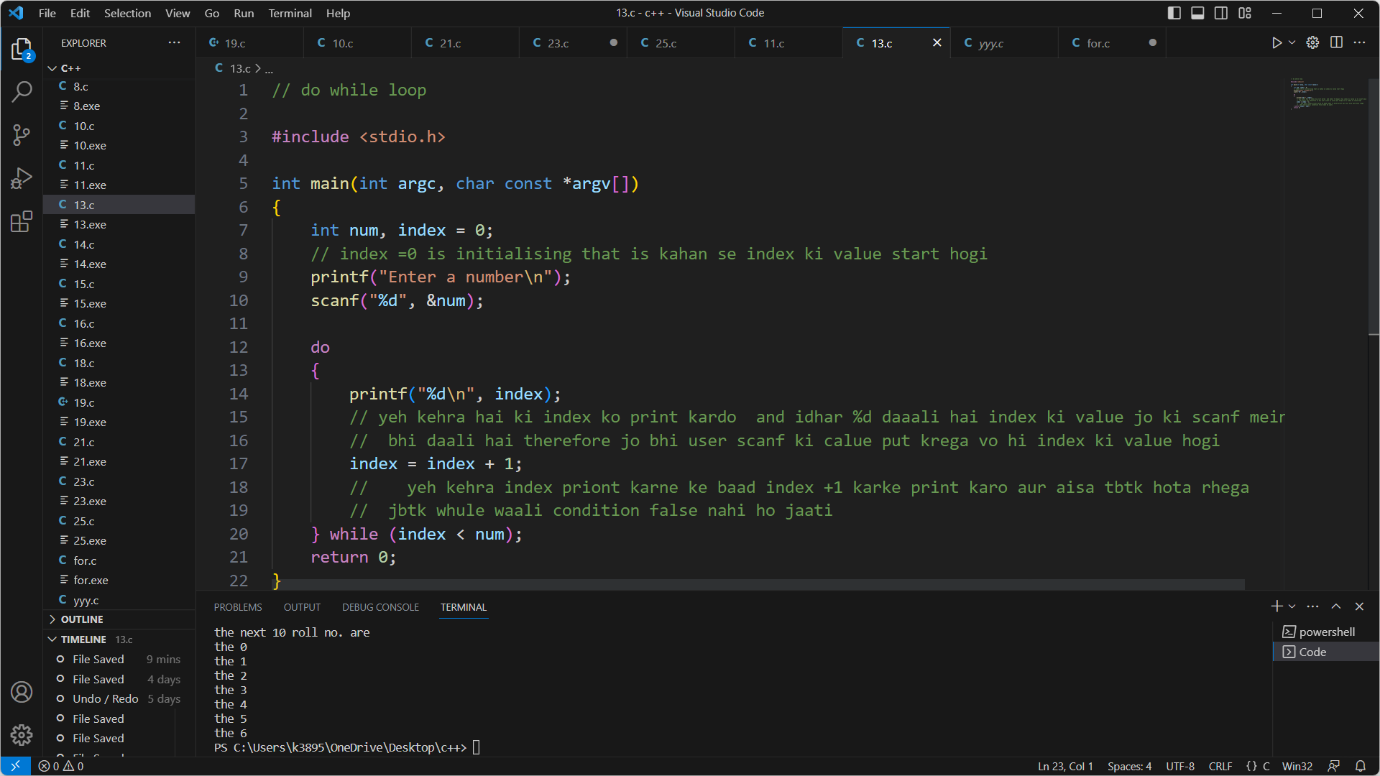
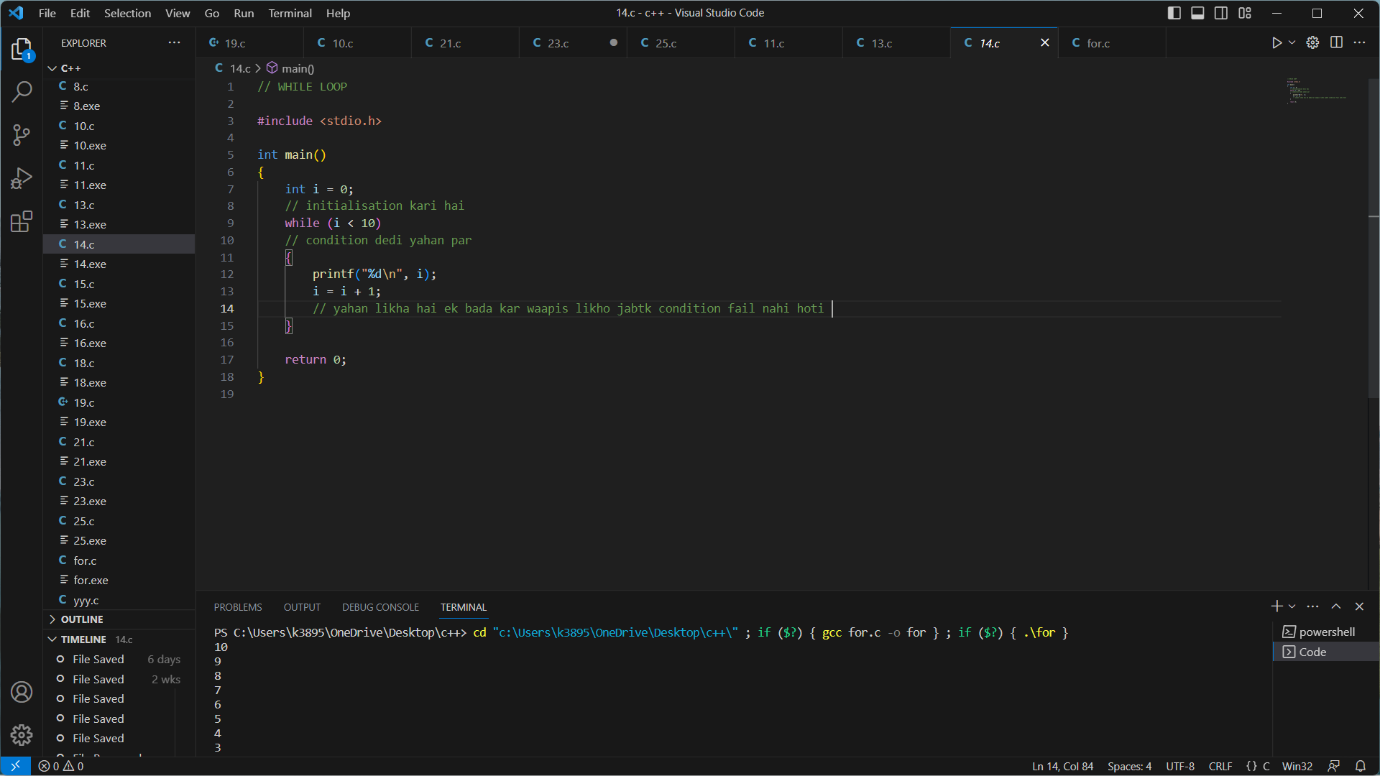
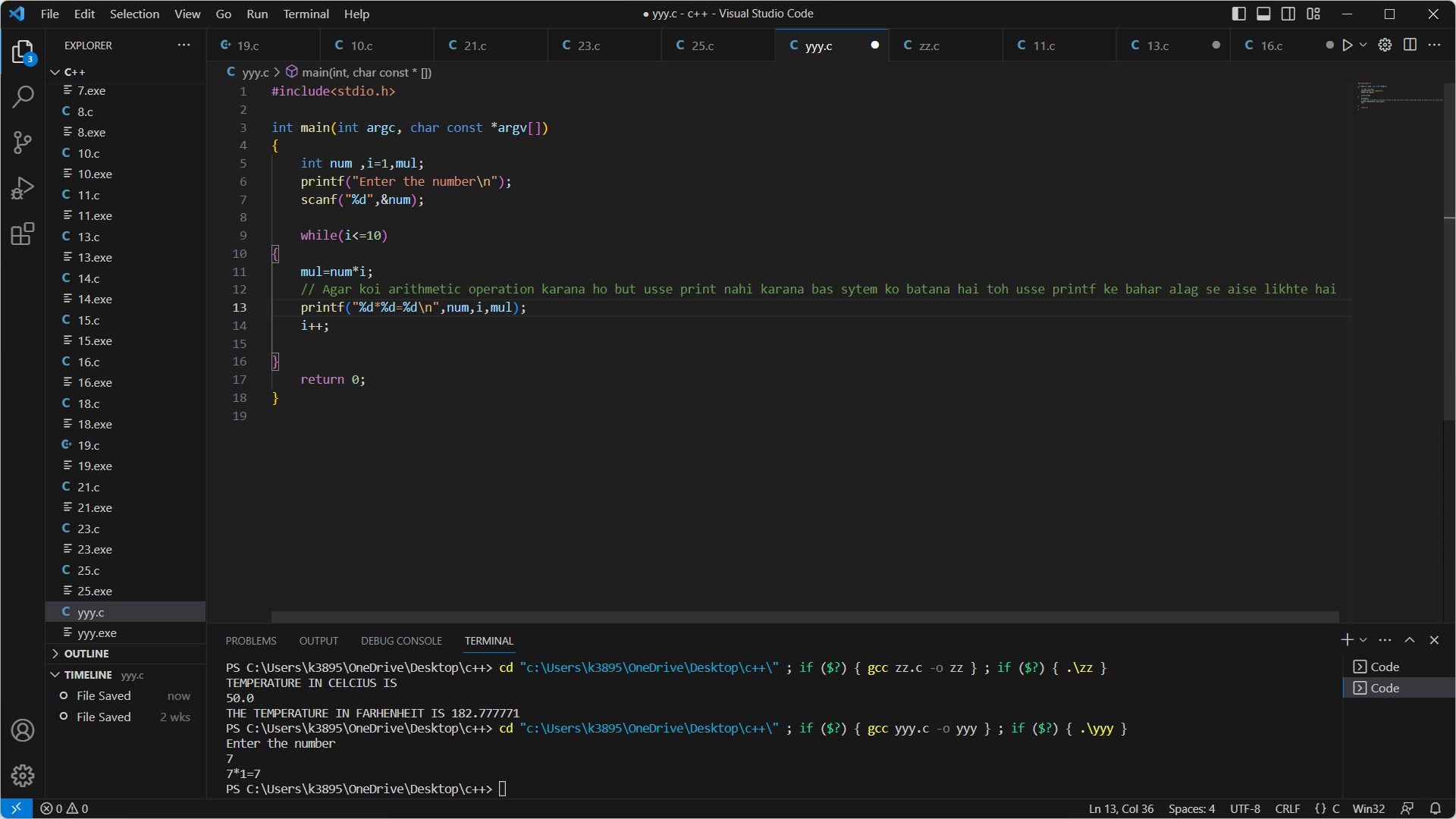
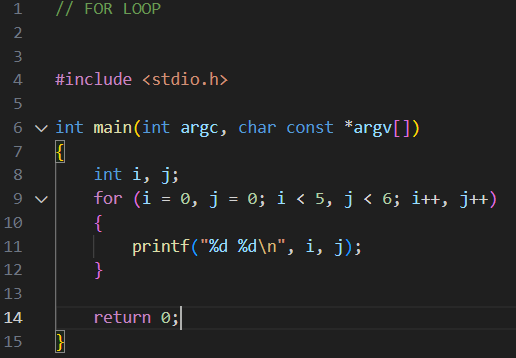
Vscode

