Gestione delle Eccezioni

1. Definizione delle eccezioni
2. Gestione in C++ e Java

Fallimento di un programma

- I programmi possino fallire per molti motivi: errori nei dati di ingresso, errori di calcolo, indici fuori range, etc.
- La gestione dei fallimenti prevede:
  - Individuazione del fallimento
  - Ripristino del funzionamento
- L'individuazione del fallimento può essere fatta da metodi di un oggetto o da funzioni. Ma cosa fare? Interrompere il programma?

> Chi individua un errore in genere non sa come ripristinare il corretto funzionamento del programma

Allora:

È necessario un meccanismo che segnali i fallimenti (**eccezioni**).

Lanciare un'eccezione

Cosa deve fare un metodo quando individua una situazione problematica?

La soluzione classica prevede che il metodo restituisca un valore che indichi il successo o il fallimento. Ma:

- Il metodo chiamante può dimenticarsi di controllare il valore ricevuto
- Il metodo chiamante può non essere in grado di far nulla in caso di fallimento

Quindi è stato introdotto un meccanismo per le eccezioni:

- Le eccezioni non devono poter essere trascurate
- Le eccezioni devono poter essere gestite da un gestore competente

Lanciare un'eccezione

Quando si individua una condizione d'errore bisogna *lanciare* (**throw**) un oggetto appropriato e basta.

```
#include <vector>
class Stack {

private:
  vector<int> elementi ;

public:
  Stack();
  int size()const;  //restituisce il numero di elementi
  void push(int);  //inserisce un elemento
  void pop();  //toglie l'elemento in cima: errore se lo stack e'vuoto
  int top() const;  //restituisce l'elemento in cima: errore se stack e'
                    //vuoto
};
```

Lanciare un'eccezione in C++

```
#include "stack.h"
#include "eccezione.h"  //classe utente che definisce un errore
Stack::Stack( ){ } //costruttore
int Stack::size( ) const { return elementi.size(); }
void Stack::push(int x){  elementi.push_back(x);  }

void Stack::pop(){
  if (elementi.size()==0)
    throw Eccezione("Stack vuoto, pop() non eseguibile");
   elementi.pop_back(); }

int Stack::top() const{
   if (elementi.size()==0)
     throw Eccezione("Stack vuoto, top() non eseguibile");
   return elementi.back( ); }
```

Esecuzione interrotta

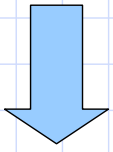
Oggetto di classe Eccezione

Catturare un'eccezione

Tutte le eccezioni dovrebbero essere gestite in qualche punto del programma.

Se un'eccezione non viene gestita **il programma termina**.

Un programma scritto professionalmente non puo' terminare bruscamente.



Gestire le eccezioni !!

Un gestore di eccezioni si installa con l'enunciato

try/catch

Catturare un'eccezione in C++

```
int main(int argc, char *argv[])
{   Stack pila;
     .....
   pila.push(...);
     .....

     try{  .....
        pila.pop();
        ..... }
     catch(Eccezione e) {
      cout << "Errore : " << e.getErrore()<< endl;
      }
    }
```

Se si genera un'eccezione, l'esecuzione si interrompe e ...

si esegue il corpo della **catch** con parametro l'ecce-zione lanciata

Catturare un'eccezione in C++

Il formato generale della try/catch è il seguente

```
     try{  .....
        ......
        ..... }
     catch(Eccezione1 e) {
       ........
       }
     catch(Eccezione2 e) {
       ........
       }
    .....
     catch(EccezioneN e) {
       ........
       }
```

Puo' generarsi piu' di una eccezione

Viene richiamato il corpo della prima catch che puo' prendere come parametro l'eccezione lanciata

Gestire le eccezioni in Java

- In Java la gestione delle eccezioni e' analoga al C++.
- Esistono le parole chiave **throw**, **try** e **catch**.
- La modalita' di uso dell' istruzione **throw** e' identica a quella del C++, cosi' come quella del costrutto **try/catch**.
- Esiste una gerarchia di classi per le eccezioni gia' pronta (es. ArithmeticException, IllegalArgumentException, IllegalStateException, IndexOutOfBoundsException, NullPointerException,...)
- E' consigliabile, per la portabilita' dei programmi, lanciare eccezioni di questa gerarchia.
- Nel caso si voglia comunque creare una nuova classe per eccezioni, bisogna che questa derivi dalla classe “Throwable” definita in **java.lang.Throwable.**