**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №3**

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

Тема: «Использование базовых языковых конструкций»

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

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

Санкт-Петербург

2024

# **Цель работы**

Изучение базовых конструкций языка на языке C++ с помощью программного продукта компании CLion.

# **Анализ задачи**

Необходимо:

1. Написать программу, конвертирующую число, соответствующее дню года, в пару «месяц – день».
2. Добавить исключения в первую программу.
3. Дополнить программу високосным годом.

# **Ход выполнения работы**

**3.1 Упражнение 1**

В ходе выполнения данного упражнения написана программа, выводящая на экран пару «месяц-день» из введенного пользователем числа.

### **3.1.1 Пошаговое описание алгоритма**

Пользователь вводит целочисленное значение.

Преобразование числа в пару «месяц-день».

На экран пользователя выводится пара «месяц-день».

### **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.

- enum class – перечислимый тип.

**3.1 Упражнение 2**

В ходе выполнения данного упражнения программа из первого упражнения дополнена исключениями (день не меньше 1 и не больше 365).

### **3.2.1 Пошаговое описание алгоритма**

Пользователь вводит целочисленное значение.

Преобразование числа в пару «месяц-день», проверка исключений.

На экран пользователя выводится пара «месяц-день».

### **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;

- main() – служит для запуска программы.

- enum class – перечислимый тип.

### **3.3 Упражнение 3**

В ходе выполнения данного упражнения программа дополняется введением года и високосным годом.

**3.3.1 Пошаговое описание алгоритма**

Пользователь вводит целочисленное значение.

Преобразование числа в пару «месяц-день», проверка исключений и проверка високосного года.

На экран пользователя выводится пара «месяц-день».

### **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.

- struct – структура;

- enum class – перечислимый тип.

### **3.3.3 Контрольный пример**

На рис. 3.2.3.1 представлены результаты выполнения программы.

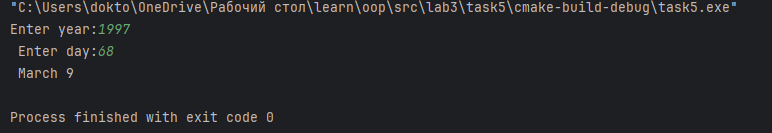
****

Рис.3.2.3.1 Контрольный пример для программы

Как видно из рисунка, пользователь ввел год и число, на выходе выведена пара «месяц-день».

**4. Листинг программы**

**whatDay.cpp**

//

// Created by dokto on 10.06.2024.

//

#include <iostream>

#include "WhatDay.h"

namespace {

constexpr auto MIN\_DAY = 1;

auto MAX\_DAY = 365;

}

/\*\*

\* Executes the main process of the WhatDay class by calling the inputYear, inputDay, calculateMonthDay, and outputMonthDay functions in sequence.

\*

\* @throws None

\*/

auto day::WhatDay::process() -> void {

this->inputYear();

this->inputDay();

this->calculateMonthDay();

this->outputMonthDay();

}

/\*\*

\* Reads a day from the user input and checks if it is within the valid range.

\*

\* @throws std::logic\_error if the day is outside the valid range

\*/

auto day::WhatDay::inputDay() -> void {

std::cout << "Enter day: ";

std::cin >> m\_day;

if(m\_day < MIN\_DAY || m\_day > MAX\_DAY) {

throw std::logic\_error( "Wrong day!" );

}

}

/\*\*

\* Prompts the user to enter a year and checks if it is a leap year.

\* If the year is not a leap year, sets `m\_isLeapYear` to false.

\* If the year is a leap year, sets `m\_isLeapYear` to true and updates `MAX\_DAY` to 366.

\*

\* @throws None

\*/

auto day::WhatDay::inputYear() -> void {

std::cout << "Enter year: ";

std::cin >> year;

if(year % 4 != 0 || (year % 100 == 0 && year % 400 != 0)) {

m\_isLeapYear = false;

} else {

m\_isLeapYear = true;

}

if(m\_isLeapYear) {

MAX\_DAY = 366;

}

}

/\*\*

\* Outputs the month and day to the console.

\*

\* @throws None

\*/

auto day::WhatDay::outputMonthDay() const -> void {

std::cout << m\_monthDay << std::endl;

}

/\*\*

\* Checks if the given day is greater than the month and updates the day accordingly.

\* If the day is greater than the month, it subtracts the month from the day and returns false.

\* Otherwise, it constructs the month and day string and returns true.

\*

\* @param temp the day to be checked

\* @param month the maximum day of the month

\* @param monthName the name of the month

\*

\* @return true if the day is less than or equal to the month, false otherwise

\*

\* @throws None

\*/

auto day::WhatDay::checkDay(int & temp, MonthName month,const std::string &monthName) -> bool {

if(temp > static\_cast<int>(month)) {

temp -= static\_cast<int>(month);

return false;

}

m\_monthDay = monthName + " " + std::to\_string(temp);

return true;

}

/\*\*

\* Calculates the month and day based on the given day of the year.

\*

\* @return void

\*

\* @throws None

\*/

auto day::WhatDay::calculateMonthDay() -> void {

int temp = m\_day;

if(checkDay(temp, MonthName::January, "January")) return;

if(m\_isLeapYear) {

if(checkDay(temp, MonthName::FebruaryLeap, "February")) return;

}else {

if(checkDay(temp, MonthName::February, "February")) return;

}

if(checkDay(temp, MonthName::March, "March")) return;

if(checkDay(temp, MonthName::April, "April")) return;

if(checkDay(temp, MonthName::May, "May")) return;

if(checkDay(temp, MonthName::June, "June")) return;

if(checkDay(temp, MonthName::July, "July")) return;

if(checkDay(temp, MonthName::August, "August")) return;

if(checkDay(temp, MonthName::September, "September")) return;

if(checkDay(temp, MonthName::October, "October")) return;

if(checkDay(temp, MonthName::November, "November")) return;

checkDay(temp, MonthName::December, "December");

}

**whatDay.h**

//

// Created by dokto on 10.06.2024.

//

#pragma once

#include <string>

namespace day{

enum class MonthName

{

January = 31,

February = 28,

FebruaryLeap = 29,

March = 31,

April = 30,

May = 31,

June = 30,

July = 31,

August = 31,

September = 30,

October = 31,

November = 31,

December = 31

};

class WhatDay {

public:

void process();

private:

auto inputDay() -> void;

auto inputYear() -> void;

auto outputMonthDay() const -> void;

auto checkDay(int & temp, MonthName month, const std::string &monthName) -> bool;

auto calculateMonthDay() -> void;

int m\_day{};

int year{};

std::string m\_monthDay{};

bool m\_isLeapYear{false};

};

}

**main.cpp**

#include "core/whatDay.h"

#include <iostream>

/\*\*

\* The main function of the C++ program.

\*

\* @return The exit status of the program.

\*

\* @throws std::logic\_error if an error occurs during the execution of the program.

\*/

int main()

{

try {

day::WhatDay whatDay;

whatDay.process();

}catch (const std::logic\_error &e) {

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

}

return EXIT\_SUCCESS;

}  
  
**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.15.0)

include\_guard(GLOBAL)

project(task5

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/whatDay.cpp

${CMAKE\_CURRENT\_SOURCE\_DIR}/src/c++/core/whatDay.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 целочисленное значение, введенное пользователем, преобразуется в пару «месяц-день» с учетом исключений, также в третьем упражнении добавлен учет года.

# **6. Выводы**

В ходе выполнения данной лабораторной работы:

* Были изучены базовые языковые конструкции языка C++.