COMPILAR

g++ -o program.exe program.cc

g++ -c program.cc torneig.cc circuit.cc conj_categories.cc jugador.cc ranking.cc partit.cc

g++ -o program.exe program.o torneig.o circuit.o conj_categories.o jugador.o ranking.o partit.o

program.exe < "input.txt"

SORT

```
#include <algorithm>

sort(v.begin(), v.end(), comp)

bool comp(int a, int b) {
    return a > b
};
```

STRUCT

```
struct info{
    int dni;
    string name;
};
info i;

i.dni
i.name
```

PAIR

```
pair<int, char> PAIR1;
PAIR1.first = 100;
PAIR1.second = 'G';

pair<string, double> PAIR2("GeeksForGeeks", 1.23);

pair<string, double> PAIR3;
PAIR3 = make_pair("GeeksForGeeks is Best", 4.56);
```

SWAP

```
swap(v[i], v[k]);
swap(a, b);
```

CLASS .hh

Conjunts → #include “subconjunts.hh”

```
#include "jugador.hh"
#include "partit.hh"
#include "ranking.hh"
#include "conj_categories.hh"
```

#include llibreries

```
#include <iostream>
#include "BinTree.hh"
#include <vector>
#include <map>
#include <cmath>
```

```
class circuit {
private:
     map<string, int> notes;
     int dni;
     string nom;
     bool existeix_torneig(string nom_torneig) const; *Es un metodo pero solo lo llamas tu (privado)
public:
     circuit(); *Constructora
     ~circuit(); *Destructora
void alta_torneig(string op, const conj_categories& cat);  *Modificadora (modifica contingut del priv.)
string consultar_nom() const; *Consultora (amb el const. Nomes consulta el valor sense modificar)
void llegir_aparellaments(ranking& rank); *Lectura/Escriptura
```

CLASS .cc

#include el .hh

```
#include "circuit.hh"
```

Utilitzar el **circuit::** para los metodos de la clase

```
bool circuit::existeix_torneig(string nom_torneig) const{
   map<string, torneig>::const_iterator it = circ.find(nom_torneig);
   if(it == circ.end()) return false;
   else return true;
}
```

VECTOR

```
#include <vector>

vector<bool> v(n, false);
vector<int> v(n, 0);
vector<int> v(n);
vector<int> v;
```

- v.size()
- v.resize()
- v.empty()
- v.push_back()
- v.pop_back()
- v.clear()

MATRIU

```
#include <vector>

typedef vector< vector<int> > matriu;

matrix mat(n, vector <int> (m))
```

MAP

```
#include <map>

map<string, int> d;
```

COUT

- for (map<string, int>::const_iterator it = d.begin(); it != d.end(); ++it);
- for (auto it = d.begin(); it != d.end(); ++it)

```
cout << it->first << it->second;
```

FIND

- d[Albert].edat / d[variable].dni
- map<string, info_persona>::const_iterator it = d.find("Albert");
- auto it = d.find("Albert")

```
if (it == d.end())…/ else…;
```

ERASE

- map<string, info_persona>::const_iterator it = d.find("Albert")
- auto it = d.find("Albert")

```
if (it != d.end()) d.erase(it);
```

- d.erase(d.find("Albert"))

OTROS

- d.insert(make_pair("Manel", info));
- d.size()

- d.empty()
- d.clear()

STACK (pila)
#include < stack >

```
stack<int> pila;
```

- pila.push(variable/numero);
- pila.pop(); //borra el element mes recent
- pila.size();
- pila.empty();
- pila.top(); //el element mes recent

QUEUE (cua)
include < queue >

```
queue<int> cua;
```

- cua::push(variable/numero);
- cua::pop();
- cua::back(); //el element mes recent
- cua::front(); //el primer de la cua (el que porta mes temps / el mes antic)
- cua::empty();
- cua::size();

PRIORITY QUEUE (cua de prioritats)
include < queue >

```
priority_queue<int> cuap;
```

- pila.push(variable/numero);
- pila.pop(); //borra el element mes gran
- pila.size();
- pila.empty();
- pila.top(); //el element mes gran

SET (no hi han elements repetits)
***Es como una lista solo con nombres para ver si existe o no. No guarda mas informacion.**
include < set >

```
set<char> a;
```

COUT
for (set < int >:: iterator it = S . begin () ; it != S . end () ; ++ it) cout << *it << endl ;
for (auto it = S . begin () ; it != S . end () ; ++ it) cout << *it << endl ;
for(auto& str: a) cout << str << ' ' ;
for (int x : a) cout << x << endl ;

ERASE
a.erase(a.find(variable/numero));
a.erase(a.begin(), a.find(30)); // borra desde begin fins al find (30)

FIND
a.find(variable/numero);
if (a.find (x) == a.end ()) //no ha trobat

OTROS

- a.insert(variable/numero);
- a.size()
- a.empty()
- a.clear()