Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 12**

*«Виняткові ситуації в мові програмування С++»*

**з дисципліни «***Об’єктно–орієнтоване програмування***»**

**Лектор:**

ст. викладач кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-11

Артех О.Ю.

**Прийняла:**

асистент кафедри ПЗ

Терендій О.В.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Тема роботи:** Виняткові ситуації в мові програмування С++.

**Мета роботи:** Ознайомитися з синтаксисом та принципами використання винятків, нав­читися передбачати виняткові ситуації, які можуть виникнути в процесі роботи програмного забезпечення, а також навчитися їх перехоплювати та опрацьовувати.

**Теоретичні відомості**

В основі обробки виняткових ситуацій у мові С++ лежать три ключових слова: try, catch і throw.

Якщо програміст підозрює, що визначений фрагмент програми може спровокувати помилку, він повинний занурити цю частину коду в блок try. Необхідно мати на увазі, що зміст помилки (за винятком стандартних ситуацій) визначає сам програміст. Це значить, що програміст може задати будь-яку умову, що приведе до створення виняткової ситуації. Після цього необхідно вказати, у яких умовах варто генерувати виняткову ситуацію. Для цієї мети призначене ключове слово throw. І нарешті, виняткову ситуацію потрібно перехопити й обробити в блоці catch. Ось як виглядає ця конструкція.

try

{

*//* *Тіло блоку try*

if(*умова*) throw *виняткова\_ситуація*

}

catch(*тип1 аргумент*)

{

*// Тіло блоку catch*

}

catch(*тип2 аргумент*)

{

*// Тіло блоку catch*

}

.

.

.

catch(*типN аргумент*)

{

*// Тіло блоку catch*

}

Розмір блоку try не обмежений. У нього можна занурити як один оператор, так і цілу програму. Один блок try можна зв'язати з довільною кількістю блоків catch. Оскільки кожен блок catch відповідає окремому типу виняткової ситуації, програма сама визначить, який з них виконати. У цьому випадку інші блоки catch не виконуються. Кожен блок catch має аргумент, що приймає визначене значення. Цей аргумент може бути об'єктом будь-якого типу.

Якщо програма виконана правильно й у блоці try не виникло жодної виняткової ситуації, усі блоки catch будуть зігноровані. Якщо в програмі виникла подія, що програміст вважає небажаним, оператор throw генерує виняткову ситуацію. Для цього оператор throw повинний знаходитися усередині блоку try або усередині функції, викликуваної усередині блоку try.

Якщо в програмі виникла виняткова ситуація, для якої не передбачені перехоплення й обробка, викликається стандартна функція terminate(), що, у свою чергу, викликає функцію abort(). Утім, іноді виняткова ситуація не є небезпечної. У цьому випадку можна виправити помилку (наприклад, привласнити нульовому знаменнику ненульове значення) і продовжити виконання програми.

**Індивідуальне завдання**

1.    Ознайомитися з основними поняттями та синтаксисом мови С++, з метою передбачення та оброблення виняткових ситуацій.

2.    Провести аналіз завдання (індивідуальні варіанти), визначити можливі виняткові ситуації, які можуть виникнути в процесі роботи програмного забезпечення, та передбачити можливість запису в log-file.

3.    Розробити програмне забезпечення для реалізації поставленої задачі.

4.    Оформити і здати звіт про виконання лабораторної роботи. Звіт має містити варіант завдання, код розробленої програми, результати роботи програми (скріншоти), висновок.

**Хід роботи**

Код програми:

**Chyslo.h**

#ifndef INC\_12LAB2PZ\_CHYSLO\_H

#define INC\_12LAB2PZ\_CHYSLO\_H

#include "radix.h"

#include <string>

*class* **Chyslo** {

*public*:

*//radix* *=* *базис*

**Chyslo**(std::string num, int radix = 10);

*//----------------------------------------------------------------------*

**Chyslo**();

*//----------------------------------------------------------------------*

Chyslo &*operator*+(Chyslo &other) ;

*//----------------------------------------------------------------------*

Chyslo &*operator*-(Chyslo &other) ;

*//----------------------------------------------------------------------*

Chyslo &*operator*\*(Chyslo &other) ;

*//----------------------------------------------------------------------*

Chyslo &*operator*=(Chyslo &other) ;

*//----------------------------------------------------------------------*

std::string **getChyslo**(radix r);

*//----------------------------------------------------------------------*

int **getChyslo**(){

*return* (int)chyslo;

