Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.7**

Дисциплина: «информатика»

Тема: Шаблоны классов

Вариант 5

Выполнил работу

студент группы РИС-20-1б

Зверев А.Д.

Проверила

Доцент кафедры ИТАС

Полякова О.А.

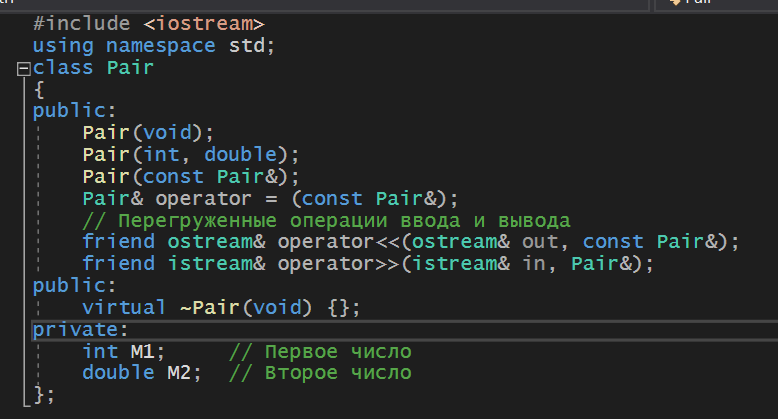
Пермь, 2021

**Постановка задачи**

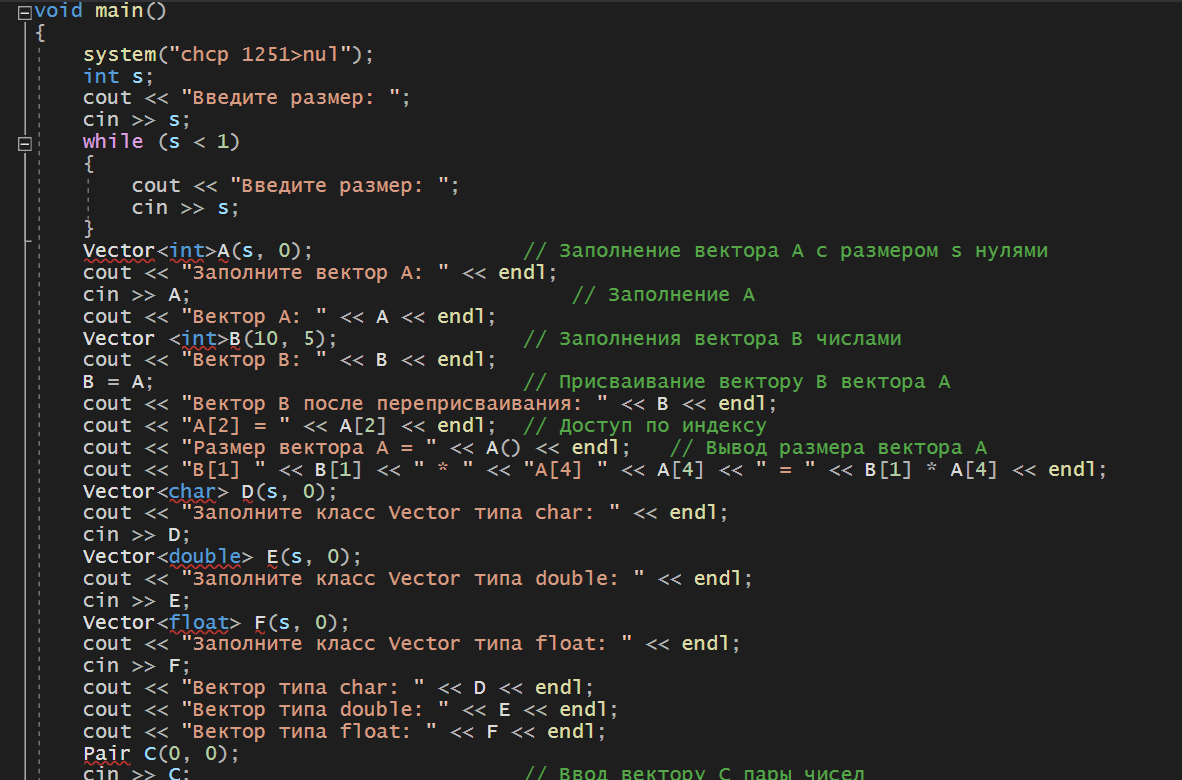
1. Определить шаблон класса—контейнера.
2. Реализовать конструкторы, деструктор, операции ввода—вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных (int, float, double).
5. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы стандартных типов данных.
6. Реализовать пользовательский класс.
7. Перегрузить для пользовательского класса операции ввода—вывода.
8. Перегрузить операции, необходимые для выполнения операций контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.
11. Класс—контейнер Вектор с элементами типа int.
12. Реализовать операции:
    1. []—доступа по индексу;
    2. Int()—определение размера вектора;
    3. \* вектор—умножение элементов векторов a[i]\*b[i];
13. 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.

