**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По лабораторной работе №18.7 на тему

«Шаблоны классов»

Вариант №11

Выполнил студент группы РИС-20-1б

Шумилов Лев Сергеевич

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –научиться работать с одним из механизмов языка программирования C++ - с шаблонами классов и шаблонными алгоритмами, которые используют возможности перегрузки операторов для обобщения алгоритмов;

**Постановка задачи**

Задача – реализовать на языке программирования C++ программу с шаблонным классом и продемонстрировать его работы с разными типами данных.

**Исходные данные для варианта №11:**

* Класс-контейнер, список из лабораторной работы №18.6;
* Переделать реализацию списка под шаблон;

**Анализ задачи**

Для решения задачи были использованы следующие средства:

1. Язык программирования C++ (Microsoft Visual C++)
2. Текстовый редактор Microsoft Visual Studio Code
3. Система контроля версий Git

Одной из особенностей работы шаблонов, объявление и реализация класса лежит в ОДНОМ заголовочном файле.

Шаблон класса задаётся следующим способом:

template <class T> //где T имя любого типа данных

class Abc

{

private:

T \_name;

public:

void setT(T val) { \_name = val; }

}

Таким образом реализация списка с учётом шаблонного подхода выглядит следующим образом, весь код, представленный ниже будет являться ПОЛНЫМ исходным кодом программы:

#pragma once

#include <iostream>

using namespace std;

template <class T>

class ListInt

{

private:

T\* \_data;

int \_length;

public:

ListInt(int length, T fillValue);

ListInt(const ListInt<T>& copied);

~ListInt();

T& operator [] (int index);

ListInt& operator = (const ListInt<T>& assigned);

ListInt operator + (const ListInt<T>& adder);

int Length() { return \_length; }

};

template<class T>

ListInt<T>::ListInt(int length, T fillValue)

{

\_length = length;

\_data = new T[\_length];

for (int i = 0; i < \_length; i++)

\_data[i] = fillValue;

}

template<class T>

ListInt<T>::ListInt(const ListInt<T>& copied)

{

\_length = copied.\_length;

\_data = new T[\_length];

for (int i = 0; i < \_length; i++)

\_data[i] = copied.\_data[i];

}

template<class T>

ListInt<T>::~ListInt()

{

delete[] \_data;

\_data = nullptr;

}

template <class T>

T& ListInt<T>::operator[](int index)

{

try

{

if (index < 0)

throw - 1;

if (index >= \_length)

throw 1;

return \_data[index];

}

catch (int code)

{

switch (code)

{

case -1:

cerr << "Error: index is out of range => less than 0;\n";

break;

case 1:

cerr << "Error: index is out of range => more than length;\n";

break;

}

return \_data[0];

}

}

template <class T>

ListInt<T>& ListInt<T>::operator=(const ListInt<T>& assigned)

{

\_length = assigned.\_length;

delete[] \_data;

\_data = new T[\_length];

for (int i = 0; i < \_length; i++)

\_data[i] = assigned.\_data[i];

return \*this;

}

template <class T>

ListInt<T> ListInt<T>::operator+(const ListInt<T>& adder)

{

try

{

if (\_length != adder.\_length)

throw - 1;

ListInt<T> temporarilyList(\_length, 0);

for (int i = 0; i < \_length; i++)

temporarilyList[i] = \_data[i] + adder.\_data[i];

return temporarilyList;

}

catch (int code)

{

if (code == -1)

cerr << "Error: list length are not the same;\n";

return \*this;

}

}

Основная программа, которая прогоняет данный шаблонный список через различные типы данных:

#include <iostream>

#include "Header files\ListInt.h"

using namespace std;

template <class T>

void PrintList(ListInt<T>& list)

{

for (int i = 0; i < list.Length(); i++)

cout << list[i] << ' ';

cout << endl;

}

int main()

{

ListInt<float> flList(10, 5.12);

ListInt<int> inList(23, 2);

ListInt<float> fl2List(10, 4.8);

PrintList(flList);

PrintList(inList);

PrintList(fl2List);

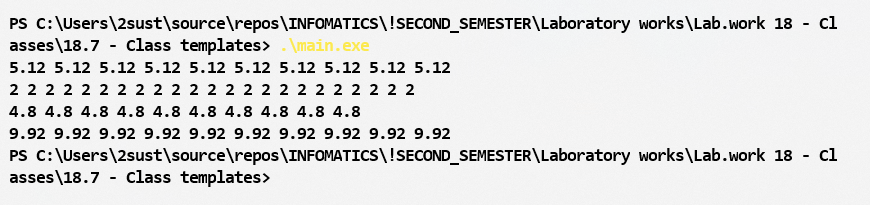
ListInt<float> summList = flList + fl2List;

PrintList(summList);

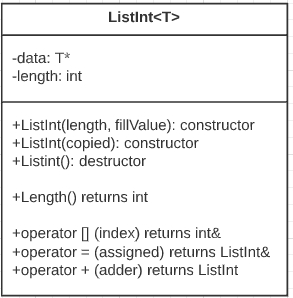
return 0;

}

**Скриншот выполненной программы:**



**UML-диаграмма: программы:**



**Ответы на вопросы:**

**В чем смысл использования шаблонов?**

С помощью шаблона функций можно отделить алгоритм от конкретных типов данных, передавая тип в качестве параметра.

**Каковы синтаксис/семантика шаблонов функций?**

template <class параметр> заголовокФункции

{тело функции}

**Каковы синтаксис/семантика шаблонов классов?**

template <class параметр> class имяКласса

{…};

**Что такое параметры шаблона функции?**

Типы данных, передаваемые в функцию

**Перечислите основные свойства параметров шаблона функции.**

Шаблоны методов (функций) не могут быть виртуальными.

**Как записывать параметр шаблона?**

template <class T>

**Можно ли перегружать параметризованные функции?**

Да.

**Перечислите основные свойства параметризованных классов.**

Шаблоны классов могут содержать статические элементы, дружественные функции и классы.

Шаблоны могут быть производными как от шаблонов, так и от обычных классов, а также являться базовыми и для шаблонов, и для обычных классов.

**Все ли компонентные функции параметризованного класса являются параметризованными?**

Да.

**Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?**

Нет.

**Могут ли шаблоны классов содержать виртуальные компонентные функции?**

Нет.

**Как определяются компонентные функции параметризованных классов вне определения шаблона класса?**

Такое определение невозможно.

**Что такое инстанцирование шаблона?**

Процесс генерации компилятором определения конкретного класса по шаблону класса и аргументам шаблона.

**На каком этапе происходит генерирование определения класса по шаблону?**

В момент создания экземпляра класса.