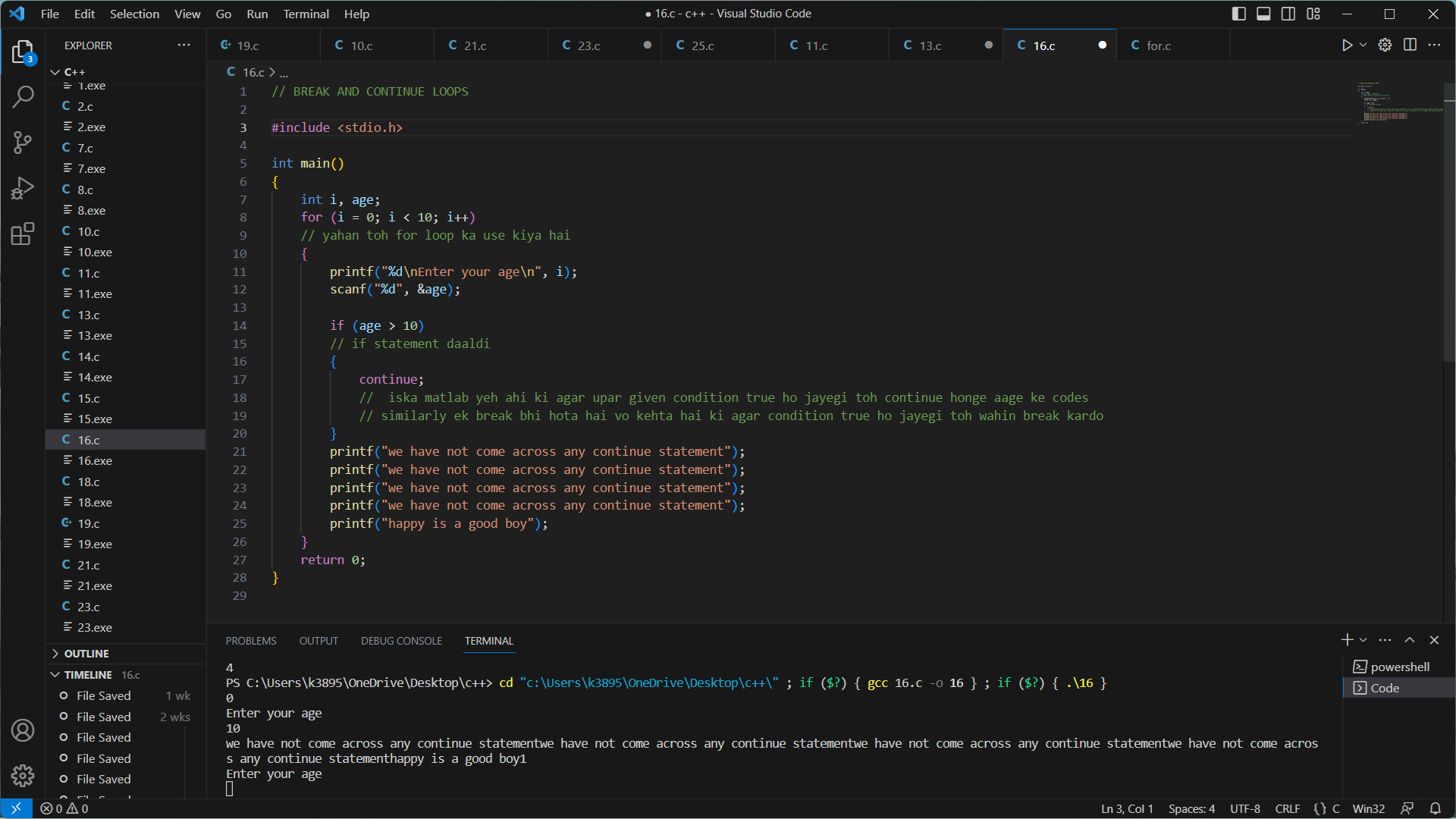
C LANGUAGE

1. It will follow on priority first “if” then “else if” and then “else”
   1. For “if” we need to provide a condition
   2. And “else” is used as, agar “if” waala nahi hua toh “else” waala hoga
   3. Whereas “else if” waala wahan use hoga jab if ke elava koi aur bhi condition deni hogi
   4. 
   5. “== “Is used to check or compare if something is equal to something
   6. “!=” means is not equal to
   7. “&&” if both the operands are true then only condition will be true finally
   8. “||” if any of the operands is true then condition will be true finally
   9. “!” used to reverse the logical state of is operand i.e true to false, false to true
   10. Using “const” key word means to make a variable constant and pc would only take its value where const is mentioned
   11. Const int a=5
   12. Int a=6
   13. Answer would be a=5
   14. “#define” also works as a “const” and anything can be defined its value
   15. “\b” backspace
   16. “\t “tab
2. In “do while loop” koi kam tab tak hota rhega jabtk humari given condition false nahi hojati still ek baar toh hoga hi phir cond n check hoti hai
3. Tabhi chalega jab condn true hogi varna ek baaar bhi nahi chalega
   1. 
   2. 
4. Comment- ctrl+/
5. /n – new line se start hoga next instructions jahan /n daala hai
6. Int-integer use %d with it
7. Float-decimal use %f with it
8. %c used for character
9. %s used for strings(characters as well)
10. The difference between character and string is

character is just one letter whereas a string is group of letters that is a word.

1. %l used for long integer
2. %lf double
3. %LF long double
4. %.10f will giv answer till 10 places and similary for any other no.
5. %10.f will give 10 spaces between answer and statement
6. “Alt + click”from mouse will bring another cursor wherever you clicked
7. “ printf("5\*1=%d\n",a\*b);”
8. Jo operation’,’ ke baad daaloge vo hoga
9. Jo bhi cheez “,” se pehle likhoge vo as it is print hogi and isme %d variable ki trah kaam krega, jo bhi uss operation aana hoga vo %d ki jagah aayega
10. Scanf allows us to put input where the print comes according to the situation
11. For eg enter your age , we can add any age through scan f and the program will react accordingly
12. scanf("%d" , &b); , b is the function jiski jagah par hum koi ek new value daalenge using scanf isliye alag se coding mein “&b” add karna padtha this is a rule
13. Switch statement mein
    1. “switch()” jo bhi cheez bracket mein daaaloge vo hi cheez case mein put kroge
    2. Eg. Switch(age)
       1. Case(age)
    3. Also ek se zyaada conditions bhi put kar sakte ho for eg.
    4. Switch(age)
    5. Case(age)
    6. Switch(marks)
    7. Case(marks)
    8. Ab jo bhi output aaayega sab mila julakar aayega
    9. Break statement hatgya to wahan rukega nahi and aage ka bhi print ho jayega
    10. Continue statement ka yeh kaam hai ki vo ki bhi loop ko bypass kar sakta hai and seedha next loop mein pahuch jaayega humara code run hone ke liye
14. 

For (expression1, expresion2,expression3)

* 1. Initialization mean to start from , therefore we put int i=0
  2. Expression 1 represents initialization
  3. Expression2 represents conditions,sabse last waali hi terminating statement hai (which means uski hi baat chalegi)
  4. “%d\*%d=%d” it means 2 assigned variable multiply hokar ek teesra variable as a answer dege
  5. 

1. There are two types of fn
   1. In this case,

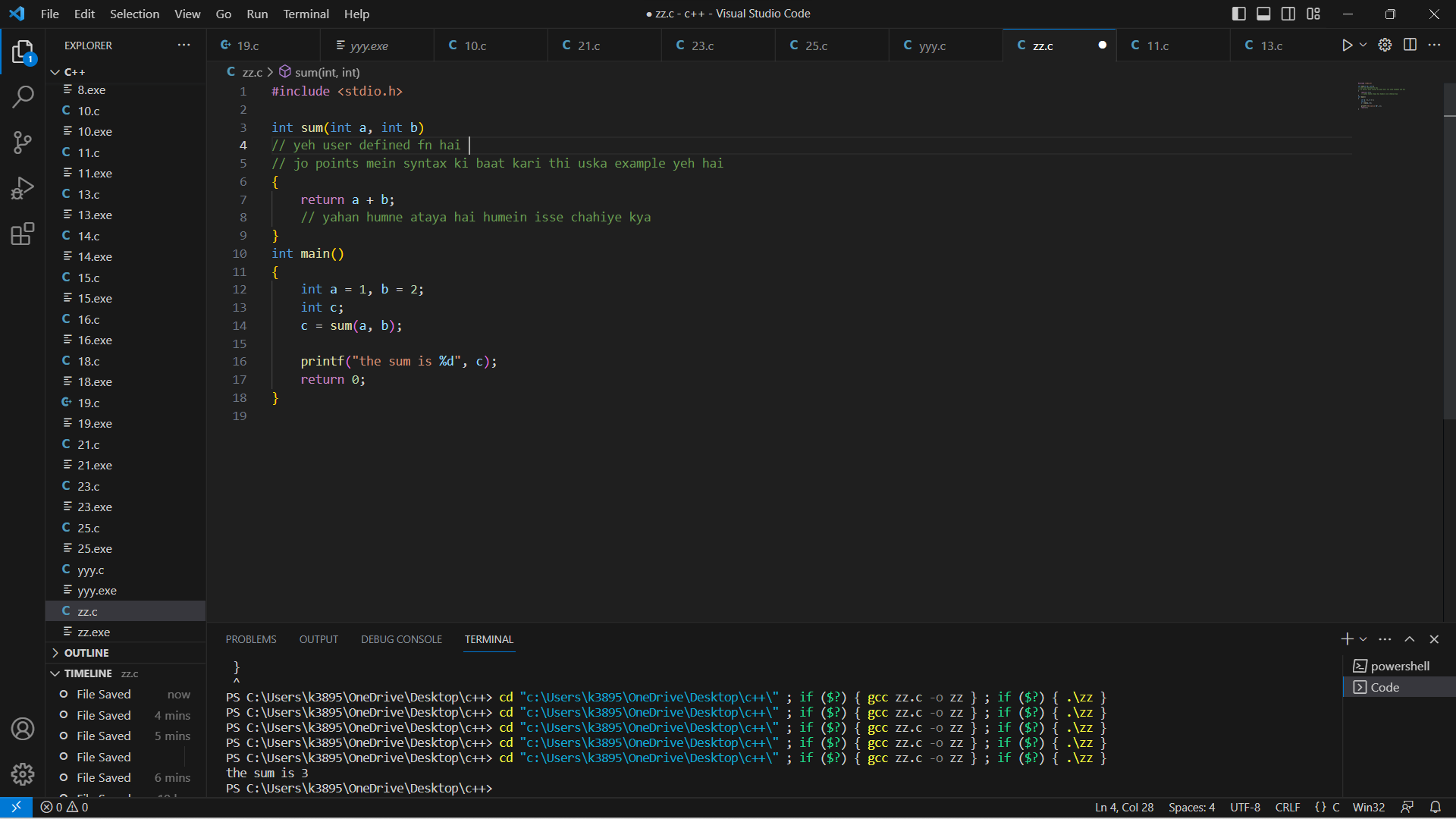
User made(int sum)and library function(printf,scanf etc)

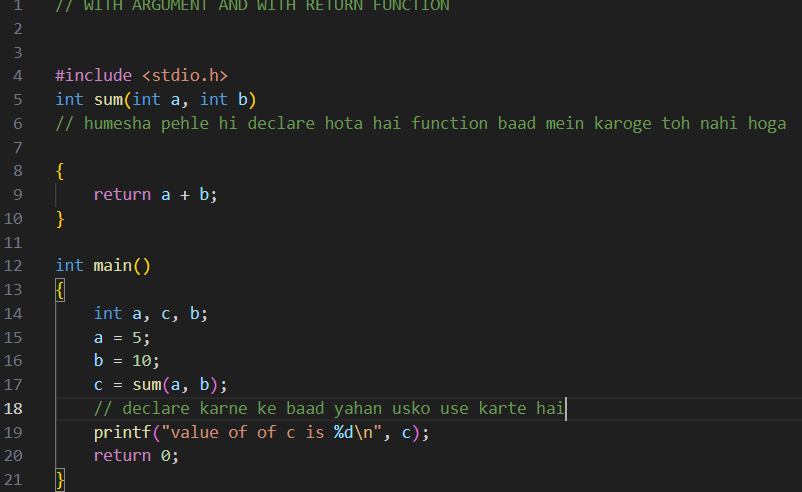
* 1. Basic syntax of fn (“return type” “function name” “data type” “parameter 1”
  2. In a user defined function we have three tasks

a)declaration : we have declared here our function “sum”

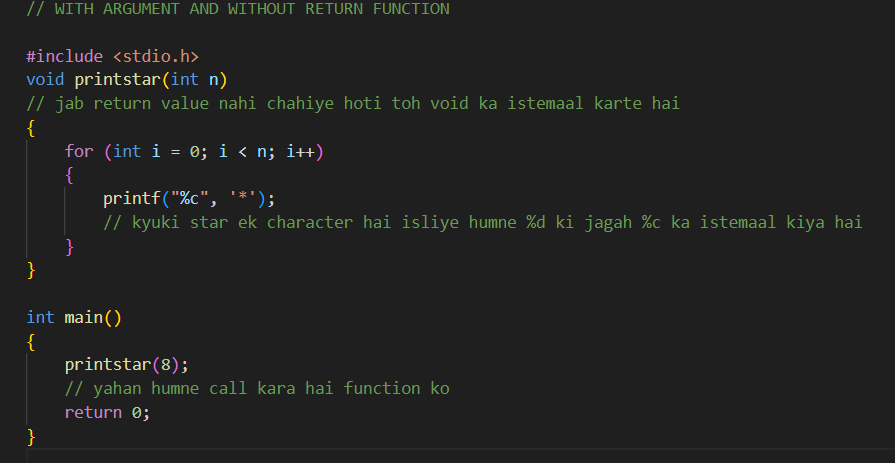
b)definition: we define our function that is we tell what a function will do

c)calling:when we have created the function and we want to use it somewhere we simply just use the name of function in this case “sum”

