Лабораторна робота №9

Тема: Вивчення механізмів оброблення виняткових ситуацій.

Мета: навчитись обробляти ситуації появлення виняткових ситуацій, вивчити механізми їх оброблення.

Завдання 1

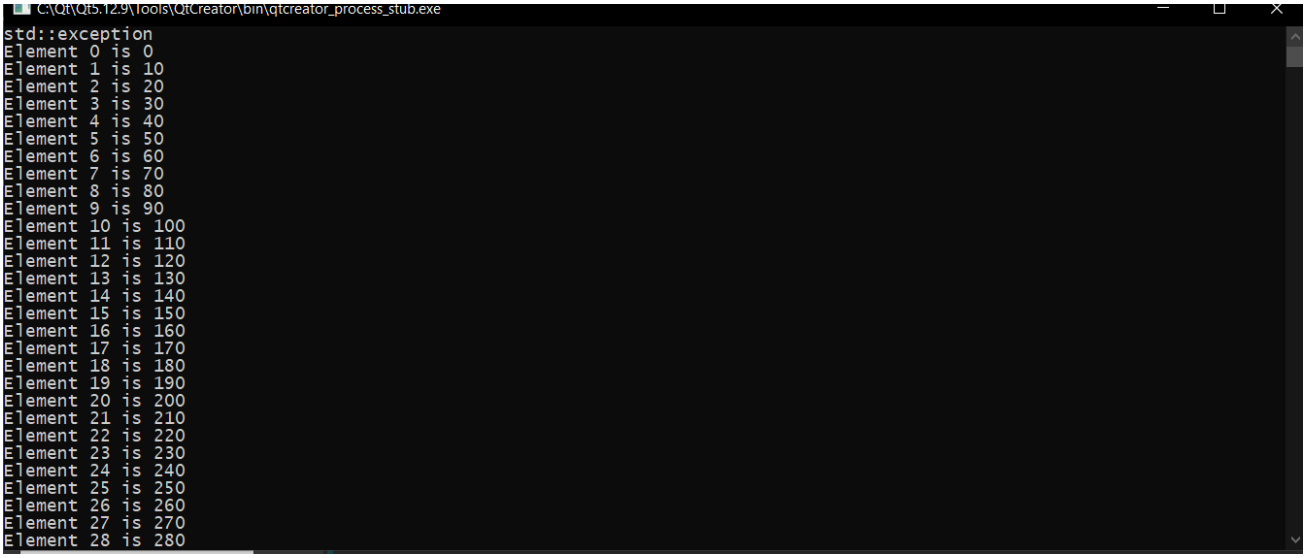
Додайте клас винятків до програми ARROVER таким чином, щоб індекси, що виходять за межі масиву, викликали генерацію винятку. Блокпастка catch може виводити користувачеві повідомлення про помилку

Код програми

```
//////program ARROVER//////
#include <iostream>
#include  <string>
using namespace std;
#include <process.h> //for exit()
const int LIMIT = 100; //array size
////////////////////////////////////////////////////////////
class safearay
{
private:
    int arr[LIMIT];
public:
    int& operator [](int n) throw (exception)
    {
        if( n< 0 || n>=LIMIT )
        {
            throw exception();
        }
        return arr[n];
    }
};
////////////////////////////////////////////////////////////
int main()
{
    int j;
    safearay sa1;

    try
    {
        for(int j=0; j<200; j++)
        {
            sa1[j] = j*10;
        }
    }
    catch (exception &ex)
    {
        cout<<ex.what()<<endl;
    }
```

```
    try
    {
        for(j=0; j<LIMIT; j++) //display elements
        {
            int temp = sa1[j]; //*right* side of equal sign
            cout << "Element " << j << " is " << temp << endl;
        }
    }
    catch (exception &ex)
    {
        cout<<ex.what()<<endl;
    }

    return 0;
}
```

