C++11: Anonymous Funcs & Blocks

jueves, 5 de mayo de 2022 06:55 a. m.

to pass by r-value in a function, we need move(x) for any l-value.

→ object by value: its main characteristic is that for any copied object, a copy is created: each is a different mem space.
→ another object

→ object by reference: any copied object points to the same memory space, that is, it points to the same object.

Apart from common constructors, other constructor is the "copy constructor"

copy: another object { create an object out of another object

```
A::A(const A &a){
  this->x = a.x;
}
```

We need this so that we control how copies of objects are done

We use them since many objects have a lot objects have matrix attributes and by this constructor we can avoid copying value by value

Since we control how copies are done

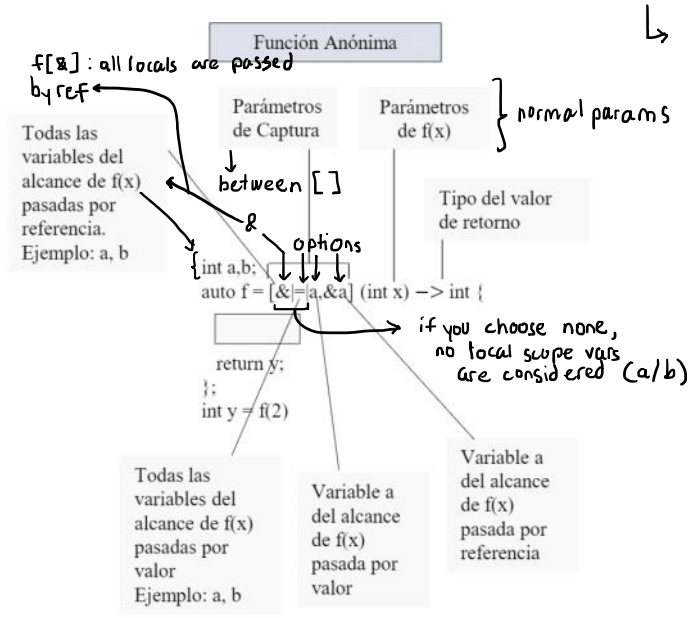
→ we can do constructors with passing by reference to l-value and to r-values

→ Another thing from c++11 are anonymous functions:

↳ a function created in any part of the code that can be passed along to be executed in any part of the code.

auto → compiler decides the type of variable

→ commonly, you use anonymous functions in threads



→ we have a thread class that creates threads by union or by separation.

```
thread h = thread(f,x);
```

So that at the end of process (main thread) end of all threads

must join main thread before it ends

name of the executing functions

params that f will receive

" f(x)"

```
h.join();
```

→ this function will wait for the thread to finish its function and then

in that line

```
thread h = thread(f,x);
h.detach();
```

main continues on its own and finishes all threads when main ends

regardless of threads finishing its function or not.

a type of synchronization — main thread can continue

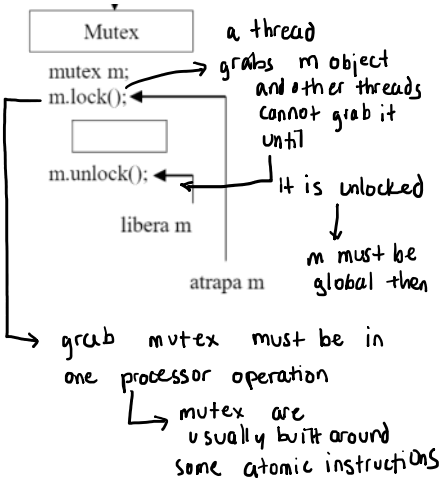
→ Synchronization : block → first synch type — shared resources' access

RAM, screen access, etc

blocks allow to block a code piece that uses a shared resource

Block types: (they do the same)

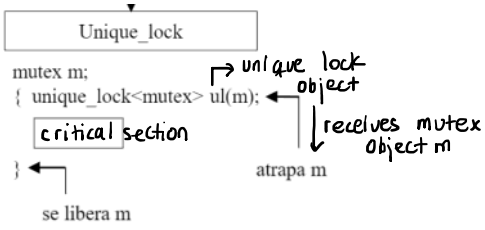
1) Mutex



grab mutex must be in one processor operation

→ mutex are usually built around some atomic instructions

2) unique_lock



→ creates a block

→ inside the block m object is grabbed

lock_guard( ) is the same only with lg object type