PROGRAMMAZIONE AVANZATA

Le eccezioni

Roberto Nardone, Luigi Romano

ROADMAP

- Cos'è un'eccezione
- Gestione delle eccezioni
  - try
  - catch
  - throw
- Funzioni ed eccezioni
- La classe exception

LE ECCEZIONI IN C++

- In un linguaggio di programmazione con il termine **eccezione** ci si riferisce ad una condizione di errore imprevista che, se non opportunamente gestita, può causare la chiusura del programma.

- Identificare all'interno di un codice tutte le condizioni che potrebbero generare errori e il set di istruzioni da eseguire per riparare ad esse non è affatto banale.

- Per questi motivi il programmatore delega la gestione delle eccezioni all'apposito meccanismo di gestione delle eccezioni messo a disposizione dal linguaggio C++.

ESEMPI DI ECCEZIONI

- Quando si parla di eccezioni ci si riferisce ad errori a **runtime**, quindi non identificabili a tempo di compilazione, ma legati al verificarsi di condizioni eccezionali in fase di esecuzione:

  - verificarsi di una divisione per zero;
  - overflow su di un dato durante una operazione aritmetica;
  - un'operazione di casting non valida;
  - riferirsi ad una periferica che in quel momento risulta scollegata;
  - fallimento nell'allocazione in memoria;
  - ……..

GESTIONE DELLE ECCEZIONI

- Quando durante l'esecuzione di un programma si verifica una condizione di errore imprevista, se è attivo il processo di gestione delle eccezioni, quest'ultimo prende il controllo del flusso di esecuzione.

- L'inclusione di tale meccanismo, noto come **catching an exception,** non è a costo zero ma prevede l'aggiunta della porzione di codice per catturare e gestire il comportamento anomalo rilevato.

- Il meccanismo di gestione delle eccezioni si serve di tre parole codice:
  - **try**
  - **catch**
  - **throw**

IL BLOCCO TRY

- La gestione di una eccezione richiede per prima cosa l'uso di un **blocco try** per testare il codice a ''rischio eccezione''.
- È buona norma inserire nel blocco try la sola porzione di codice che potrebbe generare l'eccezione.
- Se il codice nel blocco **try** verifica una condizione di errore, l'eccezione viene lanciata **(throw),** altrimenti l'esecuzione del programma è quella standard**.**
- A ciascun **blocco try** deve seguire almeno un **blocco catch,** ma sono ammessi anche più blocchi catch in sequenza.

TRY ANNIDATI

- È possibile annidare più blocchi try. Un blocco try interno manda in esecuzione il gestore catch interno, viceversa, un try esterno esegue quello del catch esterno.

```
try{
        //codice try esterno
        try{
                //codice try interno
        }
        catch(argomento){//codice catch interno}
}
catch(argomento){//codice catch esterno}
```

IL BLOCCO CATCH

- Il gestore delle eccezioni **catch** consiste di 3 parti: la parola riservata **catch**, la dichiarazione fra parentesi di un solo argomento e il blocco di istruzioni da eseguire per gestire l'eccezione.

- A differenza del blocco try, il blocco catch viene eseguito solo in condizioni particolari (quelle testate nel blocco try).

- L'argomento del catch specifica il tipo di eccezione che è in grado di catturare.

- Esiste un blocco catch che intercetta ogni tipo di eccezione, è il **blocco catch di default** e il suo parametro è sostituito da tre punti (**catch(…)**).

FORME DI UN BLOCCO CATCH

1. catch(argomento1) { **//cattura l'eccezione corrispondente all'argomento1**
   codice del 1° blocco catch
   }

2. catch(argomento2) { **//cattura l'eccezione corrispondente all'argomento2**
   codice del 2° blocco catch
   }

3. catch(…){ **//cattura qualsiasi tipo di eccezione**
   codice del blocco catch all
   }

L'ISTRUZIONE THROW

- **throw** è la parola chiave per lanciare un'eccezione.

- L'istruzione **throw** è tipicamente seguita da un'espressione che informa sul tipo di eccezione sollevata.

throw ''divisione per zero'';

- L'espressione ''divisione per zero'' informa sul motivo dell'eccezione e specifica che il catch corrispondente sarà quello in grado di gestire eccezioni di tipo **const char*.**

ESEMPIO DI GESTIONE DI UN'ECCEZIONE


```
#include <iostream>

using namespace std;

int dividi(int a, int b)
{
        if(b==0)
        {
                throw "Divisione per zero !!!";
        }
        return (a/b);
}

int main()
{
        int x, y;
        double z;
        cout<<"Inserire numeratore "<<endl;
        cin>>x;
        cout<<"Inserire denominatore "<<endl;
        cin>>y;
        try{
                z = dividi(x,y);
        }catch(const char* messaggio){
                cerr<<messaggio<<endl;
        }
        return 0;
}
```

La funzione dividi() lancia senza gestire un'eccezione quando b è nullo.

La gestione dell'eccezione avviene nel main() all'atto della chiamata alla funzione dividi() incorporata nel try.

Il catch cattura l'eccezione e stampa a video il messaggio di errore.

cerr è la funzione della libreria standard per l'output a video degli errori

FUNZIONI ED ECCEZIONI

- C++ offre la possibilità di specificare le eccezioni che una funzione può generare.

- Le eccezioni che una funzione può generare vanno specificate sia nella sua dichiarazione che nella definizione:

```
void f(argomenti) throw (T1, T2,..,Tn);
void g(argomenti) throw();
void h(argomenti);
```

- La funzione f() può lanciare solo eccezioni di tipo T1…Tn, g() non genera alcuna eccezione, h() può generare qualsiasi eccezione.

