**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №4**

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

Тема: «Использование методов»

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

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

Санкт-Петербург

2024

# **1. Цель работы**

Изучение параметров в методах в языке C++ с помощью программного продукта компании CLion.

# **2. Анализ задачи**

Необходимо:

1. Написать программу, которая сравнивает два целочисленных числа, введенных пользователем, и выводит большее;
2. Дополнить программу из первого упражнения, создать метод, который будет изменять значения праметров (параметры передаются по ссылке);
3. Дополнить программу, создать метод Factorial, который вычисляет факториал (у метода надо определить входной целочисленный параметр);
4. Дополнить программу реккурсивным вычислением факториала.

# **3. Ход выполнения работы**

## **3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, которая сравнивает два целочисленных числа, введенных пользователем, и выводит то, которое больше.

### 3.1.1 Пошаговое описание алгоритма

Создать класс и метод, определить 3 целочисленных переменных x, y и greater.

Присвоить переменной greater большее значение переменных.

На экран пользователя вывести полученное число.

### 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.

-greater() – возвращает значение наибольшего из двух чисел.

### 3.2 Упражнение 2

В ходе выполнения данного упражнения, написанная в предыдущем пункте программа, дополняется методом swap, меняющим местами значения.

## **3.2.1 Пошаговое описание алгоритма**

Создать класс и метод, определить 3 целочисленных переменных x, y и greater.

Присвоить переменной greater большее значение переменных.

Поменять местами значения.

На экран пользователя вывести полученные числа.

### 3.2.2 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.

-greater() – возвращает значение наибольшего из двух чисел.

-swap() – меняет значения двух переменных местами.

### 3.3 Упражнение 3

В ходе выполнения данного упражнения, написанная в предыдущем пункте программа, дополняется методом factorial, вычисляющим фактрориал, числа введеного пользователем.

## **3.3.1 Пошаговое описание алгоритма**

Создать класс и метод, определить 3 целочисленных переменных x, y и greater.

Присвоить переменной greater большее значение переменных.

Поменять местами значения.

Вычислить факториал.

На экран пользователя вывести полученные числа.

### 3.3.2 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.

- greater() – возвращает значение наибольшего из двух чисел.

- swap() – меняет значения двух переменных местами.

- factorial() – вычисляет факториал числа.

### 3.4 Упражнение 4

В ходе выполнения данного упражнения, написанная в предыдущем пункте программа, дополняется методом RecursiveFactorial, вычисляющий факториал реккурсивно.

## **3.4.1 Пошаговое описание алгоритма**

Создать класс и метод, определить 3 целочисленных переменных x, y и greater.

Присвоить переменной greater большее значение переменных.

Поменять местами значения.

Вычислить факториал.

На экран пользователя вывести полученные числа.

### 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.

- greater() – возвращает значение наибольшего из двух чисел.

- swap() – меняет значения двух переменных местами.

- factorial() – вычисляет факториал числа.

- recursiveFactorial() – рекурсивно вычисляет факториал.

### 3.4.3 Контрольный пример

На рис.3.1 представлены результаты выполнения программы.

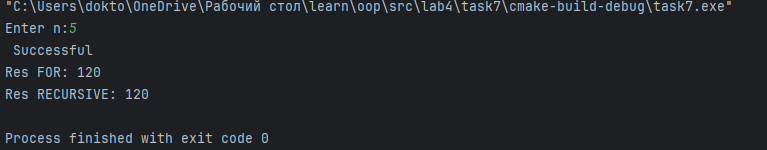
****

Рис.3.1 Контрольный пример для программы

Как видно из рисунка, на экран выведены значения, большее из них и значения факториала, вычесленного двумя способами.

# **4. Листинг программы**

**Первая программа:**

**core.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "iostream"

#include "string"

#include "core.h"

#include "utils/utils.h"

namespace {

/\*\*

\* Reads an integer value from the user input and assigns it to the given variable.

\*

\* @param var a reference to an integer variable to store the input value

\* @param str a constant reference to a string representing the prompt message

\*

\* @return void

\*

\* @throws None

\*/

auto input(int &var, const std::string &str) -> void {

std::cout << str;

std::cin >> var;

}

/\*\*

\* Outputs the given integer variable and string to the console.

\*

\* @param var the integer variable to output

\* @param str the string to output before the integer variable

\*

\* @return void

\*/

auto output(const int var, const std::string &str) -> void {

std::cout << str << var << std::endl;

}

}

/\*\*

\* Processes two integers by prompting the user to enter them,

\* finding the greater number, and swapping the two.

\*

\* @return void

\*

\* @throws None

\*/

auto root::Core::procces() -> void {

int x, y;

input(x, "Enter first number: ");

input(y, "Enter second number: ");

const int greater = utils::Utils::greater<int>(x, y);

output(greater, "Greater number: ");

output(x, "First number: ");

output(y, "Second number: ");

std::cout << "SWAP" << std::endl;

utils::Utils::swap<int>(x, y);

output(x, "First number: ");

output(y, "Second number: ");

}

**core.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

namespace root {

class Core {

public:

static auto procces() -> void;

};

}

**utils.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

namespace utils {

class Utils {

public:

/\*\*

\* Returns the greater of two values.

