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 №5

**Виконав:**

студент гр. БС-71

Воробйов І.О.

**Перевірив:**

асистент каф. БМК

Рисін С.В.

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

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

(підпис викладача)

Київ-2019

**🞏 Практичне заняття без зауважень**

**🞏 Практичне заняття має 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:**

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

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

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

**Завдання:**

1. Ознайомитися з теоретичними відомостями обробки виняткових ситуацій в мові С++.
2. Створити клас Exception, що дозволяє оброблювати 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 \*.cpp відповідно; для ініціалізації даних класів використовувати введення з клавіатури; обробка виняткової ситуації має включати виведення на екран відповідного повідомлення з текстом помилки):

|  |  |
| --- | --- |
| Номер варіанту | Завдання |
| 5 | Розробити програму обчислення та виведення на екран через статичний метод класу корені квадратного рівняння **.**  Виключити ситуацію отримання від’ємного дискримінанта і введення нульового коефіцієнта **a**. |

1. Скласти і захистити звіт по роботі.

**UML діаграма**



**Код програми:**

**equation.h:**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class equation

{

private:

double a;

double b;

double c;

public:

equation();

equation(double, double, double);

~equation();

double get\_a() const;

double get\_d() const;

double res(bool) const;

friend istream& operator >> (istream& cin, equation& obj);

};

class Exeption

{

private:

string error;

public:

Exeption(string);

string get\_error() const;

};

**equation.cpp:**

#include "equation.h"

equation::equation(){}

equation::equation(double x1, double x2, double x3) :a(x1), b(x2), c(x3) {}

equation::~equation() {}

double equation::get\_a() const

{

return a;

}

double equation::get\_d() const

{

return pow(b, 2) - 4 \* a \* c;

}

double equation::res(bool bl) const

{

if (!bl)

return (-b + sqrt(this->get\_d())) / 2 / a;

else

return (-b - sqrt(this->get\_d())) / 2 / a;

}

istream& operator >> (istream& cin, equation& obj)

{

double a, b, c;

cout << "Input a = ";

cin >> a;

cout << "Input b = ";

cin >> b;

cout << "Input c = ";

cin >> c;

obj = equation(a, b, c);

return cin;

}

Exeption::Exeption(string err) :error(err) {}

string Exeption::get\_error() const

{

return error;

}

**Source.cpp:**

#include "equation.h"

void main()

{

bool bl = true;

equation my;

while (bl)

{

try

{

cin >> my;

if (my.get\_a() == 0)

throw Exeption("Error: a = 0");

else if (my.get\_d() < 0)

throw Exeption("Error: D<0");

else

bl = false;

}

catch (Exeption error)

{

cout << error.get\_error() << endl;

system("pause");

system("cls");

}

}

cout << "x1 = " << my.res(false) << (my.res(false) == my.res(true) ? " = x2 = " : "\nx2 = ") << my.res(true) << endl;

system("pause");

}

**Результат роботи програми:**









**Контрольні запитання**

1. Що називають винятковою ситуацією?

Виняткова ситуація – це подія, що сталася під час виконання прорами, в результаті чого подальше нормальне виконання програми стає неможливим.

1. Що називають обробкою виняткової ситуації?

Після виникнення виняткової ситуації потрібно привести програму в робочий стан або виконати її коректне завершення з визволенням усіх ресурсів, які використовувалися програмою.

1. Як здійснюється обробка виняткової ситуації в мові С++?

Якщо блок try виявляє виключення усередині цього блоку коду, відбувається програмне переривання та виконується зазначена нижче послідовність дій:

програма шукає відповідний оброблювач виключення;

якщо оброблювач знайдено, стек очищується і керування передається оброблювачеві виключення;

якщо оброблювач не знайдений, викликається функція terminate() для завершення програми.

Виняткова ситуація обробляється всередині блоку catch, що повинен стояти безпосередньо за блоком try.

try

{

//код, що може генерувати виняткові ситуації

throw[expression];

}

[catch(exception-declaration)

{

//код, який у випадку відповідності exception-declaration до

//типу згенерованого в блоці try винятку

}

[catch(exception-declaration)

{

//код, що обробляє інший тип винятку

}]..]

1. Чи можна помістити блок try у функцію, залишивши блок catch для обробки виняткової ситуації у функції main()? Відповідь пояснити.

