C++

ENTRADA|SALIDA

```
#include <iostream>
std::cout << "x";
std::cin >> x;
getline(std::cin, x);
```

Namespace de toda la librería de c++

FUNCIONES

```
inline int f(int &x, int y=1)
```

inline: hace copia para evitar hacer llamada a función.

&x: parámetro por referencia

y=1: valor default

TIPO de DATOS:

bool true|false

```
#include <string>
std::string "x"
```

CLASES:

a.h

```
#ifndef A_H
#define A
class A {
  private:
    int x;
    static int y;
  protected:
  public:
    A(int x); ~A(); A(const A &a);
    virtual int f(int x);
    static int fs(int x);
};
#endif
```

Destructor

Activa Polimorfismo

constructor copia.

a.cpp

```
#include "a.h"
int A::y = 2;
A::A(int x) {
}
A::A(const A &a) {
  this->x = a.x;
}
A::~A() {
}
int A::f(int x) {
}
int A::fs(int x) {
}
```

main.cpp

```
#include "a.h"
int main() {
  A a(2);
  a.f(2);
  A::fs(2);
  A *a = new A(2);
  a->f(2);
  delete a;
  A *a2 = new A(*a);
}
```

A a = A(2); (=)

constructor copia

x = 12;
this->x = 12;
(*this).x = 12;

hereda de A

NOTA: La herencia puede ser multiple.

b.h

```
#ifndef B_H
#define B_H
class B : public A {
  public:
    B(int x);
    int f(int x);
};
#endif
```

```
#include "b.h"
B::B(int x): A(x) {
}
int B::f(int x) {
}
```

Llamada al constructor de A

INTERFAZ:

```
class I {
  public:
    virtual int f() = 0;
};
```

CLASES ABSTRACTAS:

```
class A {
  public
    virtual int f() = 0;
    int g();
};
```

Función virtual pura.

CADENAS:

```
#include <string>
std::string x;
"x" + "y" = "xy"
x.append(y) = "xy"
x.length()
x[2]
x[2] = 'a';
```

MATEMATICAS:

```
#include <cmath>
```

```
#include <math.h>
```

EXCEPCIONES:

```
try {
    throw 205;
} catch (int e) {
} finally {
}
```

CASTING:

```
tipo → string
#include <sstream>
stringstream ss;
ss << 42.3;
string s = ss.str();
```

PUNTEROS INTELIGENTES:

```
#include <memory>
unique_ptr<A> p(new A());
p→f();
```

HASHMAP:

```
#include <map>
map<string, int> m;
m["x"] = 2;
map<string, int>::iterator i;
for (i = m.begin(); i != m.end(); i++){
    k = i→first;
    v = i→second;
}
```

HASHSET:

```
#include <unorder_set>
unorder_set<string> x;
x["2"];
```

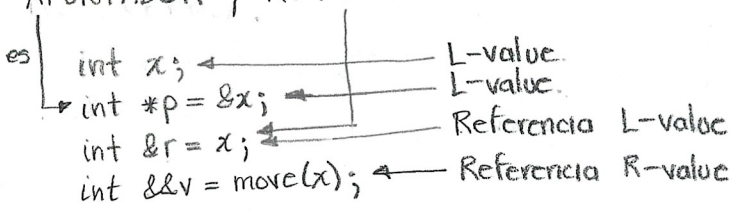
ARRALIST:

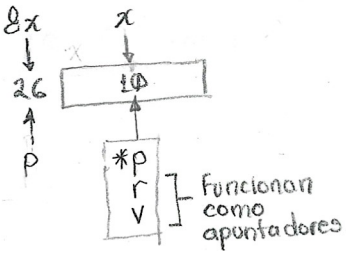
```
#include <vector>
vector<string> x;
x.push_back("x");
x[2];
x.size();
x.pop_back();
x.insert(x.begin()+2, "a");
x.remove(x.begin()+2);
```

posicion a insertar

posición a borrar

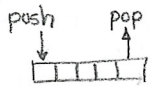
APUNTADOR y REFERENCIA:





NOTA: r es un alias de x

COLA:




```
#include <queue>
queue<int> x;
x.push(2);
x.pop();
x.size();
x.empty();
```