\*

\* @tparam T the type of the values being compared

\*

\* @param var1 the first value to compare

\* @param var2 the second value to compare

\*

\* @return the greater of var1 and var2

\*

\* @throws None

\*/

template <typename T>

static auto greater(T var1, T var2) -> T;

/\*\*

\* Swaps the values of two variables.

\*

\* @tparam T the type of the variables

\*

\* @param var1 the first variable

\* @param var2 the second variable

\*

\* @return void

\*

\* @throws None

\*/

template <typename T>

static auto swap(T &var1, T &var2) noexcept -> void;

};

template<typename T>

auto Utils::greater(T var1, T var2) -> T {

return var1 > var2 ? var1 : var2;

}

template<typename T>

auto Utils::swap(T &var1, T &var2) noexcept -> void {

std::swap(var1,var1);

}

}

**main.cpp**

#include "core/core.h"

int main() {

root::Core::procces();

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task6

VERSION 0.0.1

DESCRIPTION "task6 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/Core.cpp

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

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

)

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

PRIVATE

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

)

**Вторая программа:**

**core.cpp**

//

// Created by Юлий Максимов on 11.06.2024.

//

#include "iostream"

#include "string"

#include "core.h"

#include "utils/utils.h"

namespace {

/\*\*

\* Reads an integer from the standard input and assigns it to the variable `var`.

\*

\* @param var reference to an integer variable where the input will be stored

\* @param str a string that will be printed to the standard output before reading the input

\*

\* @throws None

\*/

auto input(int &var, const std::string &str) -> void {

std::cout << str;

std::cin >> var;

}

/\*\*

\* Outputs the given integer variable and string to the standard output.

\*

\* @param var the integer variable to be outputted

\* @param str the string to be outputted before the integer variable

\*

\* @return void

\*

\* @throws None

\*/

auto output(const int var, const std::string &str) -> void {

std::cout << str << var << std::endl;

}

}

/\*\*

\* Processes the input value and calculates the factorial using both iterative and recursive methods.

\*

\* @return void

\*

\* @throws None

\*/

auto root::Core::procces() -> void {

int n, res = 1;

input(n, "Enter n: ");

if(utils::Utils::factorialFor(n, res)) {

std::cout << "Successful" << std::endl;

}else {

std::cout << "FAIL" << std::endl;

}

output(res, "Res FOR: ");

output(utils::Utils::factorualRecursive(n), "Res RECURSIVE: ");

}

**core.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#pragma once

namespace root {

class Core {

public:

static auto procces() -> void;

};

}

**utils.h**

//

// Created by Юлий Максимов on 11.06.2024.

//

#ifndef GREATER\_H

#define GREATER\_H

namespace utils {

class Utils {

public:

/\*\*

\* Returns the greater of two values.

\*

\* @tparam T the type of the values being compared

\*

\* @param var1 the first value to compare

\* @param var2 the second value to compare

\*

\* @return the greater of var1 and var2

\*

\* @throws None

\*/

template <typename T>

static auto greater(T var1, T var2) -> T;

/\*\*

\* Swaps the values of two variables.

\*

\* @tparam T the type of the variables

\*

\* @param var1 the first variable

\* @param var2 the second variable

\*

\* @return void

\*

\* @throws None

\*/

template <typename T>

static auto swap(T &var1, T &var2) noexcept -> void;

/\*\*

\* Calculates the factorial of a given number using a for loop.

\*

\* @param n The number for which to calculate the factorial.

\* @param res The reference to the variable that will store the result.

\* @return True if the factorial calculation was successful, false otherwise.

\*/

template <typename T>

static auto factorialFor(T n, T & res) -> bool;

/\*\*

\* Calculates the factorial of a given number using recursion.

\*

\* @tparam T the type of the number

\* @param n the number for which to calculate the factorial

\* @return the factorial of n

\*/

template <typename T>

static auto factorualRecursive(T n) -> int;

};

template<typename T>

auto Utils::greater(T var1, T var2) -> T {

return var1 > var2 ? var1 : var2;

}

template<typename T>

auto Utils::swap(T &var1, T &var2) noexcept -> void {

std::swap(var1,var1);

}

template<typename T>

auto Utils::factorialFor(T n, T &res) -> bool {

for (int i = 1; i <= n; i++) {

res = res \* i;

}

return true;

}

template<typename T>

auto Utils::factorualRecursive(T n) -> int {

if (n==0) return 1;

return n\*factorualRecursive(n-1);

}

}

#endif //GREATER\_H

**main.cpp**

#include "core/core.h"

int main() {

root::Core::procces();

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task7

VERSION 0.0.1

DESCRIPTION "task6 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/Core.cpp

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

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

)

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

PRIVATE

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

)

# **5. Полученные результаты**

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

* в ходе работы программы были созданы методы и классы, с помощью которых вычислялось большее из значений, обмен значений местами и вычисление факториала.

# **6. Выводы**

В ходе выполнения данной лабораторной работы:

* были изучены параметры методов в языке C++;
* была изучена передача параметров по ссылке в языке C++.
* были изучены реккурсивные функции в языке C++.