|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №6**

**«Исключения и обработка исключений»**

**ДИСЦИПЛИНА: «Высокоуровневое программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-22Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Козина А.В. )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга , 2021 | | |

**Цель:** приобретение практических навыков и знаний по обработке ошибок и исключений в программе, во время её выполнения.

**Задачи:**

1. Познакомиться с типами ошибок;
2. Научиться обрабатывать ошибки при компиляции;
3. Познакомиться с концепцией исключений;
4. Научиться вызывать и обрабатывать исключения;
5. Получение навыков по созданию своих типов исключений;
6. Изучить дополнительные навыки по работе с ООП.

**Вариант 8**

**Задание:**

Как вы помните, на прошлой Л/Р мы закончили написание вашей программы. Теперь она работает и выполняет свои функции. Сейчас ваша задача заключается в анализе кода программы и поиска мест потенциальных ошибок. Как правило, такие места находятся в логике получения внешних данных (например, из хранилища, или ввод пользователя). Протестируйте эти места с различными сценариями работы и выявите ошибки, с которыми падает ваша программа. В Вашей программе должны быть отработаны 3-5 исключений.

**UML-диаграмма классов:**



**Рисунок 1.1.** UML-диаграмма классов



**Рисунок 1.2.** UML-диаграмма классов



**Рисунок 1.3.** UML-диаграмма классов



**Рисунок 1.4.** UML-диаграмма классов



**Рисунок 1.5.** UML-диаграмма классов

**Листинг:**

**InputException.h**

#ifndef INPUT\_EXCEPTION\_H

#define INPUT\_EXCEPTION\_H

namespace KMK

{

class InputException

{

public:

InputException(const char\* error);

const char\* GetError();

private:

const char\* m\_error{};

};

class StringToUInt16Exception : public InputException

{

public:

StringToUInt16Exception(const char\* error);

};

class StringToUInt32Exception : public InputException

{

public:

StringToUInt32Exception(const char\* error);

};

}

#endif // !INPUT\_EXCEPTION\_H

**InputException.cpp**

#include "InputException.h"

namespace KMK

{

InputException::InputException(const char\* error) { m\_error = error; }

const char\* InputException::GetError() { return m\_error; }

StringToUInt16Exception::StringToUInt16Exception(const char\* error) : InputException(error) {}

StringToUInt32Exception::StringToUInt32Exception(const char\* error) : InputException(error) {}

}

**UserInput.h**

#ifndef USER\_INPUT\_H

#define USER\_INPUT\_H

#include <string>

#include <iostream>

#include "InputException.h"

namespace KMK

{

class UserInput

{

public:

UserInput();

uint16\_t GetUInt16(uint16\_t minimum = 0, uint16\_t maximum = USHRT\_MAX);

size\_t GetUInt32(size\_t minimum = 0, size\_t maximum = UINT\_MAX);

friend std::istream& operator>>(std::istream& in, UserInput& userInput);

private:

std::string m\_input{};

};

}

#endif // !USER\_INPUT\_H

**UserInput.cpp**

#include "UserInput.h"

namespace KMK

{

UserInput::UserInput() { m\_input = {}; }

uint16\_t UserInput::GetUInt16(uint16\_t minimum, uint16\_t maximum)

{

if (m\_input.size() == 0)

{

throw StringToUInt16Exception("Error: the input is empty");

}

if (m\_input.size() > 5)

{

throw StringToUInt16Exception("Error: too many symbols");

}

for (size\_t i{}; i < m\_input.size(); ++i)

{

if (!isdigit(m\_input[i]))

{

throw StringToUInt16Exception("Error: there are not only digits in the input");

}

}

uint32\_t UInt32Input = std::stoul(m\_input);

if (UInt32Input < minimum || UInt32Input > maximum)

{

throw StringToUInt16Exception("Error: the number is out of range");

}

else

{

return (uint16\_t)UInt32Input;

}

}

size\_t UserInput::GetUInt32(size\_t minimum, size\_t maximum)

{

if (m\_input.size() == 0)

{

throw StringToUInt32Exception("Error: the input is empty");

}

if (m\_input.size() > 10)

{

throw StringToUInt32Exception("Error: too many symbols");

}

for (size\_t i{}; i < m\_input.size(); ++i)

{

if (!isdigit(m\_input[i]))

{

throw StringToUInt32Exception("Error: there are not only digits in the input");

}

}

uint32\_t UInt32Input{};

try

{

UInt32Input = std::stoul(m\_input);

}

catch (std::out\_of\_range)

{

throw StringToUInt32Exception("Error: the number is too big");

}

if (UInt32Input < minimum || UInt32Input > maximum)

{

throw StringToUInt32Exception("Error: the number is out of range");

}

else

{

return UInt32Input;

}

}

std::istream& operator>>(std::istream& in, UserInput& userInput)

{

std::getline(in, userInput.m\_input);

return in;

}

}

**Изменения в Menu.cpp**

int Menu::RunCommand() const

{

Print();

std::cout << m\_count << ". Exit\n";

std::cout << ">>> ";

UserInput input{};

std::cin >> input;

try

{

uint16\_t command = input.GetUInt16(0, m\_count);

system("cls");

if (command != m\_count)

{

return m\_items[command]->Run();

}

else

{

return 1;

}

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

return 0;

}

}

**Изменения в AuthorizationItem.cpp**

...

