

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут ІКНІ

Кафедра ПЗ

ЗВІТ

До лабораторної роботи №12

На тему: «Виняткові ситуації в мові програмування C++»

З дисципліни «Об'єктно-орієнтоване програмування»

Лектор: доцент каф. ПЗ

Коротєєва Т.О.

Виконала: ст.гр. ПЗ-23

Кохман О.В.

Прийняла: доцент каф. ПЗ

Коротєєва Т.О.

«_____» _____ 2022р.

Σ _____.

Львів – 2022

Тема: Виняткові ситуації в мові програмування C++.

Мета: 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 C++ 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 небезпечною. 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 C++, 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.
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.

2. Розробити програмне забезпечення, яке дозволяло б працювати з числами в різних форматах. Числа представляти, як об'єкти класу Chyslo. Користувач задає ціле число з клавіатури в одному з трьох форматів: двійковий, десятковий, шістнадцятковий (формат користувач теж задає при вводі). Клас повинен містити функціонал, який дозволяв би конвертувати число з одного формату в інший, а також додавати, віднімати і множити числа. Програма повинна перехоплювати та опрацьовувати такі виняткові ситуації: а) випадковий ввід користувачем символу замість цифри, б) переповнення, в) введення занадто великого числа, г) ще дві виняткові ситуації передбачити самостійно.

Всі функції повинні містити список винятків, які вони можуть генерувати.

Код програми

Назва файлу: Chyslo.h

```
#ifndef CHYSLO_H
#define CHYSLO_H
#pragma once
#include <string>
class OverflowChysloException : public std::exception {
};
class NegativeBinaryNotSupportedException : public std::exception {
};
class Chyslo {
private:
    std::string value;
    std::string type;
    const char* hexCharToBinary(char c);
    void checkOverflow(long long value);
public:
    Chyslo(std::string value, std::string type);
    Chyslo convertToDecimal();
    Chyslo convertToHex();
    Chyslo convertToBinary();
    Chyslo add(Chyslo chyslo);
    Chyslo multiply(Chyslo chyslo);
    Chyslo minus(Chyslo chyslo);
    std::string getValue();
    bool isDecimal();
    bool isHex();
    bool isBinary();
    static std::string toStandardString(System::String^ string);
};
#endif
```

Назва файлу: Chyslo.cpp

```
#include "Chyslo.h"
#include <sstream>
std::string Chyslo::getValue() {
```

```

        return value;
    }
    Chyslo::Chyslo(std::string value, std::string type) {
        this->value = value;
        this->type = type;
    }
    bool Chyslo::isBinary() {
        bool flag = true;
        for (int i = 0; i < value.size(); i++) {
            if (value[i] != '1' && value[i] != '0') {
                flag = false;
            }
        }
        return type == "binary" && flag;
    }
    bool Chyslo::isDecimal() {
        return type == "decimal" && (value[0] == '-' || value.substr(0, 1)
            .find_first_not_of("0123456789") == std::string::npos) && value
            .substr(1, value.length()).find_first_not_of("0123456789") ==
std::string::npos;
    }
    bool Chyslo::isHex() {
        int n = value.length();
        if (n == 0) {
            return false;
        }
        bool flag = true;
        for (int i = 0; i < n; i++) {
            char ch = value[i];
            if ((ch < '0' || ch > '9') && (ch < 'A' || ch > 'F')) {
                flag = false;
            }
        }
        return type == "hex" && flag;
    }
    Chyslo Chyslo::convertToBinary() {
        if (isBinary())
            return *this;
        if (isHex()) {
            std::string bin;
            for (unsigned i = 0; i != value.length(); ++i)
                bin += hexCharToBinary(value[i]);
            return Chyslo(bin, "binary");
        }
        if (isDecimal()) {
            int decimal = stoi(value);
            if (decimal < 0)
                throw NegativeBinaryNotSupportedException();
            int binary = 0, remainder, product = 1;
            while (decimal != 0) {
                remainder = decimal % 2;
                binary = binary + (remainder * product);
                decimal = decimal / 2;
                product *= 10;
            }
            return Chyslo(std::to_string(binary), "binary");
        }
    }
    Chyslo Chyslo::convertToHex() {
        if (isHex())
            return *this;
        if (isBinary()) {
            std::stringstream res;

