**Лабораторна робота №9**

**Тема:** Вивчення механізмів оброблення виняткових ситуацій.

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

1. Ознайомитись зі способами обробки виняткових ситуацій.

2. Реалізуйте програму згідно завдання 1.

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.

//////program ARROVER//////

#include <iostream>

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)

{

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(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:

#include <iostream>

using namespace std;

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

const int LIMIT = 100;

class **ARROVER**{

public:

void **exeption**(int n){

try {

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

{

throw 404;

}

} catch (int i) {

cout<<"Error-"<<i<<endl;

}

}

};

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;

ARROVER a;

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

sa1[j] = j\*10;

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

{

int temp = sa1[j];

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

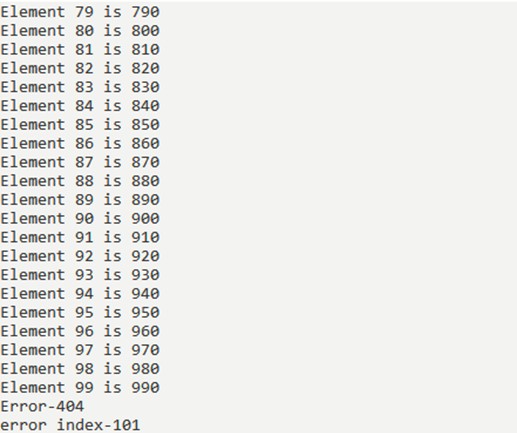
}

a.exeption(101);

a.exeption(66);

return 0;

}



Завдання 2. Модифікуйте програму з попереднього завдання таким чином,щоб в повідомлення про помилку входила інформація про значення індексу, який призвів до збою.

Код:

#include <iostream>

using namespace std;

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

const int LIMIT = 100;

class **ARROVER**{

public:

void **exeption**(int n){

try {

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

{

throw 404;

}

} catch (int i) {

cout<<"Error-"<<i<<endl<<"error index-"<<n<<endl;

}

}

};

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;

ARROVER a;

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

sa1[j] = j\*10;

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

{

int temp = sa1[j];

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

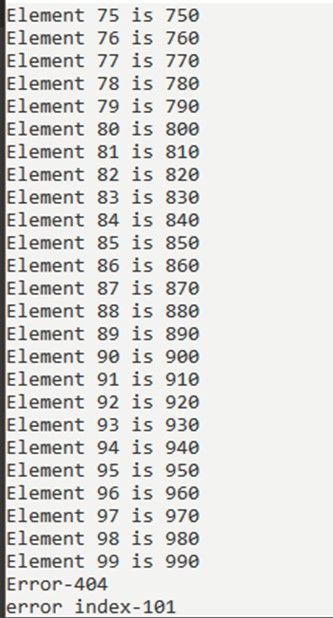
}

a.exeption(101);

a.exeption(66);

return 0;

}



4. 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>

using namespace std;

struct **Elem**

{

int data;

Elem \* next, \* prev;

};

class **List**

{

Elem \* Head, \* Tail;

int Count;

public:

**List**();

~**List**();

int **GetCount**();

Elem\* **GetElem**(int);

void operator +(int n);

void operator +=(int n);

void operator - (int pos);

void **Print**();

void operator<<(int pos);

};

List::**List**()

{

Head = Tail = nullptr; //список пустий

Count = 0;

}

List::~**List**()

{

}

void List::**Print**()

{

if(Count != 0)

{

Elem \* temp = Head;

cout << "( ";

while(temp->next != nullptr)

{

cout << temp->data << ", ";

temp = temp->next;

}

cout << temp->data << " )\n";

}

}

int List::**GetCount**()

{

return Count;

}

void List::operator<<(int pos){

if(pos < 1 || pos > Count)

{

cout << "Incorrect position !!!\n";

return;

}

Elem \* temp;

if(pos <= Count / 2)

{

temp = Head;

int i = 1;

while(i < pos)

{

temp = temp->next;

i++;

}

}

else

{

temp = Tail;

int i = 1;

while(i <= Count - pos)

{

temp = temp->prev;

i++;

}

}

cout << pos << " element: ";

cout << temp->data << endl;

}

void List::operator+(int n)

{

cout << "ok" << endl;

try {

if(n<0)

{

throw 333;

}

} catch (int i) {

cout<<"error-"<<i<<endl<<"Restart program"<<endl;

system("pause");

}

Elem \* temp = new Elem;

temp->prev = nullptr;

temp->data = n;

temp->next = Head;

if(Head != nullptr)

Head->prev = temp;

if(Count == 0)

Head = Tail = temp;

else

Head = temp;

Count++;

}

void List::operator+=(int n){

cout << "ok" << endl;

try {

if(n<0)

{

throw 333;

}

} catch (int i) {

cout<<"error-"<<i<<endl<<"Restart program"<<endl;

system("pause");

}

Elem \* temp = new Elem;

temp->next = nullptr;

temp->data = n;

temp->prev = Tail;

if(Tail != nullptr)

Tail->next = temp;

if(Count == 0)

Head = Tail = temp;

else

Tail = temp;

Count++;

}

void List::operator - (int pos)

{

if(pos == 0)

{

cout << "Input position: ";

cin >> pos;

}

if(pos < 1 || pos > Count)

{

cout << "Incorrect position !!!\n";

return;

}

int i = 1;

Elem \* Del = Head;

while(i < pos)

{

Del = Del->next;

i++;

}

Elem \* PrevDel = Del->prev;

Elem \* AfterDel = Del->next;

if(PrevDel != nullptr && Count != 1)

PrevDel->next = AfterDel;

if(AfterDel != nullptr && Count != 1)

AfterDel->prev = PrevDel;

if(pos == 1)

Head = AfterDel;

if(pos == Count)

Tail = PrevDel;

delete Del;

Count--;

}

int **main**()

{

List L;

L+=-6;

int a[10] = {-1,1,-2,3,4,5,6,7,8,9};

for(int i = 0; i < 10; i++)

if(i % 2 == 0)

L +a[i];

else

L+=a[i];

L-3;

cout << "List L:"<<endl;

L.Print();

cout << endl;

cout << "List L:"<<endl;

L.Print();

L<<2;

L<<8;

return 0;

}



Висновок: на цій лабораторній роботі я навчився обробляти ситуації появлення виняткових ситуацій,вивчив механізми їх оброблення.