ESEMPIO FUNZIONI ED ECCEZIONI


```
#include <iostream>

using namespace std;

void f(char c) throw(int, double)
{
        if(isupper(c)) throw(0);
        else if(islower(c)) throw(1.0);

}

int main()
{
        char a;
        cin>>a;

        try
        {
                f(a);
        }catch(int i){

                cerr<<i<<endl;

        }catch(double d){

                cerr<<d<<endl;

        }
}
```

La funzione **f()** può generare due eccezioni, uno di tipo **int**, l'altra **double**.

Il blocco try testa la porzione di codice a rischio eccezioni (la funzione **f()**).

Due blocchi catch, uno con **argomento intero e l'altro double** si occuperanno della gestione delle eventuali eccezioni sollevate.

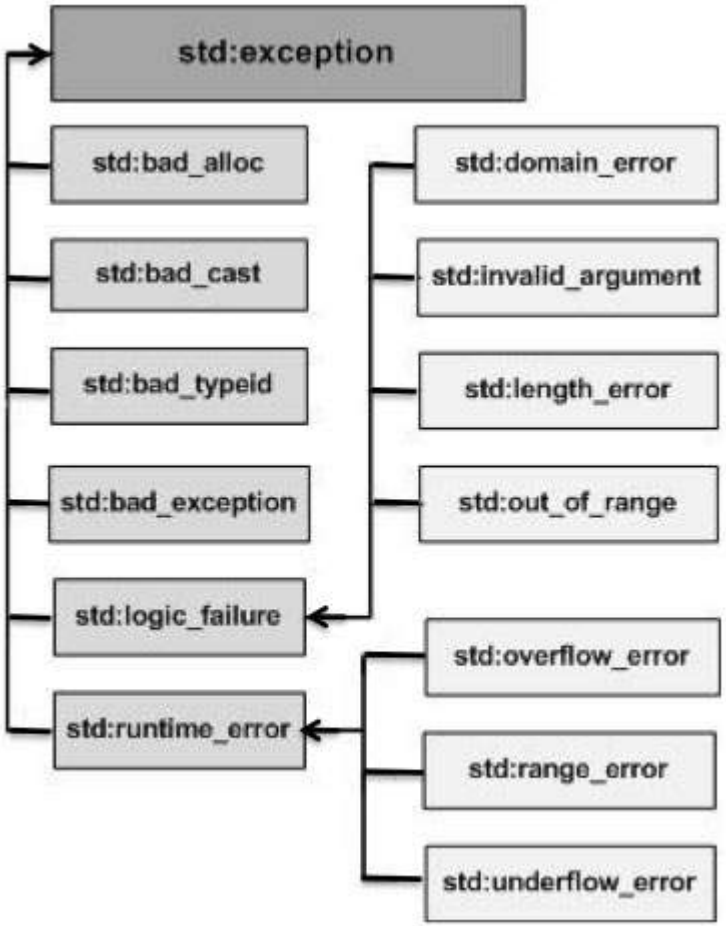
ECCEZIONI - STANDARD TEMPLATE LIBRARY

- La Standard Template Library definisce la classe base **exception** per gestire eccezioni standard.
- Le funzionalità della classe **exception** possono essere sfruttate includendo nel programma l'header file **<exception>.**
- La classe **exception** mette a disposizione una funzione **what()** che produce la stringa con l'identificativo dell'eccezione.
- Alcune eccezioni gestite dalla classe **exception** sono:

bad_alloc, bad_cast, bad_function_call, bad_typeid, invalid_argument ….

BAD_ALLOC EXCEPTION


```
#include <iostream>
#include <exception>
using namespace std;
int main () {
    try {
        int* myarray= new int[1000];
    }
    catch (exception& e)
    {
       cout << "Standard exception: " << e.what() << endl;
    }
    return 0;
}
```


std:exception
std:bad_alloc
std:bad_cast
std:bad_typeid
std:bad_exception
std:logic_failure
std:runtime_error
std:domain_error
std:invalid_argument
std:length_error
std:out_of_range
std:overflow_error
std:range_error
std:underflow_error

ESEMPIO


```
class VectorInSpecialMemory {
        int sz;
        int* elem;
    public:
        VectorInSpecialMemory(int s)
            : sz(s)
            , elem(AllocateInSpecialMemory(s))
        {
            if (elem == nullptr)
                throw std::bad_alloc();
        }
        ...
};
```

DEFINIRE NUOVE ECCEZIONI


```
#include <iostream>
#include <exception>
using namespace std;

struct MyException : public exception {
   const char * what () const throw () {
      return "C++ Exception";
   }
};

int main() {
   try {
      throw MyException();
   } catch(MyException& e) {
      std::cout << "MyException caught" << std::endl;
      std::cout << e.what() << std::endl;
   } catch(std::exception& e) {
      //Other errors
   }
}
```

ESERCIZIO

Scrivere un programma che risolva un' equazione di secondo grado.

L'applicazione dovrà stampare a video le soluzioni dell'equazione per:

- **det>0** (Soluzioni reali e distinte)
- **det=0** (Soluzioni reali e coincidenti)

L'applicazione lancerà un'eccezione nel caso in cui il è **det<0** stampando a video un messaggio di errore.

Realizzare la classe e i metodi opportuni e testare il corretto funzionamento del programma.

Contact Info

CONTACTS