```

```

        res << std::hex << stoi(value);
        return Chyslo(res.str(), "hex");
    }
    if (isDecimal()) {
        std::stringstream ss;
        ss << std::hex << stoi(value);
        std::string res(ss.str());
        return Chyslo(res, "hex");
    }
}

Chyslo Chyslo::convertToDecimal() {
    if (isDecimal())
        return *this;
    if (isBinary()) {
        int dec = 0, i = 0, rem;
        int n = stoi(value);
        while (n != 0) {
            rem = n % 10;
            n /= 10;
            dec += rem * pow(2, i);
            ++i;
        }
        return Chyslo(toStandardString(dec.ToString()), "decimal");
    }
    if (isHex()) {
        int result;
        std::stringstream ss;
        ss << std::hex << value;
        ss >> result;
        return Chyslo(toStandardString(result.ToString()), "decimal");
    }
}

const char* Chyslo::hexCharToBinary(char c) {
    switch (toupper(c)) {
        case '0': return "0000";
        case '1': return "0001";
        case '2': return "0010";
        case '3': return "0011";
        case '4': return "0100";
        case '5': return "0101";
        case '6': return "0110";
        case '7': return "0111";
        case '8': return "1000";
        case '9': return "1001";
        case 'A': return "1010";
        case 'B': return "1011";
        case 'C': return "1100";
        case 'D': return "1101";
        case 'E': return "1110";
        case 'F': return "1111";
    }
}

std::string Chyslo::toStandardString(System::String^ string) {
    using System::Runtime::InteropServices::Marshal;
    System::IntPtr pointer = Marshal::StringToHGlobalAnsi(string);
    char* charPointer = reinterpret_cast<char*>(pointer.ToPointer());
    std::string returnString(charPointer, string->Length);
    Marshal::FreeHGlobal(pointer);
    return returnString;
}

void Chyslo::checkOverflow(long long value) {
    if (value > INT_MAX) {
        throw OverflowChysloException();
    }
}

```

```

    }
}
Chyslo Chyslo::add(Chyslo chyslo) {
    if (isBinary()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(convertToDecimal().getValue()) +
stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()),
"decimal").convertToBinary();
    }
    if (isDecimal()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(value) + stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()), "decimal");
    }
    if (isHex()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(convertToDecimal().getValue()) +
stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()),
"decimal").convertToHex();
    }
}
Chyslo Chyslo::multiply(Chyslo chyslo) {
    if (isBinary()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(convertToDecimal().getValue()) *
stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()),
"decimal").convertToBinary();
    }
    if (isDecimal()) {
        auto result = stoll(getValue()) * stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()), "decimal");
    }
    if (isHex()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(convertToDecimal().getValue()) *
stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()),
"decimal").convertToHex();
    }
}
Chyslo Chyslo::minus(Chyslo chyslo) {
    if (isBinary()) {
        chyslo = chyslo.convertToDecimal();
        auto result = stoll(convertToDecimal().getValue()) -
stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()),
"decimal").convertToBinary();
    }
    if (isDecimal()) {
        auto result = stoll(getValue()) - stoll(chyslo.getValue());
        checkOverflow(result);
        return Chyslo(toStandardString(result.ToString()), "decimal");
    }
}

```

```

        if (isHex()) {
            chyslo = chyslo.convertToDecimal();
            auto result = stoll(convertToDecimal().getValue()) -
stoll(chyslo.getValue());
            checkOverflow(result);
            return Chyslo(toStdString(result.ToString()),
"decimal").convertToHex();
        }
    }
}

