C programming course

Created by : Israa Abdelghany
LinkedIn : www.linkedin.com/in/israa-abdelghany-4872b0222
GitHub : https://github.com/IsraaAbdelghany9

Introduction to programming Using C

program contains → data
                 → instructions

- In binary systems storage is called bit contains either 1 or 0
  - we can calculate number of possible compinations of zeros and ones or (max) decimal number in a number of bits using the equation :-

$(2^{\text{no of bits}}) - 1$

- In decimal systems storage is called digit contains from 0 to 9
  - we can calculate number of possible compinations of zeros and ones or (max) decimal number in a number of digits using the equation :-

$(10^{\text{no of digits}}) - 1$

the minimum storage to deal with computers is byte → 8 bit → 0 : 255

Convert binary to decimal

$(010101)_2 \rightarrow 2^0*1 + 0*2^1 + 1*2^2 + 0*2^3 + 1*2^4 + 0*2^5$

$= 1 + 0 + 4 + 0 + 16 + 0 = (21)_{10}$

to represent the negative numbers

0 → Pos
1 → neg

example:-

if most significant bit is 1 → negative

byte 8 bits → -1 : -(2^7-1) => -1 : -128

if the most significant bit is 0 → positive

byte 8 bits → 0 : (2^7-1) => 0 : 127

Programming Languages

① machine Language

0110
1001

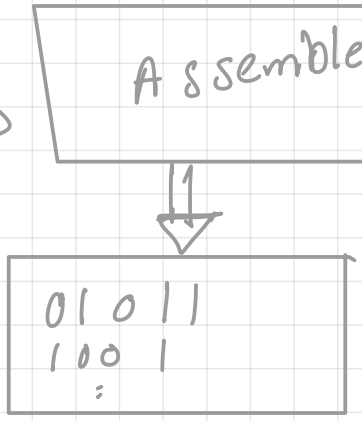
② Assembly Language

```
Load    bc, R2
Load    ac, R1
add     R1, R2
move    R1, ac
```

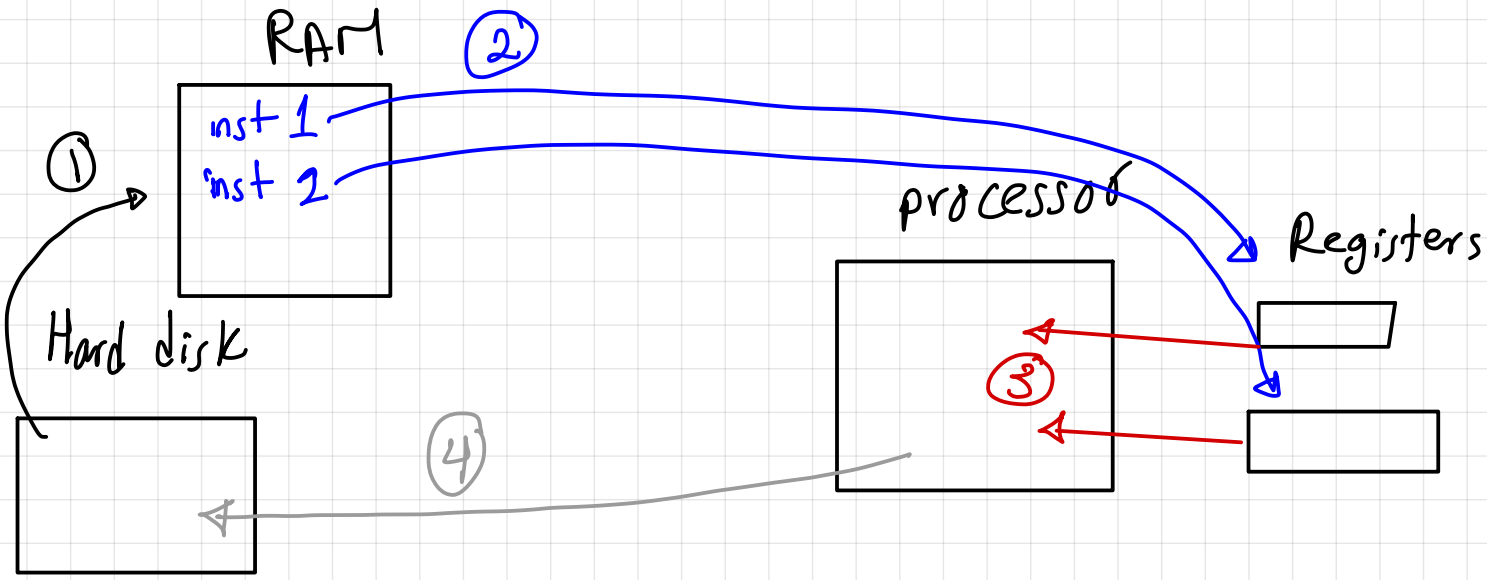
⇓

it has a small program called assembler to convert it to 0 & 1

Low Level Languages →



the general steps

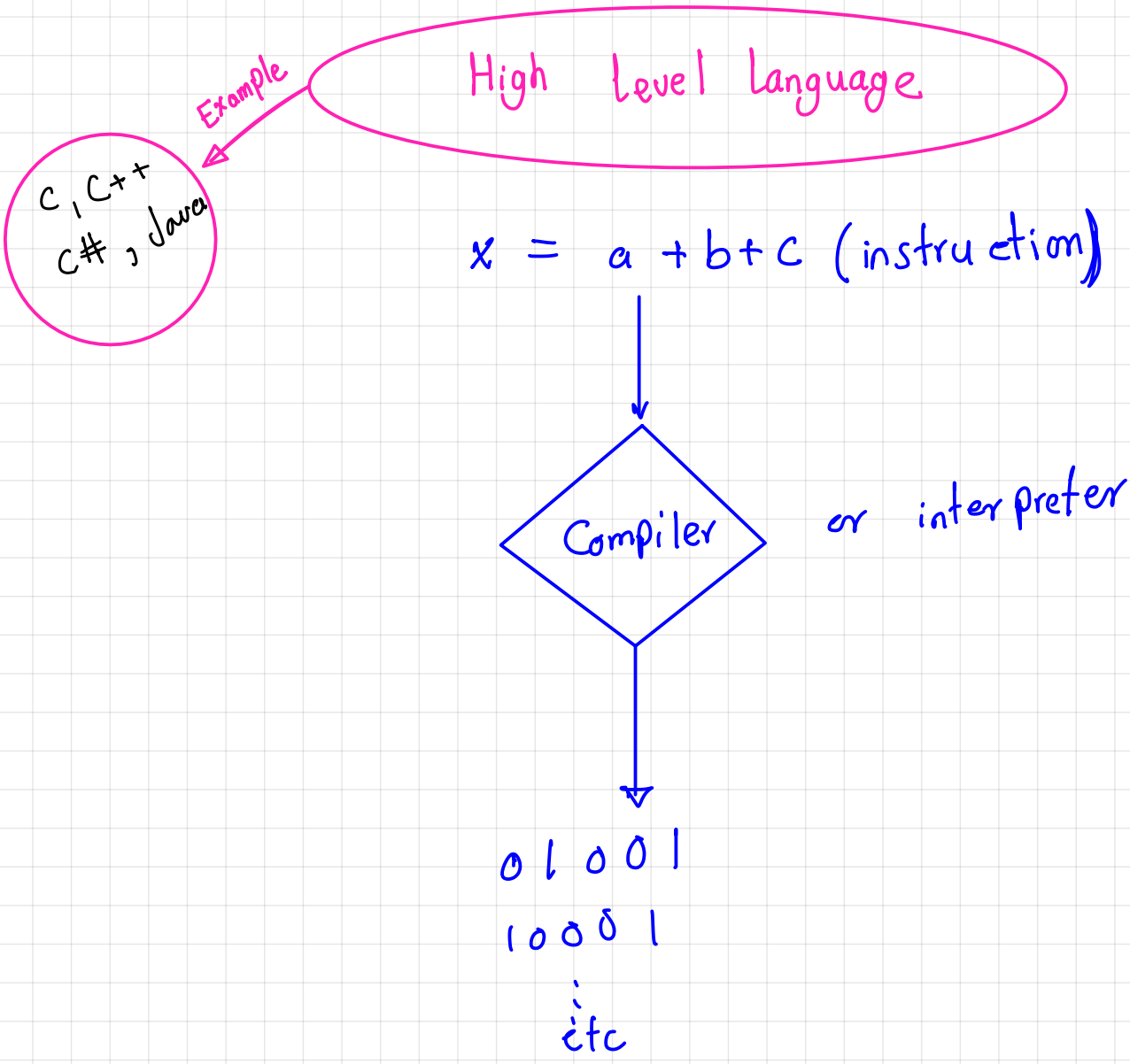


- ① Program is moved from the hard disk to the RAM (RAM is faster than Hard disk)

- ② RAM stores the instructions and move them to Registers

- ③ Registers data then is moved to processor to implement the instructions on data

④ the results of the program are then sent back to the Hard disk till you call it again

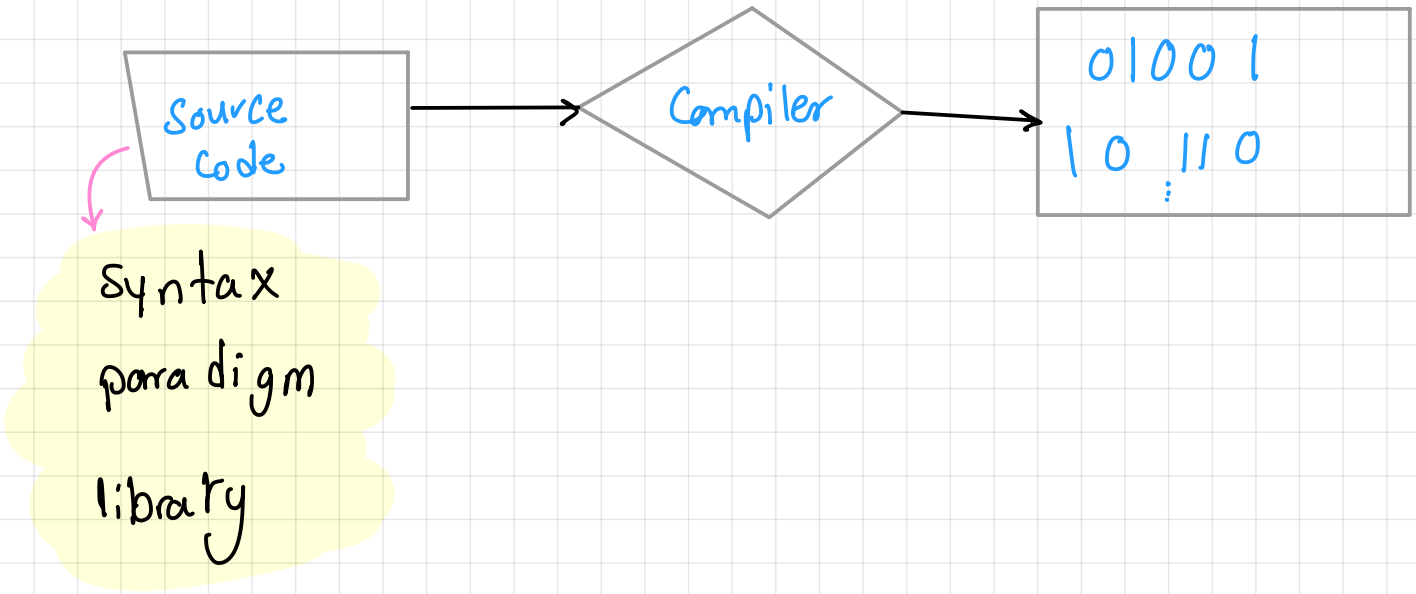


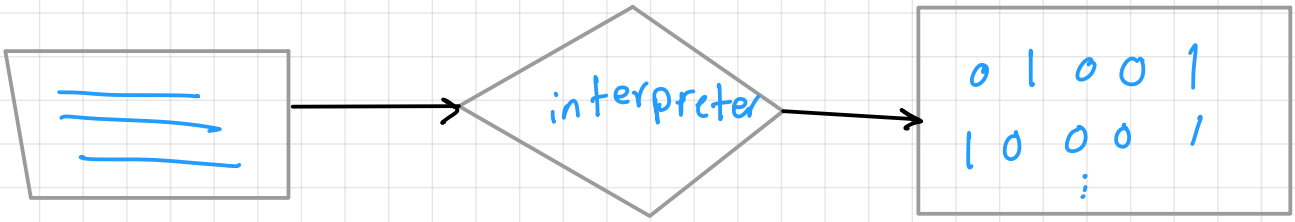
RAM is volatile ⇒ cleans data and come back

