 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №7**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Виняткові ситуації»

Варіант №15

**Виконав:**

студент гр. БС-81

Сєров О. В.

**Перевірив:**

ас. каф. БМК

Рисін С.В

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

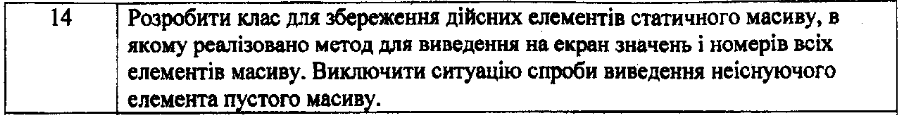
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Завдання:**

1. Ознайомитися з теоретичними відомостями обробки виняткових ситуацій в мові C++.
2. Створити клас Exception, що дозволяє обробляти зазначену в завданні виняткову ситуацію.
3. Розробити діаграму класів в нотації UML та програмний застосунок для демонстрації роботи програми в нормальному режимі та під час виникнення виняткової ситуації відповідно до свого варіанту (визначення класів та їх реалізації мають бути розташовані у файлах \*.h та \*.срр відповідно; для ініціалізації даних класів використовувати введення з клавіатури; обробка виняткової ситуації має включати виведення на екран відповідного повідомлення з текстом помилки):



1. Скласти і захистити звіт по роботі.

**🞏 Виконано без зауважень**

**🞏 Маються зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 потужність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 відсутні списки ініціалізації в конструкторах**

**🞏 константні методи**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Діаграма класів:**



**Код:**

**Matrix.h**

#pragma once

#include "Exception.h"

#include <iostream>

using namespace std;

class Matrix

{

protected:

unsigned m, n;

int\*\* data;

public:

Matrix(unsigned rows, unsigned columns);

Matrix(const Matrix& ob);

Matrix();

~Matrix();

friend Matrix operator+(Matrix& a, Matrix& b);

friend istream& operator >> (istream&, Matrix&);

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

};

**Matrix.cpp**

#include "Matrix.h"

Matrix::Matrix()

{

n = 0;

m = 0;

data = NULL;

}

Matrix::Matrix(unsigned rows, unsigned columns)

{

m = rows;

n = columns;

data = new int\* [m];

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

data[i] = new int[n];

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

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

data[i][j] = 0;

}

Matrix::Matrix(const Matrix& ob)

{

m = ob.m;

n = ob.n;

if (ob.data == NULL)

{

data = NULL;

return;

}

data = new int\* [m];

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

data[i] = new int[n];

for (unsigned i = 0; i < m; i++)

for (unsigned j = 0; j < n; j++)

data[i][j] = ob.data[i][j];

}

Matrix::~Matrix()

{

if (data != NULL)

{

for (unsigned i = 0; i < m; i++)

delete[]data[i];

delete[]data;

}

}

Matrix operator + (Matrix& ob\_a, Matrix& ob\_b)

{

if (ob\_a.m == ob\_b.m && ob\_a.n == ob\_b.n)

{

Matrix temp(ob\_a.m, ob\_a.n);

for (int i = 0; i < ob\_a.m; i++)

for (int j = 0; j < ob\_a.n; j++)

temp.data[i][j] = ob\_a.data[i][j] + ob\_b.data[i][j];

return temp;

}

else

throw Exception("Dimensions of summable matrices must match!\n");

}

istream& operator >> (istream& in, Matrix& ob)

{

if (ob.data != NULL)

{

for (int i = 0; i < ob.m; i++)

for (int j = 0; j < ob.n; j++)

in >> ob.data[i][j];

}

return in;

}

ostream& operator << (ostream& out, Matrix ob)

{

if (ob.data != NULL)

{

out << "Matrix:" << endl;

for (int i = 0; i < ob.m; i++)

{

for (int j = 0; j < ob.n; j++)

out << ob.data[i][j] << "\t";

out << endl;

}

}

else

throw Exception("Matrix is empty!\n");

return out;

}

**Exception.h**

#pragma once

#include <exception>

#include <string>

class Exception : public std::exception

{

private:

std::string \_error;

public:

Exception(const std::string& temp\_error);

virtual const char\* Excp\_massage() const;

};

**Exception.cpp**

#include "Exception.h"

Exception::Exception(const std::string& temp\_error) : \_error(temp\_error)

{

}

const char\* Exception::Excp\_massage() const

{

return \_error.c\_str();

}

**Main.cpp**

#include "Matrix.h"

void main()

{

try

{

cout << "Enter dimensions of first matrix:\n\nRows: ";

unsigned m = 0;

cin >> m;

cout << "Columns: ";

unsigned n = 0;

cin >> n;

cout << "Enter dimensions of second matrix:\n\nRows: ";

unsigned k = 0;

cin >> k;

cout << "Columns: ";

unsigned l = 0;

cin >> l;

Matrix a(m,n), b(k,l);

cout << "\nEnter data of first matrix: " << m \* n << " int's: \n";

cin >> a;

cout << "\nEnter data of second matrix: " << k \* l << " int's: \n";

cin >> b;

Matrix d = a + b;

cout << "a + b = " << d << endl;

}

catch(Exception& ex)

{

cout << "Error: " << ex.Excp\_massage() << endl;

}

catch(...)