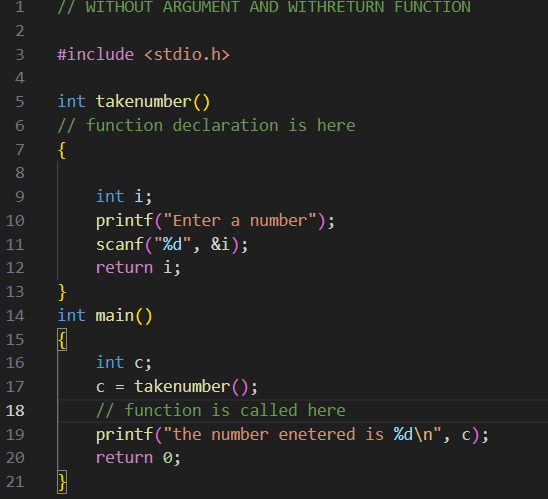




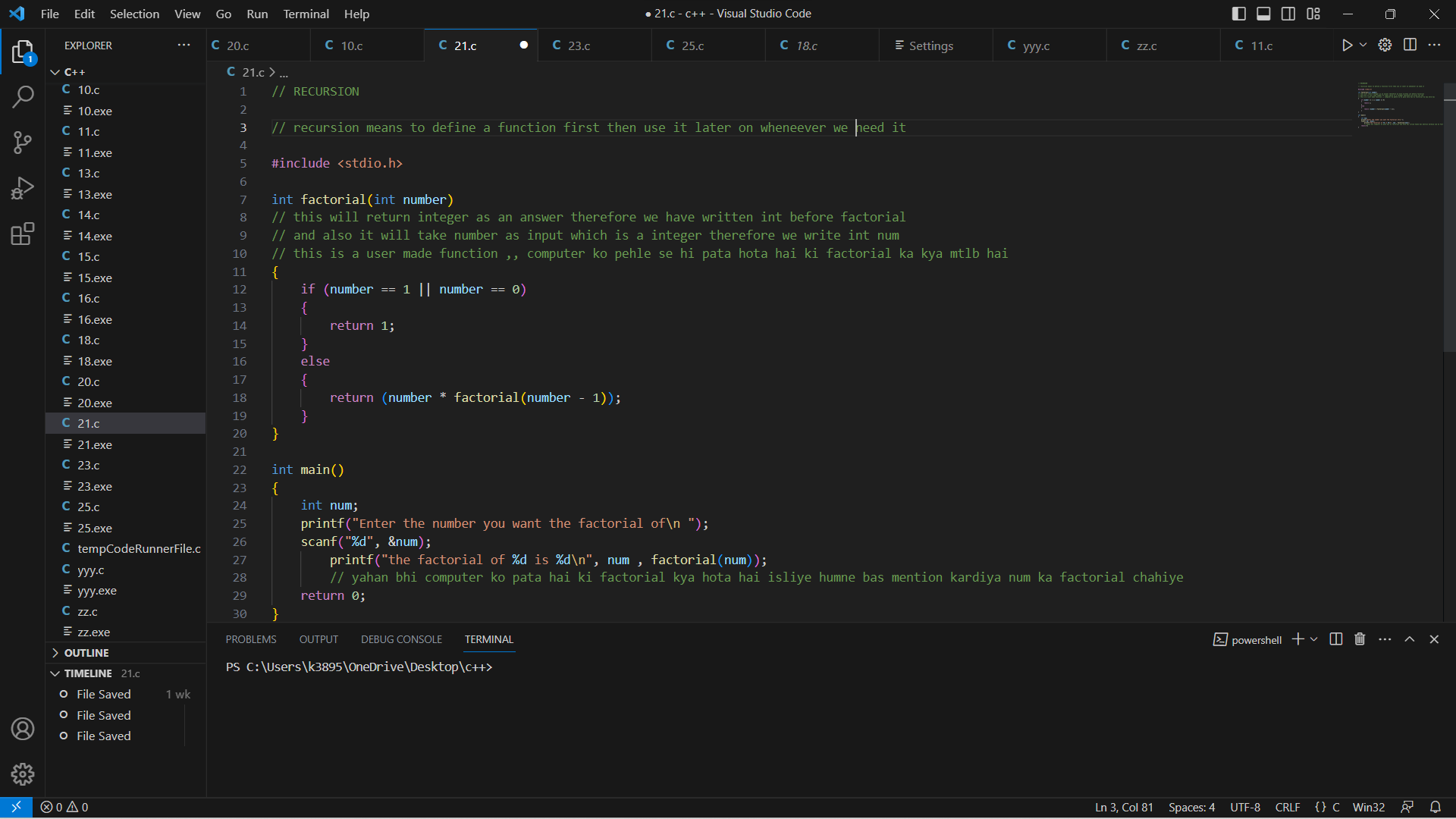
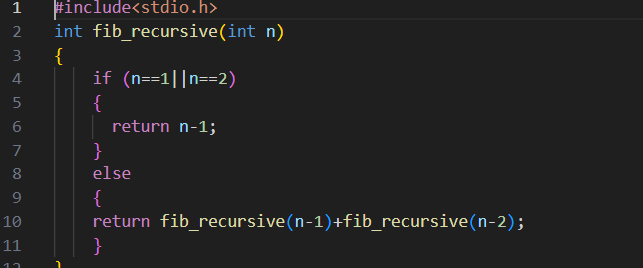
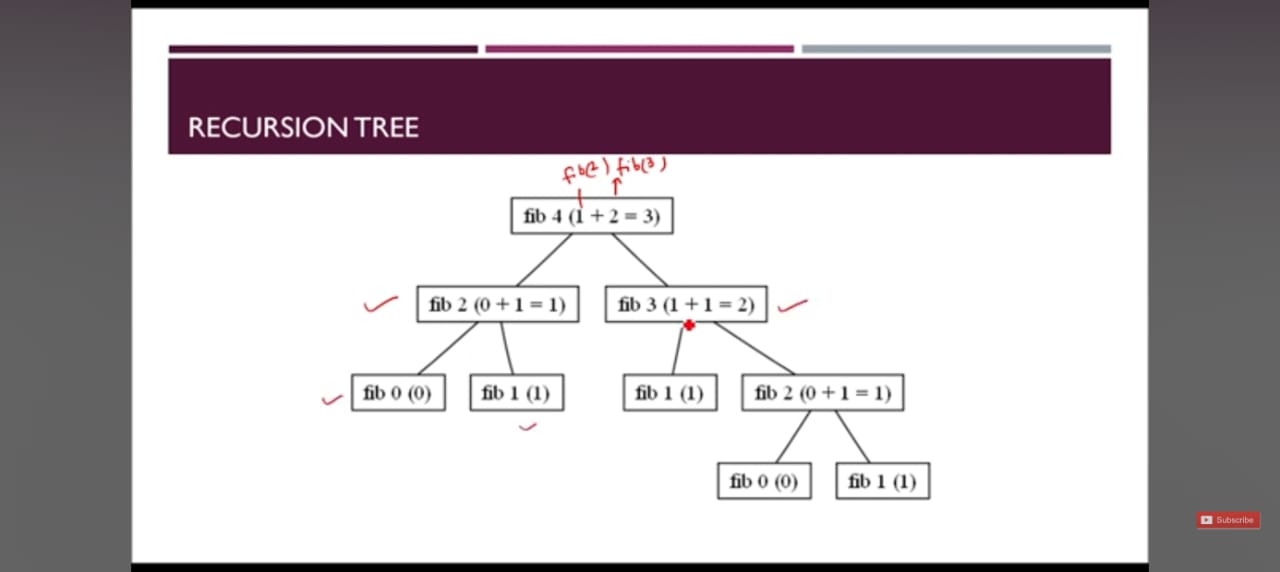
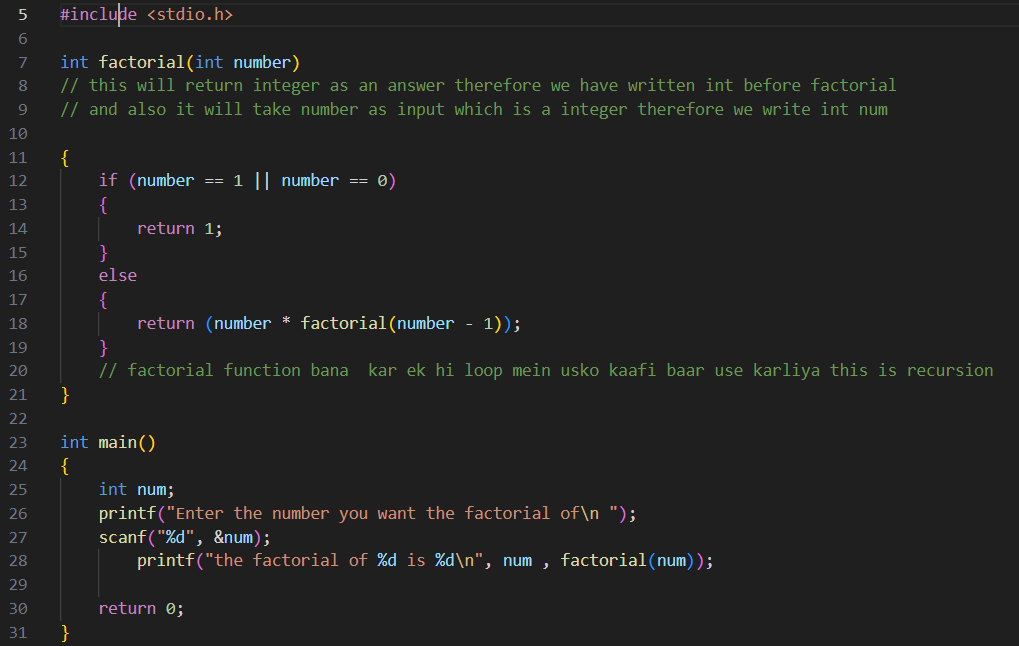
* 1. Here arguments are a and b that is the parameters present in the first line
  2. We have given a return value to our function that is a+b
  3. Later on we have called the function using its name
  4. We can also see that we have mentioned already that what kind of parameters can be used while using this function,
  5. so when we called the function we mention in brackets what the parameters are

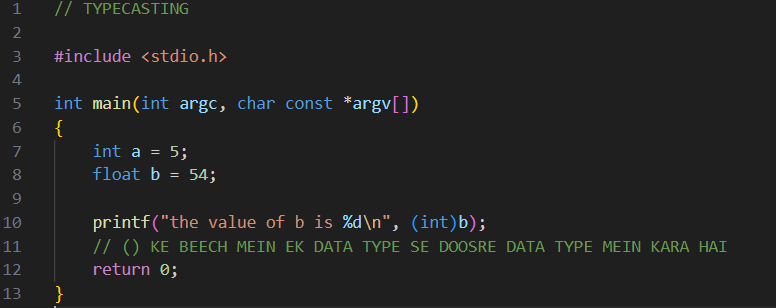
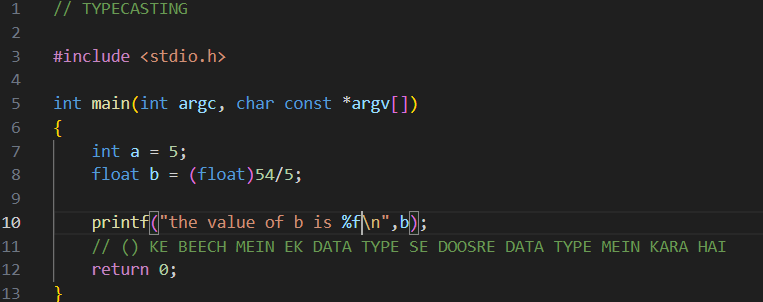


* 1. Here we can see we have a argument that is “n” in this case
  2. But there is no return value of function that is function will not return anything
  3. But,
  4. It will print star on the screens , according to our code(output on the screen isn’t thought as a return value)



* 1. Here as we can see there are no arguments (no arguments while declaring a function)
  2. But when we our defining task to function we can take arguments from user
  3. We always call a function by only writing its name

1. * 1. 
   1. Recursion means to create a function to do a specific job and use it later on as many times as we want to do the same job.
   2. Suppose there is function “prepare”
   3. We use the same “prepare” fn to make roti, to make daal ,to make raita, basically “prepare” ek function hai and uska kaam hai prepare karna .
   4. Ek hi function ko hum alag alag jagah istemaal kare hai isliye yeh recursive approach hai.
   5. Vo ek function apne andar bhi istemaal ho sakta hai baar baar( same loop mein hi )
   6. For eg:
      1. Factorial of 5=5!
         1. =5\*4!
         2. =5\*4\*3!
         3. =5\*4\*3\*2!
         4. =5\*4\*3\*2\*1!
   7. Iss code mein logic yeh hai ki kisi bhi number ka factorial nikalne ke liye base condition ka istemaal hoga
   8. In this case 5 ka nikale ke liye
   9. Hum block of code mein jaate hai
   10. Jahan vo kehtahai ki if n==1 or n==0 then return n=1
   11. Else return, number(5 in this case)\*factorial of number -1 (4 in this case)
   12. Further 4 ka factorial bhi aise hi calculate hoga that is 4\*3!
   13. And so on till we reach our base condition that is n=1 or n=0
   14. Which we will reach at factorial of 2
   15. When we get to know value of factorial of 2 using base conditions
   16. We will know value of factorial of 3 as factorial of 3 = 3\*2!
   17. Similarly we will finally get to know 5! Using same procedure.
   18. Yeh jo “!” function baaar baar istemaal hora hai , yeh hi recursive approach hoti hai.
   19. Iterative function is the one where we define a function and use it at the moment.
   20. (function ko alag alag jagah call karna recursion nahi hai balki uss function ko alag alag jagah istemalll karna output paane ke liye vo hota hai)
   21. 
   22. Recursion mein problem yeh hai ki ek hi cheez ko baar baar calculate karna padhta hai ( in this case that is the base condition i.e n==1||n==2,base condition ke elava usse calculate karna nahi aata ,vo tod tod kar baat base condition tak le aayega and usse calculate karke saare answers batayega)
   23. Tabhi humein "fib\_recursive" likhna padha else condition mei ( taki “fib\_recursive” name ke function mein state kri gyi base conditions istemaaal kar paaye tabhi we use fib\_recursive)
   24. 
   25. Taaki ghum firkar jab vo base condition par aayega toh vo "fib\_recursion"(user defined waale function) mein jo hai humne define kara hai vo usse use kar sake.
   26. Iterative approach is to create a user defined function and use it at the moment .
       1. 
2. typecasting is changing one data type to another
   1. When we do any operation bw two integers answer will always come in integer and same for float



