***ЛАБОРАТОРНА РОБОТА №9***

**Тема:** Вивчення механізмів оброблення виняткових ситуацій  
**Мета:** Навчитись обробляти ситуації появлення виняткових ситуацій, вивчити механізми їх оброблення

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.** Додайте клас винятків до програми ARROVER таким чином, щоб індекси, що виходять за межі масиву, викликали генерацію винятку. Блокпастка catch може виводити користувачеві повідомлення про помилку.

#include <iostream>

using namespace std;

#include <process.h> //for exit()

const int LIMIT = 10; //array size

class safearay

{

private:

int arr[LIMIT];

public:

int& operator [](int n)

{

if( n< 0 || n>=LIMIT )

{ cout << "\nIndex out of bounds"; exit(1); }

return arr[n];

}

};

int main()

{

safearay sa1;

for(int j=0; j<=LIMIT; j++)

sa1[j] = j\*10; //\*left\* side of equal sign

for(int j=0; j<=LIMIT; j++) //display elements

{

int temp = sa1[j]; //\*right\* side of equal sign

cout << "Element " << j << " is " << temp << endl;

}

return 0;

}

Код програми:

#include <QCoreApplication>

#include <iostream>

using namespace std;

#include <process.h> //for exit()

const int LIMIT = 10; //array size

class IndexException

{

string msg;

int index;

public:

IndexException(string str, int errorIndx){msg = str; index = errorIndx;}

void Show(){ cout << msg + ", wrong index: " + to\_string(index) + "\n"; }

};

class safearay

{

private:

int arr[LIMIT];

public:

int& operator [](int n)

{

if( n< 0 || n>=LIMIT ) { throw IndexException("\nIndex out of bounds", n);}

return arr[n];

}

};

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

{

QCoreApplication a(argc, argv);

safearay sa1;

for(int j=0; j<=LIMIT; j++)

try{

sa1[j] = j\*10; //\*left\* side of equal sign

}catch(IndexException e){e.Show();}

for(int j=0; j<=LIMIT; j++) //display elements

{

try{

int temp = sa1[j]; //\*right\* side of equal sign

cout << "Element " << j << " is " << temp << endl;

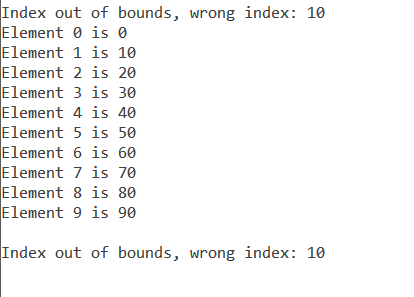
}catch(IndexException e){e.Show();}

}

return a.exec();

}

Вивід програми:



**Завдання 2.** Модифікуйте текст програми, розробленої згідно індивідуального завдання лабораторної роботи №3, добавивши до нього (в місця можливого виникнення помилок) процедури оброблення виняткових ситуацій, які будуть генерувати об’єкти класу, який міститиме такі атрибути: місце виникнення помилки; значення, яке призвело до помилки; параметризований конструктор; перевизначену операції виводу, яка виводитиме значення на екран (або зберігатиме виняткову ситуацію у файл).

Код програми:

#include <QCoreApplication>

#include <string.h>

#include <typeinfo.h>

#include <iostream>

using namespace std;

class Exception

{

string typeName;

string errorMsg;

public:

template<class T>

Exception(T type, string codeError)

{

typeName = typeid (type).name();

errorMsg = codeError;

}

friend ostream& operator<<(ostream&, Exception&);

};

ostream& operator<<(ostream& outStr, Exception& exc)

{

outStr << "Exception in " << exc.typeName << ", error code: " << exc.errorMsg;

}

class Rectangle

{

double x, y;

public:

Rectangle():Rectangle(0,0){}

Rectangle(double \_x, double \_y)

{

if(\_x <= 0 || \_y <= 0) throw Exception(this, "Not correct size, must be grater than 0, now: x=" + to\_string(\_x) + "; y=" + to\_string(\_y));

SetCornerDigonal(\_x,\_y);

}

~Rectangle(){}

double GetX() {return x;}

double GetY() {return y;}

void SetCornerDigonal(double \_x, double \_y){ x = \_x; y = \_y;}

string GetRectangle(){ return ("Rectangle diagonal is (" + to\_string(x) + "; " + to\_string(y) + ")"); }

double GetPerymetr(){ return 2\*x+2\*y; }

Rectangle& operator+(Rectangle& r)

{

Rectangle newR(x+r.x, y+r.y);

return newR;

}

Rectangle& operator-(Rectangle& r)

{

Rectangle newR(x-r.x, y-r.y);

return newR;

}

bool operator>(Rectangle& r)

{

return GetPerymetr() > r.GetPerymetr();

}

bool operator<(Rectangle& r)

{

return GetPerymetr() < r.GetPerymetr();

}

bool operator==(Rectangle& r)

{

return GetPerymetr() == r.GetPerymetr();

}

friend istream& operator<<(istream& stream, Rectangle& r);

friend ostream& operator>>(ostream& stream, Rectangle& r);

};

istream& operator>>(istream& stream, Rectangle& r)

{

double x, y;

stream >> x >> y;

r.SetCornerDigonal(x,y);

return stream;

}

ostream& operator<<(ostream& stream, Rectangle& r)

{

stream << r.GetRectangle();

return stream;

}

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

{

QCoreApplication a(argc, argv);

try {

Rectangle r1(2,3);

Rectangle r2;

} catch (Exception e) {

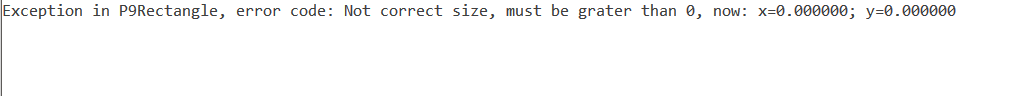
cout << e;

}

return a.exec();

}

Вивід програми:



**Висновок:** Навчився обробляти ситуації появлення виняткових ситуацій, вивчив механізми їх оброблення.