МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**Лабораторная работа №3**

по дисциплине

«Объектно-ориентированное программирование»

**Выполнил:**

Пантелеев Никита Андреевич

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

Тема: Шаблоны классов. Обработка исключительных ситуаций.

Цель работы: изучить шаблоны классов и обработчик исключительных ситуаций и научится реализовать их.

Выполнение работы:

Вариант -16

Листинг:

#include <iostream>

#include <string>

#include "Set.h"

#include "Test.h"

#include <Windows.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

// Использование шаблона Set для целых чисел

Set<int> intSet;

int inputInt;

std::cout << "Введите целые числа (введите -1 чтобы закончить):\n";

while (true) {

std::cin >> inputInt;

if (inputInt == -1) break;

intSet.insert(inputInt);

}

intSet.print();

// Использование шаблона Set для чисел с плавающей точкой

Set<double> doubleSet;

double inputDouble;

std::cout << "Введите числа с плавающей точкой (введите -1 чтобы закончить):\n";

while (true) {

std::cin >> inputDouble;

if (inputDouble == -1) break;

doubleSet.insert(inputDouble);

}

doubleSet.print();

// Использование шаблона Set для строк

Set<std::string> stringSet;

std::string inputString;

std::cout << "Введите строки (введите \"-1\" чтобы закончить):\n";

while (true) {

std::cin >> inputString;

if (inputString == "-1") break;

stringSet.insert(inputString);

}

stringSet.print();

/// Генерация исключения при доступе по индексу в Vect

Vect<int> intVect;

intVect.push\_back(1);

intVect.push\_back(2);

intVect.push\_back(3);

try {

int value = intVect[3];

std::cout << "Значение по индексу 3: " << value << std::endl;

}

catch (const std::out\_of\_range& e) {

std::cout << "Перехваченное исключение: " << e.what() << std::endl;

}

testIntegerSet();

testDoubleSet();

testStringSet();

return 0;

}

#ifndef VECT\_H

#define VECT\_H

#include <iostream>

#include <algorithm>

#include <stdexcept>

template <class T>

class Vect {

public:

Vect();

Vect(const Vect& other);

~Vect();

void push\_back(const T& value);

T& operator[](int index) const;

T\* begin() const { return data; }

T\* end() const { return data + length; }

int size() const;

void clear();

private:

T\* data;

int capacity;

int length;

};

#include "Vect.cpp"

#endif // VECT\_H

#ifndef VECT\_CPP

#define VECT\_CPP

#include "Vect.h"

template <class T>

Vect<T>::Vect() : data(nullptr), capacity(0), length(0) {}

template <class T>

Vect<T>::Vect(const Vect<T>& other) : data(nullptr), capacity(0), length(0) {

capacity = other.capacity;

length = other.length;

data = new T[capacity];

std::copy(other.data, other.data + length, data);

}

template <class T>

Vect<T>::~Vect() {

delete[] data;

}

template <class T>

void Vect<T>::push\_back(const T& value) {

if (length == capacity) {

int newCapacity = std::max(1, 2 \* capacity);

T\* newData = new T[newCapacity];

std::copy(data, data + length, newData);

delete[] data;

data = newData;

capacity = newCapacity;

}

data[length++] = value;

}

template <class T>

T& Vect<T>::operator[](int index) const {

if (index < 0 || index >= length) {

throw std::out\_of\_range("Индекс вне диапазона");

}

return data[index];

}

template <class T>

int Vect<T>::size() const {

return length;

}

template <class T>

void Vect<T>::clear() {

delete[] data;

data = nullptr;

capacity = 0;

length = 0;

}

#endif // VECT\_CPP

#ifndef SET\_H

#define SET\_H

#include "Vect.h"

template <class T>

class Set {

public:

Set();

~Set();

void insert(const T& value);

bool contains(const T& value) const;

void print() const;

private:

Vect<T> elements;

};

#include "Set.cpp"

#endif // SET\_H

#ifndef SET\_CPP

#define SET\_CPP

#include "Set.h"

template <class T>

Set<T>::Set() {}

template <class T>

Set<T>::~Set() {}

template <class T>

void Set<T>::insert(const T& value) {

if (!contains(value)) {

elements.push\_back(value);

std::sort(elements.begin(), elements.end());

}

};

template <class T>

bool Set<T>::contains(const T& value) const {

return std::binary\_search(elements.begin(), elements.end(), value);

}

template <class T>

void Set<T>::print() const {

for (const T& element : elements) {

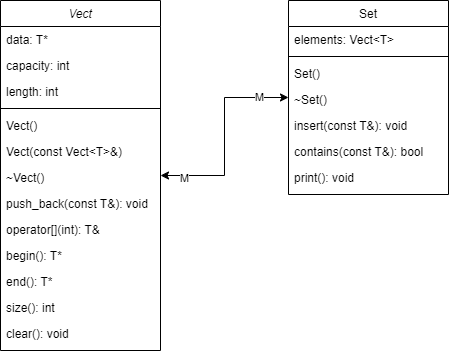
std::cout << element << " ";

}

std::cout << std::endl;

}

#endif // SET\_CPP



Ссылка на полностью сделанные задания на github: https://github.com/Filin546/OOP

Вывод: изучил шаблон классов и обработчик исключительных ситуаций и научил реализовать их.