when instruction is done it returns directly to hard disk

Compiler Languages ⟶ C, C++

interpreturer " ⟶ ds, py



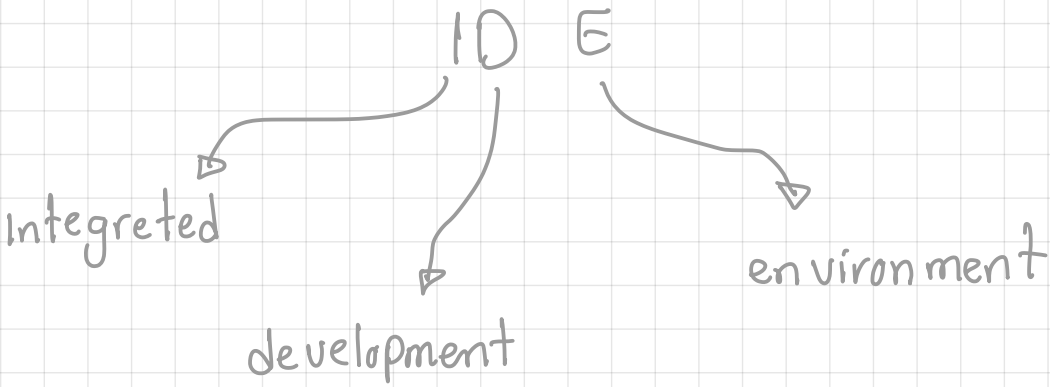


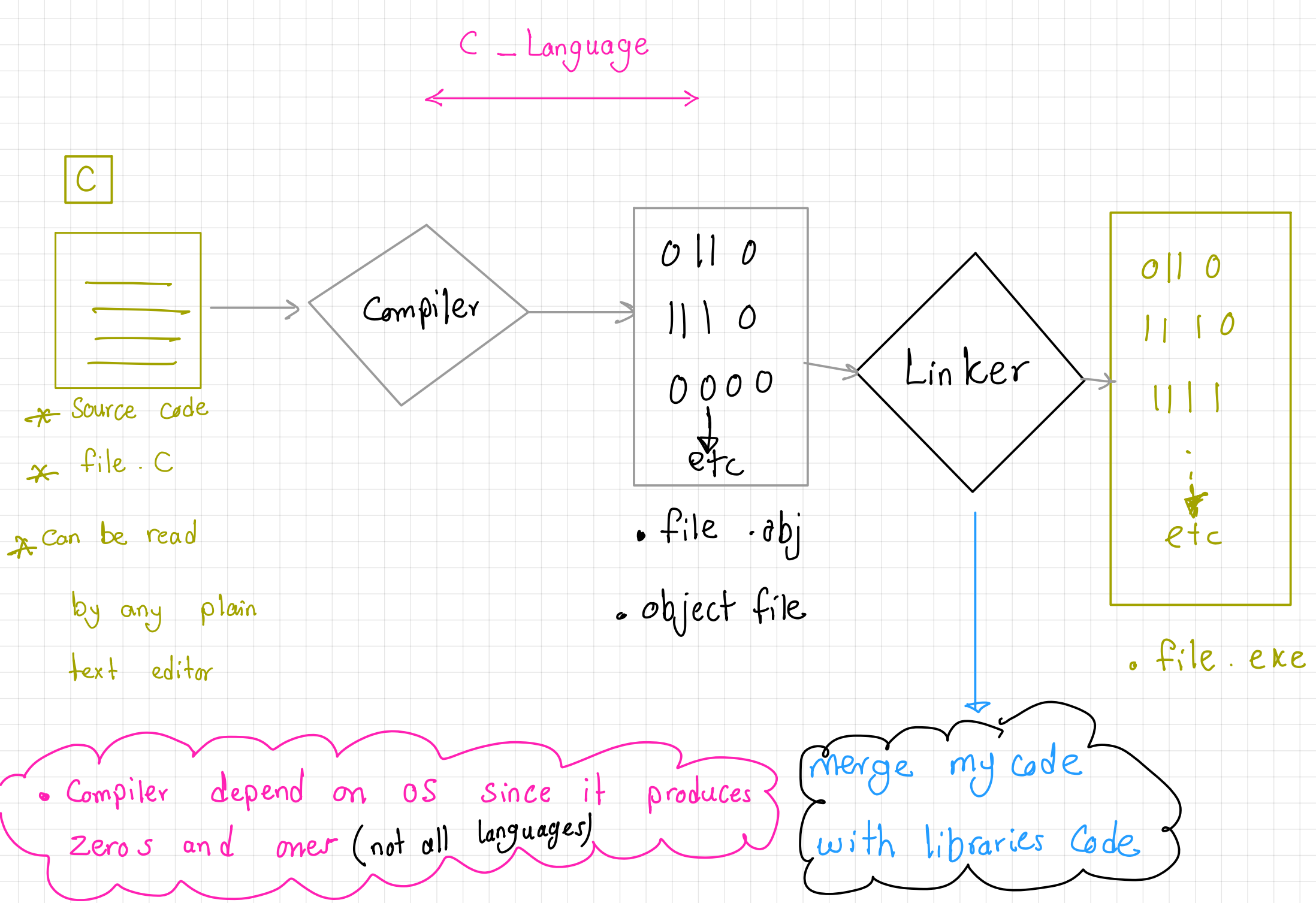
① it check line by line

② slower

③ each time I run the program it will interpret again

④ can check logical errors easier (trace code)
can check the code line by line (values)




C _ Language
C
Source code
file . C
can be read
by any plain
text editor
Compiler
0 1 1 0
1 1 1 0
0 0 0 0
etc
file . obj
object file
Linker
0 1 1 0
1 1 1 0
1 1 1 1
etc
file . exe
Compiler depend on OS since it produces
zeros and ones (not all languages)
merge my code
with libraries code

- there are languages are Compiler - Compiler and there are compiler - interpreter Languages

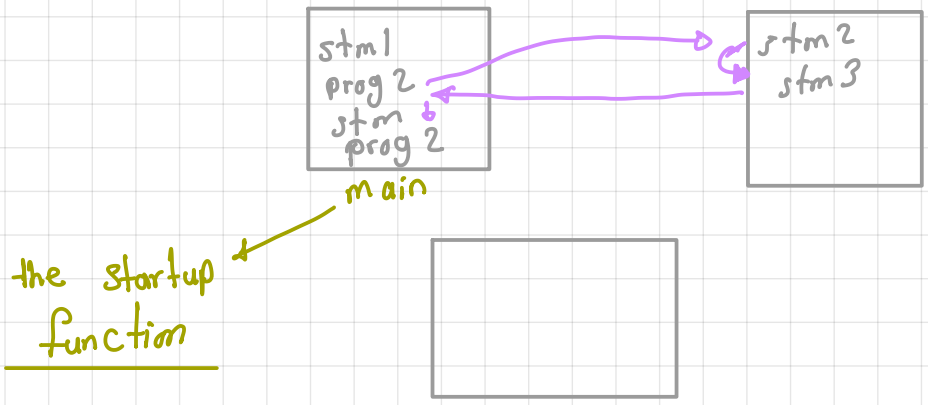
---

after break

Programming types

① Linear programming ⟶ stm 1, stm 2, stm 1, stm 2 ⟶ disadvantages: ① size in memory ② team work ③ Repettive

② structured programming.



③ OOP ⟶ will be discused in OOP Course

main function

```
#include <iostream>

#define xyz abc
```

header file → let program know that text is text or others (printf (" "))

replace each xyz by abc

pre processing (#)

```
main ()
{
    printf (" Hello World");
}
```

→ inside them all instructions need to be done (compiled)

- Before Compailing ⇒ pre processing including the data from other files.
  ⇒ in pre processing removes all Comments in the code.

- linker and pre processing explaination till 139 : 30

Errors types :-

1. Sytax errors → Compiler
2. Logical errors → trace using program in the IDE
3. Runtime errors → program is not running
   - → exception handling
4. warning → Compiler
5. Linker error

- exe file don't return to compiler it uses the result of the last compilation for the same file

---

CLR catch runtime errors and show it for developer

- Comments :-

//

OR

/* */

Eng / Ayman Elsayed Lotfy Hassan

1972 → first C versions declaration was mandatory to be in the top

- camel Case
- pascal Case

Code blocks

- How to let the code wait in C
- decleration → data_type variable_name ;
  int x ;
- int → Reserve 4 bytes in the memory
- initialization → data_type variable_name = value ;

OR

data_type variable_name ;
variable_name = value ;

printf (" x = %d \n y=%d" , x , y)

% → format specifier

%d → decimal

\ → escape character

\n → new line

%x → hexa decimal

\t → tab

\a → alart (alarm)

\b →

if x=97 ;
and I typed
printf ("%c", x);
output will be a

Char c = 'c' ;
printf ("%d", c) ; → 99
// ("%c", c) ; → c

char → 1 byte

float → 4 byte → up to 6 after .

double → 8 byte → up to 14 after .

Long int = Long → modefier → adds 4 bytes

Long double → modefier → adds 2 bytes

unsigned int → modefier → positive numbers only

After break 2

Containue Code blocks

difference between & x and x

if x = 30

scanf ("%d", x) ⇒ will go to address 30
if used by other programs
it will crash (Runtime error)

scanf ("%d", &x) ⇒ will save the value in the location of x

- scanf will read data in the buffer and save it in the location you want
- after pressing Enter it will read the buffer even if it's just

```
scanf ("%d %d", x, y);
printf ("%c : %d : %x : %x", ch, ch, ch, 16);
```

print the ch

ASCII

Hex decimal

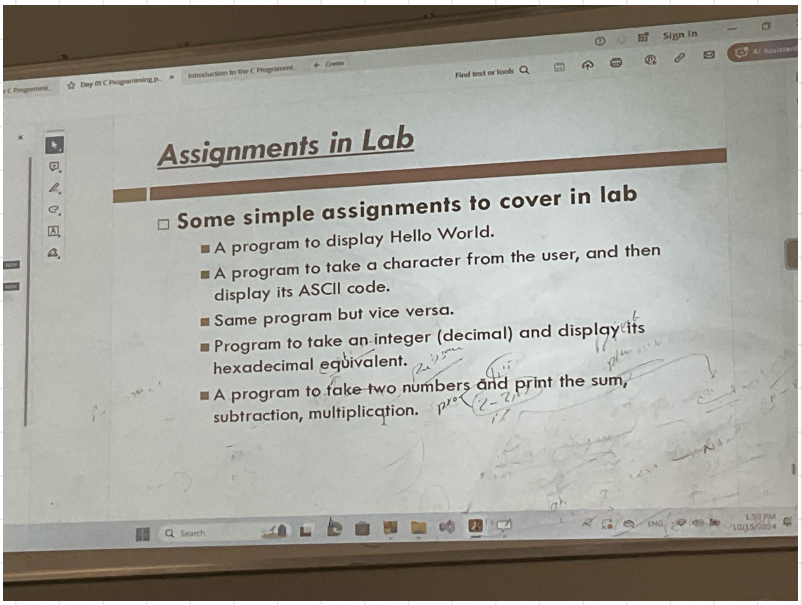
octal %o (octal not zero) 0 → 7

hexa %x 0 → 15

0 → 9 & a → F

- capital and small characters in ASCII code has difference 32 as decimal
- diffrent compilers and differen OS int size is different

requirements in lab



* operators used with int data type

* Program files(x86) → Code blocks → MinGW → include

Lecture 2 16/10/2024

- Source file → preprocessing → Compiler → obj → Linking → exe

- when we perform any mathmatical operations into numbers, the output upgrade to the highest data type of the variables in the expression (operation)

* operator percibiance
* associativity → from the left to the right
* but only assign operator (=) work from right to left

* compound operators %= 6 += 6 -= 6 /= 6 *= to reduce syntax

* unary operators ++ 6 --

post fix

x++

y=++x ⟹ x++;
y=x;

so y = new value of x

pre fix

++x

y=x++ ⟹ y=x;
x++;

so y = old value of x

Bitwise operators :- (each bit)

In C we have 32 keyword

&

|

>>

<<

Relational operators in C..

==

!=

>

<

>=

<=

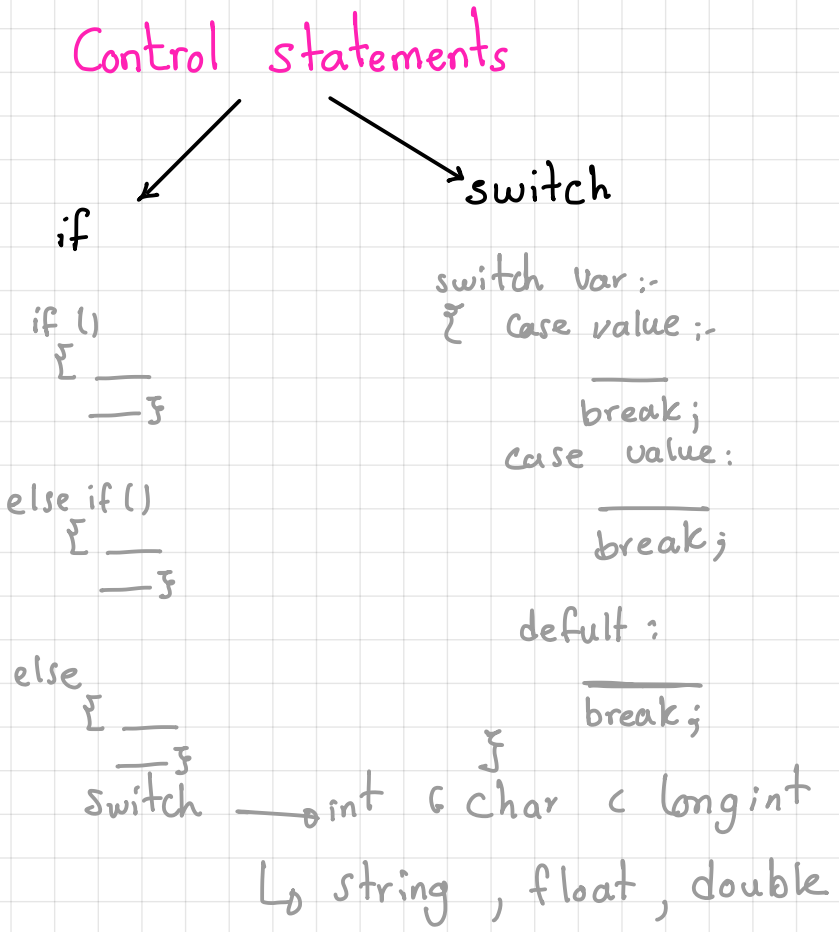
Logical operators in C :-

&&

||

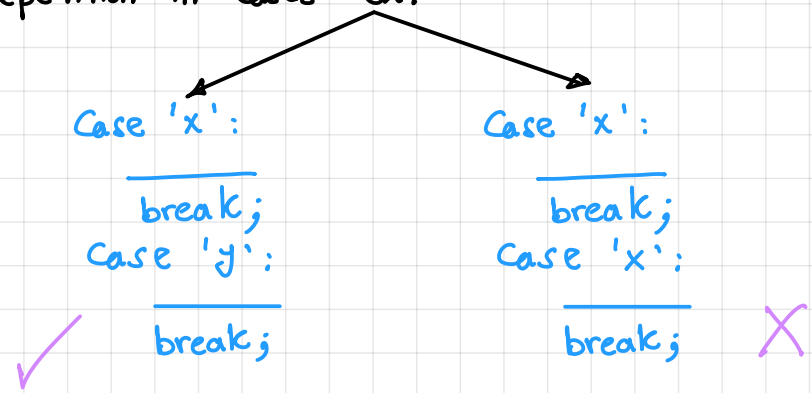
only zero is false if positive or negative it will be true