}

*//----------------------------------------------------------------------*

void **setChyslo**(long long int chyslo);

*private*:

long long chyslo;

};

#endif

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_calculate\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**others.h**

#ifndef OTHERS\_H

#define OTHERS\_H

char\* **lltoa**(long long val,char \*buf, int base){

int i = 62;

int sign = (val < 0);

*if*(sign) val = -val;

*if*(val == 0) *return* "0";

*for*(; val && i ; --i, val /= base) {

buf[i] = "0123456789abcdef"[val % base];

}

*if*(sign) {

buf[i--] = '-';

}

*return* &buf[i+1];

*class* **LetterFormatException**: *public* std::exception {

*const* char \****what***() *const* *noexcept*{

*return* "LetterFormatException";

}

};

*class* **NumberFormatException**: *public* std::exception {

*const* char \****what***() *const* *noexcept*{

*return* "NumberFormatException";

}

};

*enum* **operation** {

*plus*,

*minus*,

*multiply*,

*convert1*,

*convert2*

};

*class* **OutOfRangeException**: *public* std::exception {

*const* char \****what***() *const* *noexcept*{

*return* "OutOfRangeException";

}

};

*enum* **radix** {

*binary*=2,

*decimal*=10,

*hex*=16

};

*class* **UnsupportedBaseException**: *public* std::exception {

*const* char \****what***() *const* *noexcept*{

*return* "UnsupportedBaseException";

}

};

#endif

**Chyslo.cpp**

#include <algorithm>

#include "Chyslo.h"

#include "others.h"

*//----------------------------------------------------------------------*

Chyslo::**Chyslo**(std::string num, int radix) {

chyslo = 0;

*if* (num.empty()) {

*throw* NumberFormatException();

}

*switch* (radix) {

*case* 2:

*if*(num[0]!='1' && num[0]!='0'){*throw* NumberFormatException();}

*//Додатнє*

*if* (num[0] == '0') {

*for* (*auto* i = ++num.begin(); i < num.end(); i++) {

long long old = chyslo;

*//Множем* *на* *два.*

chyslo <<= 1;

*if* (\*i == '1' || \*i == '0') {

chyslo += \*i - '0';

} *else* {

*throw* NumberFormatException();

}

*if* (chyslo < old) {

*throw* OutOfRangeException();

}

}

*//Відємне*

} *else* {

long long power = 0;

*for* (*auto* i = ++num.begin(); i < num.end(); i++) {

long long oldChyslo = chyslo;

long long oldPower = power;

chyslo <<= 1;

*if* (\*i == '1' || \*i == '0') {

chyslo += \*i - '0';

} *else* {

*throw* NumberFormatException();

}

power <<= 1;

power += 1;

*if* (power < oldPower || oldChyslo > chyslo) {

*throw* OutOfRangeException();

}

}

power += 1;

chyslo = -power + chyslo;

}

*break*;

*case* 10:

*//Cхема* *Горнера.*

*if* (num[0] == '-') {

*for* (*auto* i = ++num.begin(); i < num.end(); i++) {

*if* ('0' <= \*i && \*i <= '9') {

long long old = chyslo;

chyslo \*= 10;

chyslo -= \*i - '0';

*if* (old < chyslo) {

*throw* OutOfRangeException();

}

} *else* {

*throw* LetterFormatException();

}

}

} *else* {

*for* (*auto* i:num) {

*if* ('0' <= i && i <= '9') {

long long old = chyslo;

chyslo \*= 10;

chyslo += i - '0';

*if* (old > chyslo) {

*throw* OutOfRangeException();

}

} *else* {

*throw* LetterFormatException();

}

}

}

*break*;

*case* 16:

*//Лямда* *вираз.* *Переводимо* *стрічку* *до* *малих* *букв.*

std::transform(num.begin(), num.end(), num.begin(), [](unsigned char c) { *return* std::tolower(c); });

*if* (!((num[0] >= 'a' && num[0] <= 'f') || (num[0] >= '0' && num[0] <= '9'))) {

*throw* NumberFormatException();

}

*if* (num[0] <= '7') {*//додатні*

chyslo += num[0] - '0';

*for* (*auto* i = ++num.begin(); i < num.end(); i++) {

*if* (!((\*i >= 'a' && \*i <= 'f') || (\*i >= '0' && \*i <= '9'))) {

*throw* NumberFormatException();