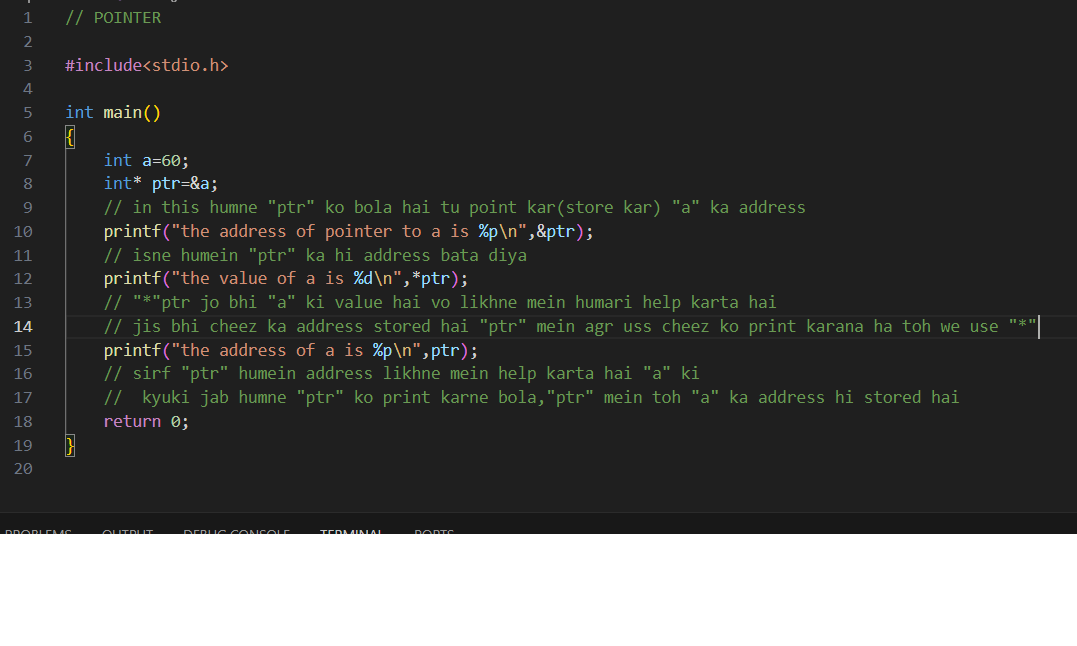
1. pointer means to point at something

“\*” is used to declare a variable as a pointer

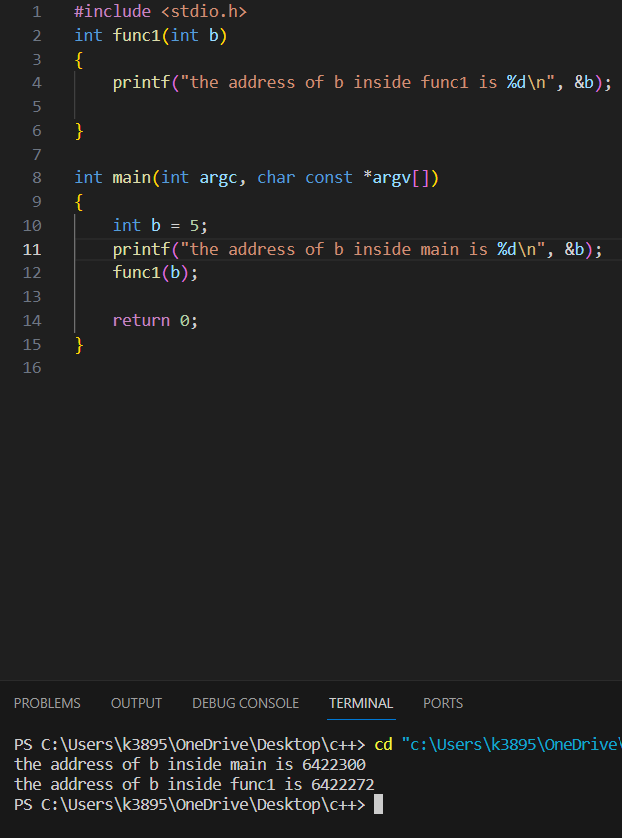
We need to declare a variable storing an address as a pointer **variable because it is not possible in C for a normal variable to hold address of anything thus pointers are important**.

P=&a(this will give an error)

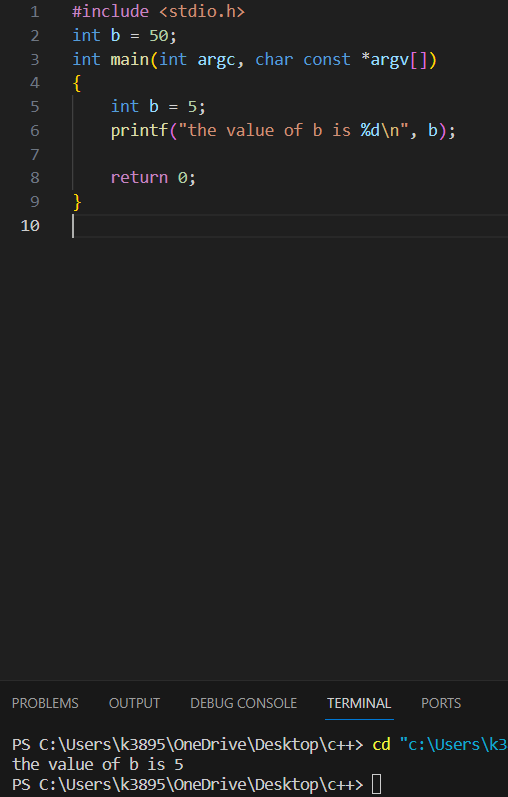
\*p=&a(this is absolutely fine)

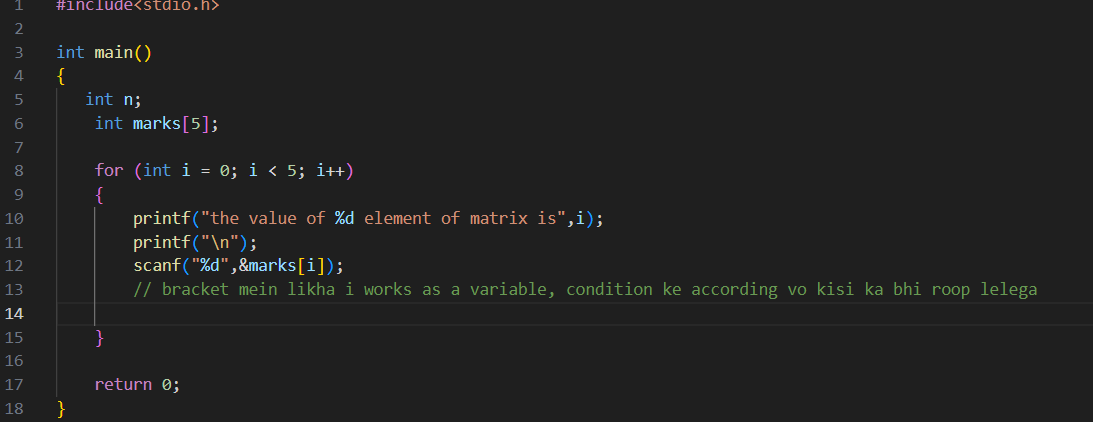
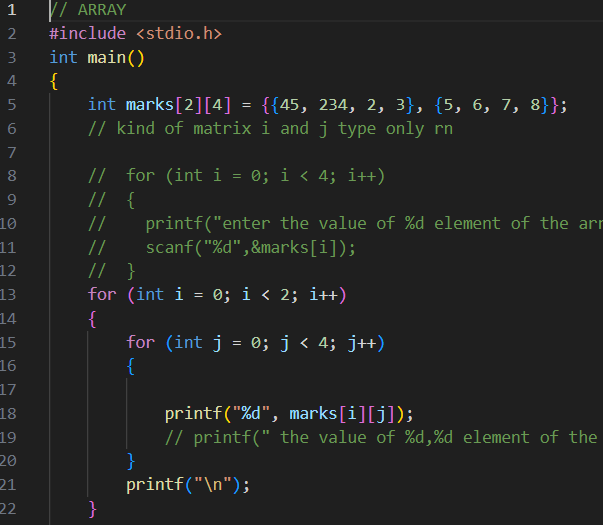
* 1. It is a variable(ptr) which points at address of another variable
  2. 
  3. 

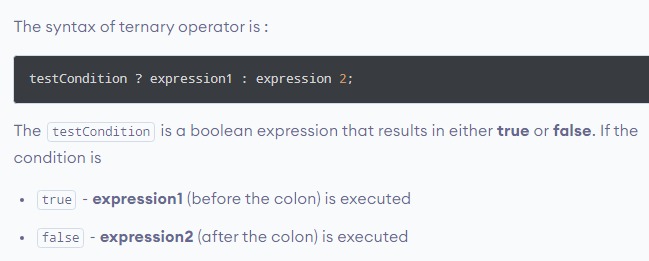
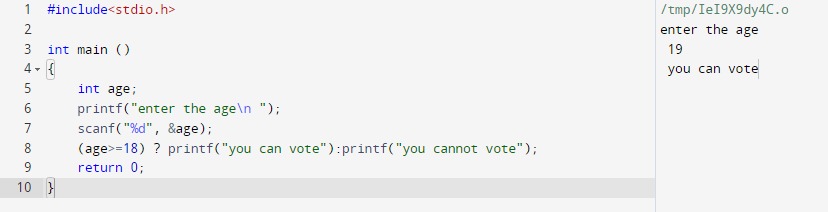
This simply shows that when variable is passed through “main function” it is copied to the “func1” and thus it is not actual variable (but a formal variable), therefore the addresses of both the “b” variable are different



This simply shows that local variable(the variable present inside the main function which can only be used by the function it is mentioned in) is given more precedence than global variable( the variable mentioned out of main function and can be used by any function)



1. Array is basically like a matrix
   1. 
   2. 
2. TERNIARY OPERATOR



1. call by reference

#include<stdio.h>

void swap (int \*x,int \*y)

/\*

idhar bracket ke andar yeh likha hua hai ki

as parameter “integer pointer” aayega

\*/

{

int temp;

temp= \*x;

// temp ko 34 kardiya

\*x=\*y;

// a=b kardiya means a=74

\*y=temp;

// before this temp = 34, now b=temp kardiya basically saying b=34

}

int main(int argc, char const \*argv[])

{

    int a=34,b=74;

    printf("%d and %d\n",a,b );

    swap(&a,&b);

/\*

In the function above “swap”

Parameters were “integer pointers”

So by writing “&a,&b” we have mentioned the “variables”

To whom those “integer pointers” will be pointing

    /\*

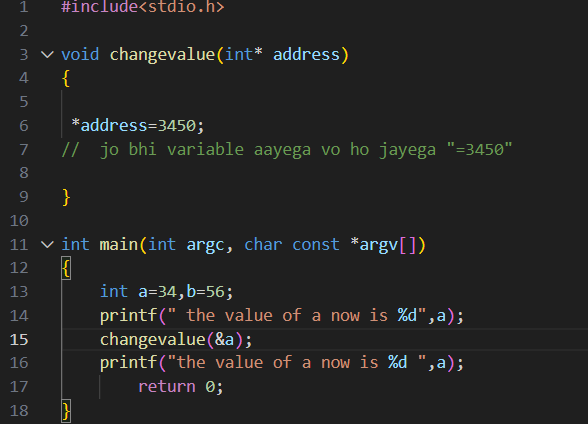
kyuki as a variable sirf address valid hai isliye "&" ka use hua hai

\*/

    printf("%d and %d\n",a,b);

        return 0;

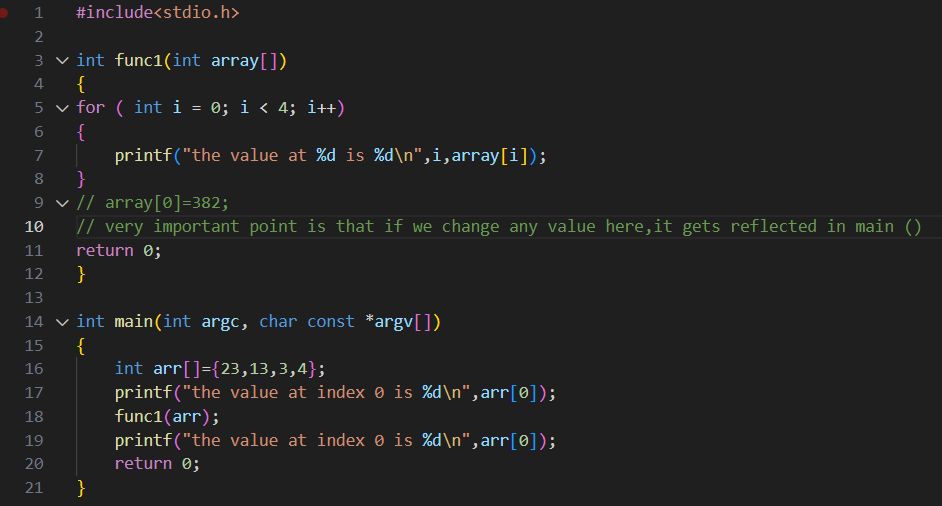
}

* 1. 