```

Назва файлу: MyForm.h

```

#pragma once
#include <string>
#include "Chyslo.h"
namespace lab12 {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    class NotSelectedTypeException : public std::exception {
    };
    class NotRecognizedNumberException : public std::exception {
    };
    class OverLimitNumberException : public std::exception {
    };
    const long long LIMIT_NUMBER = 100000;
    public ref class MyForm : public System::Windows::Forms::Form
    {
    public:
        MyForm(void)
        {
            InitializeComponent();
        }
    protected:
        ~MyForm()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::TextBox^ FirstNumberTextBox;
    protected:
    private: System::Windows::Forms::ComboBox^ FirstNumberComboBox;
    private: System::Windows::Forms::TextBox^ SecondNumberTextBox;
    private: System::Windows::Forms::ComboBox^ SecondNumberComboBox;
    private: System::Windows::Forms::Button^ addButton;
    private: System::Windows::Forms::Button^ multiplyButton;
    private: System::Windows::Forms::Button^ minusButton;
    private: System::Windows::Forms::Button^ convertToBinaryButton;
    private: System::Windows::Forms::Button^ convertToDecimal;
    private: System::Windows::Forms::Label^ firstNumberLabel;
    private: System::Windows::Forms::Label^ secondNumberLabel;
    private: System::Windows::Forms::Button^ convertToHexButton;
    private:
        System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
#pragma endregion
}

```



```

        private: System::Void
FirstNumberComboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
    }
private: System::Void MyForm_Load(System::Object^ sender, System::EventArgs^ e) {
    Form::CenterToScreen();

    FirstNumberComboBox->Items->Add("binary");
    FirstNumberComboBox->Items->Add("decimal");
    FirstNumberComboBox->Items->Add("hex");

    SecondNumberComboBox->Items->Add("binary");
    SecondNumberComboBox->Items->Add("decimal");
    SecondNumberComboBox->Items->Add("hex");
}
private: System::Void convertFirstNumberButton_Click(System::Object^ sender,
System::EventArgs^ e) { // convert to binary
    auto input = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);

    if (input) {
        auto chyslo = getChyslo(FirstNumberTextBox, FirstNumberComboBox);

        try {
            firstNumberLabel->Text = gcnew
String(chyslo.convertToBinary().getValue().c_str());
        }
        catch (NegativeBinaryNotSupportedException) {
            MessageBox::Show(this, "Negative binary numbers are not
supported", "Validation error", MessageBoxButtons::OK);
        }
    }
}
private: System::Void convertToHexButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    auto input = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);

    if (input) {
        auto chyslo = getChyslo(FirstNumberTextBox, FirstNumberComboBox);
        firstNumberLabel->Text = gcnew
String(chyslo.convertToHex().getValue().c_str());
    }
}
private: System::Void convertToDecimal_Click(System::Object^ sender,
System::EventArgs^ e) {
    auto input = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);

    if (input) {
        auto chyslo = getChyslo(FirstNumberTextBox, FirstNumberComboBox);
        firstNumberLabel->Text = gcnew
String(chyslo.convertToDecimal().getValue().c_str());
    }
}
private: System::Void addButton_Click(System::Object^ sender, System::EventArgs^
e) {
    auto inputOne = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);
    auto inputTwo = getValidChyslo(SecondNumberTextBox, SecondNumberComboBox);

    if (inputOne && inputTwo) {
        auto chysloOne = getChyslo(FirstNumberTextBox, FirstNumberComboBox);
        auto chysloTwo = getChyslo(SecondNumberTextBox,
SecondNumberComboBox);

        try {

