Tutoriat 7

Excepţii

Excepţiile sunt nişte erori neprevăzute care apar în timpul rulării unui program. De obicei, acestea duc la oprirea execuţiei acelui program şi apar din neatenţia developer-ului.

Exemplu:

```
1 int x, y;
2 cin >> x >> y;
3
4 // x = 5, y = 0
5 cout << x / y; // eroare, impartire la 0
```

Genul acesta de erori apar la rularea programului, nu la compilare şi, în cazul în care nu sunt tratate corespunzător, pot încheia execuţia soft-ului.
Exemplul de mai sus este simplist, dar în practică apar excepţii mult mai grave şi, cu adevărat, greu de prevăzut. *Spre exemplu:*

- Conexiunea la baza de date a eşuat;
- Fişierul nu a putut fi deschis cu succes;
- Conversia nu a fost realizată cu succes;
- Input-ul transmis este invalid;
- ...

Pentru că aceste feluri de erori erau rezolvate cu dificultate, a fost introdusă o modalitate mai uşoară de a le administra, fără a se încheia execuţia programului (unde acest lucru este posibil).

Astfel, de-a lungul realizării oricărui program, trebuie să avem în vedere orice posibilă eroare care ar putea fi cauzată fie de utilizator, fie de server şi în multe alte situaţii.

Block-ul de instrucţiuni try-catch-throw

Acest block de instrucţiuni, asemănător unui *do-while,* este modul în care tratăm aceste excepţii. *Exemplu:*

```
int x, y;
cin >> x;

try {
  cin >> y;
  if (y == 0) {
    throw "Exceptie";
  }
}
catch (const char* e) {
  cout << "Nr. invalid";
}
```

Acum trebuie să lămurim mai multe aspecte:

1. ***Execuţia unui try-catch***

Să vedem în ce ordine se execută instrucţiunile într-un astfel de block:

- Prima dată se execută, pe rând, ce se află în *try*;

- Dacă se întâlneşte o excepţie sau este aruncată una (folosind *throw)* se opreşte execuţia block-ului *try* şi se caută un *catch* care să prindă exact tipul excepţiei aruncate;
  - Dacă acest tip se găseşte, se va executa ce se află în interiorul acelui *catch;*
  - Altfel, se opreşte execuţia programului;

- Dacă nu s-a oprit încă execuţia, după ce se termină *catch* de executat, dacă nu a fost întâmpinată o problemă prea mare, se reia execuţia programului după block-ul *try-catch.*

2. *Tipuri de excepţii*

Acum, am observat că este foarte important *tipul excepţiilor*. Există tipuri de excepţii pe care le puteţi ”arunca”, folosind *throw:*

- *Tipuri de date obişnuite (int, double, string, char*...).*

  Ca în exemplul de mai sus, unde am aruncat un *const char*,* pe care l-am prins în catch-ul de jos.
  Putem arunca orice fel de tip de date, atât timp cât există un *catch* care să prindă acel tip de date. Altfel, programul nu rulează cum trebuie.

- *Obiecte excepţii (clasele derivate ale clasei exception)*

  Mi s-a părut important să fac distincţia aceasta, pentru că acest tip de date, *exception* este foarte util şi mai folosit în practică decât aruncarea unor tipuri de date primitive.
  Sunt multe clase care derivă din clasa *exception,* eu voi da un exemplu cu *runtime_error*. Mai multe exemple de clase la link-urile următoare:

  ➔ https://www.tutorialspoint.com/cplusplus/cpp_exceptions_handling.htm

  ➔ http://www.cplusplus.com/doc/tutorial/exceptions/

```
int x, y;
cin >> x;

try {
  cin >> y;
  if (y == 0) {
    throw runtime_error("Impartire la 0");
  }
}
catch (runtime_error e) {
  cout << e.what();
}
```

3. *Propagarea excepţiilor*

Ceva important de reţinut la excepţii este următorul lucru: *acestea se propagă de la o funcţie la alta, în funcţie de cum sunt apelate aceste funcţii.*

Exemplu:

```
1 void f() {
2    throw runtime_error("Mesaj eroare");
3 }
4
5 void g() {
6   f(); // apeleaza f care intoarce o exceptie
7   // => si g intoarce aceeasi exceptie
8 }
9
10 int main() {
11   try {
12     g();
13   } catch (runtime_error e) {
14     cout << e.what(); // Mesaj eroare
15   }
16 }
```

Aici putem observa clar următoarele lucruri:

- Funcţia *f()* nu face altceva decât să arunce o excepţie pe care nu o tratează. Din cauza lipsei, blocului *try-catch*, acea excepţie e aruncată mai departe la funcţia care o apelează pe *f,* în cazul nostru *g()*.

- Funcţia *g()* apelează funcţia *f()* dar nu tratează nici ea excepţia aruncată din f(). Astfel, excepţia este iarăşi aruncată;

- În *int main()* se apelează *g(),* dar de data asta se şi tratează excepţia, care e prinsă într-un block *catch.*

OBS. Dacă excepţia nu ar fi fost prinsă nici în *int main(),* atunci programul s-ar fi oprit cu o eroare.