**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №1**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: «Обзор языка C++»

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1335 |  | Максимов Ю.Е. |
|  |

|  |  |  |
| --- | --- | --- |
| Преподаватель: |  | Новакова Н. Е. |

Санкт-Петербург

2024

# **1. Цель работы**

Изучение консольного ввода-вывода и обработки исключений на языке C++ с помощью программного продукта компании CLion.

# **2. Анализ задачи**

Необходимо:

1. Написать программу, выводящую на экран приветственное сообщение.
2. Написать программу для деления двух целых чисел с обработкой исключений (введено не целое число или происходит деление на 0).

# **3. Ход выполнения работы**

## **3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, выводящая на экран приветственное сообщение.

### 3.1.1 Пошаговое описание алгоритма

На экран пользователя выводятся приветственное сообщение.

### 3.1.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- std::cin – ожидает следующего нажатия клавиши пользователем;

- main() – служит для запуска программы.

### 3.1.3 Контрольный пример

На рис.3.1.3.1 представлены результаты выполнения программы 1.

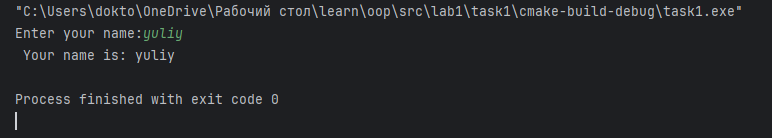


Рис.3.1.3.1 Контрольный пример для упражнения 1

Как видно из рисунка, на экран выведено приветственное сообщение с именем, введенным пользователем.

## **3.2 Упражнение 2**

В ходе выполнения данного упражнения, написанная в предыдущем пункте программа была откомпилирована и запущена с помощью командной строки.

### 3.2.1 Пошаговое описание алгоритма

Программа была откомпилирована, используя следующую команду:

cmake -GNinja -B build -DCAMKE\_BULD\_TYPE=Release && cmake --build build --config Release --parellel && cd build && ./task1

## **3.3 Упражнение 3**

В ходе выполнения данного упражнения проведена отладка пограммы с помощью CLion.

### 3.3.1 Пошаговое описание алгоритма

С помощью курсора была установлена точка останова на строке, где впревые встречается std::cout После чего была запущена отладка с помощью кнопки Debug. Для перехода на следующую строку используется кнопка F10.

## **3.4 Упражнение 4**

В ходе выполнения данного упражнения написана программа, для деления двух целых чисел с обработкой исключений (введено не целое число или происходит деление на 0).

### 3.4.1 Пошаговое описание алгоритма

На экран пользователя выводятся 2 приглашения на ввод целых чисел, после чего происходит их деление, и на экран пользователя выводится сообщение о результате (или вообщение об исключении, если такое возникло в ходе работы программы).

### 3.4.2 Используемые классы и методы

В программе, написанной в данном упражнении, используются следующие методы:

- std::cout – служит для отображения на экране строк и значений переменных, переданных в метод в качестве параметров, с переходом на новую строку;

- std::cin – ожидает следующего нажатия клавиши пользователем [1];

- main() – служит для запуска программы.

### 3.4.3 Контрольный пример

На рис.3.4.3.1 представлены результаты выполнения программы 2.

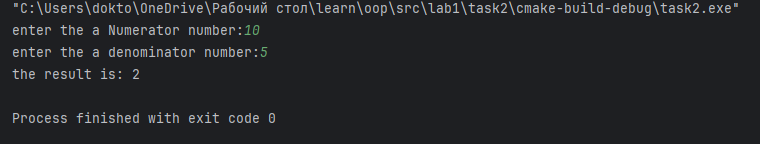
****

Рис.3.4.3.1 Контрольный пример для упражнения 2

Как видно из рисунка, пользователем были введены 2 числа, после чего на экран выведен результат их деления друг на друга.

# **4. Листинг программы**

**Первая программа:**

**handlerString.h**

//

// Created by dokto on 10.06.2024.

//

#pragma once

#include "string"

namespace handler {

class HandlerString {

public:

auto operator() () -> void;

private:

auto input\_name() -> void;

auto output\_name() const -> void;

std::string m\_name;

};

}

**handlerString.cpp**

//

// Created by dokto on 10.06.2024.

//

#include <iostream>

#include "handlerString.h"

/\*\*

\* Executes the input\_name() and output\_name() methods of the HandlerString class.

\*

\* @return void

\*/

auto handler::HandlerString::operator()() -> void {

input\_name();

output\_name();

}

/\*\*

\* Prompts the user to enter their name and stores it in the `m\_name` member variable.