Результат



Завдання 2

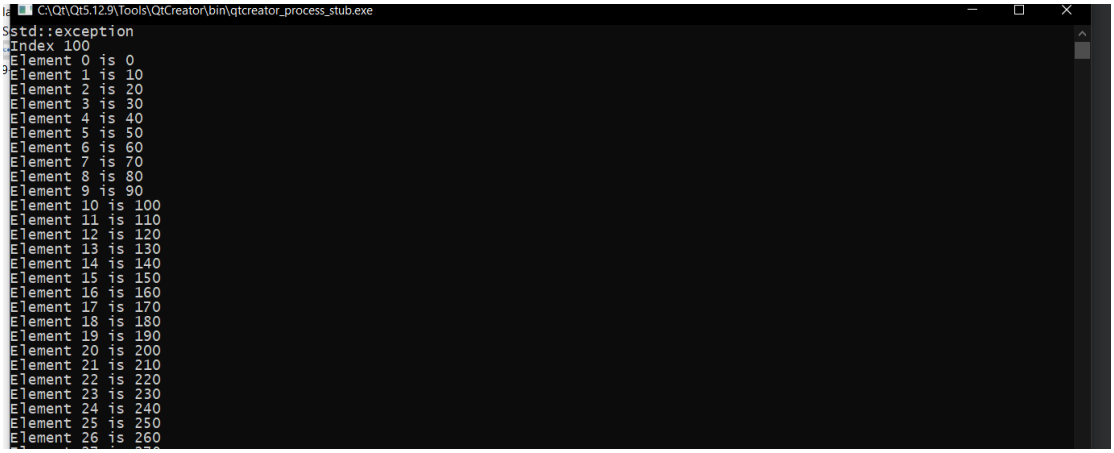
Модифікуйте програму з попереднього завдання таким чином, щоб в повідомлення про помилку входила інформація про значення індексу, який призвів до збою.

Код програми

```
//////program ARROVER//////
#include <iostream>
#include  <string>
using namespace std;
#include <process.h> //for exit()
const int LIMIT = 100; //array size
////////////////////////////////////////////////////////////
class safearay
{
private:
    int arr[LIMIT];
public:
     string str;
```

```
    int& operator [](int n)
    {
        if( n< 0 || n>=LIMIT )
        {
          str= to_string(n);
            throw   exception();
        }
        return arr[n];
    }
};
//////////////////////////////////////////////////////////////////

class myexception: public exception
{
  virtual const char* what() const throw()
  {
    return "My exception happened";
  }
} myex;
int main()
{
    int j,k;
    safearay sa1;

    try
    {
        for(int j=0; j<200; j++)
        {
            k=j;
            sa1[j] = j*10;
        }

    }
    catch (exception &ex)
    {
        cout<<ex.what()<<endl;
        cout<<"Index "<<k<<endl;
    }

    try
    {
        for(j=0; j<LIMIT; j++) //display elements
        {;
            int temp = sa1[j]; //*right* side of equal sign
            cout << "Element " << j << " is " << temp << endl;
        }
    }
    catch (exception &ex)
    {
        cout<<ex.what()<<endl;
        cout<<"Index "<<k<<endl;
    }

    return 0;
}
```

Результат



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).

