Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra Informatică Aplicată

**RAPORT**

Lucrarea de laborator nr.7

Programarea Orientata pe Obiecte

A efectuat:

st. gr. C-171 D. Melniciuc

A verificat:

dr., conf.univ. M. Oșovschi

Chişinău 2018

**Lucrarea de laborator nr. 7.**

**Tema:** Prelucrarea excepţiilor. Blocul *try{…} throw() catch()…*

**Scopul lucrării**: familiarizarea studenţilor cu prelucrarea excepţiilor, lucrul cu blocul *try{…} throw () catch()…*

**Consideraţiile teoretice necesare:**

***Tratarea excepţiilor***

Excepţiile sînt situaţiile neaşteptate apărute în cadrul sistemului care rulează un program. Programele trebuie să conţină proceduri de tratare a acestor situaţii excepţionale.

In C++ s-a realizat un mecanism de tratare a excepţiilor. Astfel, *o excepţie* este un obiect a cărui adresă este trimisă dinspre zona de cod, unde a apărut problema, către o zonă de cod, care trebuie s-o rezolve.

Paşii care trebuie, în general, urmaţi în vederea tratării excepţiilor în cadrul programelor C++ sînt:

* se identifică acele zone din program, în care se efectuează o operaţie despre care se cunoaşte că ar putea genera o excepţie şi se marchează în cadrul unui bloc de tip *try*. In cadrul acestui bloc, se tastează condiţia de apariţie a excepţiei, şi în caz pozitiv se semnalează apariţia excepţiei prin intermediul cuvîntului- cheie *throw*;
* se realizează blocuri de tip *catch,* pentru a capta excepţiile atunci cînd acestea sînt întîlnite.

Blocurile *catch* urmează un bloc *try*, în cadrul cărora sînt tratate excepţiile.

Sintaxa pentru *try*:

*try {*

// cod

*throw TipExcepţie;*

*}*

Sintaxa pentru *throw*:

*throw TipExcepţie;*

Sintaxa pentru *catch*:

*catch(TipExcepţie)  
{*

// cod tratare excepţie

*}*

Dacă *TipExcepţie* este "*...*", este captată orice excepţie apărută.

După un bloc *try*, pot urma unul sau mai multe blocuri *catch*. Dacă excepţia corespunde cu una din declaraţiile de tratare a excepţiilor, aceasta este apelată. Dacă nu este definită nici o funcţie de tratare a excepţiei, sistemul apelează funcţia predefinită, care încheie execuţia programului în curs. După ce funcţia este executată, programul continuă cu instrucţiunea imediat următoare blocului *try*.

*TipExcepţie* nu este altceva decît instanţierea unei clase vide (care determină tipul excepţiei), care poate fi declarată ca:

*class TipExcepţie {};*

În continuare vom prezenta un exemplu de program care utilizează tratarea excepţiilor.

*#include <iostream.h>*

*#define MAXX 80*

*#define MAXY 25*

*class Point*

*{ public:*

*class xZero {};*

*class xOutOfScreenBounds {};*

*Point(unsigned x1, unsigned y1)*

*{ x =x1; y =y1; }*

*unsigned GetX() { return x; }*

*unsigned GetY() { return y; }*

*void SetX(unsigned x1)*

*{ if(x1 > 0)*

*if(x1 < = MAXX) x =x1;*

*else throw xOutOfScreenBounds();*

*else throw xZero();*

*}*

*void SetY(unsigned y1)*

*{ if( y1 > 0)*

*if( y1 < = MAXY) y =y1;*

*else throw xOutOfScreenBounds();*

*else throw xZero();*

*}*

*protected:*

*int x, y;*

*};*

*void main()*

*{ Point p(1, 1);*

*try*

*{ p.SetX(5);*

*cout<<"p.x successfully set to "<<p.GetX()<<"."<< endl;*

*p.SetX(100);*

*}*

*catch(Point::xZero)*

*{ cout << "Zero value!\n"; }*

*catch(Point::xOutOfScreenBounds)*

*{ cout << "Out of screen bounds!\n"; }*

*catch(...)*

*{ cout << Unknown exception!\n"; }*

*}*

Rezultatul îndeplinirii programului:

*p.x successfuly set to 5.*

Datorită faptului că excepţia este instanţierea unei clase, prin derivare pot fi realizate adevărate ierarhii de tratare a excepţiilor. Trebuie de avut însă în vedere posibilitatea de apariţie a unor excepţii chiar în cadrul codului de tratare a unei excepţii, situaţii care trebuie evitate.

**Varianta 4:**

Scrieţi un program care formează un fişier nou selectîndu-se din trei fişiere date mai întîi numerele negative, zerourile, apoi numerele pozitive.

