Excepcions en C++.

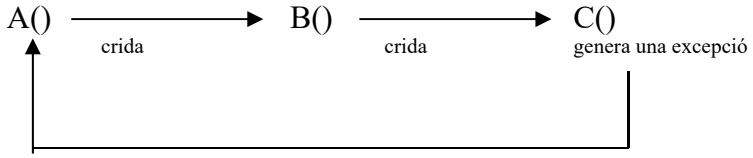
1. Concepte.

Les excepcions són el mecanisme que proporciona el C++ pel tractament dels errors que es puguin generar en un aplicatiu. Permet un tractament **distribuït** de les excepcions mitjançant directives de llançament i recollida d'excepcions. Aquest mecanisme segueix una metodologia de P.O.O.

Una excepció es defineix com un succés inesperat que es genera durant l'execució del programa i que trenca el flux normal d'execució.

2. Avantatges de l'ús de les excepcions.

- Aconseguir un aplicatiu *robust* prevenint-lo dels errors que podrien provocar una finalització inesperada.
- Aconseguir una separació entre el codi específic pel tractament de les excepcions i la resta.
- Possibilitat de propagar els errors a altres funcions:



3. Llançament d'excepcions. La sentència *throw*.

En la forma tradicional, quan una funció detecta un error, retorna un codi d'error a través del valor de retorn de la funció o sobre un paràmetre de sortida, que l'usuari de la funció ha de comprovar per esbrinar si tot ha anat correctament. Mitjançant les excepcions, la funció crearà i llançarà excepcions sense la necessitat de paràmetres d'error que fan l'ús de la funció una mica pesada i poc clara. La sintaxi és:

throw especificació_excepció_llançar

Exemple

```
void g(){
        .....
        if (error)  throw "S'ha produït un error";
            else
                .....
}
```

- La sentència ***throw*** actua com un return, no escriu res, i provoca l'acabament de l'execució de la funció g().
- Acompanyant el ***throw*** ha d'anar un argument que pot ser de qualsevol tipus. En l'exemple anterior és del tipus cadena, que és el tipus de l'excepció que es crearà.
- En una funció hi poden haver tantes sentències ***throw*** com es vulguin.
- Si una funció crea i genera excepcions (té una o més sentències ***throw***) s'ha d'indicar en l'especificació de la funció:

tipus_retorn nom_funcio(<paràmetres>) ***throw*** **(**llista_tipus_excepcions)

En Java no sempre és necessari que el prototipus del mètode indiqui que llança excepcions, tot depèn del tipus d'aquestes. Si cal indicar-ho, en Java cal usar la paraula throws. En Java només es poden llençar objectes, és a dir instàncies de classes.

En l'exemple anterior:

void g() ***throw*** **(char *)**
si la llista és buida vol dir que la funció no pot llançar excepcions

Exemple

```
void g() throw (char*, int){      Sintaxi diferent de Java
        char p[100];
        int dim;
        //sentències que omplen i treballen amb la taula
        if (dim >= 100)
                throw "Error la taula ja està plena";
        else {p[dim]='m';dim++;} //inserim
        ...
        float f,g;
        ...
        if (g==0.0)                    // divisió per zero ?
                throw 4;       // llançament de l'enter 4
        f=f/g;
}
```

- Les excepcions llançades en una funció arriben a la funció que ha cridat a aquesta.

4. Recollida d'excepcions. Sentències *try* i *catch*.

La funció que rep les excepcions pot optar per recollir-les i donar-les-hi el tractament adient o pot optar per deixar-les passar. Per recollir excepcions cal l'ús de la sentència ***try***. Per donar-les-hi tractament cal l'ús de sentències ***catch***. La sintaxi és:

```
try{
        .....      sentències que poden generar excepcions
}
catch (tipus1 excepcio1){
        .....      sentències pel tractament de les
                   excepcions de tipus 1
}
catch (tipus2 excepcio2 ){
        .....      tractament d'excepcions de tipus 2
}
```

- No pot haver cap sentència entre el final del bloc ***try*** i el començament del primer ***catch***.
- Si es tenen varies sentències que poden provocar excepcions es pot:
  - ➔ Agrupar-les totes a dins el bloc ***try*** (estructura mostrada anteriorment), o bé
  - ➔ Posar cada sentència individual dins un bloc ***try*** amb el seu bloc ***catch***

- Cada bloc ***catch*** va acompanyat d'un argument que indica el tipus d'excepció que tracta el bloc ***catch.***
- Funcionament: quan es genera una excepció en una de les sentències situades dins del bloc ***try*** s'abandona l'execució del bloc i es busca seqüencialment el primer bloc ***catch*** amb el mateix tipus de paràmetre que l'excepció que s'ha generat, s'executen totes les sentències especificades dins d'aquest bloc trobat. Un cop executades es continua l'execució amb la primera sentència després de tots els ***catch***.
- Per tant, és important l'ordre de col·locació dels blocs ***catch***.
- Si no es genera cap excepció, no s'executa cap bloc ***catch***. Si no existeix cap ***catch*** adequat al tipus d'excepció que s'ha produït l'execució passa l'excepció a un nivell superior. I en aquest cas, la cerca del bloc ***catch*** comença al mètode superior o pare del que ha produït l'excepció. Si aquest no se n'ocupa, comença una cerca cap amunt.

Exemple

```
void f(){
        .....
        g();  //no vol recollir les excepcions
        // generades en l'execució d'aquesta funció, es passen a un nivell
        // superior  (p.e. la funció que ha cridat f())
        ...
        try {
                g();  // es volen tractar

                ....
        }
        catch (char *param1) {
                // sentències pel tractament de les
                // excepcions tipus cadena
        }
        catch (int param2) {
                //sentències pel tractament de les excepcions tipus enter
                 switch (param2){
                    case 0:
                    .....
                }
        }
        ...
}
```

La lògica de funcionament de les excepcions en C++ i Java és la mateixa. Les diferències bàsicament estan en alguns aspectes de la sintaxi i que, en Java els elements llançats sempre han de ser objectes de la classe o subclasses de Exception.