**Анализ задачи**

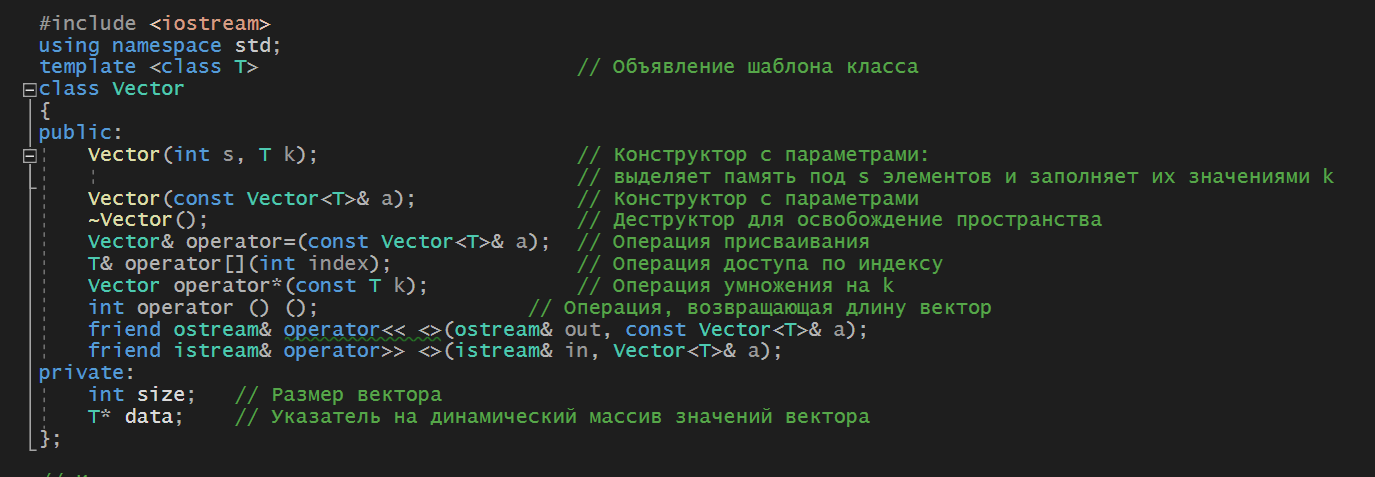
1. Для решения задачи необходимо…
   1. Создать класс Pair в файле Pair.h, где будут храниться данные, используемые внутри программы;



* 1. Написать функцию void main, где будут вызываться основные операции, производимые с векторами;



* 1. В заголовочном файле Header.h описать класс с нужными переменными;



1. В программе были использованы следующие типы данных:
   1. Тип данных int для хранения введенных данных;



**Блок—схема программы**



**Решение**

**18.7.cpp**

#include "Header.h"

#include "Pair.h"

#include <iostream>

using namespace std;

void main()

{

system("chcp 1251>nul");

int s;

cout << "Введите размер: ";

cin >> s;

while (s < 1)

{

cout << "Введите размер: ";

cin >> s;

