МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**Лабораторная работа №7**

по дисциплине

«Наследование»

**Выполнил:**

Ермолаев Кирилл Александрович

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

**Тема:** Основы объектно-ориентированного программирования на ЯП С++.

**Цель работы:** Наследование.

**Ход работы:**

**Вариант 7** SymbString DecString operator +(T&, T&)

Написать программу, демонстрирующую работу с объектами двух типов, T1 и T2, для чего создать систему соответствующих классов. Каждый объект должен иметь идентификатор (в виде произвольной строки символов) и одно или несколько полей для хранения состояния (текущего значения) объекта.

Клиенту (функции main) должны быть доступны следующие основные операции (методы): создать объект, удалить объект, показать значение объекта и прочие дополнительные операции (зависят от варианта). Операции по созданию и удалению объектов инкапсулировать в классе Factory. Предусмотреть меню, позволяющее продемонстрировать заданные операции. При необходимости в разрабатываемые классы добавляются дополнительные методы (например, конструктор копирования, операция присваивания и т.п.) для обеспечения надлежащего функционирования этих классов.

//DecString.h:

#ifndef DECSTRING\_H

#define DECSTRING\_H

#include <string>

class DecString {

public:

    std::string identifier;

    int value;

    DecString(const std::string& id, int val);

    void showValue() const;

};

#endif // DECSTRING\_H

//DecString.cpp:

#include "DecString.h"

#include <iostream>

DecString::DecString(const std::string& id, int val) : identifier(id), value(val) {}

void DecString::showValue() const {

    std::cout << "DecString " << identifier << " has value: " << value << std::endl;

}

//SymbString.h:

#ifndef SYMBSTRING\_H

#define SYMBSTRING\_H

#include <string>

class SymbString {

public:

    std::string identifier;

    SymbString(const std::string& id);

    void showIdentifier() const;

};

#endif // SYMBSTRING\_H

//SymbString.cpp:

#include "SymbString.h"

#include <iostream>

SymbString::SymbString(const std::string& id) : identifier(id) {}

void SymbString::showIdentifier() const {

    std::cout << "SymbString identifier: " << identifier << std::endl;

}

//main.cpp:

#include "classes.h"

#include "DecString.h"

#include "SymbString.h"

int main() {

    T1\* t1 = Factory::createT1("T1\_Object");

    T2\* t2 = Factory::createT2("T2\_Object", 10);

    DecString\* decString = Factory::createDecString("DecString\_Object", 25);

    SymbString\* symbString = Factory::createSymbString("SymbString\_Object");

    // Демонстрация операции создания объектов

    std::cout << "Created objects with identifiers: " << t1->identifier << ", " << t2->identifier << ", "

              << decString->identifier << ", " << symbString->identifier << std::endl;

    // Демонстрация операции сложения

    T2 result = (\*t1) + (\*t2);

    std::cout << "Result of T1 + T2: " << result.identifier << " with value: " << result.value << std::endl;

    // Демонстрация методов DecString и SymbString

    decString->showValue();

    symbString->showIdentifier();

    // Освобождение памяти

    Factory::deleteObject(t1);

    Factory::deleteObject(t2);

    Factory::deleteObject(decString);

    Factory::deleteObject(symbString);

    return 0;

}

//unitTest.cpp

// Тестирование DecString

TEST(DecStringTest, ShowValue) {

    DecString\* decString = Factory::createDecString("DecString\_Object", 25);

    // Проверка идентификатора

    EXPECT\_EQ(decString->identifier, "DecString\_Object");

    // Проверка значения

    testing::internal::CaptureStdout();

    decString->showValue();

    std::string output = testing::internal::GetCapturedStdout();

    EXPECT\_EQ(output, "DecString DecString\_Object has value: 25\n");

    // Освобождение памяти

    Factory::deleteObject(decString);

}

// Тестирование SymbString

TEST(SymbStringTest, ShowIdentifier) {

    SymbString\* symbString = Factory::createSymbString("SymbString\_Object");

    // Проверка идентификатора

    EXPECT\_EQ(symbString->identifier, "SymbString\_Object");

    // Проверка вывода

    testing::internal::CaptureStdout();

    symbString->showIdentifier();

    std::string output = testing::internal::GetCapturedStdout();

    EXPECT\_EQ(output, "SymbString identifier: SymbString\_Object\n");

}

<https://github.com/MoeTomatoki/OOP>

**Вывод:** изучил базовые понятия в использование паттернов проектирования; технология проектирования программ с учетом будущих изменений. А также более подробно изучил тему наследования классов и полиморфизм.