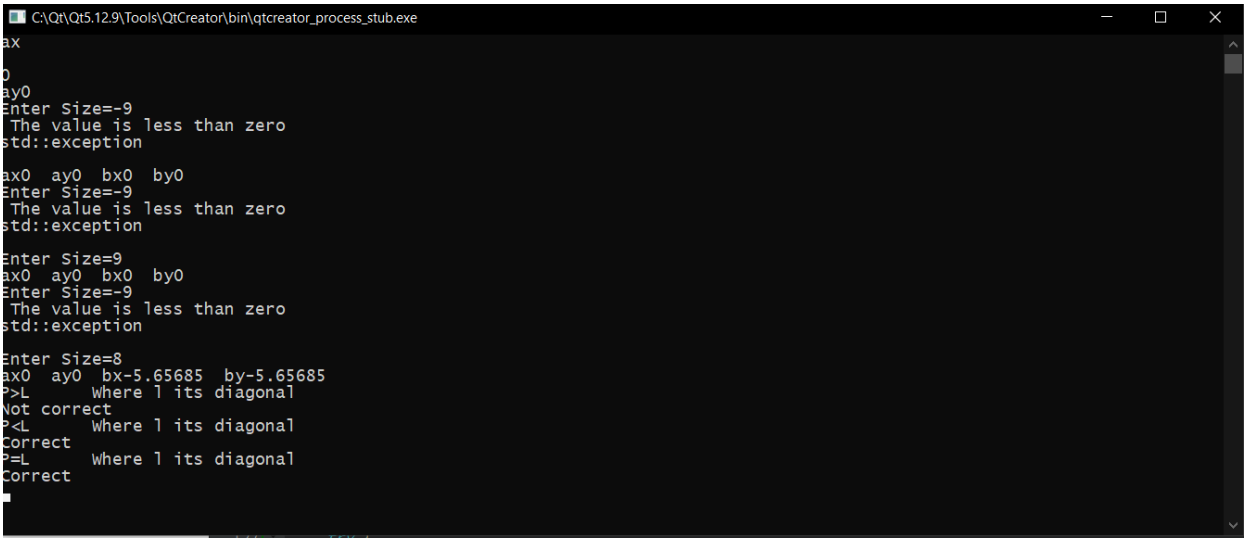
Код програми

```
#include <iostream>
#include <math.h>

using namespace std;


class kub
{
private:
    double ax, ay, bx, by, Size,newSize;
public:
    kub()
    {
        this->ax=0;
        this->ay=0;
        this->bx=0;
        this->by=0;
        this->Size=0;
    };

    void Start()
    {
        cout<<"ax";
        cin>>this->ax;
        cout<<"ay";
        cin>>this->ay;
        cout<<"Enter Size=";
        cin>>this->Size;
        if(Size<0)
        throw exception();
        this->bx=this->by=Size/(sqrt(2));
```

```
    }


    void getpoint()
    {
        cout<<"ax"<<ax<<"  ay"<<ay<<"  bx"<<bx<<"  by"<<by<<endl;
    }

    void NewSize()
    {
        cout<<"Enter Size=";
        cin>>newSize;
        if(newSize<0)
            throw exception();


    }

    kub operator =(const kub& obj)
    {
        bx=obj.bx;
        by=obj.by;
        Size=obj.Size;
        return *this;
    }
    kub operator +(const kub& obj) const
    {
        kub temp;
        temp.bx=(obj.ax+(obj.Size+obj.newSize)/(sqrt(2)));
        temp.by=(obj.ay+(obj.Size+obj.newSize)/(sqrt(2)));
        temp.Size=Size+newSize;
        return temp;
    }
    kub operator -(const kub& obj) const
    {
        kub temp;
        temp.bx=(obj.ax+(obj.Size-obj.newSize)/sqrt(2));
        temp.by=(obj.ay+(obj.Size-obj.newSize)/sqrt(2));
        temp.Size=Size+newSize;
        return temp;
    }

    kub operator >(const kub& obj) const
    {
        double P=4*((this->Size-this->newSize)/sqrt(2));
        if(P>obj.Size)
            cout<<"Correct"<<endl;
        else cout<<"Not correct"<<endl;
    }
    kub operator <(const kub& obj) const
    {
        double P=4*((this->Size-this->newSize)/sqrt(2));
        if(P<obj.Size)
            cout<<"Correct"<<endl;
        else cout<<"Not correct"<<endl;
    }
    kub operator ==(const kub& obj) const
    {
        double P=4*((this->Size-this->newSize)/sqrt(2));
        if( (P=obj.Size))
            cout<<"Correct"<<endl;
        else cout<<"Not correct"<<endl;
    }

};
```

```
int main()
{

    kub k;
point1:
    try {
    k.Start();
    }
    catch (exception &ex)
    {
        cout<<" The value is less than zero"<<endl;
        cout<<ex.what()<<endl;
        goto point1;

    }
    k.getpoint();
point2:
    try {

        k.NewSize();
    }
    catch (exception &ex )
    {
        cout<<" The value is less than zero"<<endl;
        cout<<ex.what()<<endl;
        goto point2;
    }
    k=k+k;
    k.getpoint();
    point3:
    try {

        k.NewSize();
    }
    catch (exception &ex )
    {
        cout<<" The value is less than zero"<<endl;
        cout<<ex.what()<<endl;
        goto  point3;
    }
    k=k-k;
    k.getpoint();
    cout<<"P>L\t Where 1 its diagonal"<<endl;
    k>k;
    cout<<"P<L\t Where 1 its diagonal"<<endl;
    k<k;
    cout<<"P=L\t Where 1 its diagonal"<<endl;
    k==k;



    return 0;
}
```

Результат



Висновок:

Навчився обробляти ситуації появлення виняткових ситуацій, вивчити механізми їх оброблення.