# POO - Excepții

# Excepții

# Excepții - concept

În momentul în care un program de C++ rulează pot să apară diferite erori, erori introduse de programatori, erori ce apar de la un input greșit sau alte exemple neprevăzute. În momentul în care o astfel de eroare apare programul se oprește și va genera o eroare. Termenul folosit este că C++ va arunca o eroare (throw an error).

Unul din avantajele limbajului C++ față de C sunt managementul excepțiilor. Acestea sunt anomalii care apar la runtime.

Excepțiile oferă o metodă de transfer a controlui dintr-o parte a programului la alta. În C++ managementul excepțiilor se realizează cu trei cuvinte cheie:

**try, catch, throw**

# try, catch, throw

**try** - un bloc care conține try, identifică o bucată de cod pentru care excepții vor fi activate. Este urmată de mai multe blocuri de **catch**

**catch** - reprezintă o bucată de cod care va fi executată în momentul în care apare o eroare în blocul **try**

**throw** - excepția din cadrul unui bloc **try** este aruncată către blocul **catch**

În cazul în care un bloc aruncă o excepție, o metodă prinde excepția folosind combinația de try - catch. Codul din interiorul blocurilor try catch este considerat cod protejat.

# Sintaxă

```
try {
  // cod de validat
  throw exception; // arunca o exceptie cand apar problemele
}
catch () {
  // cod de management al erorilor
}
```

Exemplu try-catch

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

float impartire(int a, int b)
   if( b == 0 ) {
      throw "Impartire la zero
   }
   return (a/b);
}

int main () {
    int x = 30;
    int y = 0;
    float z;
    try {
        z = impartire(x, y);
        cout << z << endl;
    } catch (const char* msg)
        cerr << msg << endl;
    }

    return 0;
}

```

# Avantaje

1. Separarea managementului de erori de codul normal - codul de *error handling* este separat ducând la o mai bună menținere a codului și lizibilitate a acestuia
2. Funcțiile și metodele pot face management la orice - o funcție poate arunca mai multe excepții, dar poate să aleagă cum face management la ele
3. Gruparea erorilor - atât primitive cât și obiecte pot fi aruncate ca excepții

# Exemple și observații

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

int main()
{
   int x = -1;

   cout << "Inainte de try" << endl;
   try {
      cout << "In interiorul blocului de try" << endl;
      if (x < 0)
      {
         throw x;
         cout << "Dupa throw (nu se executa) \n";
      }
   }
   catch (int x ) {
      cout << "Exceptie prinsa" << endl;
   }

   cout << "Dupa catch";
   return 0;
}
```

Code will never be executed

Obs - după throw se sare direct la **catch**, nu pot exista blocuri de cod între

# catch all

În cazul în care vrem să avem o bloc de cod care să prindă mai multe/orice tip de eroare putem folosi următoarea sintaxă:

```
try {
    // cod
} catch(...) {
   // cod pentru orice exceptie
}
```

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

int main()
{
    try  {
       throw 10;
    }
    catch (string excp)  {
        cout << "Caught " << excp;
    }
    catch (...)  {
        cout << "Default Exception\n";
    }
    return 0;
}
```

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

int main()
{
    try  {
       throw 10;
    }
    catch (int excp)  {
        cout << "Caught " << excp;
    }
    catch (...)  {
        cout << "Default Exception\n";
    }
    return 0;
}
```

Exemplu catch all

Dacă o excepție este aruncată și nu e prinsă nicăieri, compilatorul va arunca următoarea eroare:

**terminating with uncaught exception of type ...**

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      try  {
7          throw 'x';
8      }
9      catch (int x)  {
10         cout << "Prins ";
11     }
12     return 0;
13 }
```

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

int main()
{
    try {
        try {
            throw 10;
        }
        catch (int n) {
            cout << "Handle 1";
            throw;
        }
    }
    catch (int n) {
        cout << "Handle 2";
    }
    return 0;
}
```

Blocurile try-catch pot fi încapsulate (nested)

În momentul în care un bloc conține obiecte, toate obiectele sunt distruse înainte să se treacă la blocul de catch.

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

class Example {
public:
    Example() { cout << "Constructor called " << endl; }
    ~Example() { cout << "Destructor called " << endl; }
};

int main()
{
    try {
        Example examplu;
        throw 10;
    }
    catch (int x) {
        cout << "Caught " << x << endl;
    }
}
```

De asemenea, o excepție care aparține de o clasă derivată trebuie prinsă înainte de excepția clasei de bază

```
1  #include<iostream>
2  using namespace std;
3
4  class Parent {};
5  class Child: public Parent {};
6  int main()
7  {
8     Child copil;
9     try {
10        throw copil;
11    }
12    catch(Parent p) {
13         cout<<"Caught Parent Exception";
14    }
15    catch(Child c) {                    ⚠ Exception of type 'Child' will be caught by earlier handler
16         cout<<"Caught Child Exception";
17    }
18    getchar();
19    return 0;
20 }
```

# Summary

# De Reținut

**try** - un bloc care conține try, identifică o bucată de cod pentru care excepții vor fi activate. Este urmată de mai multe blocuri de **catch**

**catch** - reprezintă o bucată de cod care va fi executată în momentul în care apare o eroare în blocul **try**

**throw** - excepția din cadrul unui bloc **try** este aruncată către blocul **catch**

# Exerciții

# Exerciții

1. Realizați o aplicație de căutare și rezervare zboruri. Aplicația trebuie să aibă următorii actori:
   - Operator: activități de login și adăugare/ștergere zboruri (contul este deja existent în sistem cu user și parolă)
   - Utilizator: activități de creare cont, autentificare, căutare zbor, rezervare

Implementați metodele de management a excepțiilor pentru acțiunile următoare:

- Operator: login neurește (user sau parolă greșită), adăugare detalii greșite la zbor (format dată greșită, dată în trecut, țară să aibă caractere neexistente în numele țărilor)
- Utilizator: login nereușit, autentificare nereușită (format email necorect, parolă prea slabă, parola repetată nu e corectă), zbor inexistent, introducere detalii greșite la rezervare zbor

Read me - in care sa prezentati cum se face login-ul, autentificarea in mod normal