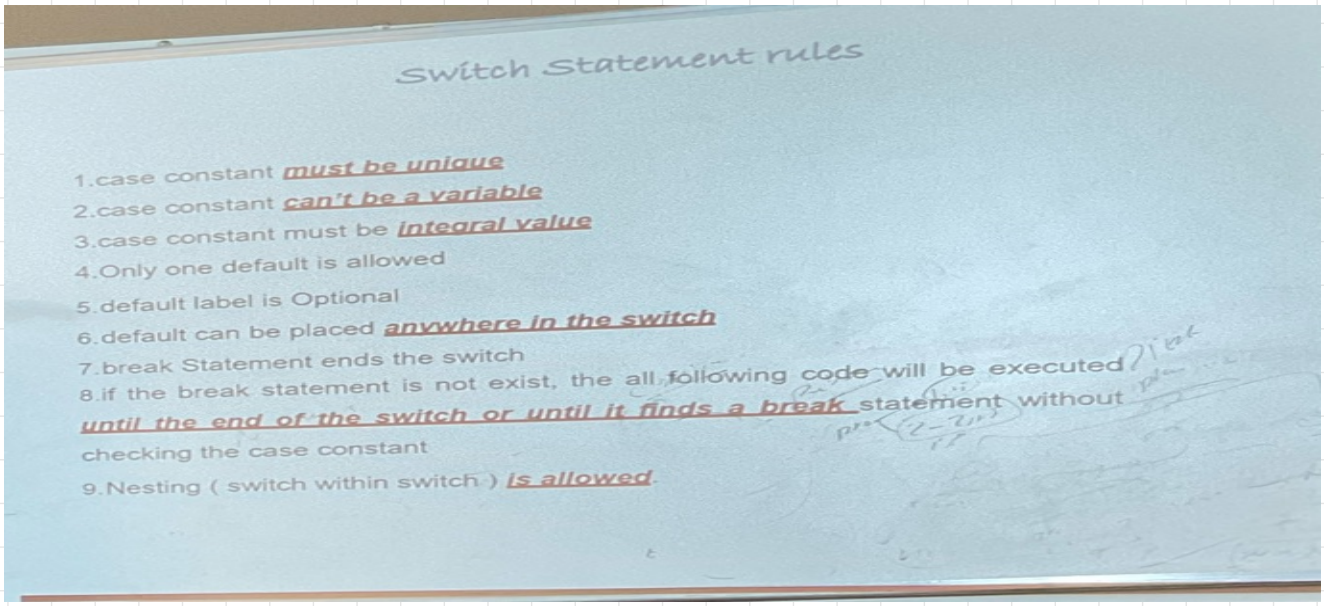
Control statements



Switch is perferred in some apps like calculator but if Comparison if is better

- switch should take lateral value (static value) not a Variable.
- no repetition in Cases ex:-





Flow chart

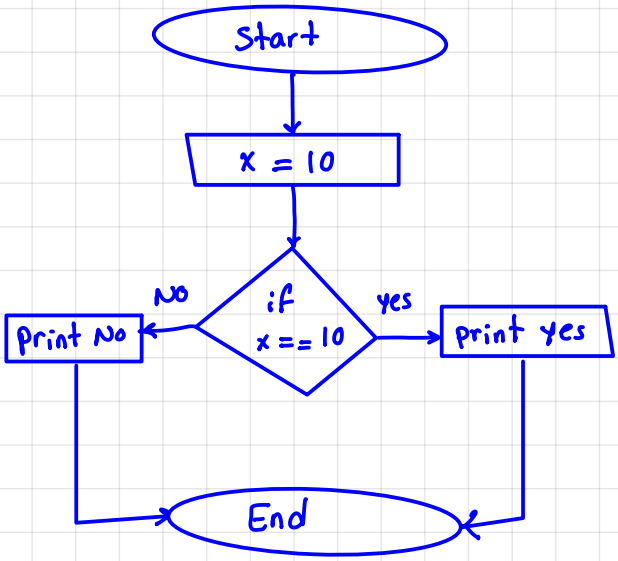
statement

Condition

End

Start

Example of if



break ;

After break

Loop statements

- while ( ) { } ⇒ Loop till false
- for (;;) { } ⇒ I know no. of iterations
- do{ } _ while ( ) ⇒ performed at least once

```
for (int i=0 ; i<5 ; i++)
 {
    printf ("Hello");
 }
```

Note

i++ is done 5 times

i<5 " " 6 "

i = 0 is done once

Another example

```
for (int i=0 ; i<5 ; i++)
 {
    printf ("Hello %d", i+1) ;
 }
```

no ++i is equal i=i+1

same as i+1 not assigning the value to i

i = 1
i = 2
i = 3
i = 4
i = 5

i = 1
i = 3
i = 5

- break is used with Loops or switch to exit Loops

* another representation of for

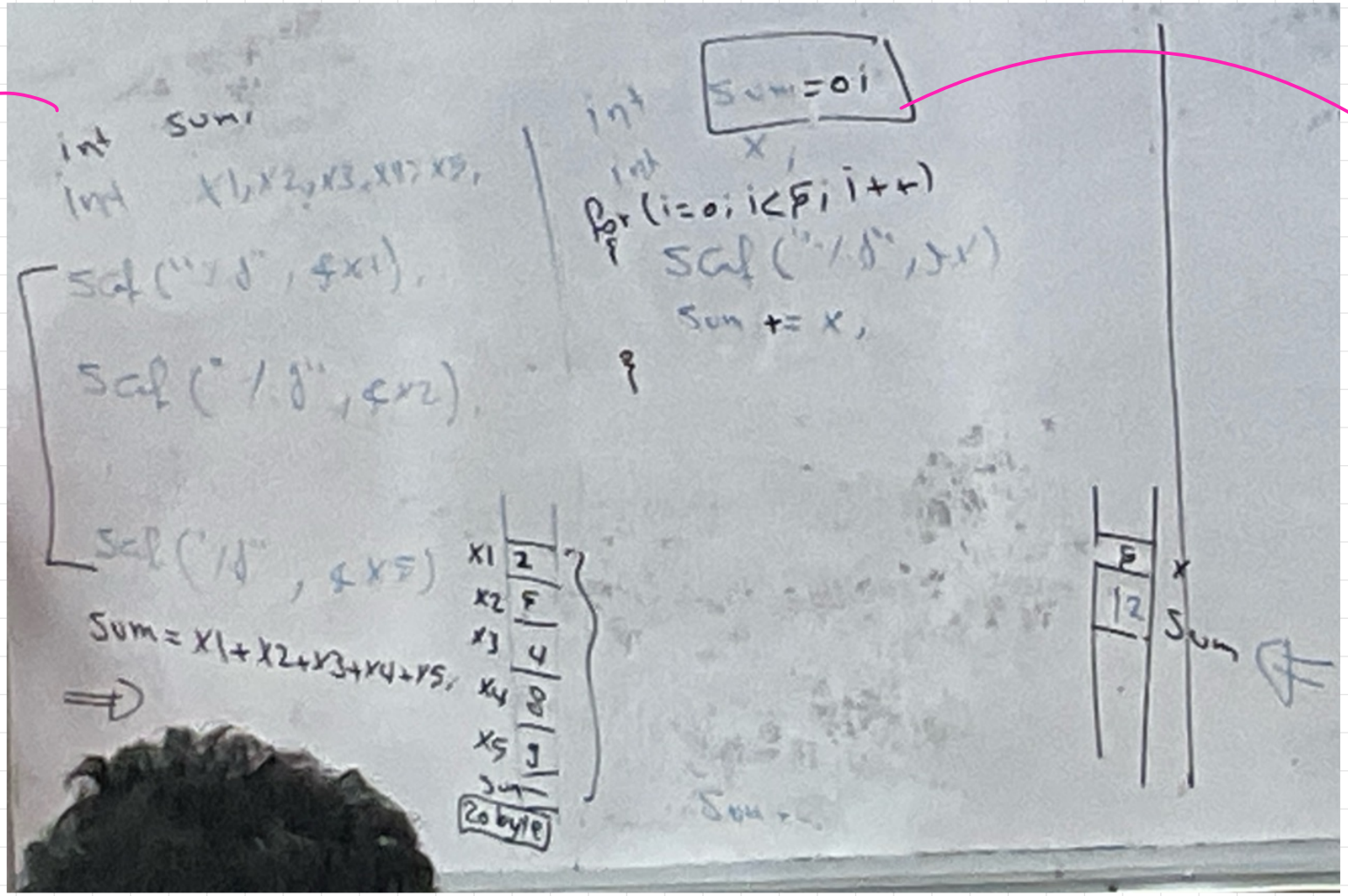
```
int i=0
for ( ; ;)
{   if (i==5)
        break;
    i++;
}
```

not readable

Example of for

```
int y;
printf ("Enter the i") y
scanf (" %d", &y);
for (int i=0; i<y; i++)
{   printf (" %d ", i);
}
```

write alot of code



don't save values of x. it replace Last value by new value.

Each of the two problems Solutim is arrays.

3rd way using do-while

performed at least once and number of Loops is unknown

```
int sum = 0;
do {
    scanf ("%d", &x);
    sum += x;
}
while (sum < 100);
```

4th way using while

```
int sum = 0;
while (sum < 100)
    {
    scanf ("%d", &x);
    sum += x;
    }
```

while is done zero times or more not like do-while performed at least once

— flushall ( ); ⇒ remove data from buffer
(enter data)

System('cls'); ⇒ clear screen

```
do {system ('cls')
   gotoxy (5, 5);
   printf  menu;
     flushall ();
     scanf ("%c", ch);
     Switch
        {

        }
```

Lab :-

1. degree ⇒ grade (excellent 100-85
   v Good 75
   Good 65
   acceptable 50
   fail)

   (if)

2. Calculator n1, n2, op switch

3. No of month 1 → 31 days
   2 → 28 "
   3 → 31 "
   4 → 30 "

4. sum of 5 numbers

5. menu (new, display, exit

6. magic box → tomorrow

7. factorial

8. number → prim
   → not prim

⑨ reverse number ex int i = 789

987

Day 3

On line

Arrays

Day 4 Friday 18/10/2024

we use _getch(); to scanf ("%c", ch);

then

Difference between Scan a char and using _getchar

scanf needs to press Enter after typing the character not like _getchar → no need to press Enter after the char (Both need printf to print the data

- to get the average of each subject (Column)

```
for (int j=0 ; j<4 ; j++)
    for (int i=0 ; i< 3 ; i++)
```

Student	sub1	sub2	sub3	sub 4
"				
"				