} *else* {

long long old = chyslo;

chyslo <<= 4;

*if* (\*i <= '9') {

chyslo += \*i - '0';

} *else* {

chyslo += \*i - 'a' + 10;

}

*if* (old > chyslo) {

*throw* OutOfRangeException();

}

}

}

} *else* {

long long power = 15;

*if* (num[0] <= '9') {

chyslo += num[0] - '0';

} *else* {

chyslo += num[0] - 'a' + 10;

}

*for* (*auto* i = ++num.begin(); i < num.end(); i++) {

*if* (!((\*i >= 'a' && \*i <= 'f') || (\*i >= '0' && \*i <= '9'))) {

*throw* NumberFormatException();

} *else* {

long long oldChyslo = chyslo;

long long oldPower = power;

chyslo <<= 4;

*if* (\*i <= '9') {

chyslo += \*i - '0';

} *else* {

chyslo += \*i - 'a' + 10;

}

*//число* *яке* *заповнене* *все* *одиницями,* *щоб* *інвертувати.*

power <<= 4;

power += 15;

*if* (power < oldPower || oldChyslo > chyslo) {

*throw* OutOfRangeException();

}

}

}

*//1000*

power += 1;

chyslo = -power + chyslo;

}

*break*;

*default*:

*throw* UnsupportedBaseException();

}

}

*//----------------------------------------------------------------------*

std::string Chyslo::**getChyslo**(radix r) {

char buffer [64]={0};

*if*(r>=*binary*){

*return* std::string(lltoa(chyslo,buffer,r));

}

*else* *throw* UnsupportedBaseException();

}

*//----------------------------------------------------------------------*

void Chyslo::**setChyslo**(long long int chyslo) {

Chyslo::chyslo = chyslo;

}

*//----------------------------------------------------------------------*

Chyslo& Chyslo::*operator*+(Chyslo &other) {

*auto* \*result = *new* Chyslo();

result->chyslo = chyslo + other.chyslo;

*if* (chyslo >= 0 && other.chyslo >= 0 && (result->chyslo < chyslo || result->chyslo < other.chyslo)) {

*throw* OverflowException();

}

*if* (chyslo <= 0 && other.chyslo <= 0 && (result->chyslo > chyslo || result->chyslo > other.chyslo)) {

*throw* OverflowException();

}

*return* \*result;

}

*//----------------------------------------------------------------------*

Chyslo& Chyslo::*operator*-(Chyslo &other) {

*auto* \*result = *new* Chyslo();

result->chyslo = chyslo - other.chyslo;

*if* (chyslo <= 0 && other.chyslo >= 0 && (result->chyslo > chyslo || result->chyslo > other.chyslo)) {

*throw* OverflowException();

}

*if* (chyslo >= 0 && other.chyslo <= 0 && (result->chyslo < chyslo || result->chyslo < other.chyslo)) {

*throw* OverflowException();

}

*return* \*result;

}

*//----------------------------------------------------------------------*

Chyslo& Chyslo::*operator*\*(Chyslo &other) {

*auto* \*result = *new* Chyslo();

result->chyslo = chyslo \* other.chyslo;

*if* (chyslo >= 0 && other.chyslo >= 0 && (result->chyslo < std::max(chyslo, other.chyslo))) {

*throw* OverflowException();

}

*//різні* *знаки*

*if* (((chyslo >= 0 && other.chyslo <= 0) || (chyslo <= 0 && other.chyslo >= 0)) &&

(result->chyslo > std::min(chyslo, other.chyslo))) {

*throw* OverflowException();

}

*if* (chyslo <= 0 && other.chyslo <= 0 && (result->chyslo < -std::min(chyslo, other.chyslo))) {

*throw* OverflowException();

}

*return* \*result;

}

*//----------------------------------------------------------------------*

Chyslo &Chyslo::*operator*=(Chyslo &other) {

*this*->chyslo=other.chyslo;

*return* \**this*;

}

*//----------------------------------------------------------------------*

Chyslo::**Chyslo**() {

chyslo = 0;

}

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "others.h"

#include "QMessageBox"

#include "Chyslo.h"

#include <exception>

#include <ctime>

#include <fstream>

#include <string.h>

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

*//----------------------------------------------------------------------*

void **f\_logging**(*const* char \* info)

{

std::ofstream LOG;

char\* filename = "log";