}

Vector<int>A(s, 0); // Заполнение вектора A с размером s нулями

cout << "Заполните вектор A: " << endl;

cin >> A; // Заполнение A

cout << "Вектор А: " << A << endl;

Vector <int>B(10, 5); // Заполнения вектора B числами

cout << "Вектор B: " << B << endl;

B = A; // Присваивание вектору B вектора A

cout << "Вектор B после переприсваивания: " << B << endl;

cout << "A[2] = " << A[2] << endl; // Доступ по индексу

cout << "Размер вектора A = " << A() << endl; // Вывод размера вектора A

cout << "B[1] " << B[1] << " \* " << "A[4] " << A[4] << " = " << B[1] \* A[4] << endl;

Vector<char> D(s, 0);

cout << "Заполните класс Vector типа char: " << endl;

cin >> D;

Vector<double> E(s, 0);

cout << "Заполните класс Vector типа double: " << endl;

cin >> E;

Vector<float> F(s, 0);

cout << "Заполните класс Vector типа float: " << endl;

cin >> F;

cout << "Вектор типа char: " << D << endl;

cout << "Вектор типа double: " << E << endl;

cout << "Вектор типа float: " << F << endl;

Pair C(0, 0);

cin >> C; // Ввод вектору С пары чисел

cout << C << endl;

C = { 1,0 }; // Другое присваивание вектору C чисел

cout << C;

~A();

~B();

~D();

~E();

~F();

}

**Pair.cpp**

#include "Pair.h"

// Зануление элементов

Pair::Pair(void)

{

M1 = M2 = 0;

}

// Присваивание чисел

Pair::Pair(int chislo1, double chislo2)

{

M1 = chislo1;

M2 = chislo2;

}

Pair::Pair(const Pair& t)

{

M1 = t.M1;

M2 = t.M2;

}

// Операция присваивания

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

{

M1 = t.M1;

M2 = t.M2;

return\*this;

}

// Вывод чисел

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

{

out << t.M1 << " : " << t.M2;

return out;

}

// Ввод чисел

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

{

system("chcp 1251>nul");

cout << "\nПервое число? ";

in >> t.M1;

cout << "\nВторое число? ";

in >> t.M2;

return in;

}

**Header.h**

#include <iostream>

using namespace std;

template <class T> // Объявление шаблона класса

class Vector

{

public:

Vector(int s, T k); // Конструктор с параметрами:

// выделяет память под s элементов и заполняет их значениями k

Vector(const Vector<T>& a); // Конструктор с параметрами

~Vector(); // Деструктор для освобождение пространства

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

T& operator[](int index); // Операция доступа по индексу

Vector operator\*(const T k); // Операция умножения на k

int operator () (); // Операция, возвращающая длину вектор

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

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

private:

int size; // Размер вектора

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

};

// Конструктор с параметрами

template <class T>

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

{

size = s; // Размер вектора

data = new T[size]; // Создание динамического массива для элементов вектора

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

data[i] = k; // Заполнение k элементами

}

// 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 освобождает память,

// Так как это класс и элементы класса находятся в private,

// То программа не может автоматически удалить эти элементы

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; // Удаление старого массива

// если в нем есть элементы

data = new T[size]; // И создание нового

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 умножения

template <class T>

Vector<T> Vector<T>::operator\*(const T k)//\*k

{

Vector<T> temp(size, k);

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

temp.data[i] = data[i] \* k;

return temp;

}

// Возвращение размера вектора

template <class T>

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

{

return size;

}

// Операция 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;

}

// Операция ввода

template <class T>

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

{

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

in >> a.data[i];

return in;

}

**Pair.h**

#include <iostream>

using namespace std;

class Pair

{

public:

Pair(void);

Pair(int, double);

Pair(const Pair&);

Pair& operator = (const Pair&);

// Перегруженные операции ввода и вывода

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

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

public:

virtual ~Pair(void) {};

private:

int M1; // Первое число

double M2; // Второе число

};

**Скриншоты**