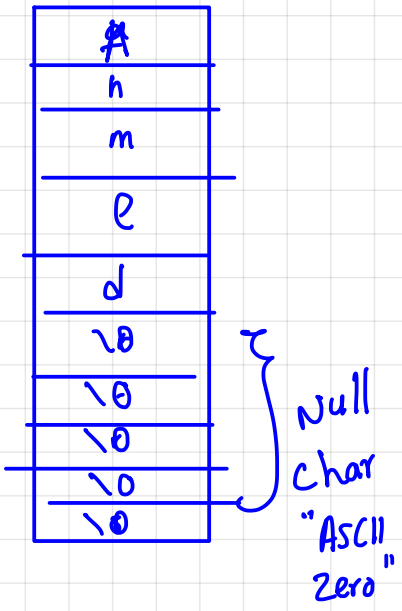
- Array of char "string"

```
char name[10] = "Ahmed";
printf("%s", name);
```

prints data till it finds Null stops

Same as

```
i=0
while (name[i] != '\0')
{ printf("%c", name[i]);
  i++;
}
```



```
scanf("%s", name);
```

name → without &

but if `scanf("%s", &nam[0]);`

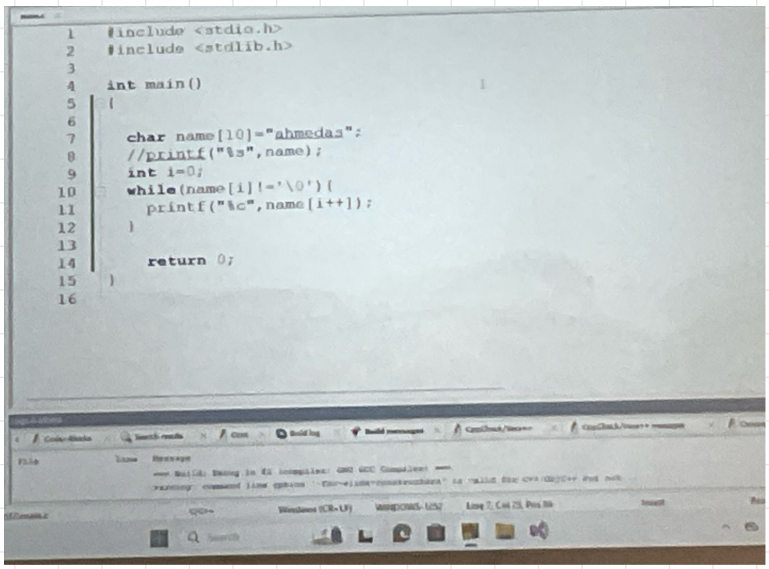
we need &

---

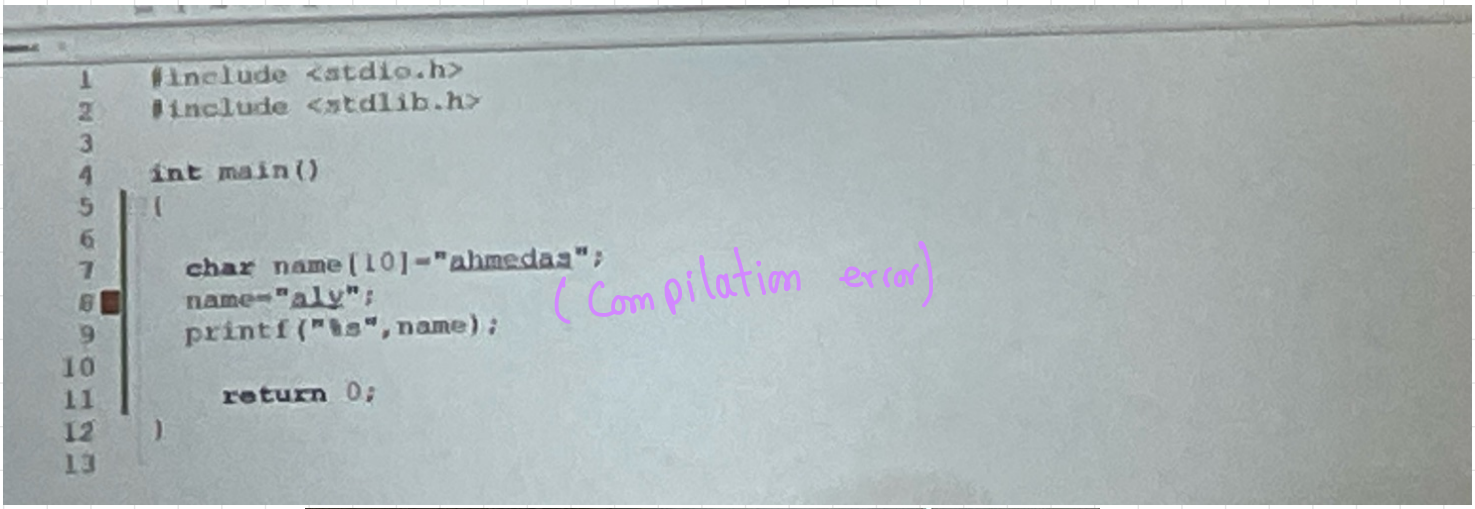
if I didn't write the size of the array but initialize the array size +1 (Null is involved)

if I am saving Israa I need size of array 6 not 5 israa + 1 which is \'0'

Print array of character



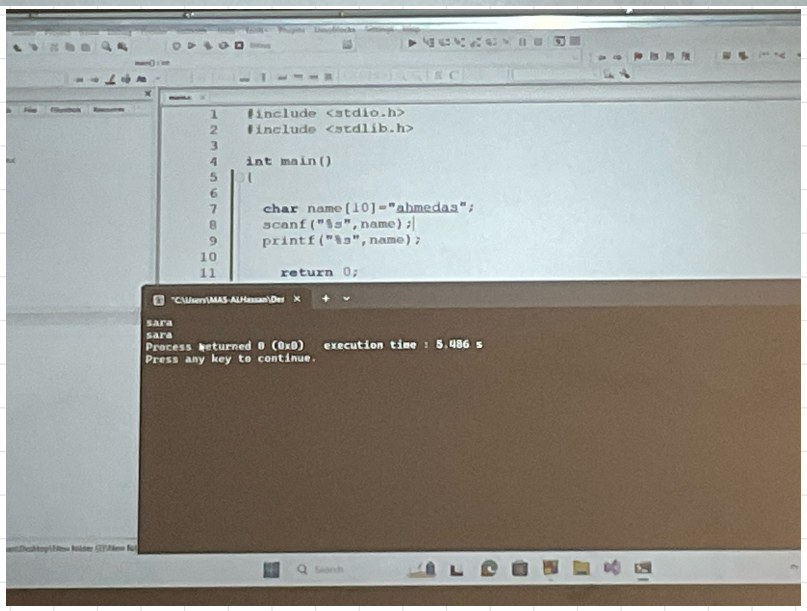
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

  char name[10]="ahmedaa";
  //printf("%s",name);
  int i=0;
  while(name[i]!='\0'){
    printf("%c",name[i++]);
  }

    return 0;
}
```



```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

  char name[10]="ahmedaa";
  name="aly";
  printf("%s",name);

    return 0;
}
```

(Compilation error)

Scan array of characters



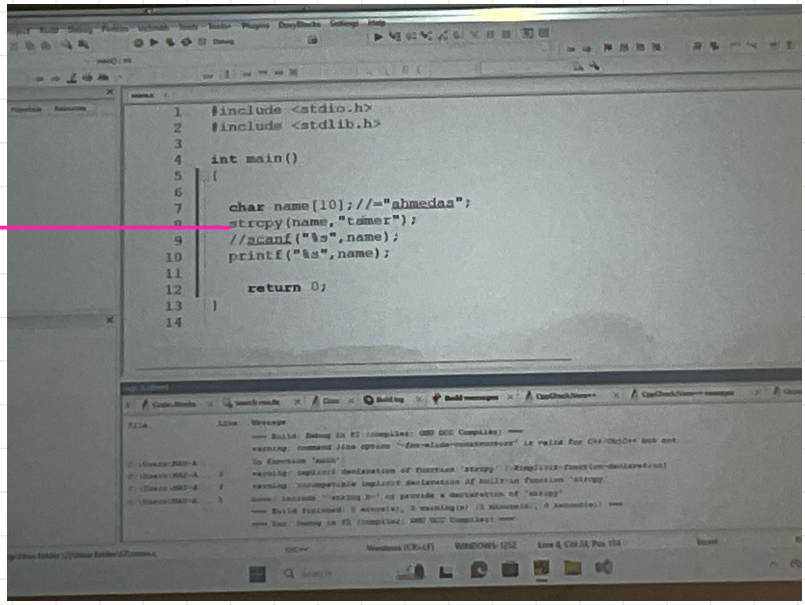
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

  char name[10]="ahmedaa";
  scanf("%s",name);
  printf("%s",name);

    return 0;
```

```
sara
sara
Process returned 0 (0x0)   execution time : 5.486 s
Press any key to continue.
```

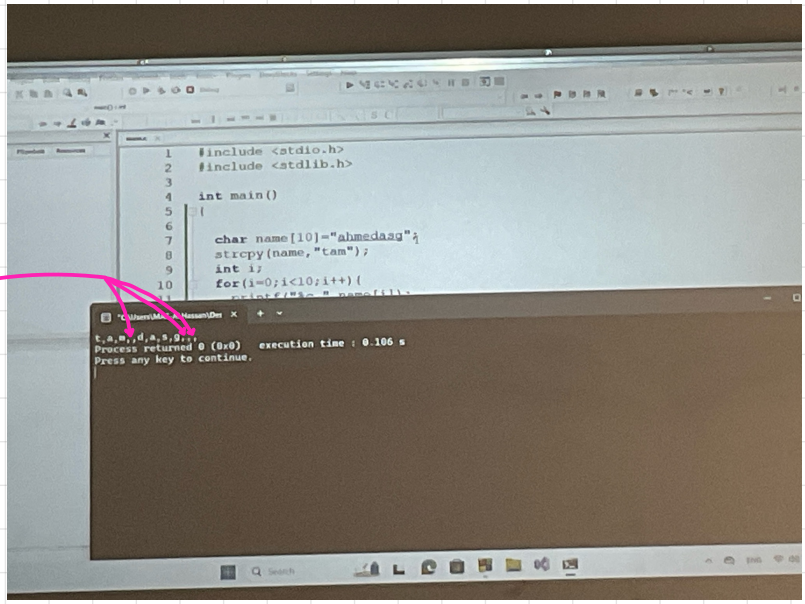
if size of array < number of characters it's runtime error

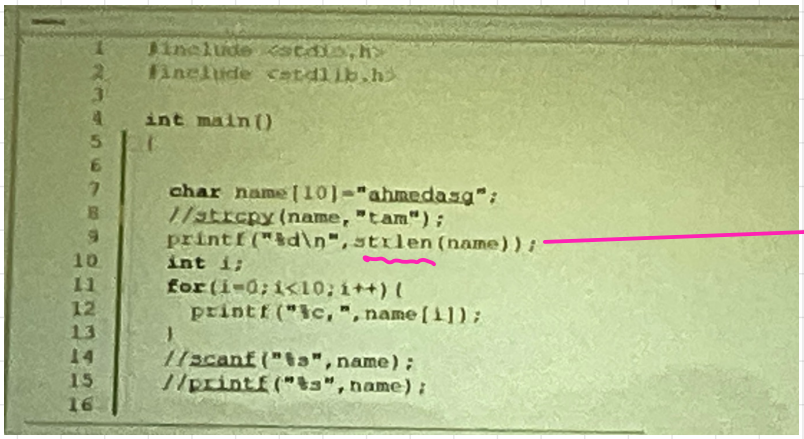
strcpy changes the value of the name to tamer instead of ahmedas

Same as name = tamer



these are for '\0'

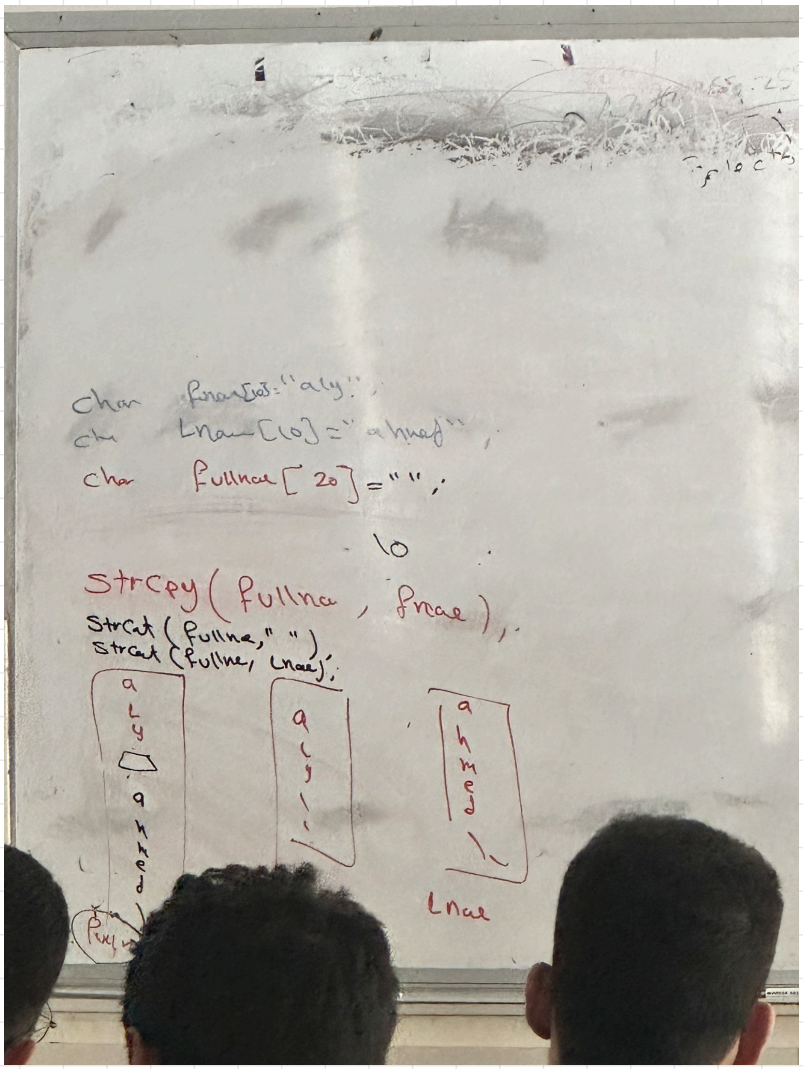


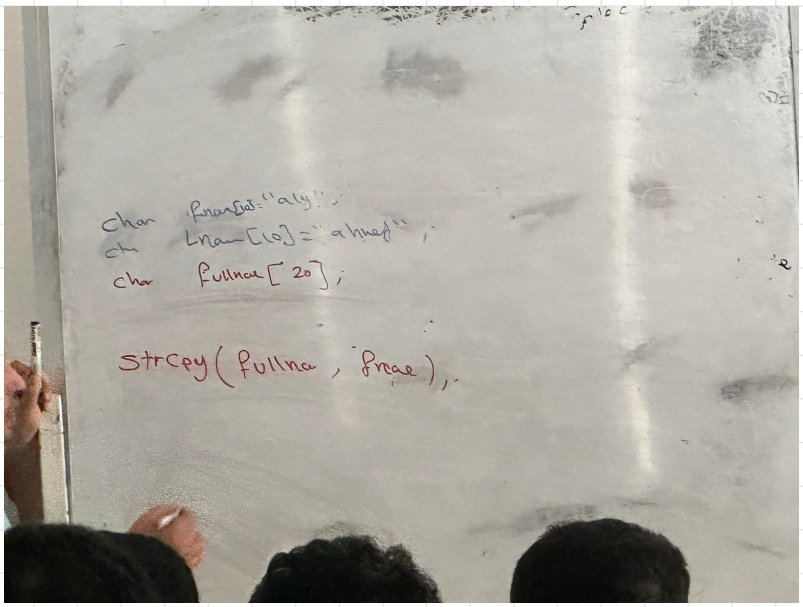


strlen stop counting when facing '\0'