```

```

        firstNumberLabel->Text = gcnew
String(chysloOne.add(chysloTwo).getValue().c_str());
    }
    catch (OverflowChysloException) {
        MessageBox::Show(this, "There was overflow!", "Validation
error", MessageBoxButtons::OK);
    }
}

private: System::Void multiplyButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    auto inputOne = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);
    auto inputTwo = getValidChyslo(SecondNumberTextBox, SecondNumberComboBox);

    if (inputOne && inputTwo) {
        auto chysloOne = getChyslo(FirstNumberTextBox, FirstNumberComboBox);
        auto chysloTwo = getChyslo(SecondNumberTextBox,
SecondNumberComboBox);

        try {
            firstNumberLabel->Text = gcnew
String(chysloOne.multiply(chysloTwo).getValue().c_str());
        }
        catch (OverflowChysloException) {
            MessageBox::Show(this, "There was overflow!", "Validation
error", MessageBoxButtons::OK);
        }
    }
}

private: System::Void minusButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    auto inputOne = getValidChyslo(FirstNumberTextBox, FirstNumberComboBox);
    auto inputTwo = getValidChyslo(SecondNumberTextBox, SecondNumberComboBox);

    if (inputOne && inputTwo) {
        auto chysloOne = getChyslo(FirstNumberTextBox, FirstNumberComboBox);
        auto chysloTwo = getChyslo(SecondNumberTextBox,
SecondNumberComboBox);

        try {
            firstNumberLabel->Text = gcnew
String(chysloOne.minus(chysloTwo).getValue().c_str());
        }
        catch (OverflowChysloException) {
            MessageBox::Show(this, "There was overflow!", "Validation
error", MessageBoxButtons::OK);
        }
    }
}

private: Chyslo getChyslo(System::Windows::Forms::TextBox^ textBox,
System::Windows::Forms::ComboBox^ comboBox) {
    return Chyslo(Chyslo::toStandardString(textBox->Text->ToString()),
Chyslo::toStandardString(comboBox->SelectedItem->ToString()));
}

private: std::string* getValidChyslo(System::Windows::Forms::TextBox^
textBox, System::Windows::Forms::ComboBox^ comboBox) {
    try {
        auto selectedType = comboBox->SelectedItem;
        if (!selectedType) {
            throw NotSelectedTypeException();
        }
    }
}

```

```

        auto chyslo = Chyslo(Chyslo::toStandardString(textBox->Text->ToString()), Chyslo::toStandardString(selectedType->ToString()));

        if (!chyslo.isBinary() && !chyslo.isDecimal() && !chyslo.isHex()) {
            throw NotRecognizedNumberException();
        }

        if (stoll(chyslo.getValue()) > LIMIT_NUMBER) {
            throw OverLimitNumberException();
        }

        return &chyslo.getValue();
    }
    catch (NotRecognizedNumberException) {
        MessageBox::Show(this, "Not recognized number type for the
following input: " + textBox->Text + ". Please try again", "Validation error",
MessageBoxButtons::OK);
    }
    catch (NotSelectedTypeException) {
        MessageBox::Show(this, "Not selected type. Please try again",
"Validation error", MessageBoxButtons::OK);
    }
    catch (OverLimitNumberException) {
        MessageBox::Show(this, "Input number is too large. Please try
again", "Validation error", MessageBoxButtons::OK);
    }
    return nullptr;
}
};
}