LOG.open (filename, std::ios\_base::out | std::ios\_base::app);

time\_t now = time(0);

char\* dt = ctime(&now);

int len = strlen(dt);

dt[len-1] = '\0';

LOG << "[" << dt << "] " << info << std::endl;

LOG.close();

}

void MainWindow::**on\_calculate\_clicked**()

{

radix radix1, radix2, radix3;

operation flag=(operation)-1;

Chyslo result;

*if*(ui->plus->isChecked()){

flag = *plus*;

}

*else* *if*(ui->minus->isChecked()){

flag = *minus*;

}

*else* *if*(ui->multiply->isChecked()){

flag = *multiply*;

}

*else* *if*(ui->changeFirst->isChecked()){

flag = *convert1*;

}

*else* *if*(ui->changeSecond->isChecked()){

flag = *convert2*;

}

*else* {QMessageBox::warning(*this*,"Операція введення","Похоже на те, що ви не вибрали операцію.");

*return*;

}

*//*

*if*(ui->first\_double->isChecked()){

radix1 = *binary*;

}

*else* *if*(ui->first\_tens->isChecked()){

radix1 = *decimal*;

}

*else* *if* (ui->first\_sixteenth->isChecked()){

radix1 = *hex*;

}

*else* radix1 = (radix)0;

*//*

*if*(ui->second\_double->isChecked()){

radix2 = *binary*;

}

*else* *if*(ui->second\_tens->isChecked()){

radix2 = *decimal*;

}

*else* *if* (ui->second\_sixteenth->isChecked()){

radix2 = *hex*;

}

*else* radix2 = (radix)0;

*//*

*if*(ui->result\_double->isChecked()){

radix3 = *binary*;

}

*else* *if*(ui->result\_tens->isChecked()){

radix3 = *decimal*;

}

*else* *if* (ui->result\_sixteenth->isChecked()){

radix3 = *hex*;

}

*else* radix3 = (radix)0;

*//*

*try* {

*switch* (flag) {

*case* *plus*:{

Chyslo chyslo1(ui->first->text().toStdString(),radix1);

Chyslo chyslo2(ui->second->text().toStdString(),radix2);

result = chyslo1 + chyslo2;

*break*;

}

*case* *minus*:{

Chyslo chyslo1(ui->first->text().toStdString(),radix1);

Chyslo chyslo2(ui->second->text().toStdString(),radix2);

result = chyslo1 - chyslo2;

*break*;

}

*case* *multiply*:{

Chyslo chyslo1(ui->first->text().toStdString(),radix1);

Chyslo chyslo2(ui->second->text().toStdString(),radix2);

result = chyslo1 \* chyslo2;

*break*;

}

*case* *convert1*:{

Chyslo chyslo1(ui->first->text().toStdString(),radix1);

result = chyslo1;

*break*;

}

*case* *convert2*:{

Chyslo chyslo2(ui->second->text().toStdString(),radix2);

result = chyslo2;

*break*;

}

}

ui->result->setText(QString::fromStdString(result.getChyslo(radix3)));

}

*catch*(NumberFormatException& a){

QMessageBox::critical(*this*,"Помилка вводу","Неправильно введені числа, повторіть ще раз.");

f\_logging("NumberFormatException");

*return*;

}

*catch*(LetterFormatException& a){

QMessageBox::critical(*this*,"Помилка вводу","Введені букви замість чисел, повторіть ще раз.");

f\_logging("LetterFormatException");

*return*;

}

*catch*(OutOfRangeException& a){

QMessageBox::critical(*this*,"Помилка вводу"," Введене завелике число, повторіть ще раз.");

f\_logging("OutOfRangeException");

*return*;

}

*catch*(OverflowException& a){

QMessageBox::critical(*this*,"Невідома поведінка","Переповнення, перевірте та повторіть ще раз.");

f\_logging("OverflowException");

*return*;

}

*catch*(UnsupportedBaseException& a){

QMessageBox::critical(*this*,"Помилка","Невибрані кнопки, повторіть ще раз.");

