Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Лабораторная работа №7 по ООП

Семестр: 2

Выполнил студент ИВТ-23-2б:

Синицын Владислав Сергеевич

(дата, подпись)

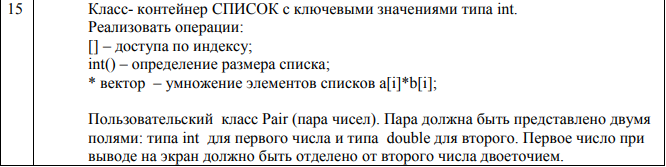
Проверила:

Полякова Ольга Андреевна

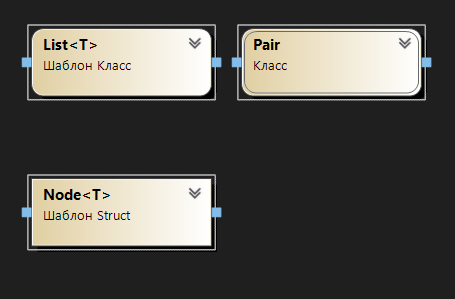
(дата, подпись)

Пермь 2024

**Задание**



**UML-Диаграмма**

****

**Программное решение**

**Pair.h**

#pragma once

#include <iostream>

using namespace std;

class Pair {

int first;

double second;

public:

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

Pair(int, double);

Pair();

Pair(const Pair&);

~Pair();

Pair& operator=(const Pair&);

Pair operator-(const Pair&);

Pair& operator+(int);

Pair& operator+(double);

Pair& operator++();

Pair operator++(int);

int GetFirst() const;

double GetSecond() const;

void SetFirst(int m\_first);

void SetSecond(double m\_second);

};

**Pair.cpp**

#include "Pair.h"

Pair::Pair(int first, double second) {//конструктор с параметрами

this->first = first;

this->second = second;

}

istream& operator>>(istream& in, Pair& m\_pair) {//перегрузка оператора ввода

in >> m\_pair.first;

in >> m\_pair.second;

return in;

}

ostream& operator<<(ostream& out, const Pair& m\_pair) {//перегрузка оператора вывода

return (out << m\_pair.first << " : " << m\_pair.second);

}

Pair::Pair() {//конструктор без параметров

first = 0;

second = 0;

}

Pair::~Pair() {//деструктор

}

Pair::Pair(const Pair& m\_pair) {//конструктор копирования

first = m\_pair.first;

second = m\_pair.second;

}

Pair& Pair::operator=(const Pair& m\_pair) {//перегрузка оператора присваивания

if (&m\_pair != this) {

first = m\_pair.first;

second = m\_pair.second;

}

return \*this;

}

Pair Pair::operator-(const Pair& m\_pair) {//перегрузка оператора вычитания

Pair result(first - m\_pair.first, second - m\_pair.second);

return result;

}

Pair& Pair::operator+(int first) {//перегрузка оператора сложения - целый аргумент

this->first += first;

return \*this;

}

Pair& Pair::operator+(double second) {//перегрузка оператора сложения - дробный аргумент

this->second += second;

return \*this;

}

Pair& Pair::operator++() {//перегрузка оператора ++ префикс

++first;

++second;

return \*this;

}

Pair Pair::operator ++(int) {//перегрузка оператора ++ постфикс

Pair temp = \*this;

this->first++;

this->second++;

return temp;//сначала нужно вывести без изменений

}

int Pair::GetFirst() const{

return first;

}

double Pair::GetSecond() const{

return second;

}

void Pair::SetFirst(int m\_first) {

first = m\_first;

}

void Pair::SetSecond(double m\_second) {

second = m\_second;

}

**List.h**

#pragma once

#include <string>

#include <iostream>

using namespace std;

template <class T>

struct Node {

T key;

Node<T>\* next;

};

template <class T>

class List {

public:

List() {};

List(int count) {

n = count;

head = new Node<T>;

head->key = 0;

lastNd = head;

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

current = new Node<T>;

current->key = 0;

lastNd->next = current;

lastNd = current;

}

lastNd->next = NULL;

};

~List() {

lastNd = head;

while (lastNd != NULL) {

current = lastNd->next;

delete lastNd;

lastNd = current;

}

n = 0;

};