```

Назва файлу: MyForm.cpp

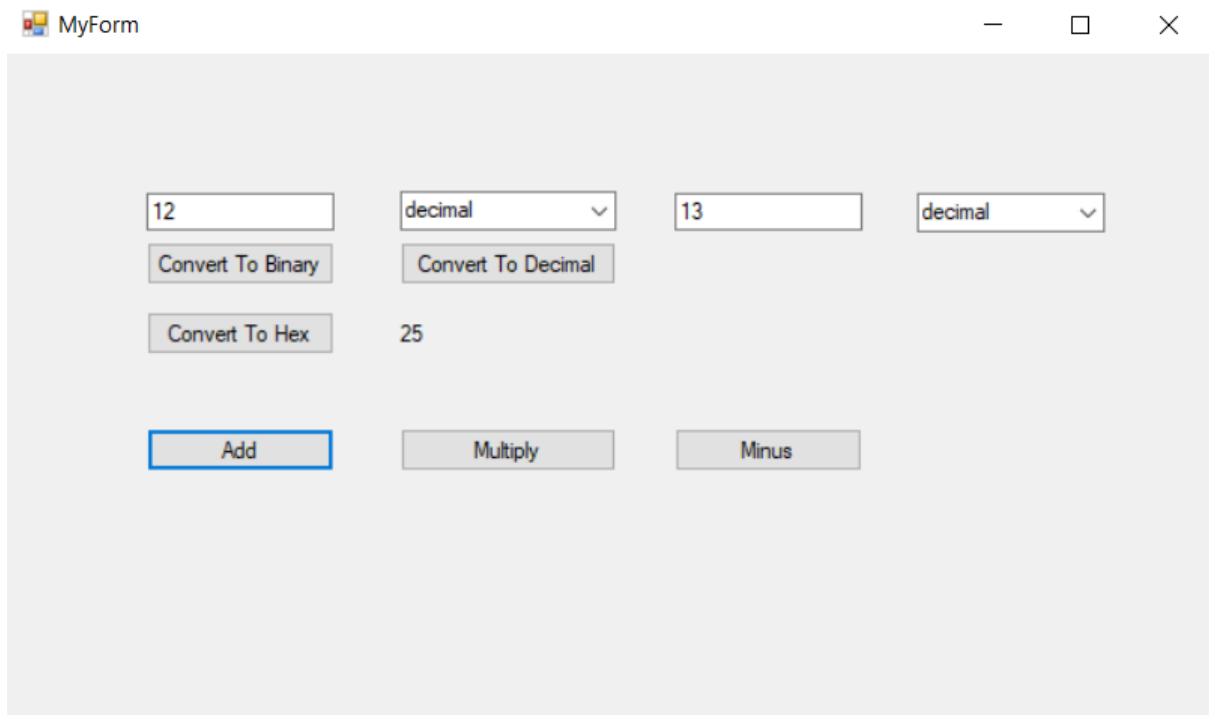
```

#include "MyForm.h"

using namespace lab12;
int main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew MyForm());
    return 0;
}

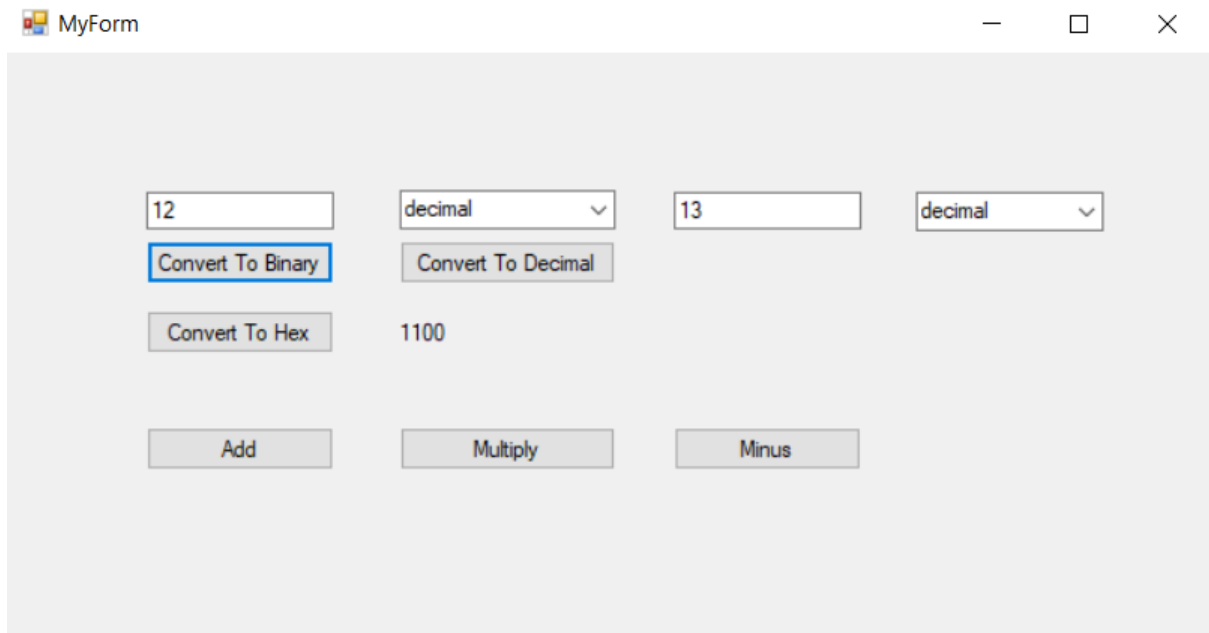
```

Протокол роботи



The screenshot shows a window titled "MyForm" with standard Windows window controls (minimize, maximize, close). The interface contains two input fields with values "12" and "13", both set to "decimal" in the dropdown menus. Below the first input field are buttons for "Convert To Binary" and "Convert To Hex". Below the second input field is a "Convert To Decimal" button. In the center, the number "25" is displayed. At the bottom, there are three buttons: "Add" (highlighted with a blue border), "Multiply", and "Minus".