Strcpy size higher than value

* strcat ⇒ Compine more than string after the last \0





scanf breaks in space ali mohamed

saves only ali

⇒ gets solves that problem of scanf

puts (fname); but printf ("%s", fname);

strcat puts space between arrays

```
fname = "ali";
lname = "Ahmed";

fullname = "  ";
strcpy (fullname, fname);
strcat (fullname, ' ');
strcat (full name, lname);
```

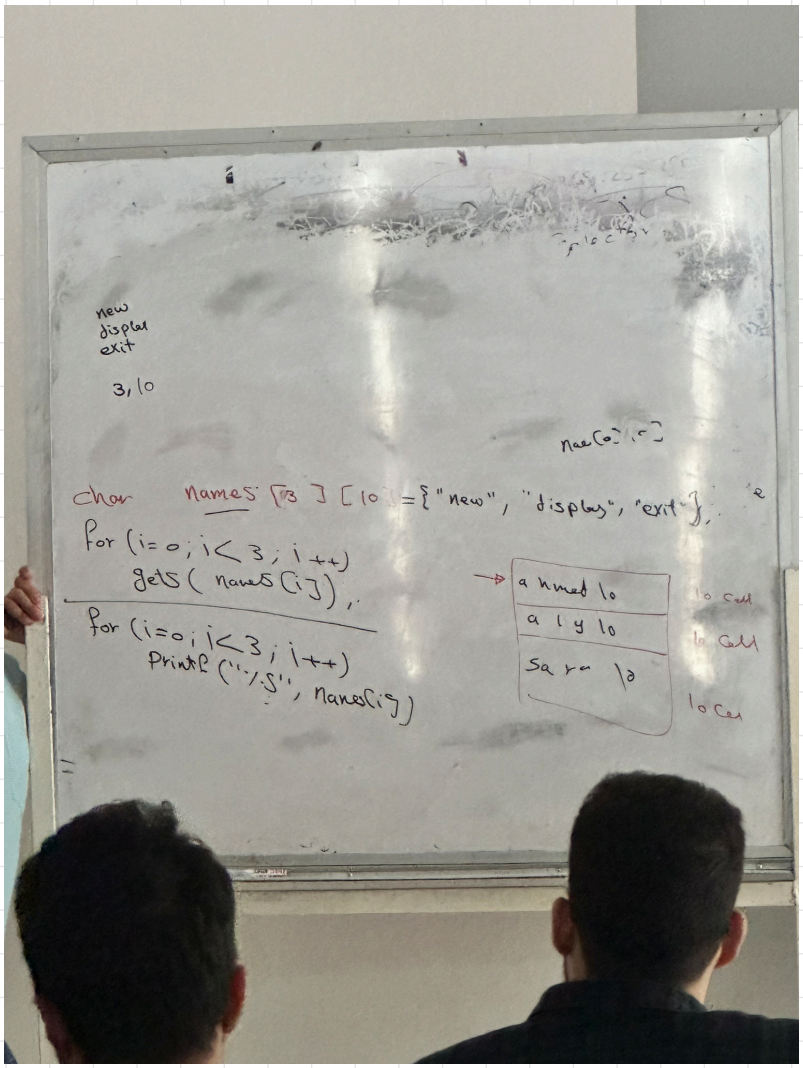
Day 4 after break

create a list of names (array of char)

```
char name[3][10];   // max char is 9
for (int i=0; i<3; i++)
{   gets(name[i]);
}
   //gets(name[1]);
   //gets(name[2]);
```

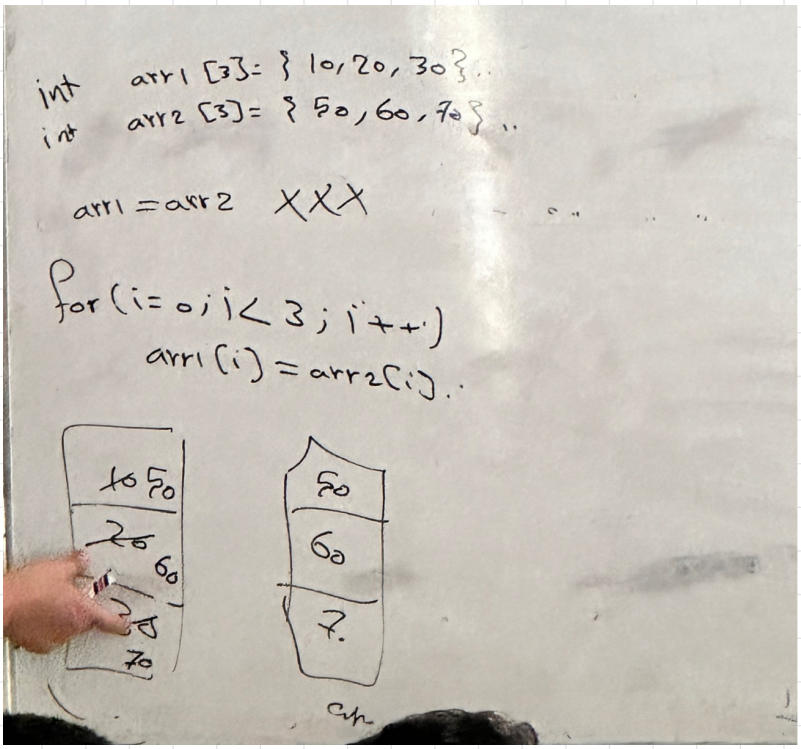
works only for array of characters

Ali
Samer
Israa



can't apply = sign between arrays

must by element by element



String Compi

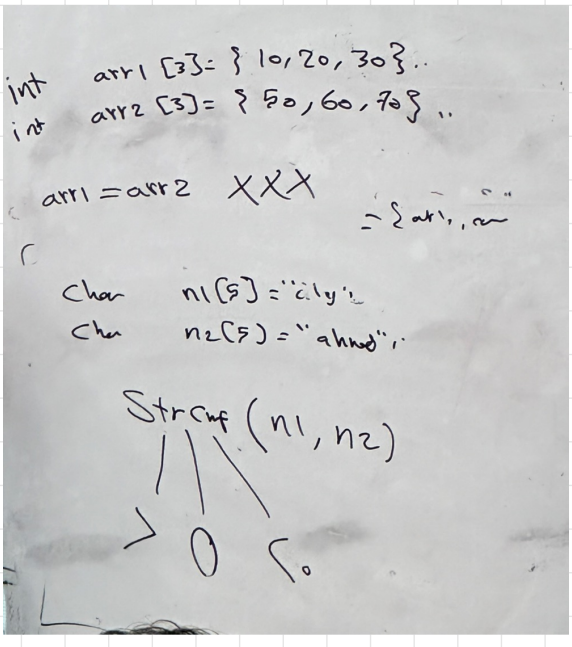
ignores Case

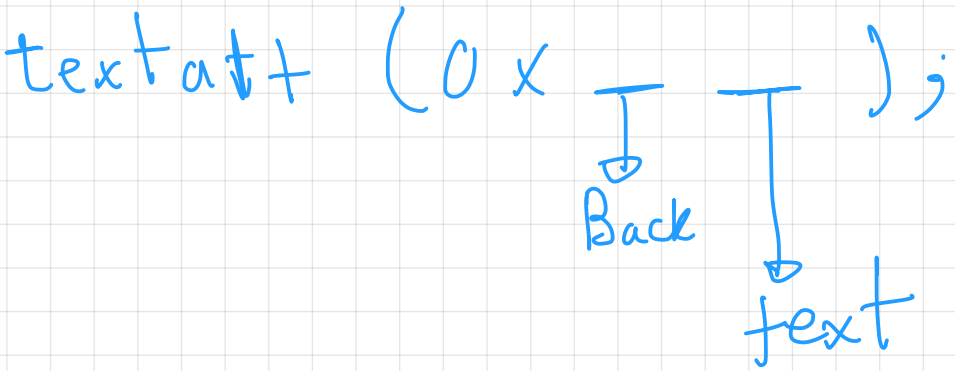
A = a

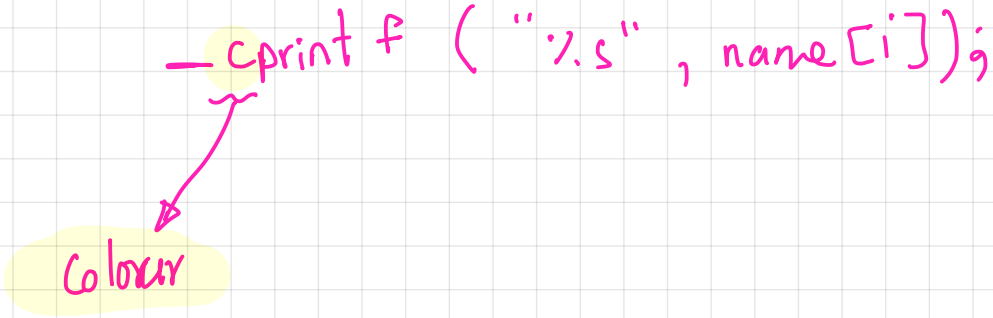
String Comp

don't ignore

A ≠ a







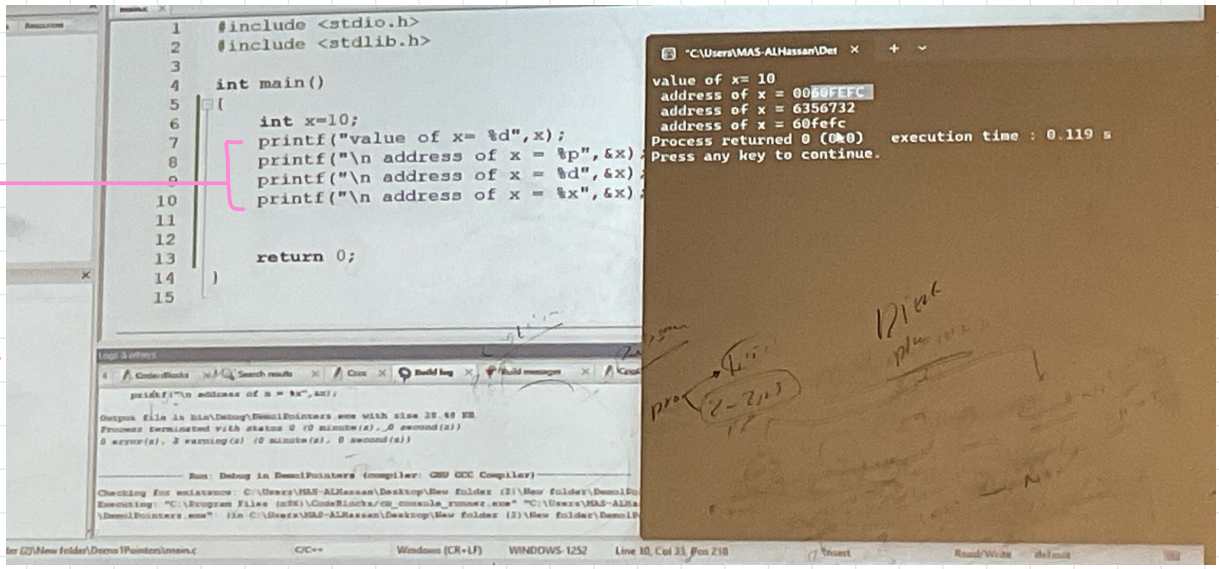
keys are 256 only (ASCII) But later there's an extended keys

Day 5

Pointers

- each variable has a name .
- you select the data type of variable according to data assigned in it so that defines the size of variable in the memory and range of values allowed for it
- operating system (OS) saves the data of the program using pointers
- ech byte has it's own address
- & ⇒ And operator used to bring the address of variable.
- % p ⇒ is the format specifier for the address .

