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.

Întrebările pentru verificarea cunoştinţelor:

1. Cum sînt realizate excepţiile în cadrul programelor C++ ? Daţi exemple.

Temele pentru acasă:

1. Ce rezultate vor fi obţinute la rularea următorului program:

```
void main()
{ cout<<"Început";
try{    // începutul blocului try
cout<<"interiorul blocului try \n";
throw 10; // generarea erorii
cout<<"Aceasta nu se va executa";
}
catch (int i)  {
cout<<"Este prelucrată eroarea: ";
cout<<I<<"\n";}
cout << "Sfîrşit";   }
```

Teme pentru lucrări de laborator:

1. Scrieţi un program care compară două fişiere date.

2. Scrieţi un program care determină numărul maximal şi cel minimal dintr-un şir de numere aleatoare dintr-un fişier. Subconsecutivitatea de elemente dintre numărul maximal şi cel minimal determinat să se înregistreze într-un nou fişier.

3. Scrieţi un program care tipăreşte toate cuvintele diferite de ultimul cuvînt dintr-un fişier. Cuvintele din fişier sînt separate prin virgulă, iar.după ultimul cuvînt se pune punct.

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

5. Scrieţi un program care ordonează lexicografic o consecutivitate de înregistrări (dintr-un fişier) de tipul

```
struct { char nume [30];
int ani}  înregistrare;
```

6. Scrieţi un program care formează un fişier nou din trei fişiere date după următoarea legitate: se selectează mai întîi numerele divizibile la 3, la 5 şi la 7, apoi numerele pozitive pare de pe locuri impare.

7. Scrieţi un program care sortează lexicografic cuvintele dintr-un text. Pentru fiecare element sortat să se indice numărul de repetări ale cuvîntului în textul dat.

8. Scrieţi un program care din două fişiere ordonate descrescător se vor uni în unul nou în care se va păstra ordinea descrescătoare de sortare.

9. Scrieţi un program care va tipări în ordine inversă subconsecutivitatea de numere dintre valoarea minimă şi maximă ale unei consecutivităţi de numere citită dintr-un fişier.

10. Scrieţi un program care calculează suma înmulţirii numerelor vecine pe locuri pare dintr-un fişier dat.

11. Scrieţi un program care determină frecvenţa cu care a fost generat fiecare element în fişierului creat. Valorile elementelor sînt cuprinse între 1 şi 100.

12. Scrieţi un program care determină numărul maximal şi cel minimal din numerele unui fişier dat. Să se determine elementele mai mari ca cel minimal şi mai mici ca numărul maximal.

13. Scrieţi un program care formează un fişier nou din cel dat după următoarea legitate: elementele fişierului nou se obţine din inversul numerelor din fişierul dat.

14. Scrieţi un program care să formeze un fişier nou care conţine elementele a două fişiere ordonate crescător.Elementele fişierului format va păstra ordinea crescătoare de sortare.

16. Scrieţi un program care efectuează reformatarea unui fişier textual în felul următor. Lungimea rîndului de caractere în fişierul nou are lungimea de 60 de caractere. Dacă în rîndul dat se depistează punctul, restul rîndului din fişierul dat se scrie din rînd nou.

Bibliografia

1. D. Somnea, D. Turturea. Introducere în C++. Programarea obiect orientata. – Bucureşti, ed. Teora, 1993.