Рис. 1 Результат додавання двох чисел.



The screenshot shows the same "MyForm" window. The input fields still contain "12" and "13" in decimal mode. The "Convert To Binary" button under the first input field is now highlighted with a blue border. The central display now shows the binary value "1100". The "Add", "Multiply", and "Minus" buttons remain at the bottom.

Рис. 2 Результат конвертування.

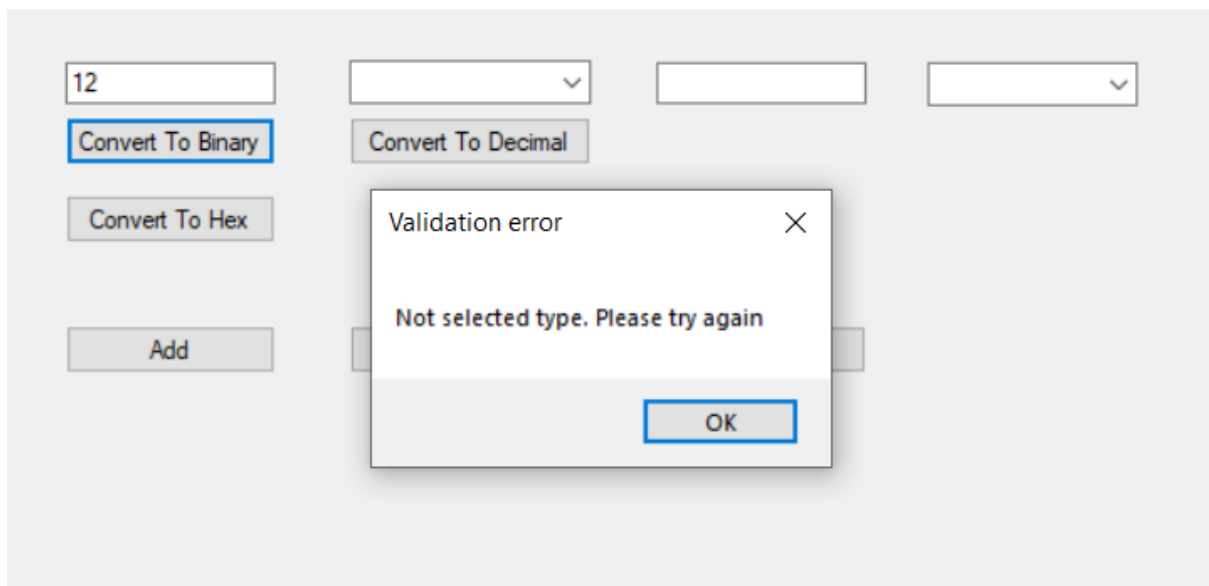


Рис. 3 Виняткова ситуація через необраний тип числа.