Different format specifier to print Addresses



- if %p but without & with the variable it will print the data saved in the variable (whatever its type) in the format of address (Hexa decimal) not its actual address

```
int   x = 10 ;
int * ptrx = & x ;
* ptrx = 30
printf ( "%d", x ) ;
printf ( "%d", *p) ;
printf (
```

data type of pointer needs to be same as variable

the address size depending on the local machine

```
printf ( "  ", x )  ⇒ 30
printf ( "  ", &x )  ⇒ Address of x
printf ( "  ", *ptrx)  ⇒ 30 (value of x)
printf ( "  ", &ptrx)
printf ( "  ", ptrx)
scanf ( "%d ", x)  ⇒ Run time error
```

No Refrence

$x = *ptrx$

$\&x = ptrx$

if I tried to print a value of pointer that is not initialized yet ⇒ Run time error

int* y , x ; ⇒ x is a normal int

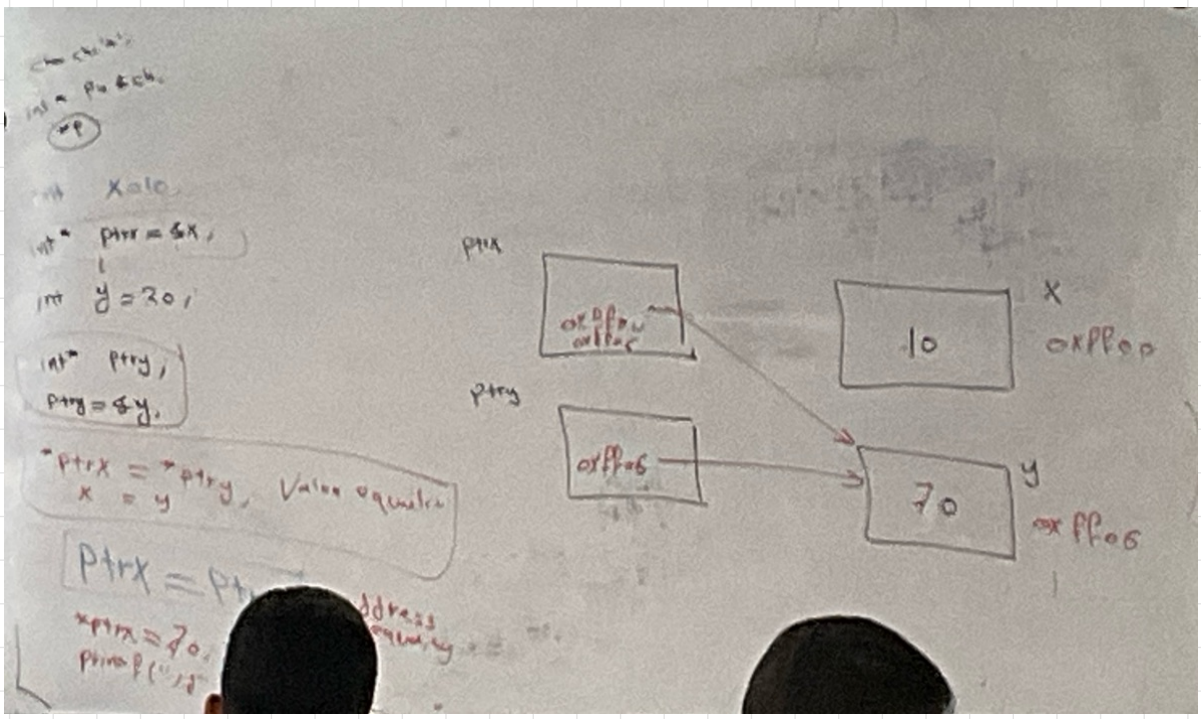
* ptry ⇒ Refferes to the place that pointer is reffering to (المكان الى بتشاور عليه)

*ptr y = *ptr x ⇒ value equality (x = y)

ptr x = ptry ⇒ ptr y will be copied to ptrx
so ptr x will reffer to y instead of x

↳ address equality

*ptrx = 70 ;
printf ("%d", x) ; ⇒ output will be 70



initialization of pointer

int * ptrx = & x

decleration ptr = & x

* ptr = x ⇒ wrong (runtime error)

int* ptr x ;

ptr = & x ; ⇒ this is the same as * ptr always

undetermined (expected) behaviour

↳ run time error

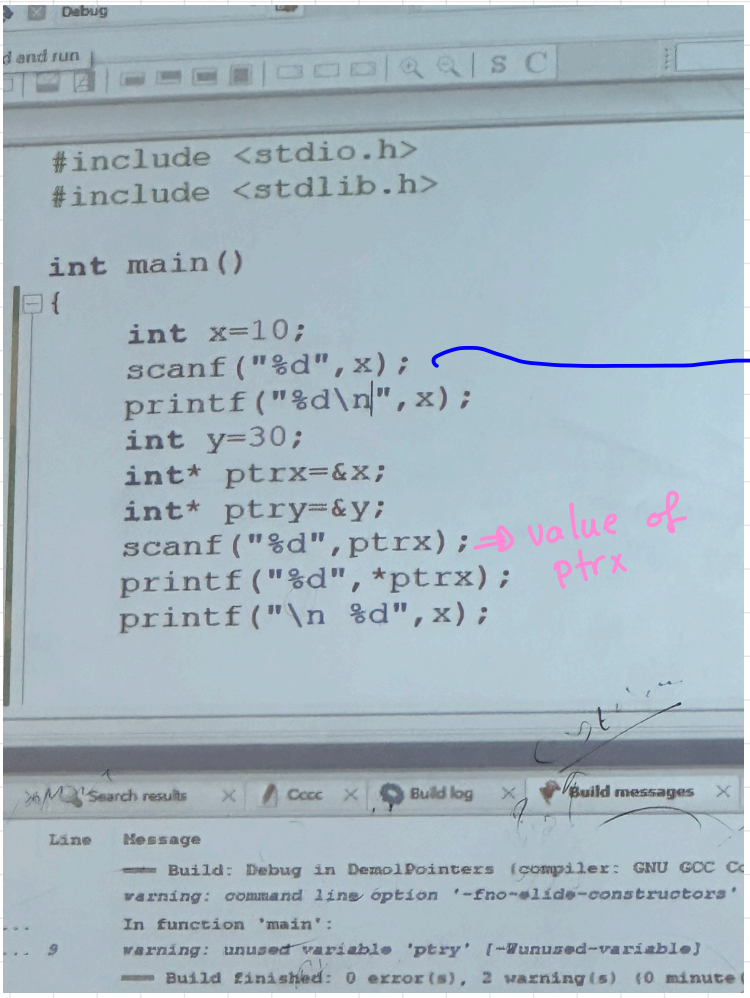
trying to access place may not be accessable ⇒ Run time error

- in same scope ⇒ you can not have variables with same name
- x = y ⟶ value equality at this line only copy y in x
-

After break.

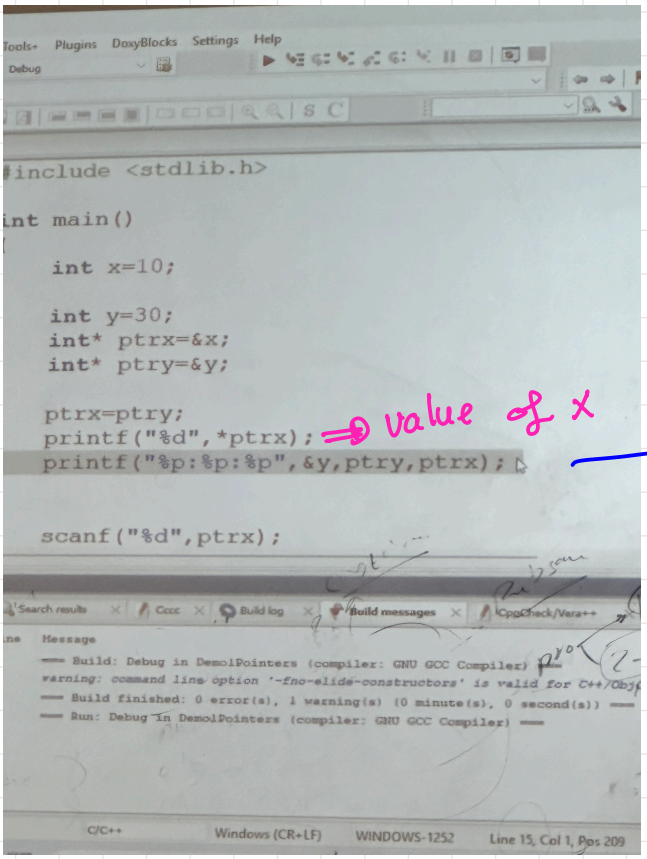
search Stack Learn

Code blocks



value of ptrx

Runtime error may run but still considered as Run time error



value of x

Same

min

10

Quiz

What will be the output of the following lines:

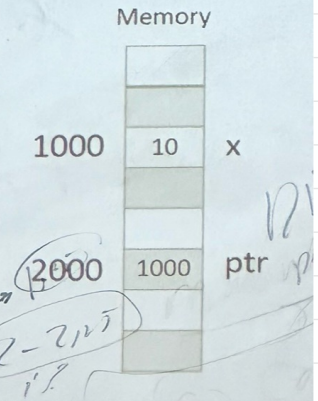
a. `printf("%d",x);` → 10

b. `printf("%p", &x);` → 1000

c. `printf("%p", ptr);` → 1000

d. `printf("%d", *ptr);` → 10

e. `printf("%p",&ptr);` → 2000



Note: %p used to print an address

---

*ptrx = *ptry → value equality

ptrx = ptry → address equality

---

if 1 typed

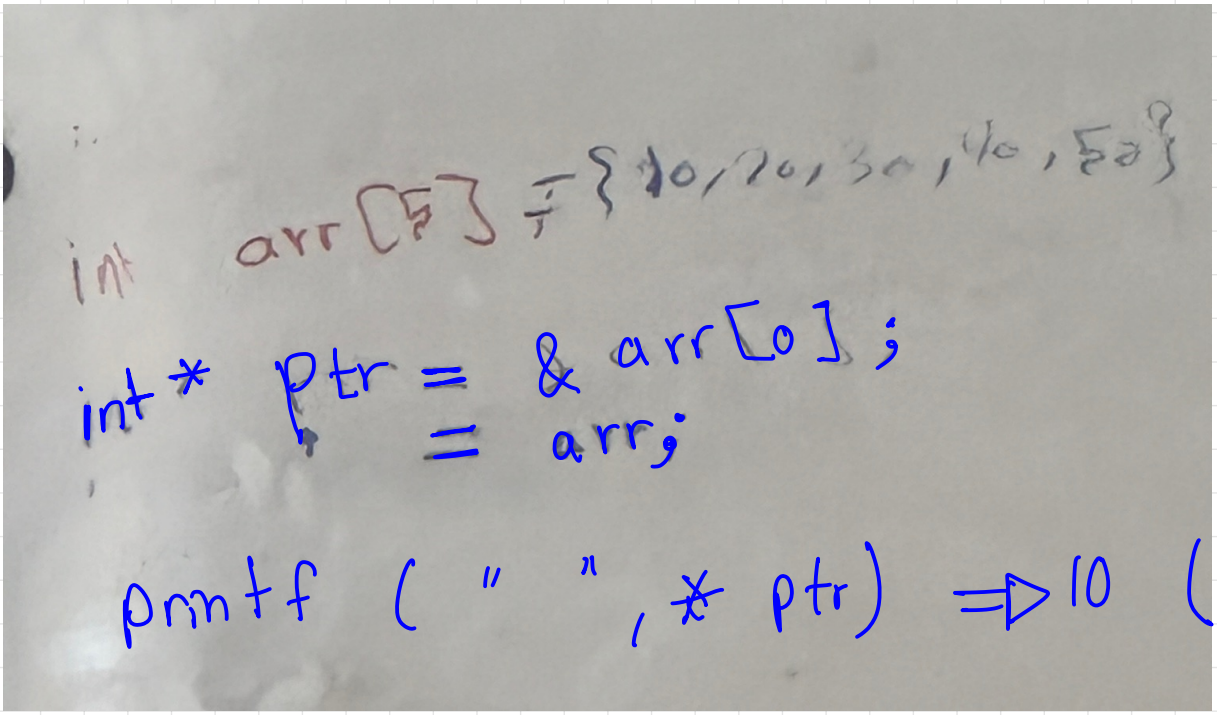
ptrx = ptrx + 1 or ptrx ++

the 1 depends on the data type

if int → it will increased by 4 not one if double will increase by 8 and so on

Pointers & Arrays

- name of array is a pointer for the first element in the arrays



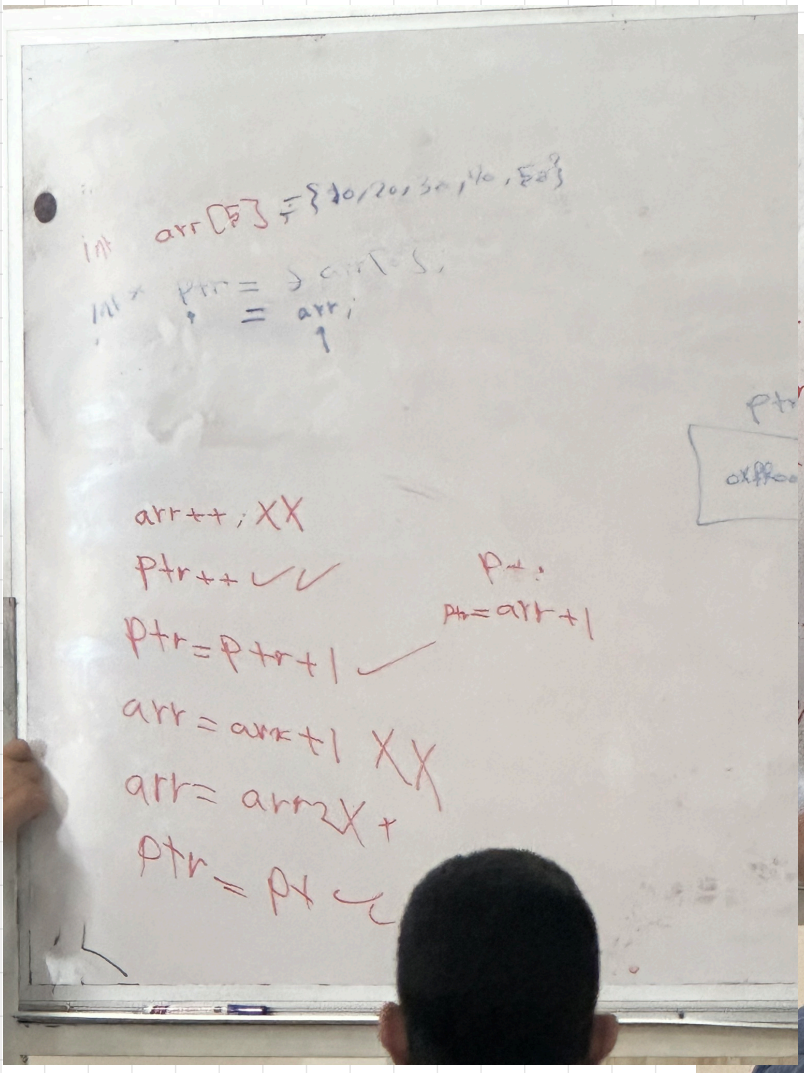
printf (" " , *(ptr + 1)) ⇒ 20 (0xff04)