Блоки try і catch нерозривні. Не можна помістити блок try у функцію, залишивши блок catch у функції main(). Необхідно або обробити виняткову ситуацію усередині функції, як це зроблено в більш ранньому прикладі, або перенести обробку в модуль виклику. У першому випадку функція, завершивши обробку, повертає визначене її специфікацією значення, а в другому — виняткову ситуацію

1. До чого призведе передавання винятку в блок catch за посиланням?

catch за значенням викликає дві копії об'єкта, catch за посиланням буде викликати один

1. Для чого використовують вираз though, що не містить операндів?

Вираз throw без операндів повторно створює виключення, що обробляється в даний момент

1. До чого призведе генерація похідної виняткової ситуації за умови наявності блоку catch, що призначений для обробки базової виняткової ситуації? Відповідь пояснити.

Оскільки виняткова ситуація може бути об'єктом класу, у мові С++ існує можливість створювати ієрархію виняткових ситуацій. У цьому випадку блок catch перехоплює об'єкти не тільки базового, але і похідних класів. При генерації похідних виняткових ситуацій це приводить до непорозумінь — їх перехоплює блок catch, призначений для обробки базових виняткових ситуацій. catch для похідного виключеня стоїть до базового.

Друга ситуація: тотальне перехоплення виняткових ситуацій

Оскільки блок catch(…) не має аргументів, у ньому важко передбачити точну реакцію на виняткову ситуацію, яка виникла. Єдине, що залишається завершити роботу програми і вивести на екран відповідне повідомлення. Похідне виключення стоїть після базового.

1. Назвіть і опишіть стандартні винятки з бібліотеки std.

bad\_alloc - помилка розподілу пам'яті.

bad\_cast - некоректне приведення типів .

bad\_typeid - неправильне застосування оператора typeid .

bad\_exception - непередбачена виняткова ситуація .

логічні помилки

out\_of\_range - вихід за межі припустимого діапазону .

invalid\_argument - невірний аргумент функції .

length\_error - перевищення припустимих розмірів об'єкта .

помилки при виконанні програми

domain\_error - вихід за межі припустимого діапазону.

overflow\_error - переповнення .

underflow\_error - утрата значимості .

domain\_error - вихід за межі припустимого діапазону

1. Для чого під час обробки виняткових ситуацій використовую призначені функції terminate(), unexpected(), abort()?

Если в программе нет предложения catch, способного обработать исключение, оно остается необработанным. Но исключение – это настолько серьезная ошибка, что программа не может продолжать выполнение. Поэтому, если обработчик не найден, вызывается функция terminate() из стандартной библиотеки C++. По умолчанию terminate() активизирует функцию abort(), которая аномально завершает программу. (В большинстве ситуаций вызов abort() оказывается вполне приемлемым решением. Однако иногда необходимо переопределить действия, выполняемые функцией terminate().  Функция terminate() предоставляет механизм времени выполнения, который извещает пользователя о том, что подходящего обработчика не нашлось.

нарушения спецификации исключений функции могут быть обнаружены только во время выполнения. Если функция возбуждает исключение, не указанное в спецификации, то вызывается unexpected() из стандартной библиотеки C++, а та по умолчанию вызывает terminate(). (В некоторых случаях необходимо переопределить действия, выполняемые функцией unexpected().  
Необходимо уточнить, что unexpected() не вызывается только потому, что функция возбудила исключение, не указанное в ее спецификации. Все нормально, если она обработает это исключение самостоятельно, внутри функции.

1. Навести приклад коду, за допомогою якого можна вивести на екран текстове повідомлення з номером строчки, під час виконання якої сталася виняткова ситуація, та назвою файлу, що містить строчку.

class my\_exception : public std::runtime\_error {

std::string msg;

public:

my\_exception(const std::string &arg, const char \*file, int line) :

std::runtime\_error(arg) {

std::ostringstream o;

o << file << ":" << line << ": " << arg;msg = o.str();

}

~my\_exception() throw() {}

const char \*what() const throw() {return msg.c\_str();}

};

#define throw\_line(arg) throw my\_exception(arg, \_\_FILE\_\_, \_\_LINE\_\_);

void f() {

throw\_line("Oh no!");

}

int main() {

try {

f();

}

catch (const std::runtime\_error &ex) {

std::cout << ex.what() << std::endl;

}

}