\*

\* @return void

\*/

auto handler::HandlerString::input\_name() -> void {

std::cout << "Enter your name: ";

std::cin >> m\_name;

}

/\*\*

\* Outputs the name stored in the `m\_name` member variable to the console.

\*

\* @throws None

\*/

auto handler::HandlerString::output\_name() const -> void {

std::cout << "Your name is: " << m\_name << std::endl;

}

**main.cpp**

#include "Core/handlerString.h"

int main() {

handler::HandlerString handlerString;

handlerString();

return EXIT\_SUCCESS;

}  
  
**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task1

VERSION 0.0.1

DESCRIPTION "task1 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/Core/handlerString.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/Core/handlerString.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

**Вторая программа:**

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task2

VERSION 0.0.1

DESCRIPTION "task2 for OOP"

LANGUAGES C CXX

)

if(NOT CMAKE\_CXX\_STANDARD)

message(STATUS "[${PROJECT\_NAME}] setting c++ standard to c++23")

set(CMAKE\_CXX\_STANDARD 23)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

endif()

add\_executable( ${PROJECT\_NAME}

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/main.cpp

src/c++/Core/dividingNumbers.cpp

src/c++/Core/dividingNumbers.h

)

target\_include\_directories(${PROJECT\_NAME}

PRIVATE

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++

)

**dividingNumbers.cpp**

//

// Created by dokto on 10.06.2024.

//

#include <iostream>

#include "dividingNumbers.h"

/\*\*

\* Constructor for the DividingNumbers class.

\*

\* Initializes the m\_numbers member variable with a unique pointer to a Numbers object.

\*

\* @return None.

\*/

nambers::DividingNumbers::DividingNumbers():

m\_numbers(std::make\_unique<Numbers>())

{

}

/\*\*

\* Divides the numerator by the denominator and prints the result.

\*

\* @throws std::logic\_error if the denominator is zero.

\*/

void nambers::DividingNumbers::dividingNumbers() const {

addingNumerator();

addingDenominator();

m\_numbers->k = m\_numbers->i / m\_numbers->j;

std::cout << "the result is: " << m\_numbers->k << std::endl;

}

/\*\*

\* Prompts the user to enter a numerator number and stores it in the `m\_numbers` object.

\*

\* @throws None

\*/

void nambers::DividingNumbers::addingNumerator() const {

std::cout << "enter the a Numerator number:";

std::cin >> m\_numbers->temp;

m\_numbers->i = std::stoi(m\_numbers->temp);

}

/\*\*

\* Prompts the user to enter a denominator number and stores it in the `m\_numbers` object.

\*

\* @throws std::logic\_error if the denominator is zero.

\*/

void nambers::DividingNumbers::addingDenominator() const {

std::cout << "enter the a denominator number:";

std::cin >> m\_numbers->temp;

if(m\_numbers->temp == "0") {

throw std::logic\_error("denominator can't be zero");

}

m\_numbers->j = std::stoi(m\_numbers->temp);

}

**dividingNumbers.h**

//

// Created by dokto on 10.06.2024.

//

#pragma once

#include "string"

#include "memory"

namespace nambers {

struct Numbers {

int i;

int j;

int k;

std::string temp;

};

class DividingNumbers {

public:

DividingNumbers();

void dividingNumbers() const;

private:

void addingNumerator() const;

void addingDenominator() const;

std::unique\_ptr<Numbers> m\_numbers;

};

}

**main.cpp**

#include "Core/dividingNumbers.h"

#include "iostream"

/\*\*

\* The main function of the program.

\*

\* This function initializes a `DividingNumbers` object and calls its `dividingNumbers` method.

\* If an exception of type `logic\_error` is thrown during the execution of `dividingNumbers`,

\* the error message is printed to the standard output.

\*

\* @return EXIT\_SUCCESS if the program executes successfully.

\*/

int main()

{

try {

const nambers::DividingNumbers dividingNumbers;

dividingNumbers.dividingNumbers();

}catch (std::logic\_error &error) {

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

}

return EXIT\_SUCCESS;

}

# **5. Полученные результаты**

В ходе выполнения данной лабораторной работы нами были получены следующие результаты:

* в ходе работы программы 1 на экран было выведенно приветственное сообщение вида “Hello , <name>”
* в ходе работы программы 2 были введены 2 целых числа, и на экран был выведен результат их деления.

# **6. Выводы**

В ходе выполнения данной лабораторной работы:

* были изучены простейшие конструкции языка C++;
* были изучины способы консольного ввода-вывода.
* была изучена конструкция try-catch для обработки исключений.