Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа № 7**

Информатика

за 2 семестр

Вариант № 1

Выполнил:

Студент группы РИС 20-1-бз

Акиев В.В.

20-ЭТФ-641

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

2021

**Шаблон классов**

1. **Цель задания**

1.1 Создание консольного приложения, состоящего из нескольких файлов в системе программирования VisualStudio.

1.2. Реализация шаблона класса-контейнера.

1. **Задание**

Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

+ число – добавляет константу ко всем элементам вектора;

Пользовательский класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием.

1. **Файл Лабораторная\_7.cpp**

#include "Vector.h"

#include <iostream>

#include "Time.h"

using namespace std;

void main()

{

Time t;

cin >> t;

cout << t;

Vector<Time>A(5, t);

cin >> A;

cout << A << endl;

Vector <Time>B(10, t);

cout << B << endl;

B = A;

cout << B << endl;

cout << A[2] << endl;

cout << "size=" << A() << endl;

B = A + t;

cout << B << endl;

B = B + 532;

cout << B << endl;

}

1. **файл Time.cpp**

#include "Time.h"

Time::Time(void)

{

min = sec = 0;

}

Time::Time(int M, int S)

{

min = M; sec = S;

}

Time::Time(const Time& t)

{

min = t.min;

sec = t.sec;

}

Time& Time::operator =(const Time& t)

{

min = t.min;

sec = t.sec;

return\*this;

}

Time Time::operator+(Time k)

{

int t = min \* 60 + sec;

int kt = k.min \* 60 + k.sec;

t += kt;

Time temp(t / 60, t % 60);

return temp;

}

//добавляем число к времени

Time Time::operator+(int k)

{

int t = min \* 60 + sec;

t += k;//добавляем секунды

Time temp(t / 60, t % 60);

return temp;

}

ostream& operator<<(ostream& out, const Time& t)

{

out << t.min << " : " << t.sec;

return out;

}

istream& operator>>(istream& in, Time& t)

{

cout << "\nmin: "; in >> t.min;

cout << "\nsec: "; in >> t.sec;

return in;

}

1. **Файл Time.h**

#pragma once

#include <iostream>

using namespace std;

class Time

{

public:

Time(void);

Time(int, int);

Time(const Time&);

Time& operator=(const Time&);

Time operator+(Time k);

Time operator+(int k);

//перегруженные операции ввода-вывода

friend ostream& operator<< (ostream& out, const Time&);

friend istream& operator>> (istream& in, Time&);

public:

virtual ~Time(void) {};

private:

int min, sec;

};

1. **Файл Vector.h**

#include <iostream>

using namespace std;

template <class T> //T - параметр шаблона

class Vector

{

public:

//конструктор с параметрами: выделяет память под s элементов и заполняет их

//значением k

Vector(int s, T k);

//конструктор с параметрами

Vector(const Vector<T>& a);

//деструктор

~Vector();

//оператор присваивания

Vector& operator=(const Vector<T>& a);

//операция доступа по индексу

T& operator[](int index);

//операция для добавление константы

Vector operator+(const T k);

Vector operator+(const int k);

//операция, возвращающая длину вектора

int operator()();

//перегруженные операции ввода-вывода

// <> - указывают на то, что функция является шаблоном

friend ostream& operator<< <>(ostream& out, const Vector<T>& a);

friend istream& operator>> <>(istream& in, Vector<T>& a);

private:

int size;//размер вектора

T\* data;//укзатель на динамический массив значений вектора

};

//опрeделение функций

//конструктор с параметрами

template <class T>

Vector<T>::Vector(int s, T k)

{

size = s;

data = new T[size];

for (int i = 0; i < size; i++)

data[i] = k;

}

//конструктор копирования

template <class T>

Vector<T>::Vector(const Vector& a)

{

size = a.size;

data = new T[size];

for (int i = 0; i < size; i++)

data[i] = a.data[i];

}

//деструктор

template <class T>

Vector<T>::~Vector()

{

delete[]data;

data = 0;

}

//операция присваивания

template <class T>

Vector<T>& Vector<T>::operator=(const Vector<T>& a)

{

if (this == &a)return \*this;

size = a.size;

if (data != 0) delete[]data;

data = new T[size];

for (int i = 0; i < size; i++)

data[i] = a.data[i];

return \*this;

}

//операция доступа по индексу

template <class T>

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

{

if (index < size) return data[index];

else cout << "\nError! Index>size";

}

//операция для добавления константы

template <class T>

Vector<T> Vector<T>::operator+(const T k)//+k

{

Vector<T> temp(size, k);//инициализируем временный вектор любым значением

for (int i = 0; i < size; ++i)

temp.data[i] = data[i] + k;

return temp;

}

//операция для добавления числа

template <class T>

Vector<T> Vector<T>::operator+(const int k)//+k

{

if (size > 0)

{

Vector<T> temp(size, data[0]);//инициализируем временный вектор любым значением

for (int i = 0; i < size; ++i)

temp.data[i] = data[i] + k;

return temp;

}

else cout << "\nError! size = 0";

}

//операция для получения длины вектора

template <class T>

int Vector<T>::operator ()()

{

return size;

}

//операции для ввода-вывода

template <class T>

ostream& operator<< (ostream& out, const Vector<T>& a)

{

for (int i = 0; i < a.size; ++i)

out << a.data[i] << " ";

return out;

}

template <class T>

istream& operator>> (istream& in, Vector<T>& a)