***Codul programului:***

*#include <iostream>*

*#include <fstream>*

*#include <stdio.h>*

*#include <array>*

*using namespace std;*

*int main(int argc, char const \*argv[]) {*

*system("cls");*

*cout <<"lista erorilor:\n";*

*cout <<"eroarea 1: problema cu negative.txt\n";*

*cout <<"eroarea 2: problema cu positive.txt\n";*

*cout <<"eroarea 3: problema cu zero.txt\n";*

*system("pause"); system("cls");*

*int arrN[10000]; //negative*

*int arrZ[10000]; //zero*

*int arrP[10000]; //positive*

*int arrN2[10000];*

*int arrZ2[10000];*

*int arrP2[10000];*

*int i = 0;*

*int n = 0;*

*ifstream inFile1;*

*inFile1.open("negative.txt");*

*try{*

*if (inFile1.fail()) {*

*throw 1;*

*}*

*}*

*catch(int x){*

*cout << "\n\n\n[-] erroarea cu numarul: " << x <<endl;*

*cerr << "erroare cu text file\n\n" << endl;*

*system("pause");*

*exit(1);*

*}*

*while( !inFile1.eof() ){*

*inFile1 >> arrN[i];*

*i++; n++;*

*}*

*int nn = n;*

*inFile1.close();*

*cout << "matricea din negative.txt:\n";*

*for (i = 0; i < n; i++) {*

*if (arrN[i] < 0)*

*arrN2[i] = arrN[i];*

*cout << arrN[i] << " ";*

*}*

*ifstream inFile2;*

*inFile2.open("positive.txt");*

*try{*

*if (inFile2.fail()) {*

*throw 2;*

*}*

*}*

*catch(int x){*

*cout << "\n\n\n[-] erroarea cu numarul: " << x <<endl;*

*cerr << "erroare cu text file\n\n" << endl;*

*system("pause");*

*exit(1);*

*}*

*i = 0; n = 0;*

*cout << "\n\n";*

*while( !inFile2.eof() ){*

*inFile2 >> arrP[i];*

*i++; n++;*

*}*

*int np = n;*

*inFile2.close();*

*cout << "\nmatricea din positive.txt:\n";*

*for (i = 0; i < n; i++) {*

*if (arrP[i] > 0)*

*arrP2[i] = arrP[i];*

*cout << arrP[i] << " ";*

*}*

*ifstream inFile3;*

*inFile3.open("zero.txt");*

*try{*

*if (inFile3.fail()) {*

*throw 3;*

*}*

*}*

*catch(int x){*

*cout << "\n\n\n[-] erroarea cu numarul: " << x <<endl;*

*cerr << "erroare cu text file\n\n" << endl;*

*system("pause");*

*exit(1);*

*}*

*i = 0; n = 0;*

*cout << "\n\n";*

*while( !inFile3.eof() ){*

*inFile3 >> arrZ[i];*

*i++; n++;*

*}*

*int nz = n;*

*inFile3.close();*

*cout << "\nmatricea din zero.txt:\n";*

*for (i = 0; i < n; i++) {*

*if (arrZ[i] == 0)*

*arrZ2[i] = arrZ[i];*

*cout << arrZ[i] << " ";*

*}*

*ofstream file("array2.txt");*

*if (file.is\_open())*

*{*

*file << "matricea cu numere negative:\n";*

*for (int i = 0; i < nn; i++){*

*if (arrN2[i] < 0)*

*file << arrN2[i] << " ";*

*}*

*file << "\n\nmatricea cu numere pozitive:\n";*

*for (int i = 0; i < np; i++){*

*if (arrP2[i] > 0)*

*file << arrP2[i] << " ";*

*}*

*file << "\n\nmatricea cu numere egale cu 0:\n";*

*for (int i = 0; i < nz; i++){*

*if (arrZ2[i] == 0)*

*file << arrZ2[i] << " ";*

*}*

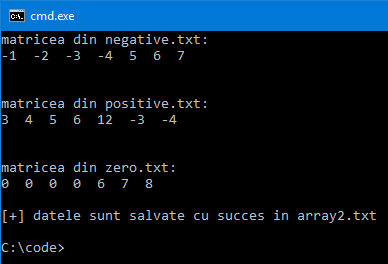
*cout << "\n\n[+] datele sunt salvate cu succes in array2.txt\n";*

*file.close();*

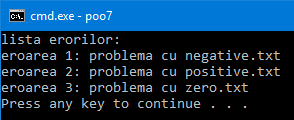
*} else cout << "erare cu fileul array2.txt";*

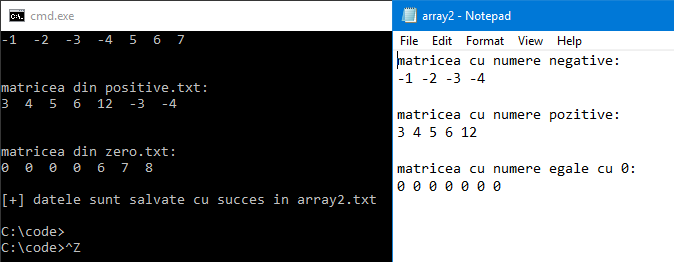
*return 0;*

*}*



***ScreenShot-uri:***





***Concluzie:***

*Dupa efectuarea laborotorului am dobintid experienta in limbajul de programere C++ lucrind cu exceptii.*