UserInput input{};

system("cls");

std::cout << "0. Log in\n";

std::cout << "1. Sign in\n";

std::cout << "2. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

command = input.GetUInt16(0, 2);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "Your nickname: " << userNickname << "\n\n";

std::cout << "0. Dialogs\n";

std::cout << "1. Interests\n";

std::cout << "2. Reminders\n";

std::cout << "3. Themes\n";

std::cout << "4. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

userCommand = input.GetUInt16(0, 4);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "0. Open dialog\n";

std::cout << "1. Find user\n";

std::cout << "2. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

dialogCommand = input.GetUInt16(0, 2);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\n0. Write a message\n";

std::cout << "1. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

dialogWithUserCommand = input.GetUInt16(0, 1);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\n0. Create new interest\n";

std::cout << "1. Delete interest\n";

std::cout << "2. Find by interest\n";

std::cout << "3. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

interestCommand = input.GetUInt16(0, 3);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\nInput interest number: ";

size\_t interestNumber{};

std::cin >> input;

try

{

interestNumber = input.GetUInt32(0, indexes.size() - 1);

}

catch (StringToUInt32Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\n0. Create new reminder\n";

std::cout << "1. Delete reminder\n";

std::cout << "2. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

reminderCommand = input.GetUInt16(0, 2);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

Entity::Date newDate{};

std::cout << "\nInput day: ";

std::cin >> input;

try

{

newDate.day = input.GetUInt16(1, 31);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

std::cout << "Input month: ";

std::cin >> input;

try

{

newDate.month = input.GetUInt16(1, 12);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

std::cout << "Input year: ";

std::cin >> input;

try

{

newDate.year = input.GetUInt16();

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\nInput reminder number: ";

size\_t reminderNumber{};

std::cin >> input;

try

{

reminderNumber = input.GetUInt32(0, indexes.size() - 1);

}

catch (StringToUInt32Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\n0. Open theme\n";

std::cout << "1. Create theme\n";

std::cout << "2. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

themeCommand = input.GetUInt16(0, 2);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\nInput theme number: ";

size\_t themeNumber{};

std::cin >> input;

try

{

themeNumber = input.GetUInt32(0, m\_storage->GetThemeListSize() - 1);

}

catch (StringToUInt32Exception& exception)

{

std::cout << exception.GetError() << "\n";

std::cin >> input;

continue;

}

...

std::cout << "\n0. Write a message\n";

std::cout << "1. Exit\n";

std::cout << ">>> ";

std::cin >> input;

try

{

themeDialogCommand = input.GetUInt16(0, 1);

}

catch (StringToUInt16Exception& exception)

{

std::cout << exception.GetError() << "\n";

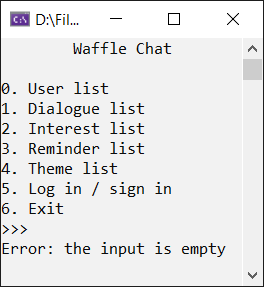
std::cin >> input;

continue;

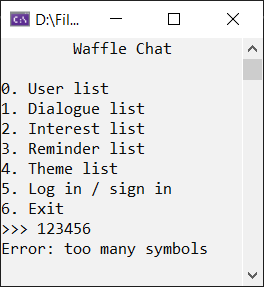
}

...

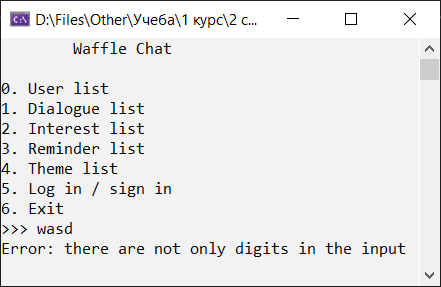
**Демонстрация:**



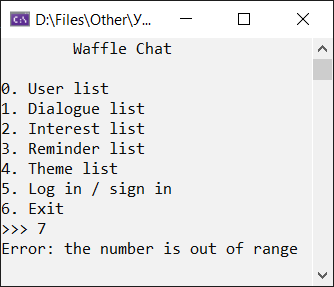
**Рисунок 2.** Ошибка пустого ввода

****

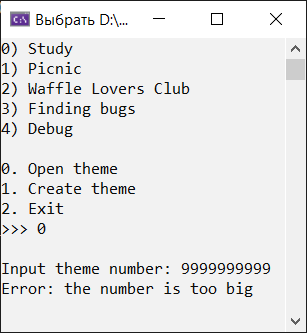
**Рисунок 3.** Ошибка слишком большого количества символов

****

**Рисунок 4.** Ошибка ввода не цифр

****

**Рисунок 5.** Ошбика выхода за границы

****

**Рисунок 6.** Ошибка слишком большого числа

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки создания собственных классов исключений, выбрасывания исключений и их обработки.