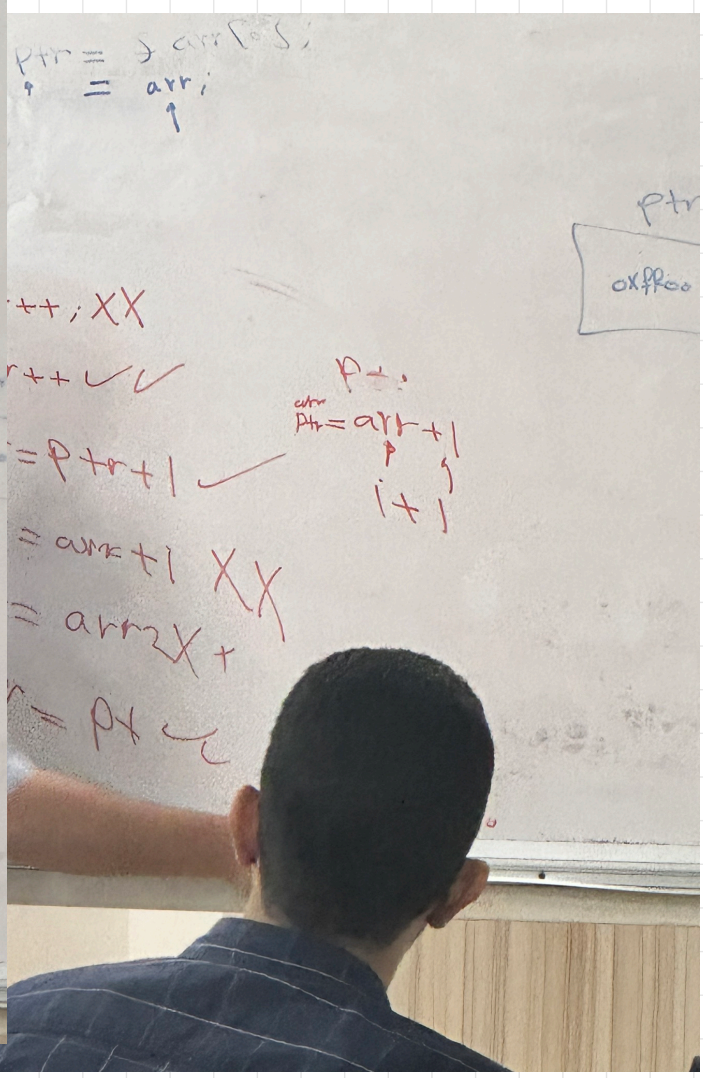
(" " , *(ptr + 2) ⇒ 30 (0xff08)

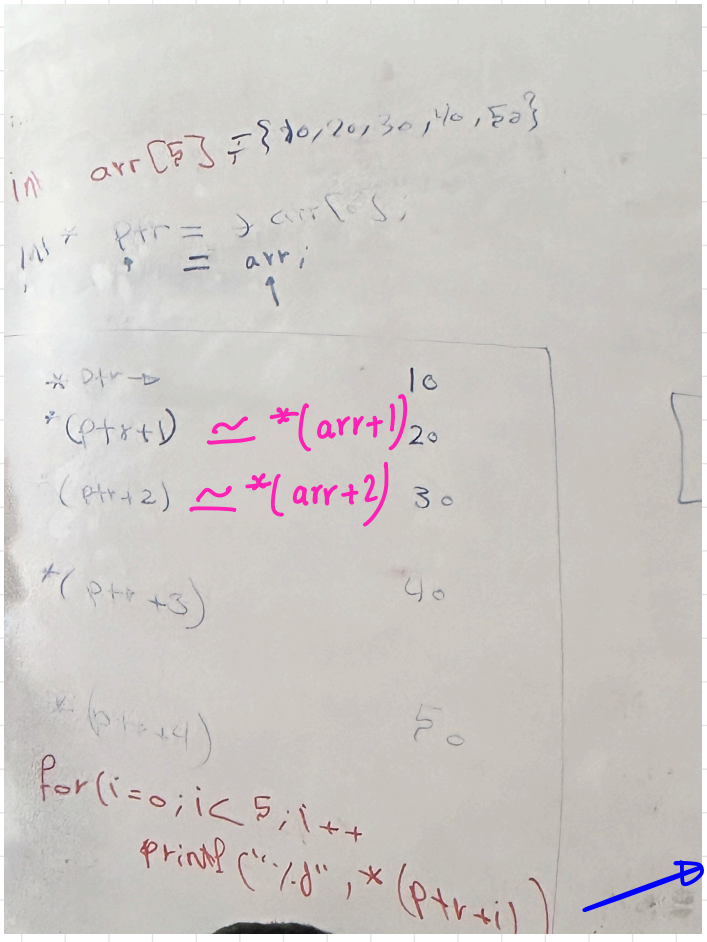
arr1 = arr2 → Compilation error (Const pointer)

Can't change all element

because the name of the array is a Const pointer Can't be changed because I will not be able to access the rest of elements if I changed the pointer of the array


arr[5] = {10, 20, 30, 40, 50};
int * ptr = arr;
arr++; XX
ptr++
ptr = ptr + 1
arr = arr + 1 XX
arr = arr + 2 XX
ptr = ptr


ptr = arr;
ptr
0xff00
++; XX
ptr++
ptr = ptr + 1
arr = arr + 1 XX
arr = arr + 2 XX
ptr = ptr
ptr + 1;
arr
ptr = arr + 1
i + 1


arr[5] = {10, 20, 30, 40, 50};
int * ptr = arr;
* ptr → 10
*(ptr+1) ~ *(arr+1) 20
(ptr+2) ~ *(arr+2) 30
*(ptr+3) 40
*(ptr+4) 50
for(i=0; i<5; i++
printf("%d", *(ptr+i))

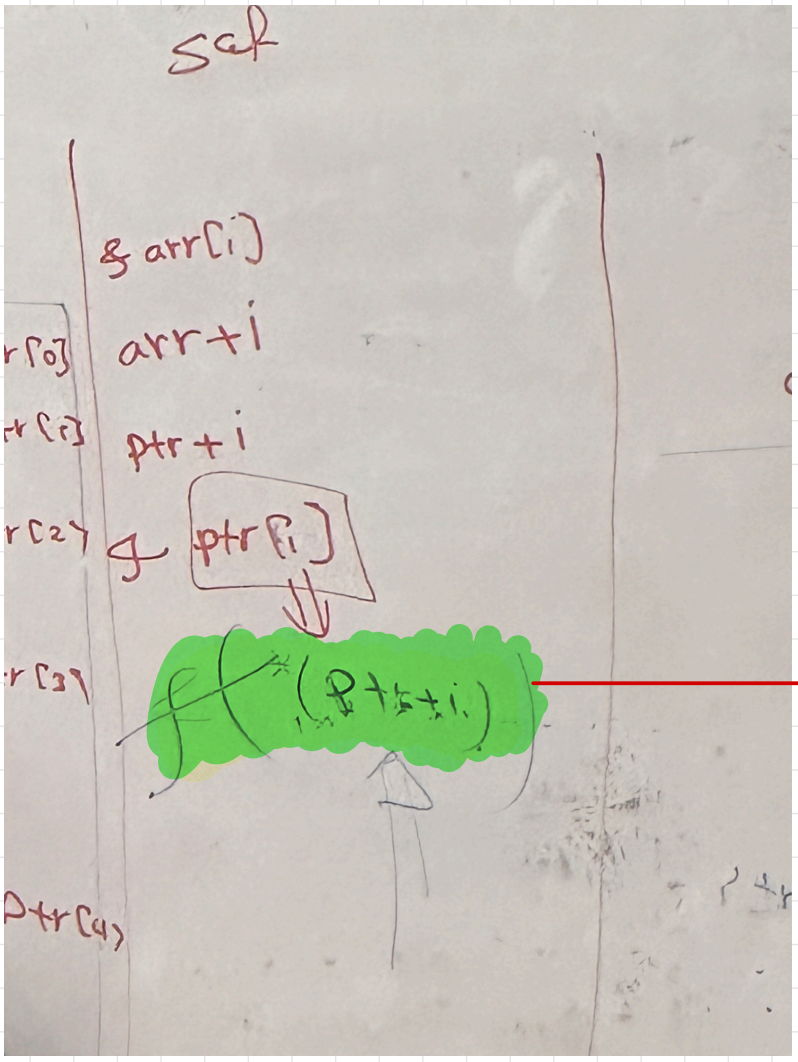
scanf ("%d", (ptr + i))

- dealing with arrays through pointers is much faster than arr[element]

- *ptrx ≃ ptrx[0]
- *(ptrx+1) ≃ ptrx[1]
- *(ptrx+2) ≃ ptrx[2]

and so on (they are the same

&arr[2] = ~~&~~(*(arr+2)) → location of second element of the array



→ which is the location of second element

- arr = arr+1 ⟹ Compilation error

- Be careful when using ptr++ or ptr--
  Because it will increase the valuse by one and
  Save it ⟹ So don't use it with scanf but
  use ptr+i or ptr+1 instead

LAB!

[1] scan and print data of array using pointer

[2] save prime number or not and

[3] Continue of Menu ⟹ build sub menu

[4] struct for employee (id, age, name)

[5] struct for x, y values

[6] struct for imagenary

Struct :-

name of the struct

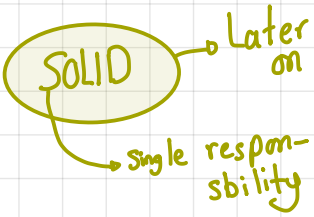
I want

نموذج
"
blue print

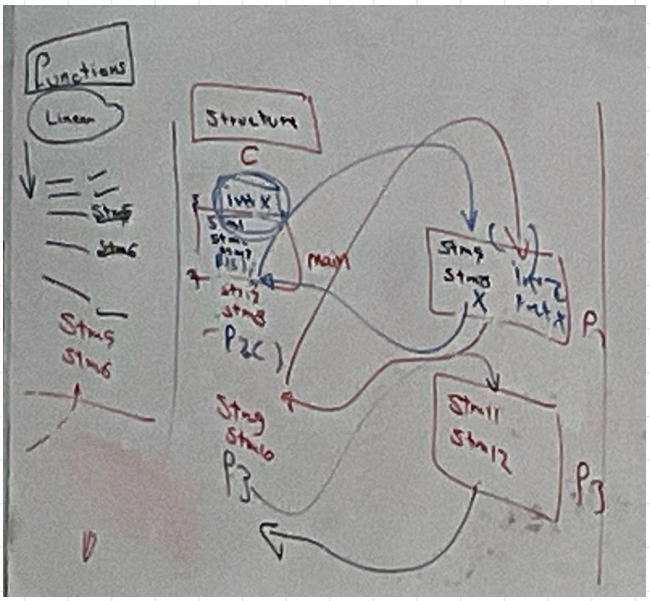
```
struct emp
{
    int id;
    char name[30];
    int age;
};

int main()
{
  struct emp el;
    el.age = 30;
   el.id = 10;

  strcp ( el.name, "aly" );

  printf ("%d : %s :%d", el.id , el.name, el.age);
```

DAY 6

Continue pointers :- && Functions :-



↳ Group instructions to use it later



- the standard is to write the function after the main function but create a decleration before the main. (function prototype)

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   //structures
4   void printv1();// declare function, function prototype OR header
5
6   int main()
7   {
8       printv1();//call
9       printf("Hello world!\n");
10      return 0;
11  }
12  //impl define
13  void printv1()
14  {
15      printf("\n**********");
```

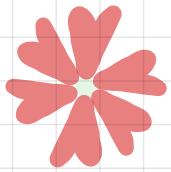
→ the disadvantage of it is not reuseable

- Some books calls the variables in the function as parameter.

But in the calling or prototype it's called argument

→ void add ( ) ≃ void add (void) ←

no parameters



- no more than one return. ⇒ but can be done using pointers

break → Switch
for

return → exit function.