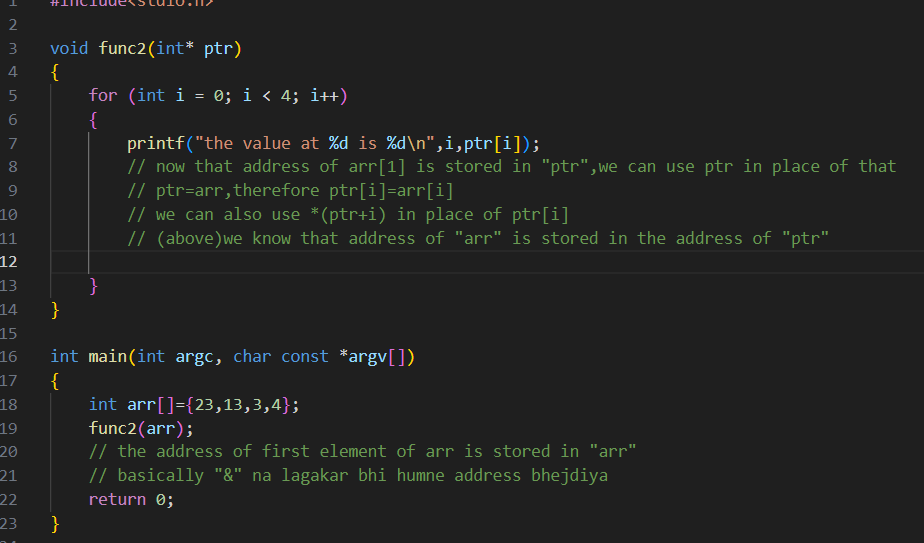
Jo bhi variable ka address function mein jaayega ,,,uss addresss mein stored value hogi 3450

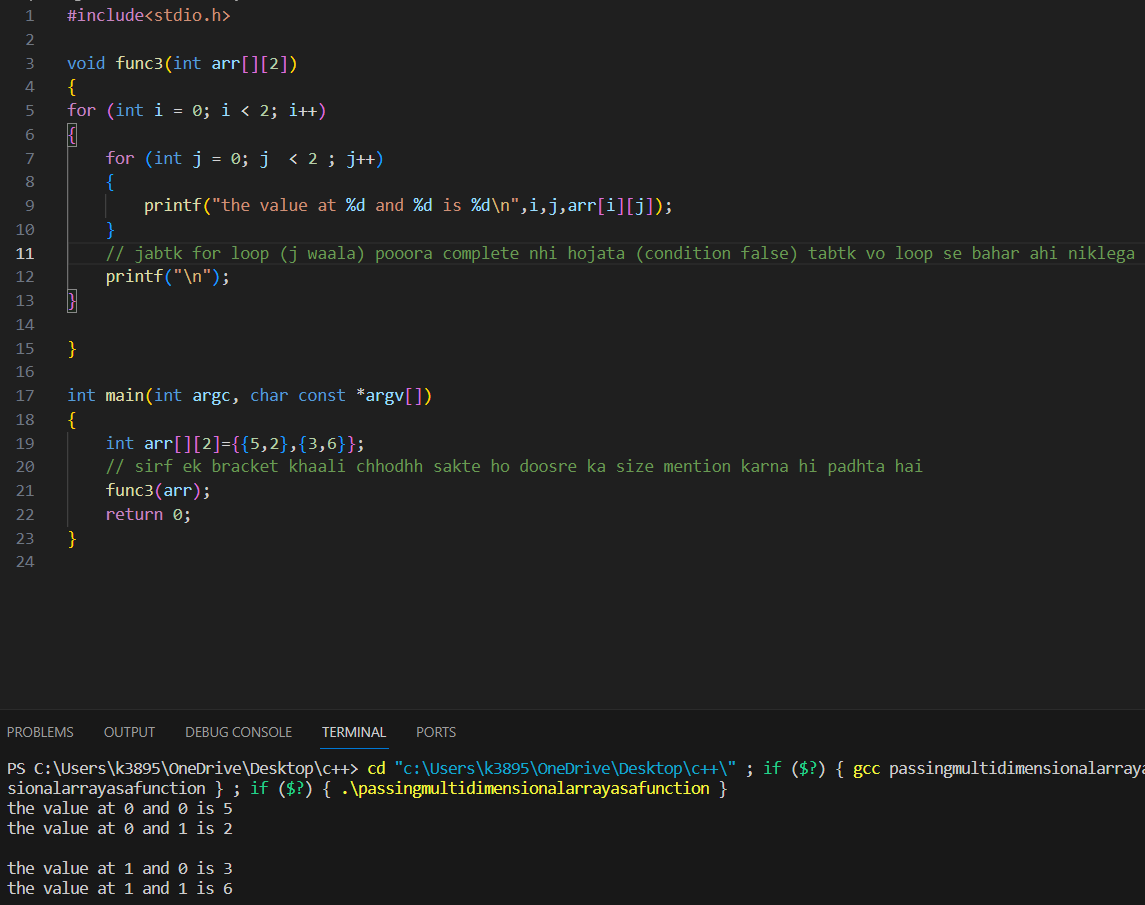
( what I have written in comments of above code is not particularly right)

1. We can pass an array to a function by two ways(it cannot be passed as a simple variable by mentioning it as a parameter in the user defined function), we need to use proper form array is potrayed as <arrayname>[],,, “[]”is very important)
   1. First, to mention it in the function



Pehle 23 show hogi value uske baad 382

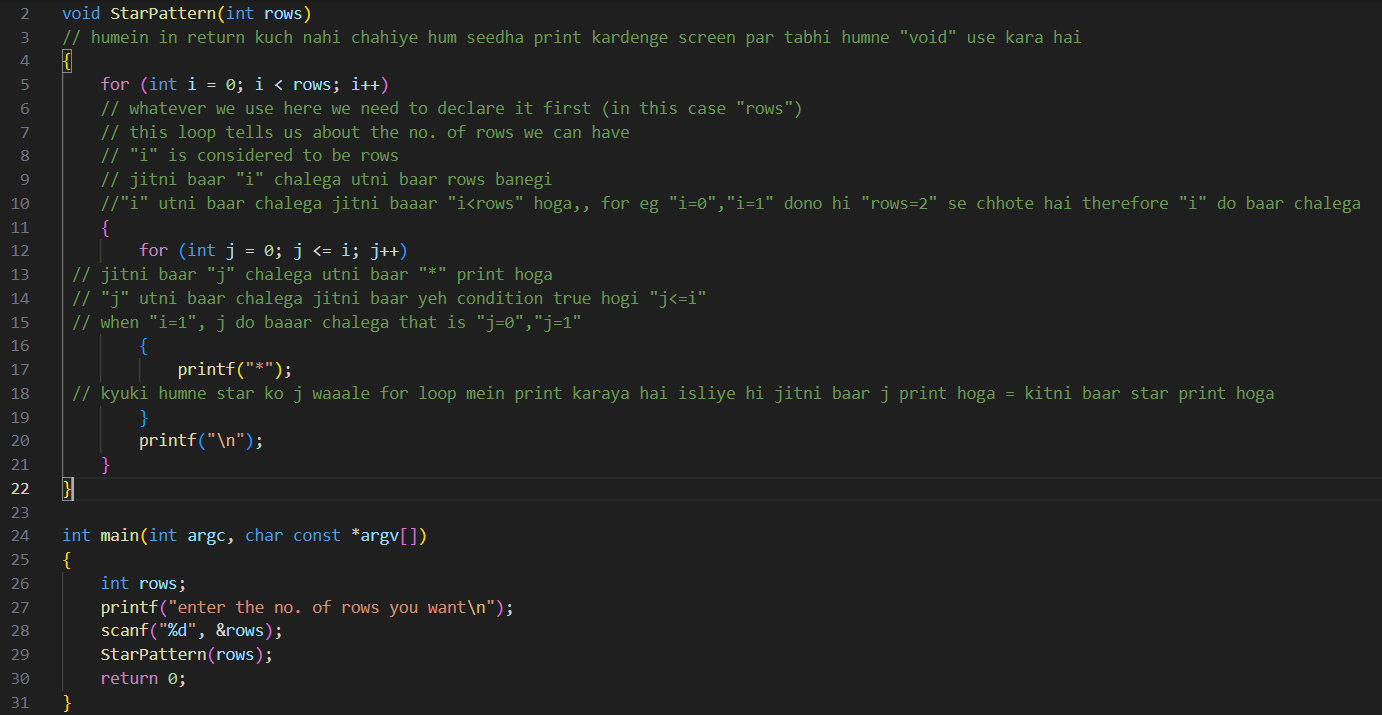
* 1. Second, use a pointer which points towards the array
  2. 



1. Starpattern

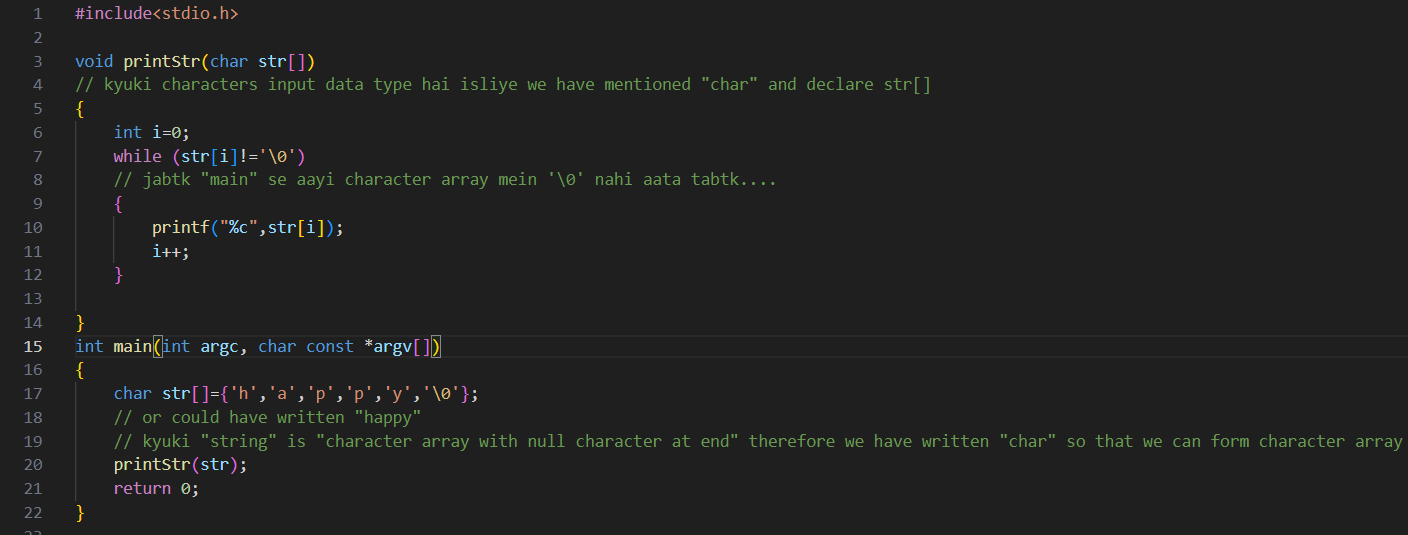
**Working of for loop**

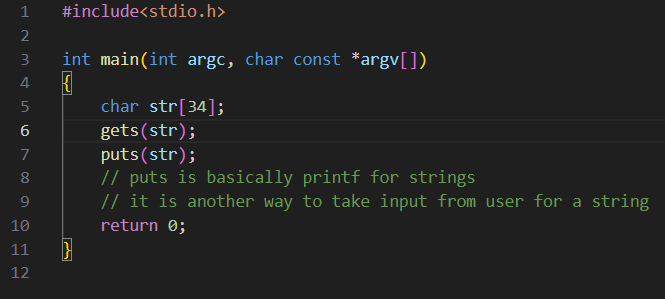
* 1. The initialisation step is executed first, and only once. This step allows you to declare
  2. Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
  3. After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement.
  4. The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

