Structs

function pointer

Smart Pointer

44

Q. What are Structs ??

→ If we want to store data which comprises of multiple values of non-similar type then we use struct.

also called as Structures

And this is considered as user defined type

keyword

struct <name of struct> {

float s;

int t;

member fields

any no. of member fields

};

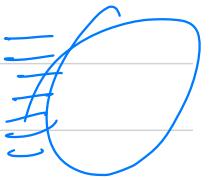
member function

We can access the member fields of a struct using "." operator

Can we give default values to member fields ??

→ Before C++11 this was not allowed. But now we can

In C++ you have constructors for struct



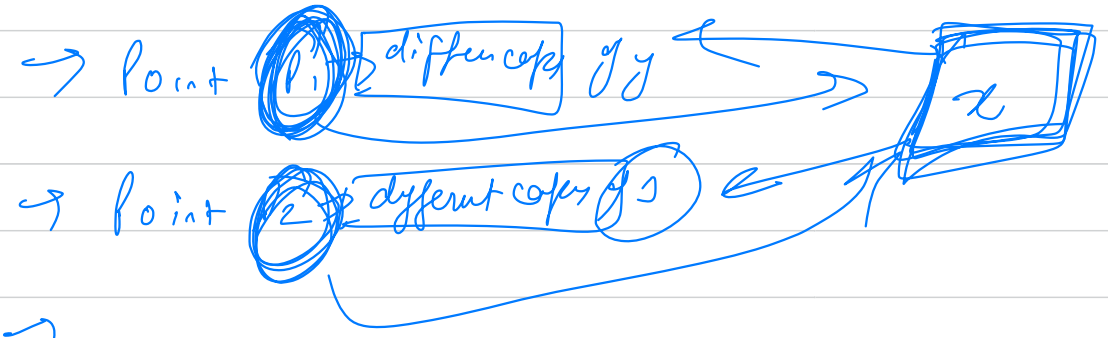
What are static variables ??

struct Point {

int y = 0

static int x;

}



int Point::x = 0;

$a \times c + i a \times d + i b \times c + i^2 bd$ $\quad i^2 = -1$

$\Rightarrow ac + iad + ibc - bd$

real $\rightarrow$ $ac - bd$

img $\rightarrow$ $ad + bc$

Special pointers –

Program → Process → OS

(1) Function Pointer → It will point to functions in memory

It will point to a part of code

It stores the start of that code part.

→ You cannot allocate or de-allocate memory using function pointer

we can pass a function pointer as argument & can return it from any func°

Q. If you don't manually delete any pointer what happens ??

Java has its own garbage collector

delete

Smart Pointers → This helps us such that we don't need to explicitly delete pointers from memory

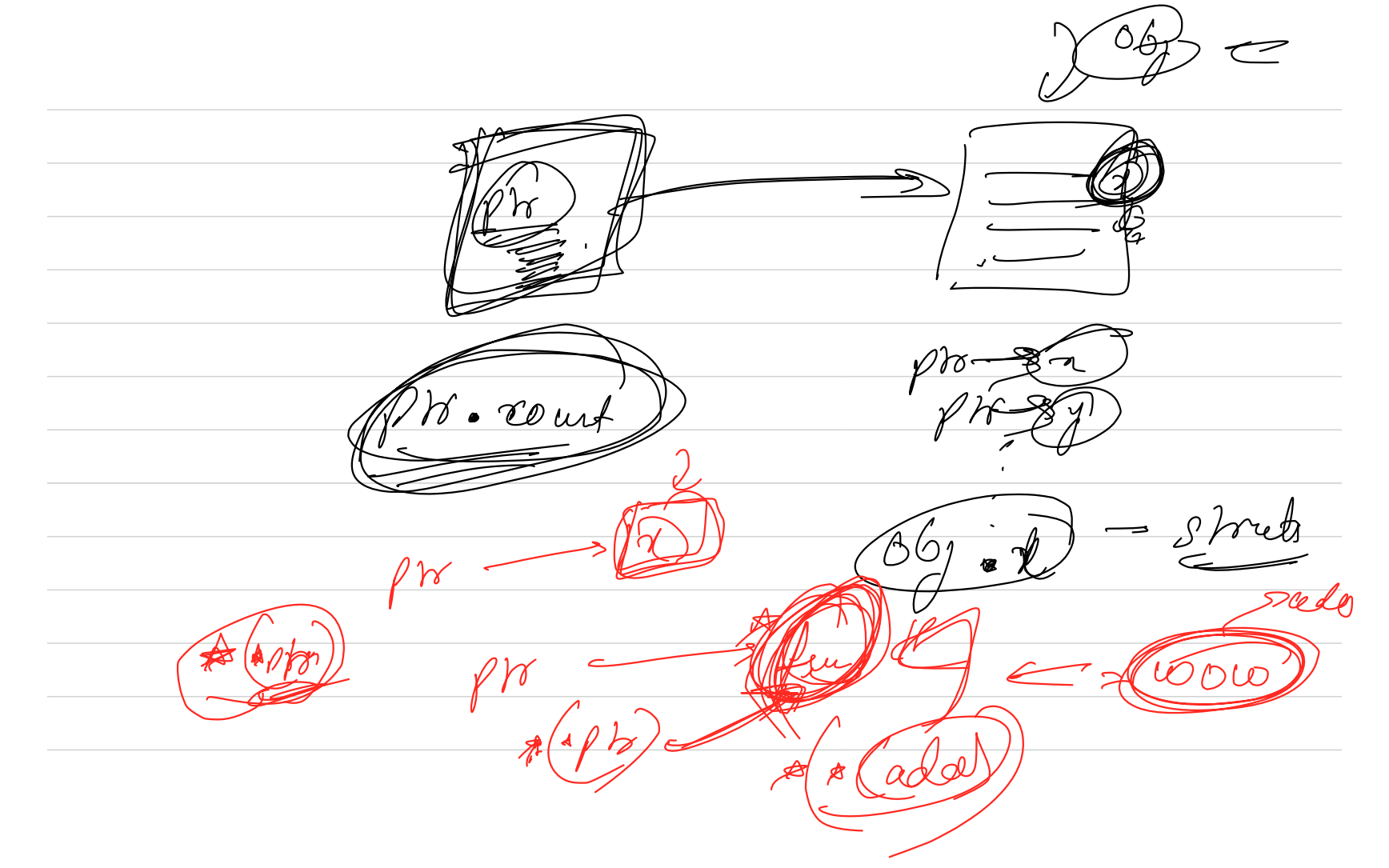
Handles destruction of pointers automatically

↳ unique_ptr → It is type of smart pointer, which handles auto destruction of pointer memory & if you use unique_ptr the if one object is created & ptr points to it then we cannot assign more pointer to point at that memory location.

* Shared_ptr → In this smart pointer all the automatic destruction is done for ptrs as well as you can make multiple pointers point to same location

↳ Reference counter → It keeps count of the ptrs that are maintained to point to a location

use_count() method



Enums in C++

↳ Enumerations → user defined datatype

↳ they map string → value

enum <name> { m=0, f=1 }