{

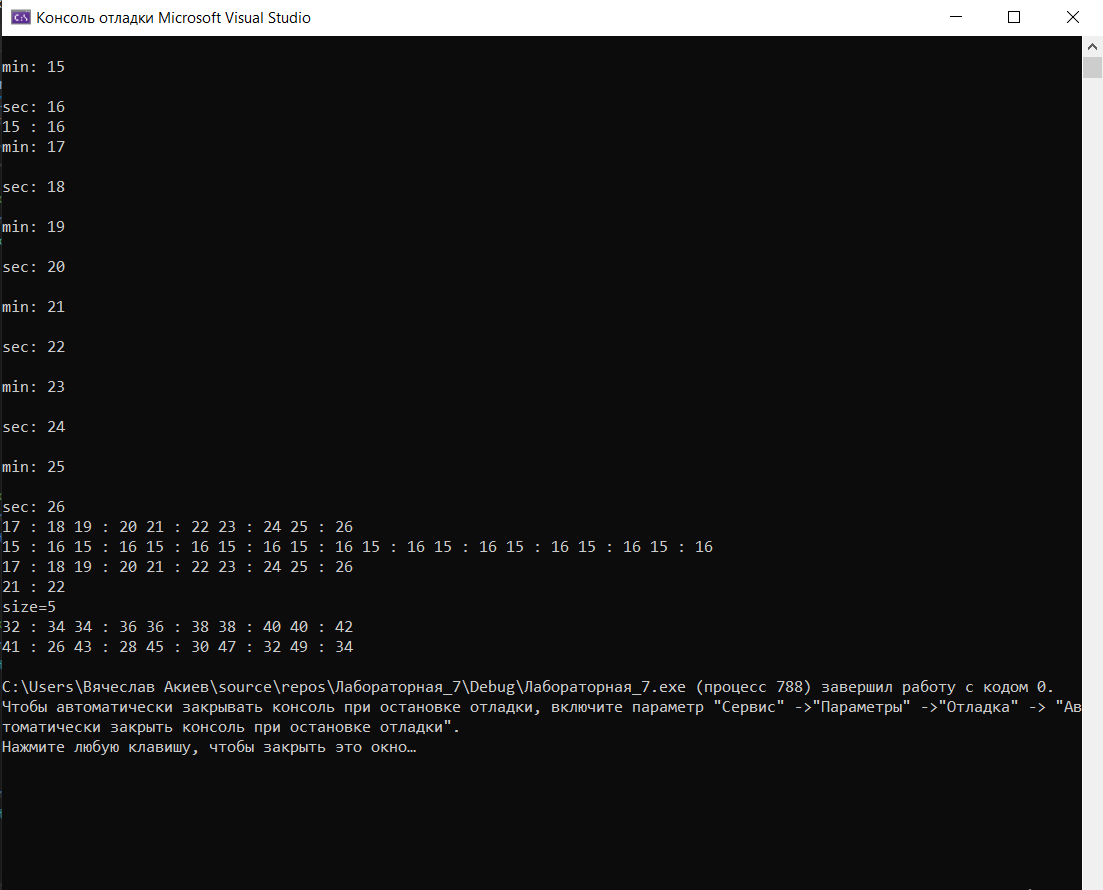
for (int i = 0; i < a.size; ++i)

in >> a.data[i];

return in;

}

1. **Результат работы программы**

****

1. **Контрольные вопросы.**

**1. В чем смысл использования шаблонов?**

С помощью шаблона можно отделить алгоритмы от конкретных типов

данных, передавая тип в качестве параметра, или создавать араметризированные классы для создания

**2. Каковы синтаксис / семантика шаблонов функций?**

Формат шаблона:

template <параметры\_шаблона>

заголовок\_функции

{ тело функции }

**3. Каковы синтаксис / семантика шаблонов классов?**

Формат шаблона:

template <параметры шаблона>

class имя\_класса

{ … };

**4. Что такое параметры шаблона функции?**

Параметр, благодаря которому можно вызвать функцию от перемеренной с любым типом.

**5. Перечислите основные свойства параметров шаблона функции.**

1. Имена параметров шаблона должны быть уникальными во всем определении шаблона.

2. Список параметров шаблона функции не может быть пустым, так как при этом теряется возможность параметризации и шаблон функций становится обычным определением конкретной функции.

3. В списке параметров шаблона функций может быть несколько параметров. Каждый из них должен начинаться со служебного слова class. Например, допустим такой заголовок шаблона :

template <class type1, class type2>

Соответственно, неверен заголовок :

template < class type1, type2, type3>

4. Недопустимо использовать в заголовке шаблона параметры с одинаковыми именами.

**6. Как записывать параметр шаблона?**

template<class Имя класса>

**7. Можно ли перегружать параметризованные функции?**

Да, можно, достаточно поменять тип данных у параметров функции.

**8. Перечислите основные свойства параметризованных классов.**

1. Компонентные функции параметризованного класса автоматически являются параметризованными.

2. Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями

3.С одной стороны, шаблоны могут быть производными (наследоваться) как от шаблонов, так и от обычных классов,

с другой стороны, они могут использоваться в качестве базовых для других шаблонов или классов.

4. Статические переменные шаблонов классов необходимо инициализировать для каждого используемого типа данных.

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

Нет, шаблоны классов могут содержать статические элементы, дружественные функции и классы.

**10. Являются ли дружественные функции, описанные в параметризованном**

**классе, параметризованными?**

Не всегда, зависит от функции и её логики

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

Нет не могут.

**12. Как определяются компонентные функции параметризованных классов вне**

**определения шаблона класса?**

Все определение шаблонного класса размещается в заголовочном файле.

**13. Что такое инстанцирование шаблона?**

Процесс генерации компилятором определения конкретного класса по шаблону класса и

аргументам шаблона.

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

На момент объявления объекта либо объявления указателя.