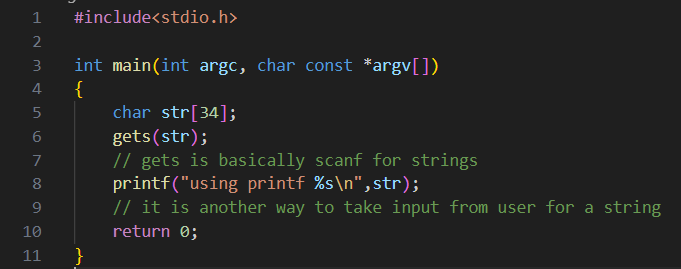


1. STRING

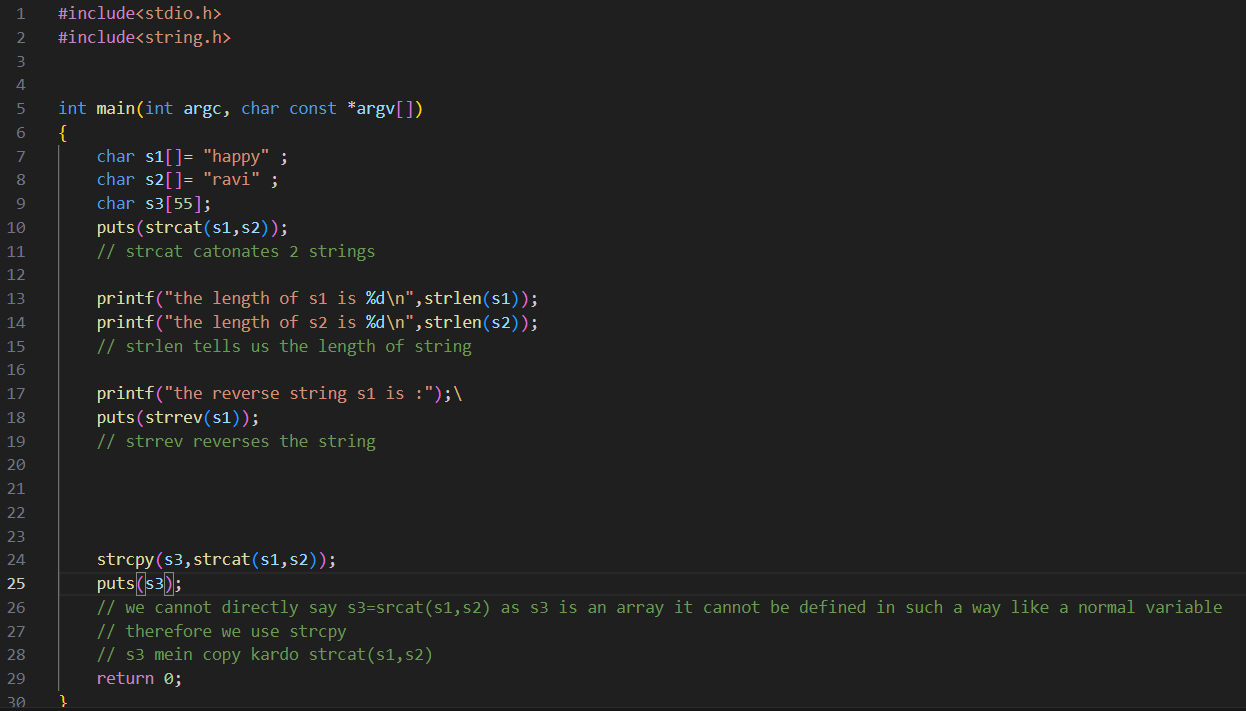
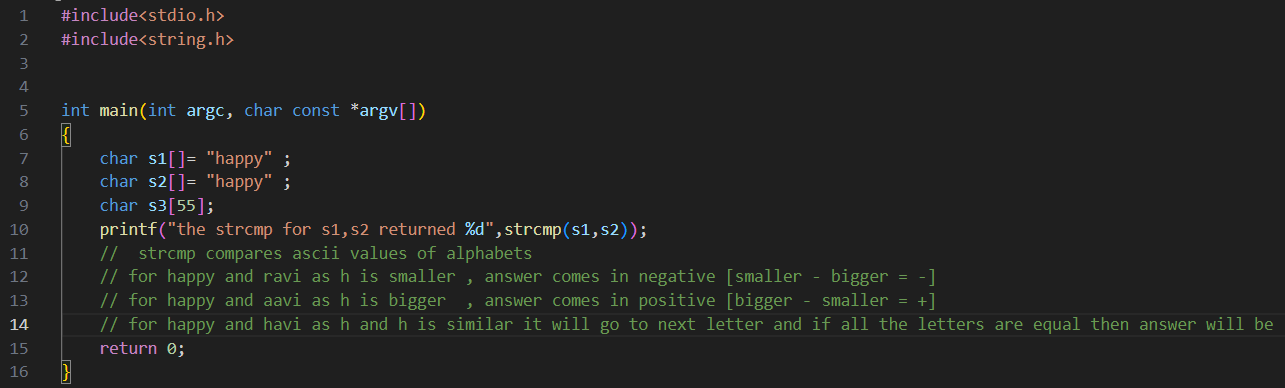
Printing a string on monitor



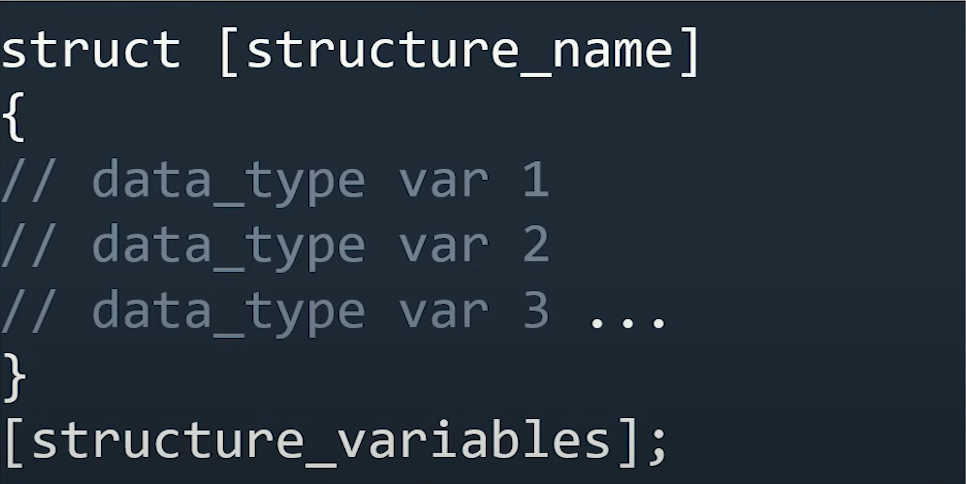




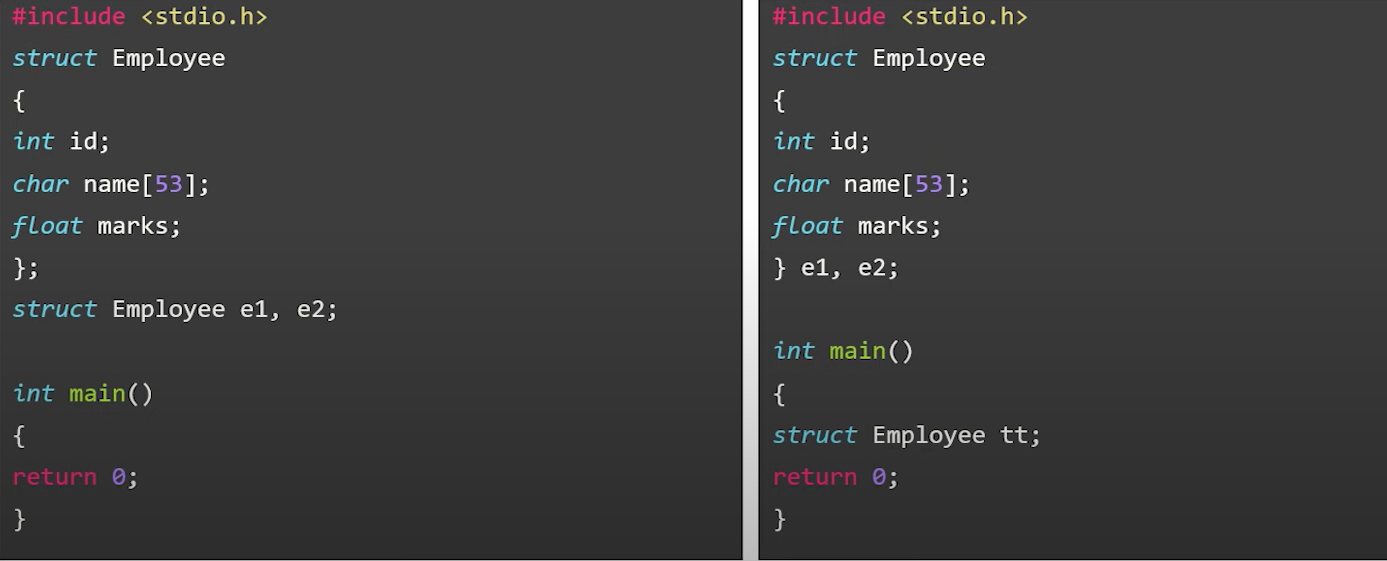
1. String.h functions



1. 
2. Structures is the data type which is user defined and is made by combining any other data type (int, char,long, etc….)
   1. Syntax of structure

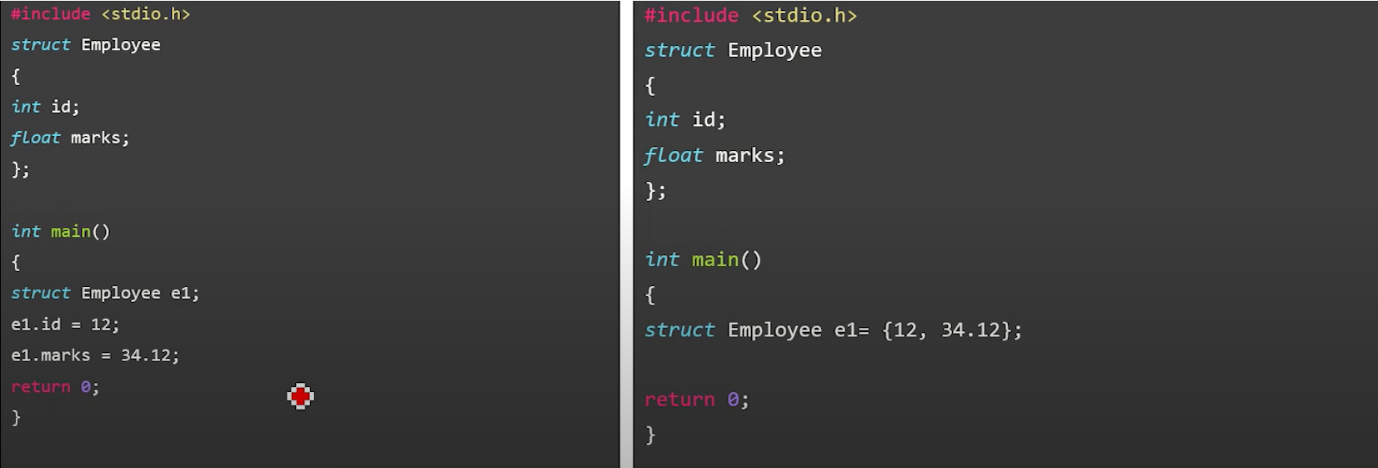


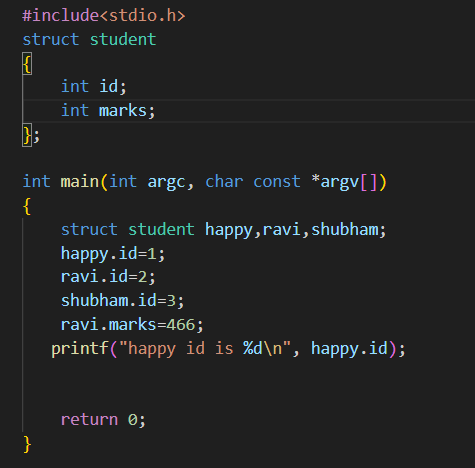
* 1. We create a new data type structure then to use it we create its variables which can be defined in 2 ways (btw e1,e2 ek kisam se ek row/column of data(which has all data typs) ko depict karti hai



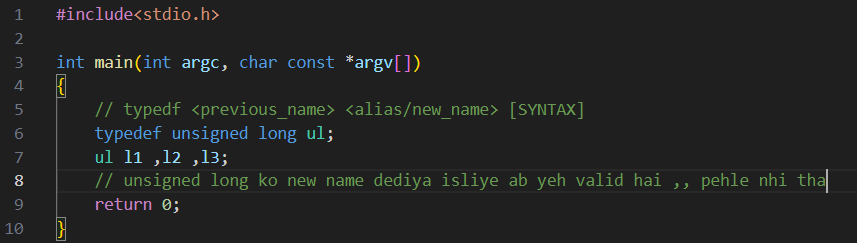
* 1. Structure members are accessed using dot operator[.] or like an array

For accessing using dot operator,we use the operator in between “structure name” and “member name”



* 1. 

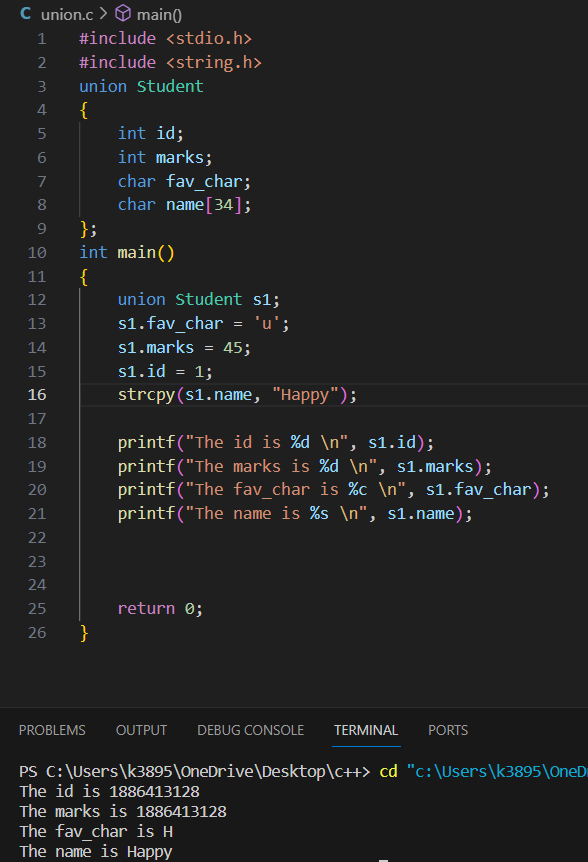
1. The typedef is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program.

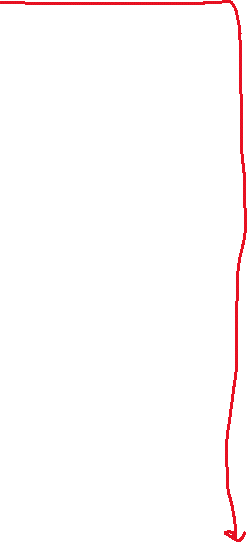
xxxxz

1. The difference between union and structure is

union sirf at a time one variable ko use karne ke liye hai(chahe 4 variables ke option ho lekin 1 define kardiya to baaki saare finish because same memory mein hote hai saaare and vo memory koi ek variable khaa gya)

(alt+down arrow key=code ki ek line neeche (interchange)





Only one worked

Inme se kisi ek ko hi value milegi and jaise hi kisi ek ko value milgyi toh baaki saare bekar hojayenge

Union memory space bachata hai kyuki kisi ek quantity ki hi space banata hai.

1. STATIC VARIABLE

static variable preserves its previous value and is not initialized again.

1. 

