

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.

```
"C:\Users\dokto\OneDrive\Рабочий стол\learn\oop\src\lab1\task1\cmake-build-debug\task1.exe"  
Enter your name:yuliy  
Your name is: yuliy  
  
Process finished with exit code 0  
|
```

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 -DCMAKE_BUILD_TYPE=Release && cmake --  
build build --config Release --parallel && cd build && ./task1
```

3.3 Упражнение 3

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

3.3.1 Пошаговое описание алгоритма

С помощью курсора была установлена точка останова на строке, где впервые 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 вообще об исключении, если такое возникло в ходе работы программы).

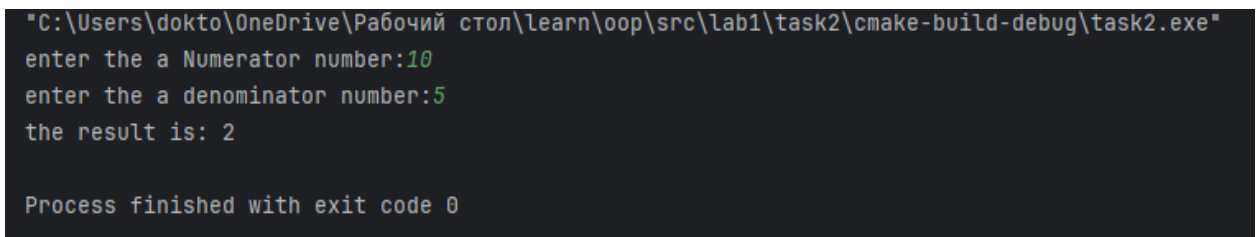
3.4.2 Используемые классы и методы

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.



```
"C:\Users\dokto\OneDrive\Рабочий стол\learn\oop\src\lab1\task2\cmake-build-debug\task2.exe"
enter the a Numerator number:10
enter the a denominator number:5
the result is: 2

Process finished with exit code 0
```

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.
 */
```

```
numbers::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 numbers::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 numbers::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 numbers::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 numbers {
    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 numbers::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 выведено 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 изучены способы консольного ввода-вывода.
- была изучена конструкция try-catch для обработки исключений.