List& operator = (List<T>& l) {

if (this != &l) {

if (this != 0) {

lastNd = head;

while (lastNd != NULL) {

current = lastNd->next;

delete lastNd;

lastNd = current;

}

n = 0;

}

lastNd = head = new Node<T>;

l.lastNd = l.head->next;/////

lastNd->key = l.head->key;

while (l.lastNd != NULL) {

lastNd->next = new Node<T>;

lastNd->next->key = l.lastNd->key;

l.lastNd = l.lastNd->next;

lastNd = lastNd->next;

}

lastNd->next = NULL;

n = l.n;

}

return \*this;

};

int& operator[] (int index) {

if (index < n) {

lastNd = head;

for (int i = 0; i < index; i++) {

lastNd = lastNd->next;

}

return lastNd->key;

}

else {

cout << "Ошибка: Индекс за пределами диапазона!";

}

};

int operator () () {

return n;

};

friend ostream& operator << (ostream& out, List<T>& l) {

if (l.n) {

l.lastNd = l.head;

while (l.lastNd != NULL) {

out << l.lastNd->key << ' ';

l.lastNd = l.lastNd->next;

}

}

else {

out << "Список пуст!";

}

return out;

};

friend istream& operator >> (istream& in, List<T>& l) {

l.lastNd = l.head;

while (l.lastNd != NULL) {

in >> l.lastNd->key;

l.lastNd = l.lastNd->next;

}

return in;

};

void operator << (T t) {

if (head == NULL) {

head = new Node<T>;

head->key = t;

head->next = NULL;

}

else {

lastNd = head;

while (lastNd->next != NULL) {

lastNd = lastNd->next;

}

lastNd->next = new Node<T>;

lastNd->next->key = t;

lastNd = lastNd->next;

lastNd->next = NULL;

}

n += 1;

};

private:

Node<T>\* lastNd, \* current, \* head;

int n = 0;

};

**OOPLab7.cpp**

#include <iostream>

using namespace std;

#include "List.h"

#include "Pair.h"

int main() {

List <int> l;

l << 5;

l << 4;

cout << "List<int>:\n";

cout << l << '\n';

List <Pair> p;

Pair c;

c.SetFirst(9);

c.SetSecond(9.4);

p << c;

cout << "List<Pair>:\n";

cout << p << '\n';

return 0;

}



**Контрольные вопросы**

1. В чем смысл использования шаблонов?

Возможность использовать один класс или функцию для различных типов данных

1. Каковы синтаксис/семантика шаблонов функций?

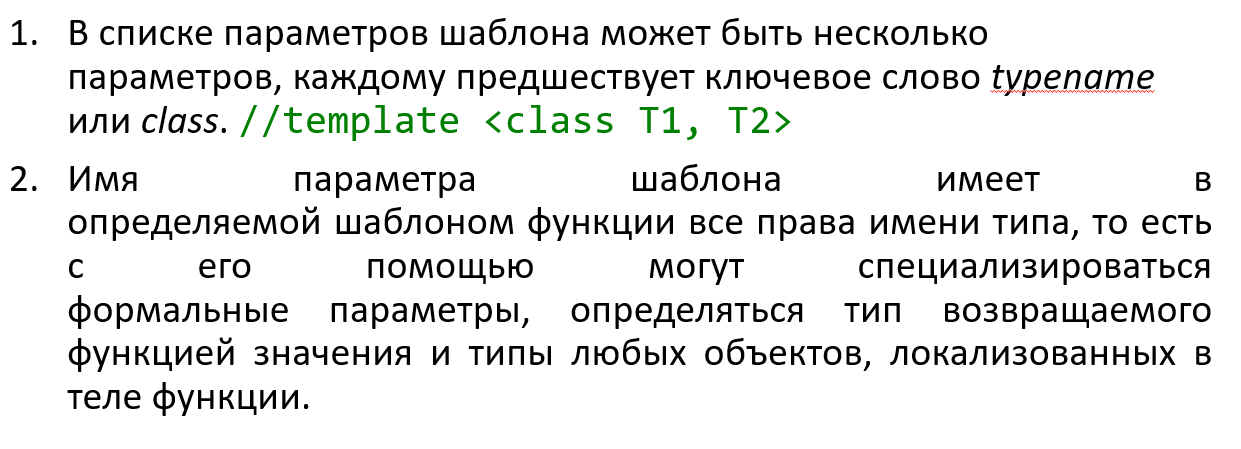
template<параметры> возвращаемое значение имя (параметры) {...}