VS



BASICALLY, the difference is “static int var” ek baar initialise hogya and value “0” hogyi then value uske baad “0” na hoke uski value increment hokar aayega (waapis 0 nhi aayega)

Whereas,

In normal “int” value waapis “0” hogi kyuki loop wahan se start hota hai toh jitna baar bhi vo loop enter hoga value “var” ki 0 hi hogi.

DRY RUN TO UNDERSTAND MORE

1. Dynamic memory location

When you dynamically allocate memory using malloc or similar functions, you’re asking the system to reserve a block of memory for you without knowing its size at compile-time. This is because the size of the memory block depends on the data being stored, which is known only at runtime.

The only way to keep track of this dynamically allocated memory is through a pointer. The pointer holds the memory address of the allocated block, allowing you to access and manipulate the memory.(we know pointer, Instead of storing a value, a “pointer” stores the “address of a variable” for eg: int \*ptr=&a;

,,where a is an integer variable

But if we only make a “pointer” without assigning an address of a variable to it

We indirectly reserve some space

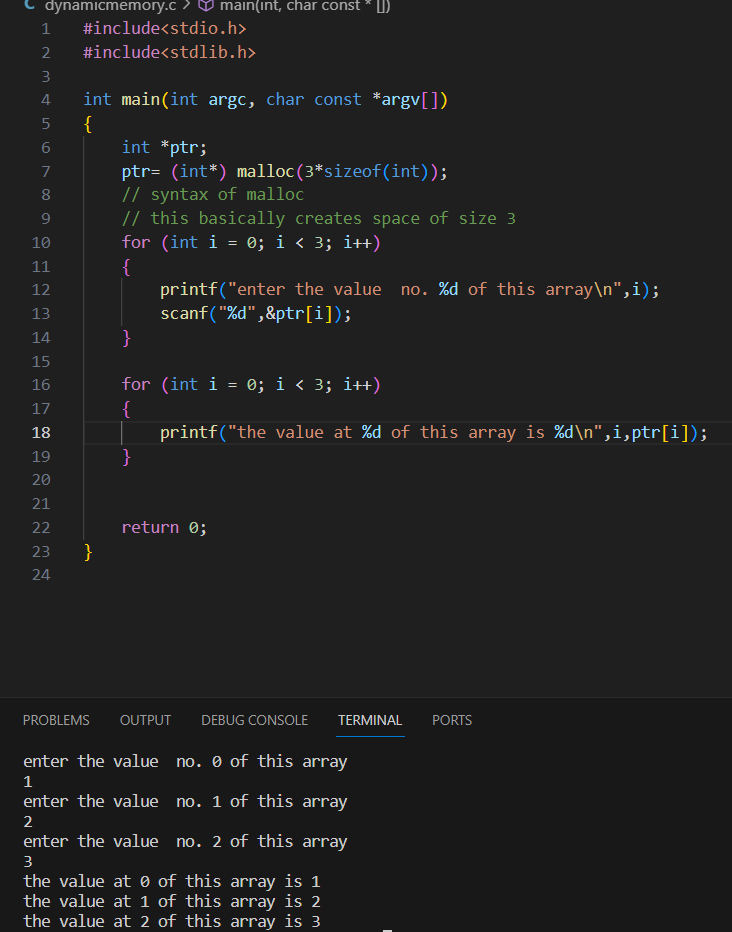
For eg:

Int \*ptr;

In this example “ptr” is a “pointer” of data type “int” which doesn’t store “address of any variable” but indirectly reserving space as a “pointer” holds the “memory address”

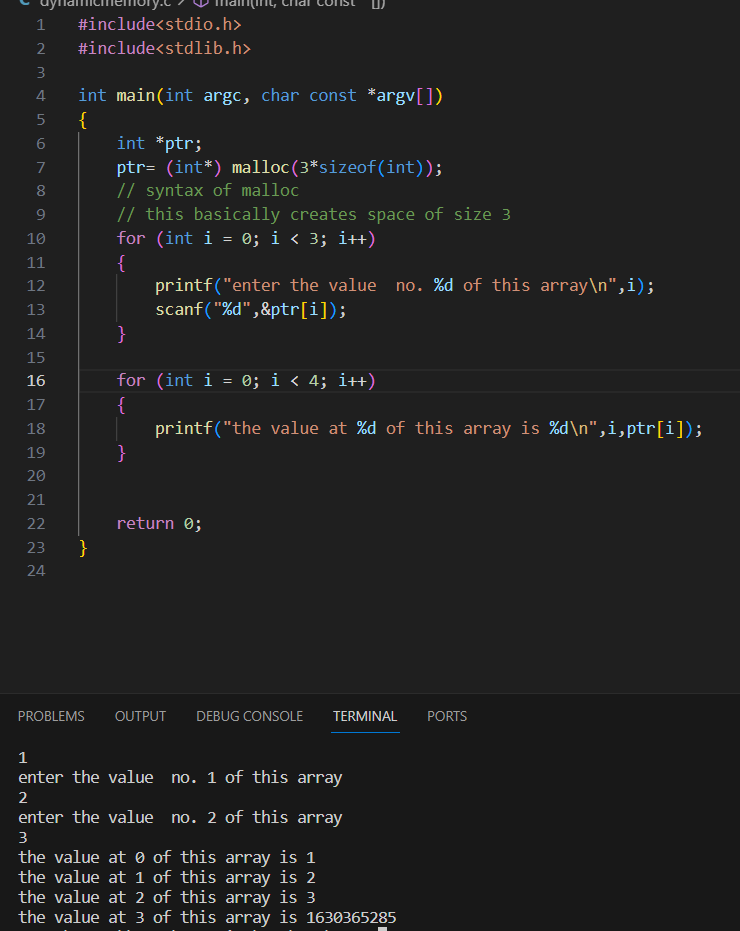
MALLOC- it reserves a block of memory with given amount of bytes

Syntax:(ptr-type\*) malloc(size\_in\_bytes)



\*\*\*\*\*\* space of size 3 array

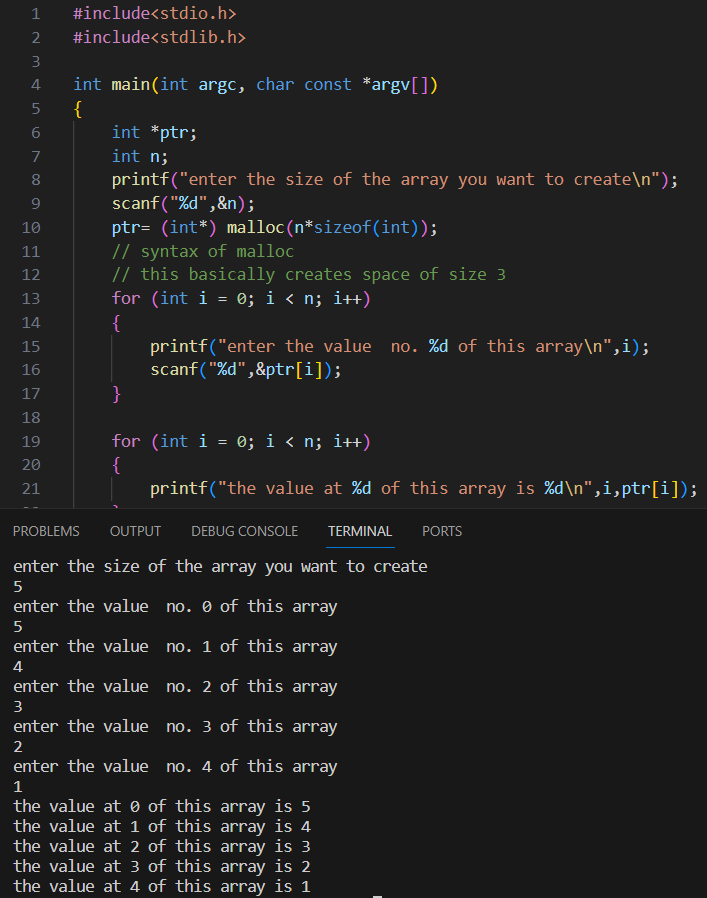
And then, usss block of space mein ,that is, the array mein values store hongi



Garbage value due to,

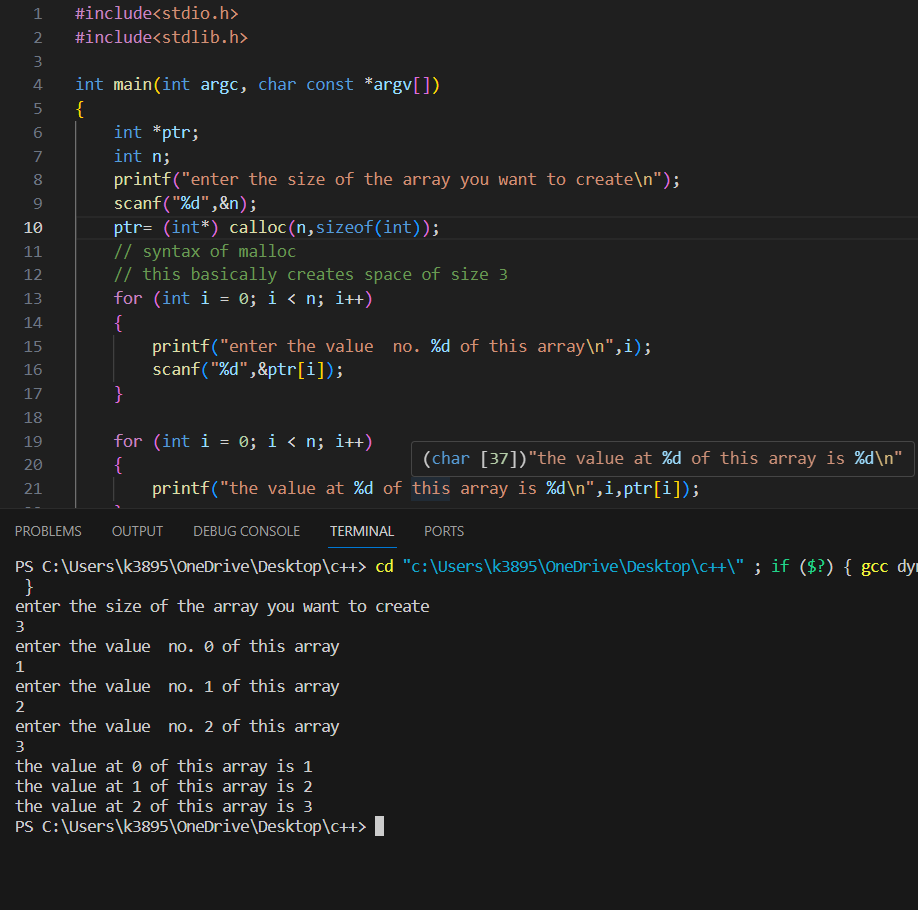
In 16 th line , kyuki humne array banaya hai 3 ka then hum usse zyaada space kaise maang sakte hai

DYNAMIC MEMORY IS MORE EFFICIENT IN TERMS OF MEMORY USAGE AS THE SIZE OF MEMORY TO BE RESERVED CAN BE CHOSEN WHILE RUNNING THE PROGRAM

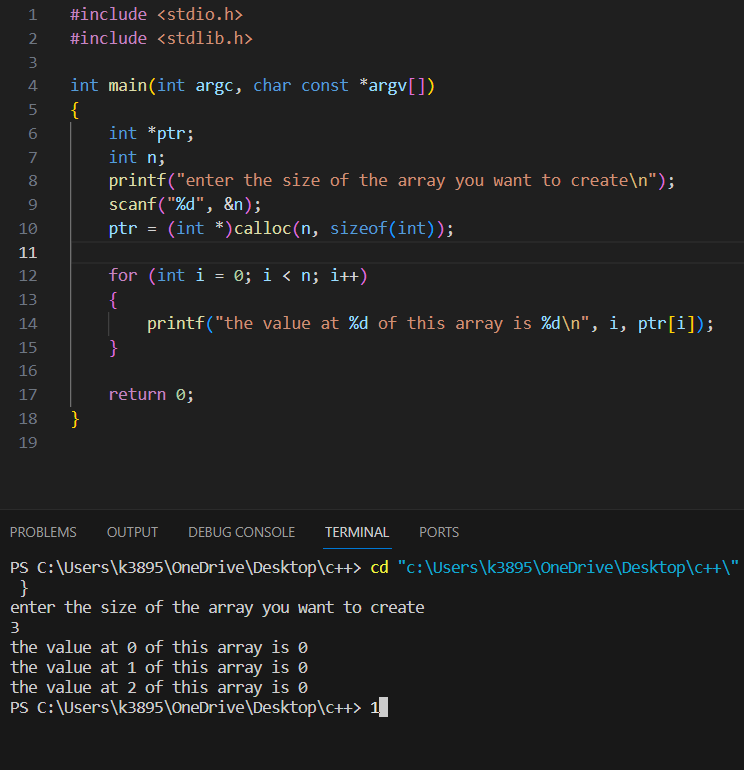


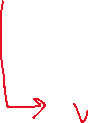
CALLOC-It reserves n block of memory with the given amount of bytes

Syntax: ptr=(ptr-type\*) calloc(n,size\_in \_bytes)