```
14      scanf("%d %d",&a,&b);
15
16      int c=add(a,b);
17      printf("%d",c*2);
18      return 0;
19  }
20  int add(int x,int y)
21  {
22      int z;
23
24      z=x+y;
25      float d=2.3;
26      return d;
27      //printf("%d",z);
28  }
29  void printv3(char ch,int n)
```

will cast it and Return just 2.

Search

params #

Not in our Course

record

After break

ternary operator :- `x>y ? x :y;`

- variables are local to its function .
  if I need variable to be global
  ↳ I should declare it before main function

- Global variables are initialized by zero always even if I didn't type that

- Code structure :-

  - #include
  - #define
  - struct
  - global data
  - function proto type
  - main function
  - function implementation

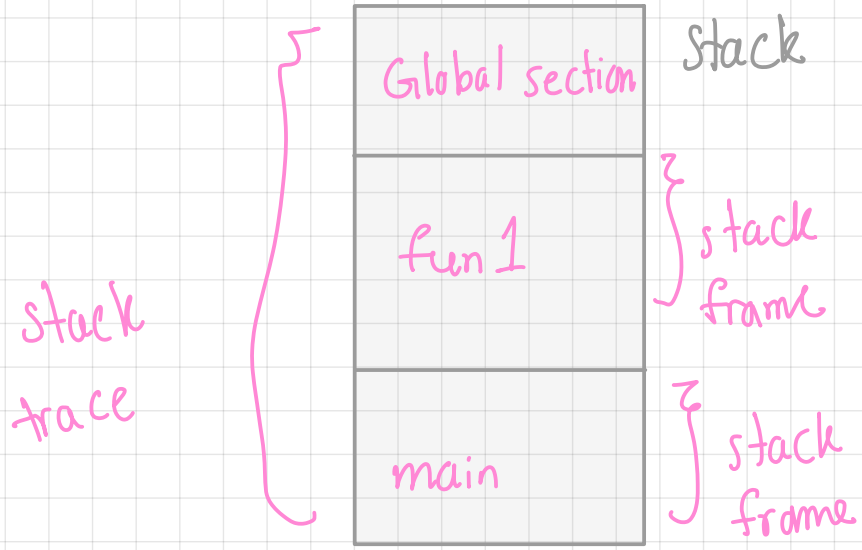
C++ Refrence

* myfun(z) ⇒ Call by value .

⇒ 2 types of Call available:-

① Call by value
myfun (x);

② Call by reference
my fun (&x);

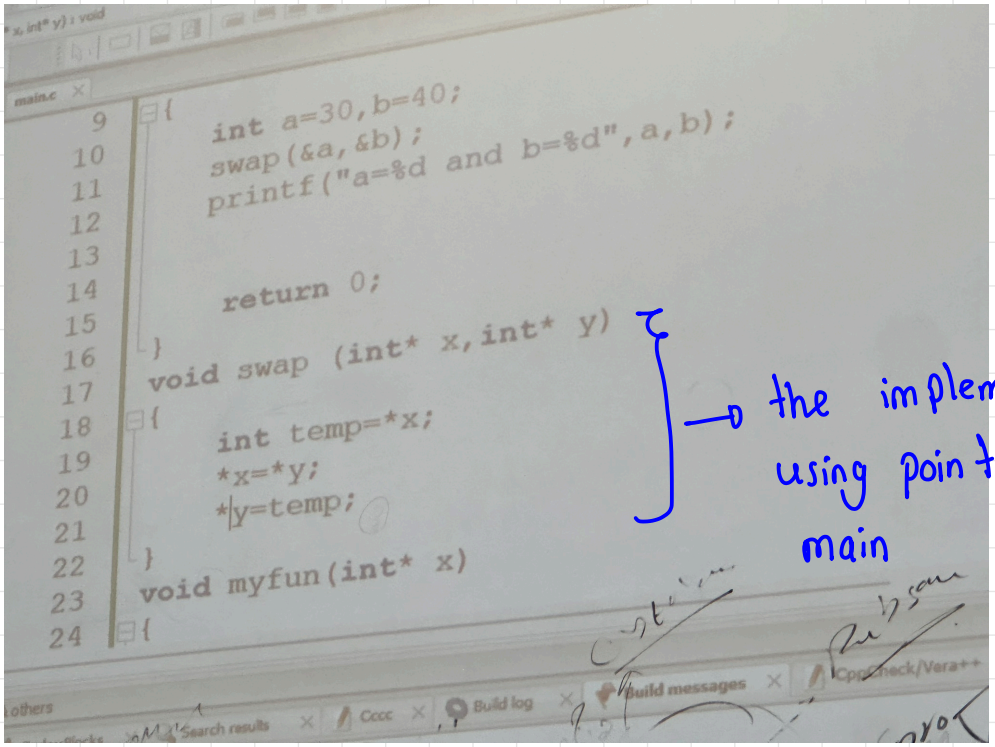


⇒ if

```
int g=3000;

int main ()
{   int g = 30;
    printf ("%d", g);
}   my fun();

my fun ()
{
    g=10;
    printf ("%d", g);
}
```

⇒ 30 which is in its scope

⇒ work with the global so it has been changed to 10



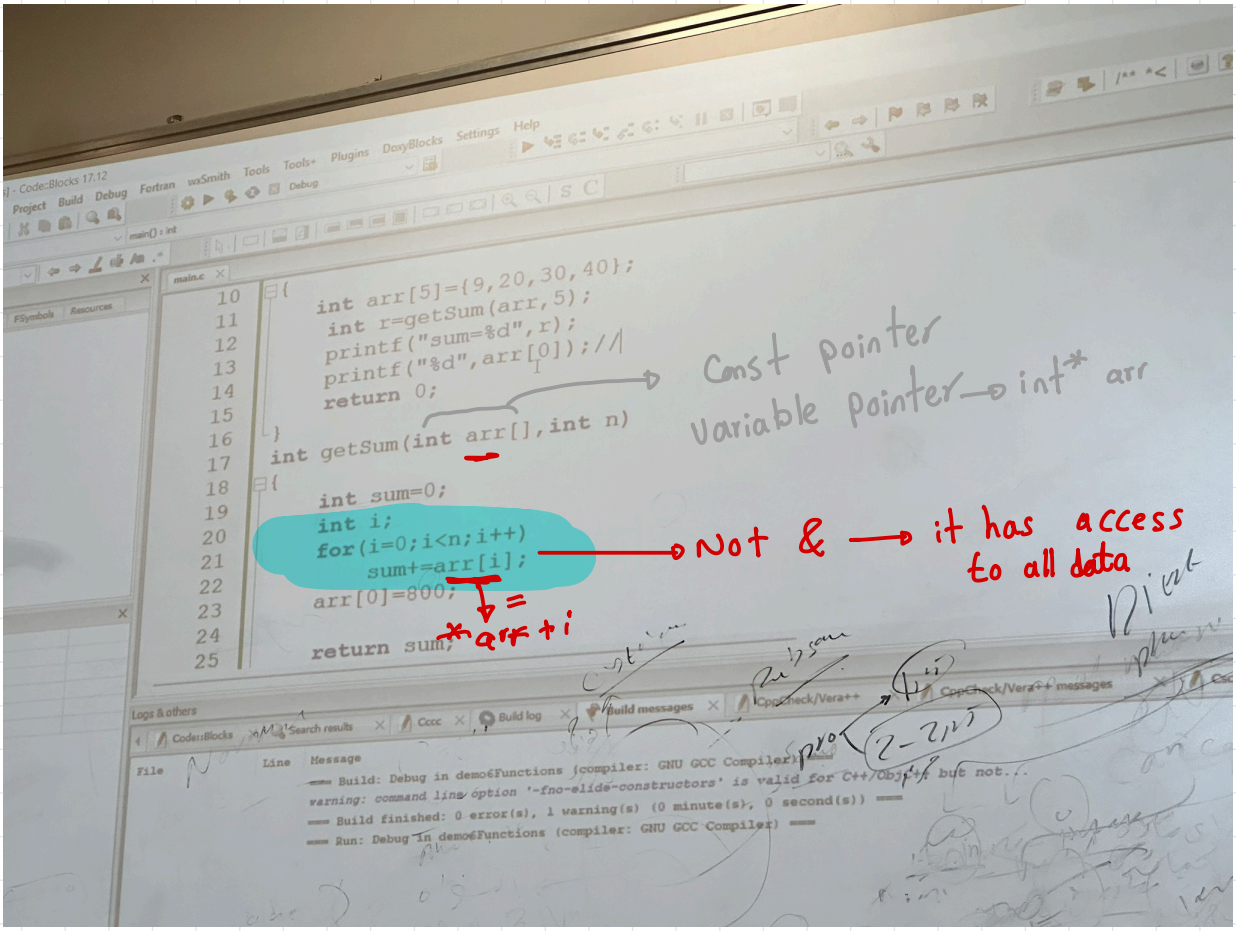
```
9   {
10      int a=30,b=40;
11      swap(&a,&b);
12      printf("a=%d and b=%d",a,b);
13
14
15      return 0;
16  }
17  void swap (int* x,int* y)
18  {
19      int temp=*x;
20      *x=*y;
21      *y=temp;
22  }
23  void myfun(int* x)
24  {
```

the implementation using pointers affect the main

This is a way to make functions return more than one variable

سؤال امتحان

implement function return more than one value in diffrent ways


int arr[5]={9,20,30,40};
int r=getSum(arr,5);
printf("sum=%d",r);
printf("%d",arr[0]);//1
return 0;
int getSum(int arr[],int n)
int sum=0;
int i;
for(i=0;i<n;i++)
sum+=arr[i];
arr[0]=800;
return sum;
Const pointer
Variable Pointer → int* arr
Not & → it has access to all data
*arr+i
Build messages
Build log
Search results
Code::Blocks
Line
Message
=== Build: Debug in demo6Functions (compiler: GNU GCC Compiler) ===
warning: command line option '-fno-elide-constructors' is valid for C++/ObjC++ but not...
=== Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===
=== Run: Debug in demo6Functions (compiler: GNU GCC Compiler) ===

LAB

- factorial function
- get max function
- swap 2 numbers function

*(emp+1)
↳ 38 byte

- scan array function
- get array sum //
- print array //

→ Same file

- scan & print data of array of struct.
- pointer to struct scan & print

to build up struct pointer :-

⇒ struct emp e1 = {10, "aly", 50};
struct *ptr = & e1;

⇒ to print id or other values using pointers (*ptr).id

equal →

(*ptr).id OR ptr → id Not Ptr.id or *ptr.id

printf ("%d", ptr → id);

scanf ("%d", &(ptr → id));

printf ("%d %s %d", ptr → id, ptr →

Address

&ptr → id or &(ptr → id)

Search

DAY 7

```
int main() {
    const int x = 30;
    const int *ptr = &x;
    int y = 40;
    *ptr = 90;
    ptr = &y;
}
```

→ will give compilation error

→ will be okay

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    const int x=30;
    const int *ptr=&x;
    int y=40;
    *ptr=90;
    printf("%d",x);
    ptr=&y;
    y=90;
    *ptr=90;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x=30;
    int *const ptr=&x;
    int y=60;
    ptr=&y;
}
```

→ Can not change the location pointer is reffering to

→ Compilation error

```
int main()
{
    int x = 20;      // 4 byte
    double d = 3.2;  // 8 byte
    d = x;           // implicit ( 4 byte in 8 byte)

    x = d;           // explicit   take the first 4 bytes only
      ↳ x = (int) d
```

bette to write it as x=(int) d;

without explicit ⇒ compilation error.

(malloc ≅ new (other than c))

```
int* ptr = (int*) malloc (size * sizeof(int))
```

explicit casting

void *

Great question! These declarations deal with how pointers and the values they point to are treated.

1. `const int x = 30;`
   - Here, `x` is a constant integer. Its value cannot be changed after initialization.
2. `const int *ptr = &x;`
   - This declares a pointer to a constant integer. The pointer `ptr` can point to different integers, but you cannot change the value of the integer it points to through `*ptr`. However, you can make `ptr` point to another address.
3. `int *const ptr = &x;`
   - This declares a constant pointer to an integer. Here, `ptr` must always point to the same integer (in this case, `x`), but you can change the value of `x` through `*ptr`.

So, the first one focuses on protecting the value of `x`, the second protects the value pointed to by `ptr (but not ptr` itself), and the third keeps the pointer itself constant, but allows modification of the value it points to.

version 2 of malloc

```
int size;
Scanf ("d", size);
int* ptr = (int*) malloc (size * sizeof(int));
if (ptr != Null)
  { Print & Scan)
```

cast

malloc (1)

```
int size;
scanf ("d", size);
int* ptr = malloc (size * sizeof (int));

for (int i=0; i<size)i++)
    scanf ("%d", ptr+i
        or &ptr[i]);

for (int i=0; i<size; i++)
  { printf ("%d", *(ptr+i)
        or ptr[i])

free (ptr); → to release it
    for OS to use
    it later (dead lock)
```

if I typed the cast then it's explicit

if no → implicit

Search about dynamic Array in struct

pointer to pointer

1. `int** *ptp = &ptr;`

2. `int x = 30;`
   `*ptr;` → 30
   `**ptp;` → 30

if malloc didn't find the locations or find it separated ⇒ it returns Null

so we add:-

```
if (ptr != NULL) {
{for (int i=0; i<size; i++)
    Scan
    and print;
}
```

malloc note

LAB

1- one dimension array dynamic

2- one struct array dynamic

3- Menu program

- first ask user size of array.
- `struct emp* ptr = malloc`
- ask what index you want to enter data
- if there any data
- Add search by ID&name&All