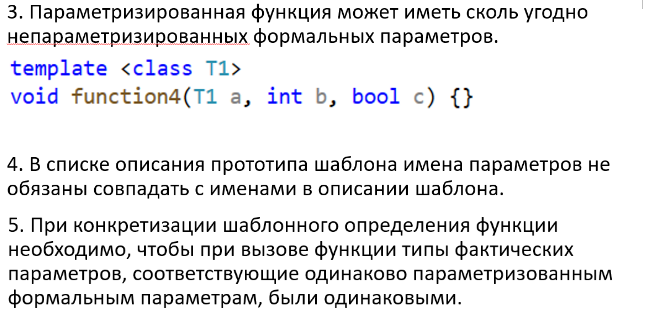
1. Каковы синтаксис/семантика шаблонов классов?

template<параметры> class имя {...}

1. Что такое параметры шаблона функции?

Типы данных, которые будут передаваться в функцию

1. Перечислите основные свойства параметров шаблона функции.

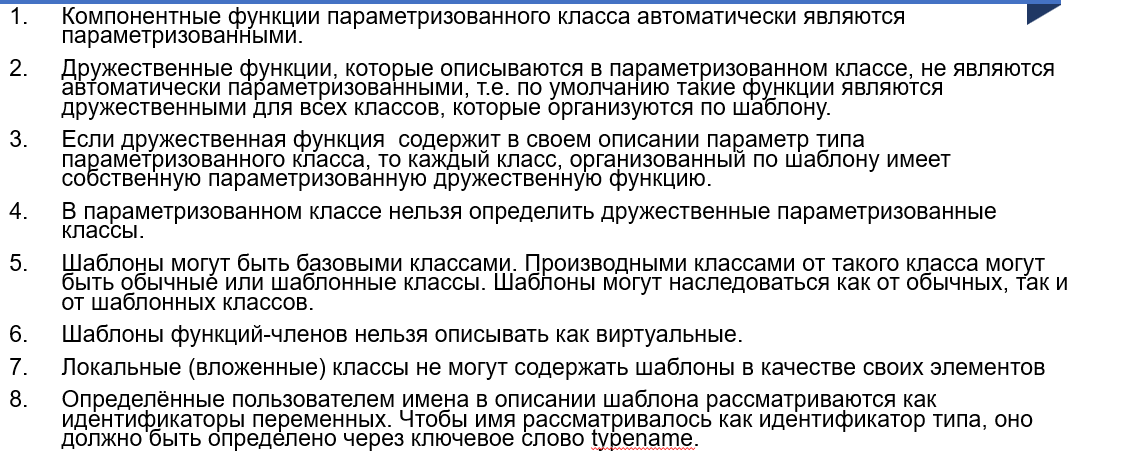


1. Как записывать параметр шаблона?

template <class/typename TYPE>

1. Можно ли перегружать параметризованные функции?

Да

1. Перечислите основные свойства параметризованных классов.
2. Все ли компонентные функции параметризованного класса являются параметризованными?

Да

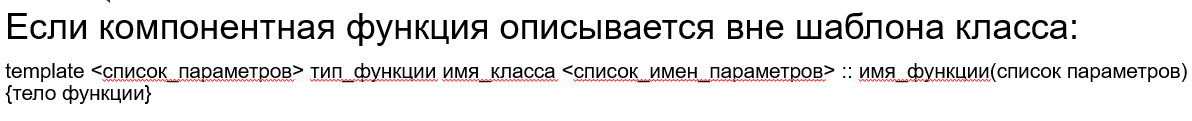
1. Являются ли дружественные функции, описанные в параметризованном  
   классе, параметризованными?

Нет

1. Могут ли шаблоны классов содержать виртуальные компонентные функции?

Нет

1. Как определяются компонентные функции параметризованных классов вне определения шаблона класса?



1. Что такое инстанцирование шаблона?

Процесс генерации компилятором конкретного класса по шаблону при создании экземпляра параметризованного класса.

1. На каком этапе происходит генерирование определения класса по шаблону?

На этапе создания экземпляра класса