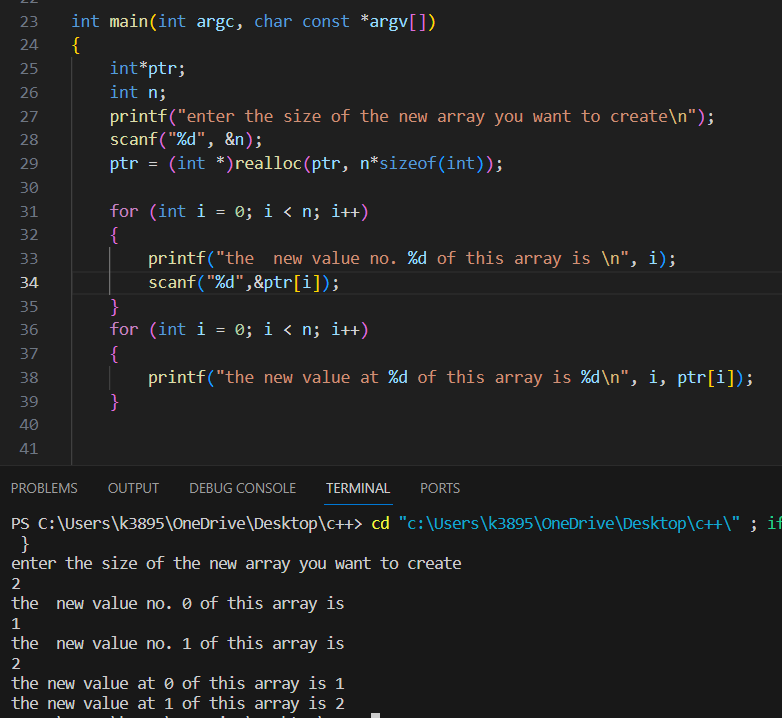
Since we haven’t provided any value it will automatically initialise with a 0 (unlike malloc which would initiialise with garbage value)





REALLOC:if the dynamically allocated memory is insufficient we can change the size of previously allocated memory using realloc() function

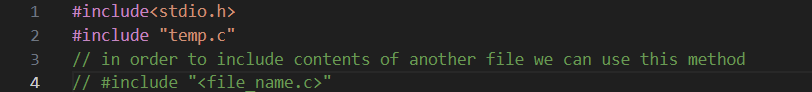
Syntax: ptr=(ptr\*type) realloc(ptr, new\_size\_in\_bytes)



FREE: when the dynamically allocated memory is not required anymore we can free it using free()

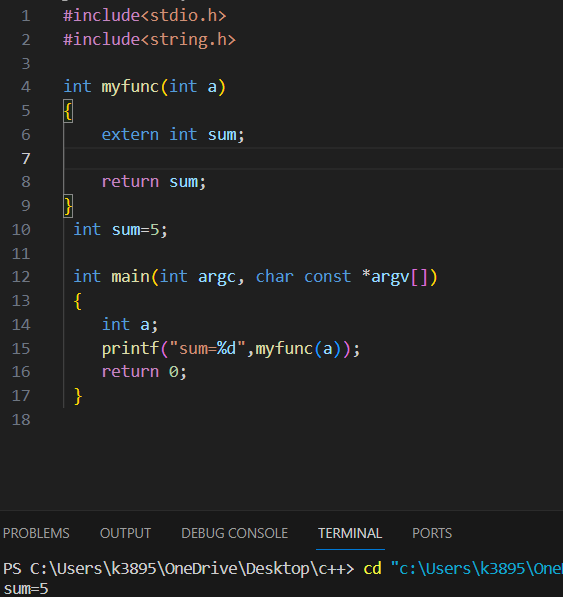
TO USE ALL SUCH FNCTIONS WE NEED TO INCLUDE HEADER FILE STDLIB.H

1. STORAGE CLASSES

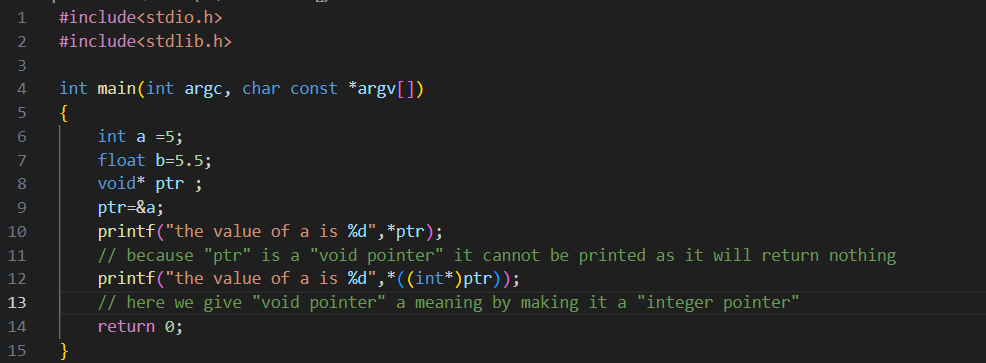


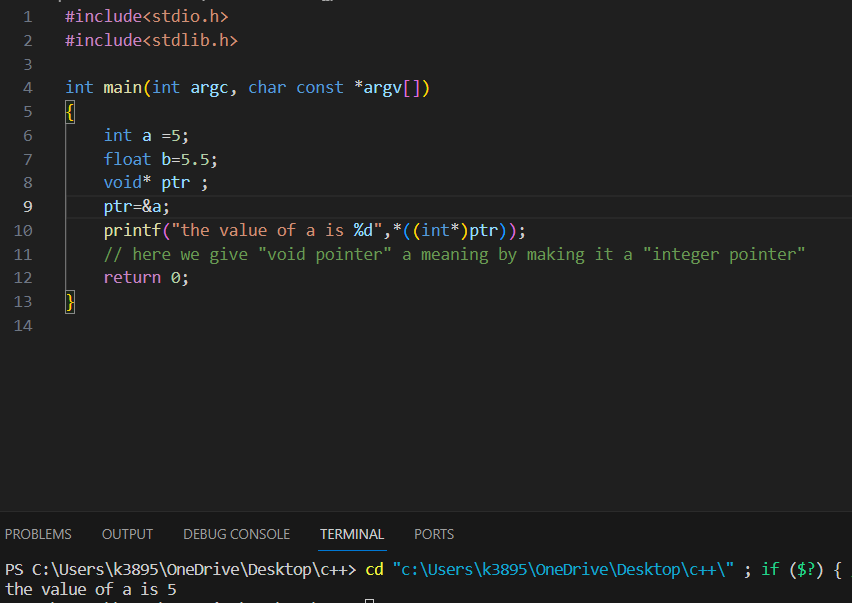
In order to use global variable we use “extern”

(but if local variable is present it will be given precedence)



1. VOID POINTER:







Syntax :

Print whatever is present inside the stored address

Since general syntax of pointer is <data\_type\*> <pointer\_name>

Here also we have given this void pointer a new meaning



(int\*)ptr



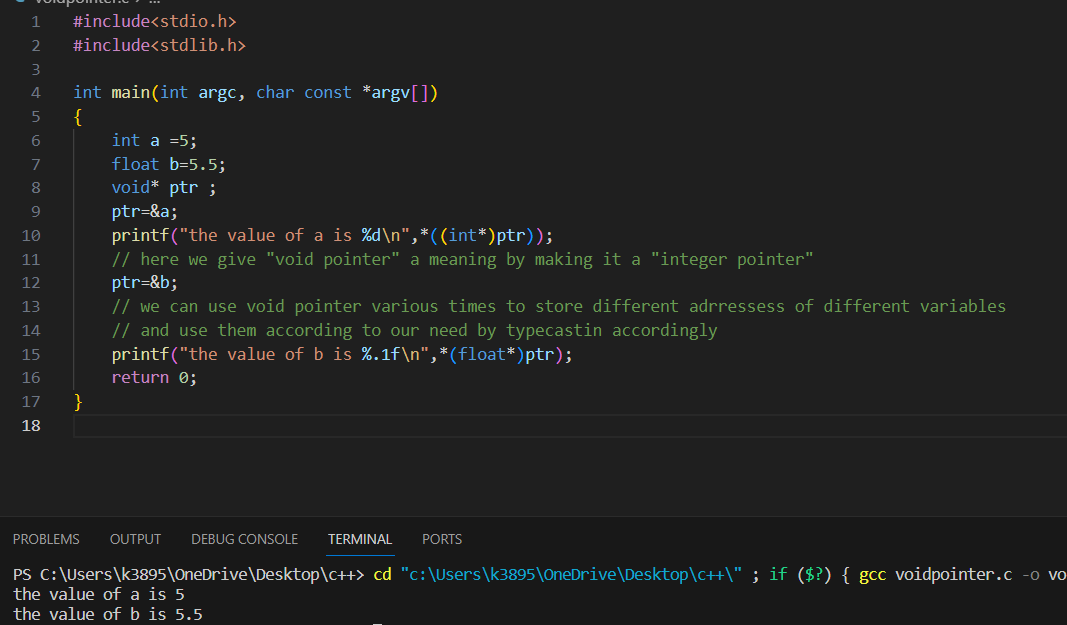
Void pointer is converted to int pointer

\*((int\*)ptr)



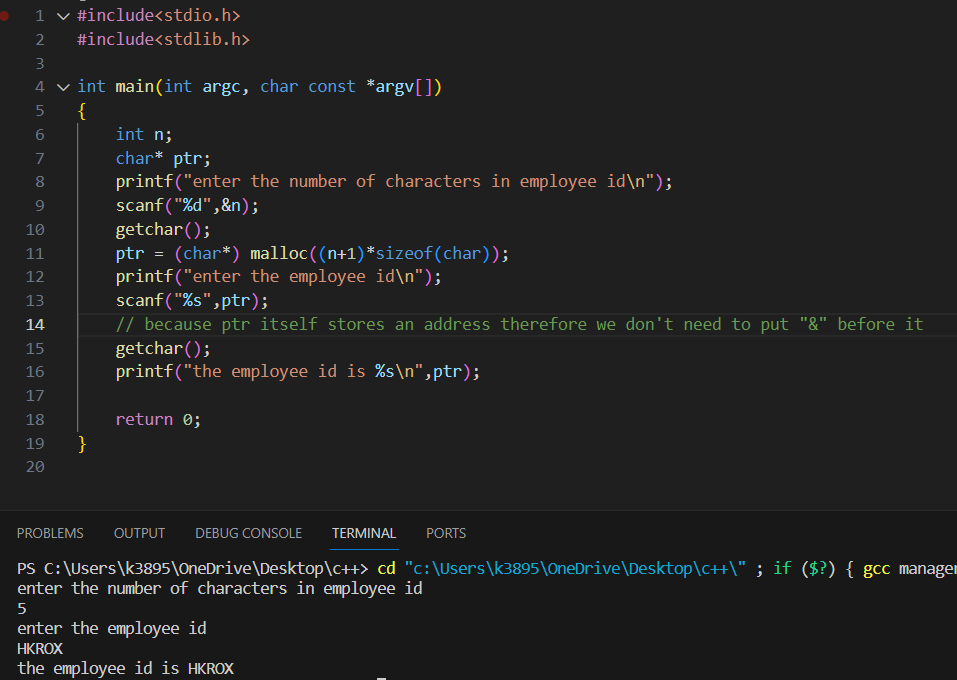
Use of “\*” tells us

Print whatever is stored in int\* ptr



1. Getchar() use is to read one character at a time given in keyboard

(in situations like scanf and printf ek cheez aati hai aur ek cheeez apne aap aaage badh jaati hai we used to think it is because of “\n” but it is because when we press enter after putting the values the computer stores that enter and use it for next “scanf” value and therefore next “scanf” is skipped and we used to think it is because of “\n”)





To allocate memory dynamically we neeed to do it in a pointer(in this case “ptr”)

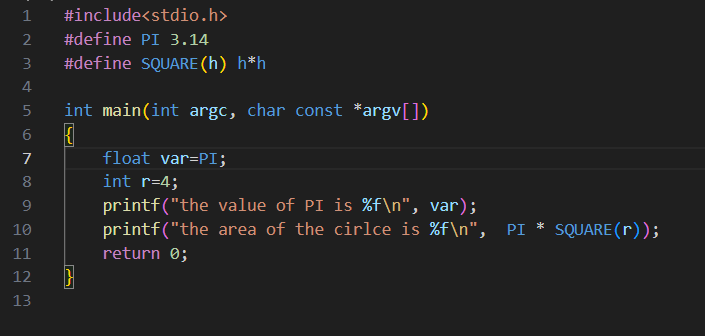
Therefore We need to initialize the pointer first (in this case char\* ptr)



We have written “n+1” because n represents the no. of characters in the employee id and +1 is the null character

Dangling pointer understand from somewhere else than harry.

1. PREPROCESSOR



**fputc():**fputc() is used to write**characters** to the file

****

If it couldn't do so in case of any other issue, it would display an EOF exception. EOF stands for End of File.

#### fputs(): fputs() is used to write a null-terminated string to the file

#### 

#### fgetc():fgetc() works exactly the opposite of fputc()

#### It reads only **one character at a time**.

#### 

#### fgets():fgets() is used to read a string from a file.

#### It takes**three parameters**

#### the second one is the count of variables we want to get from the files

#### the first one is the storage array

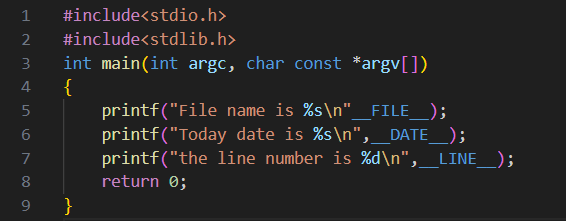
#### The third one is the file pointer

#### 

#### 

1. In C, a **macro** is a piece of code in a program that is replaced by the value of the macro

Whenever a macro name is encountered by the compiler, it replaces the name with the definition of the macro. Macro definitions need not be terminated by a semi-colon(**;**).



1. “->” basically means kisi structure member ke andar ki cheez ko point karna