{

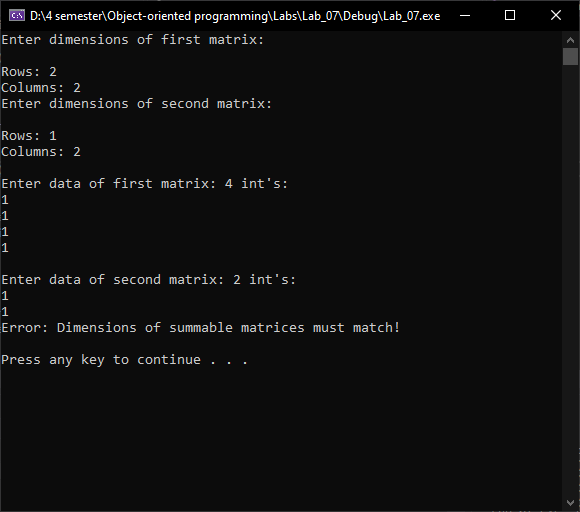
cout << "Error: something goes wrong!" << endl;

}

system("pause");

}

**Результати:**

****

**Контрольні питання:**

1. **Що називається винятковою ситуацією?**Виняткова ситуація – це аномальна поведінка при виконанні, яке програма може виявити. Наприклад, ділення на 0, вихід за границі масиву або закінчення памяті. Такі винятку порушують нормальний хід програми і на них треба негайно відреагувати.
2. **Що називається обробкою виняткової ситуації?**

Перевірка умов, що описують виняткову ситуацію, і реакція на її виникнення називається обробкою виняткової ситуації.

1. **Як здійснюється обробка виняткової ситуації в С++?**Обробка виняткових ситуацій в С++ використовує 3 основних ключових слова: try, catch і throw. Ті інструкції програми, де очікується можливість появи виняткових ситуацій розміщується в блоці try. Якщо в блоці try виникає виняток, тобто помилка, то генерується виняток. Виняток перехвачується, використовуючи catch і оброблюється.
2. **Чи можна помістити блок try у функцію, залишивши блок catch для обробки виняткових ситуацій у функції main()? Відповідь пояснити.**Блок catch повинен розміщуватися відразу після блоку try, щоб компілятор міг знати, що йому робити при виникненні випадкової ситуації. Бо у блоці try генерується виняток, а за допомогою блока catch відбувається перехват цього винятку. Виняток не може бути повернений з функції.
3. **До чого призведе передавання винятку в блок catch за посиланням?**При передаванні винятку за посиланням, catch буде напряму посилатися на об’єкт-виняток, а не створювати його локальну копію. Це запобігає непотрібному копіюванню великих об’єктів-винятків.
4. **Для чого використовують вираз throw, що не містить операндів?**Оператор throw може використовуватися без аргумента в тому випадку, якщо він розміщений в блоці catch. В цьому випадку він повторно «викидує» помилку, перехвачену оператором catch, в якому він розміщений. Якщо оператор throw який знаходиться за межами блоку catch призводить до виклику функції terminate().
5. **До чого призведе генерація похідної виняткової ситуації за умови наявності блоку catch, що призначений для обробки базової виняткової ситуації? Відповідь пояснити.**Похідна виняткова ситуація буде перехвачена за допомогою блока catch, який призначений для базової виняткової ситуації. Якщо присутній catch і для базового класу і для похідного, то необхідно розташовувати 1й похідний, а другий базовий, щоб кожен з них «спіймав» свою помилку.
6. **Назвіть і опишіть стандартні типи винятків з бібліотеки std.**В бібліотеці С++ є два заголовки, які мають відношення до обробки виняткових ситуацій або винятків: <exception> і <stdexcept>. Заголовок <exception> визначає класи, типи і функції, які зв’язані з обробкою винятків. Клас exception – це базовий клас для всіх винятків, визначених в стандартній бібліотеці С++. Клас bad\_exception представляє собою тип винятку, який генерується функцією unexpected(). З класу exception виведено ряд інших важливих класів. Клас bad\_alloc використовується у випадку невдалого виконання оператору new. Клас bad\_typeid корисний при недопустимому виразі typeid. Клас bad\_cast використовується при неправильній операції динамічного приведення типу.
7. **Для чого під час обробки виняткових ситуацій використовують призначені функції terminate(), unexpected(), abort()?**

Для того щоб реагувати на виняткові ситуації, обробка яких програмістом не передбачена, в мові С++ використовують ці функції. Механізм роботи виглядає наступним чином. Виняткову ситуацію, яку не перехватив жоден з операторів catch, передають вгору по ланцюжку викликів.

– unexpected() викликається, якщо функція вказує винятки, які може викидати, але викидує не вказане воно викликає функцію, яка зареєстрована за допомоги set\_unexpected(), або якщо вона є не є зареєстрованою, вона викликає terminate();  
– terminate() викликається з unexpected(), або якщо викидається виключення, яке не ловиться у коді програми, вона викликає функцію, яка зареєстрована за допомоги set\_terminate(), або вона є не є зареєстрованою, вона викликає abort();  
– abort() призводить до аварійного завершення програми, викликає \_exit();  
– \_exit() завершує виконання програми без закриття файлів, очищення буферів та виклика функцій

1. **Навести приклад коду, за допомогою якого можна вивести на екран текстове повідомлення з номером стрічки, під час виконання якої сталася виняткова ситуація, та назвою файлу, що містить цю стрічку.**  
   struct temp

{

char \*str;

int n;

temp(char \*s, int i): str(s), n(i)

{ }

};

try

{

throw temp(\_\_FILE\_\_, \_\_LINE\_\_);

}

catch(temp &t)

{

cout << t.str << " " << t.n << endl;

}