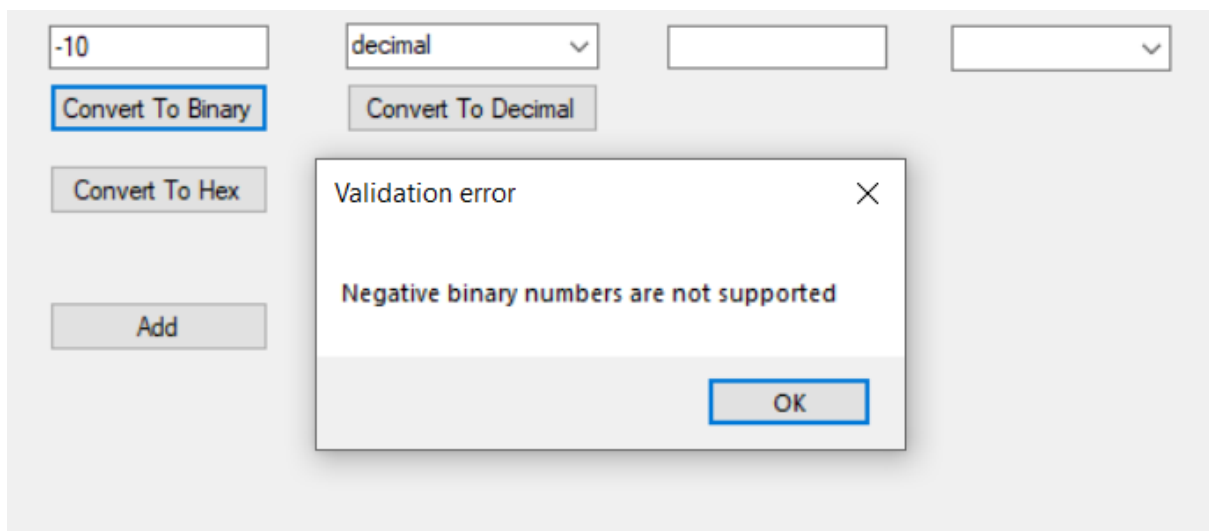


Рис. 4 Виняткова ситуація через від'ємне значення числа.

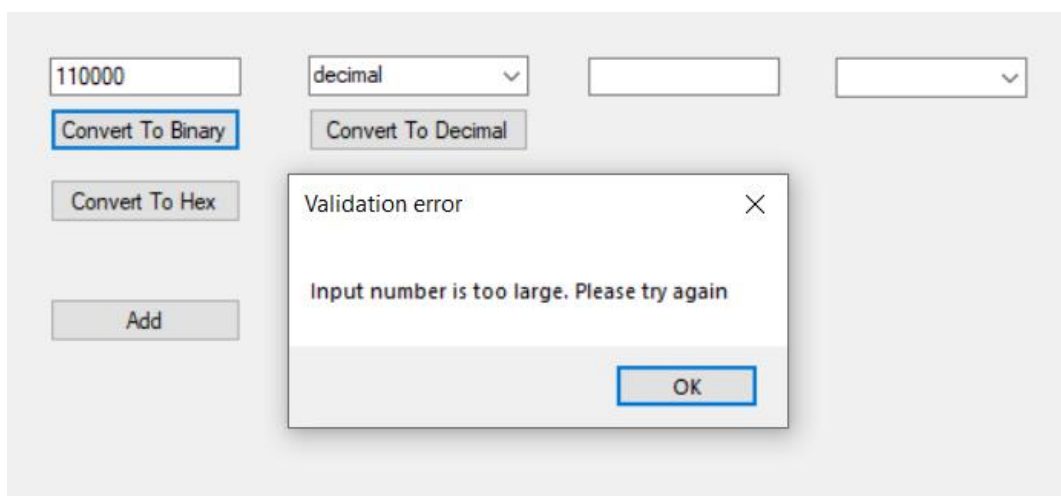


Рис. 5 Виняткова ситуація через занадто велике число.