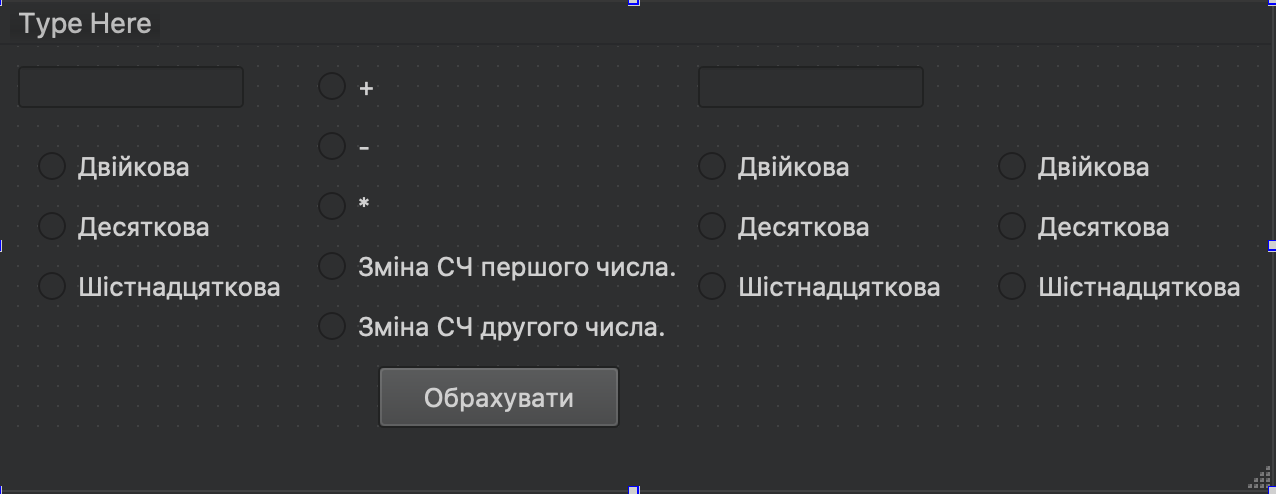
f\_logging("UnsupportedBaseException");

*return*;

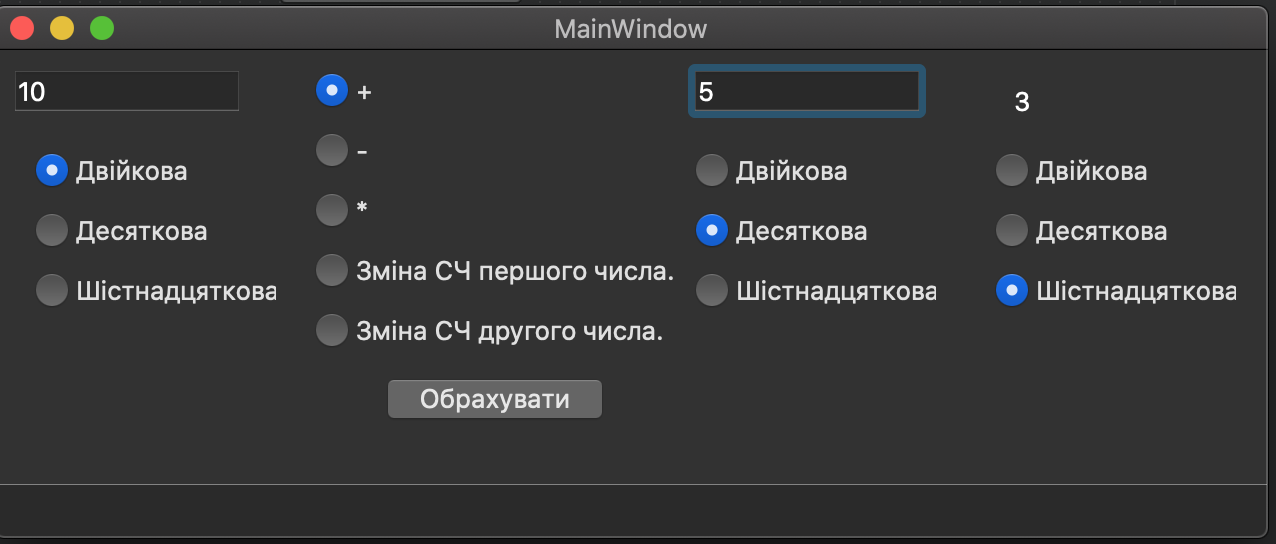
}

}

Форма MainWindow:



Результат виконання:



**Висновки**

Під час виконання лабораторної роботи я ознайомився з синтаксисом, та принципами використання винятків, навчився передбачати виняткові ситуації, які можуть виникнути в процесі роботи програмного забезпечення, а також навчився їх перехоплювати та опрацьовувати. Отримані знання закріпив на практиці, розробивши програму для обчислення арифметичних дій над числами у різних форматах.