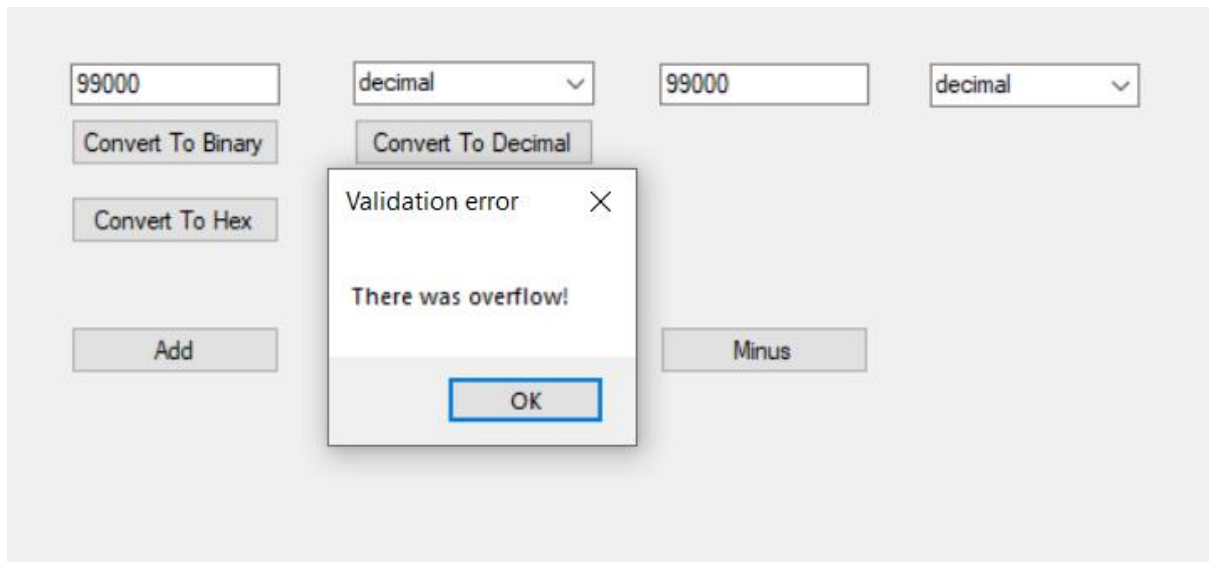


Рис. 6 Виняткова ситуація через переповнення.

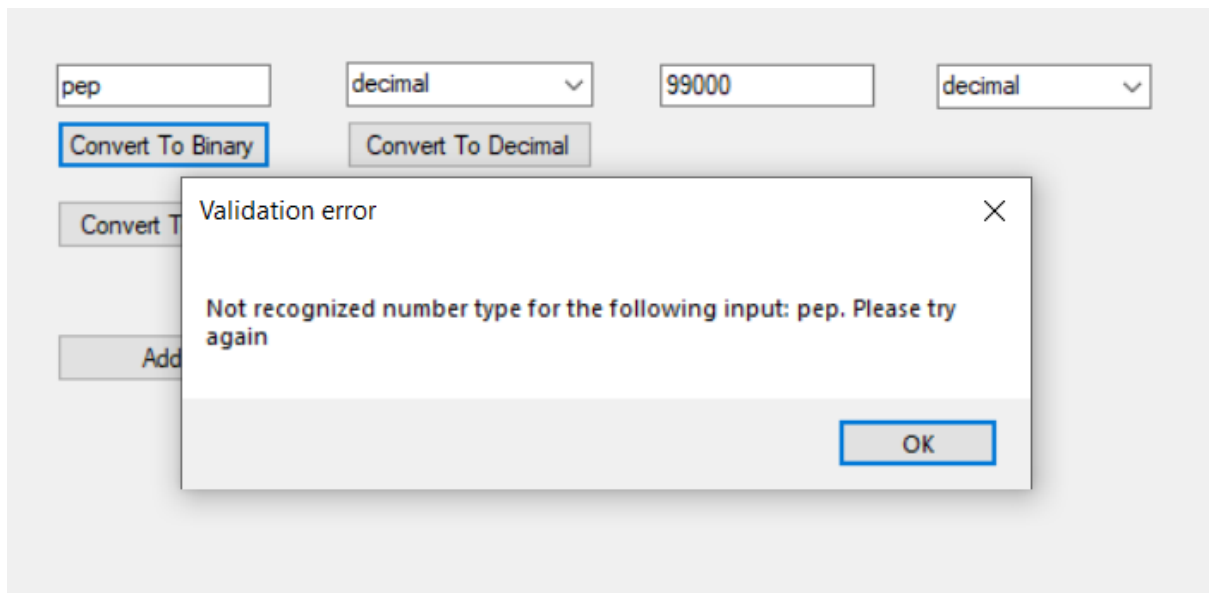


Рис. 7 Виняткова ситуація через заборонені символи в числі.

Висновок

На цій лабораторній роботі я дізналась про виняткові ситуації в c++, також реалізувала програму і використала там виняткові ситуації та продемонструвала результати роботи програми на формі у Visual Studio 2022.