Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 13

ОСНОВЫ АЛГОРИТМИЗАЦИИ ЗА II СЕМЕСТР

Выполнил студент:

Тимолянов Григорий Констнатинович

Группа РИС-21-1бз

Проверила:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

ПЕРМЬ 2024

# Лабораторная работа №13 "Стандартные обобщенные алгоритмы библиотеки STL"

**Вариант №15**

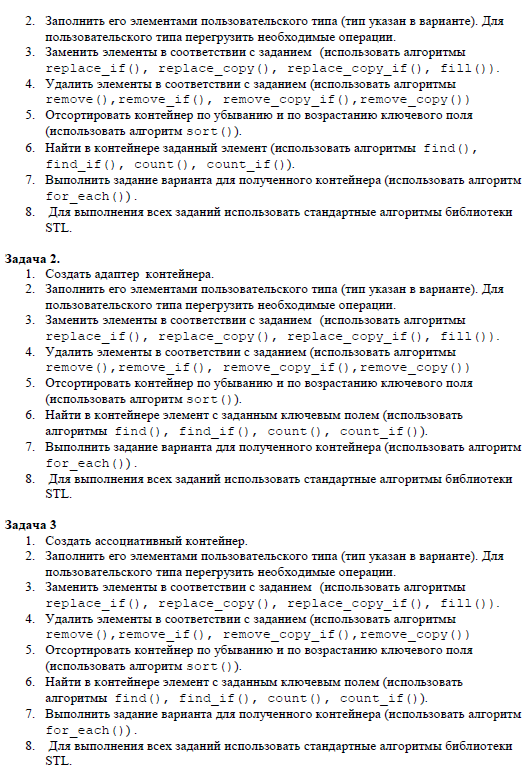
**Цель:**

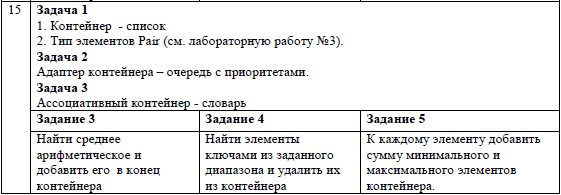
1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Использование стандартных обобщенных алгоритмов из библиотеки STL в ОО программе.

## **Задача:**







**Решение:**

# Файл Обобщенные алгоритмы STL.cpp

#include "C:\Users\arsenii.stashkov\Documents\Политех\Лабораторные работы C++\2 семестр\Основы алгоритмизации\Лабораторная работа №11\Task2\Pair.h"

#include <list>

#include <iostream>

//#include <algorithm>

using namespace std;

typedef list <Pair> TList;

// Формируем список

TList MakeList(int n) {

Pair a;

TList l;

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

cin >> a;

l.push\_back(a);

}

return l;

}

// Выводим список на экран

void PrintList(TList l) {

for (auto i = l.begin(); i != l.end(); ++i) {

cout << \*i << "\n";

}

cout << endl;

}

// Находим среднее арифметическое и добавляем в список

TList Average(TList l) {

Pair avg;

for (auto i = l.begin(); i != l.end(); ++i) {

avg = avg + \*i;

}

Pair& p = avg;

p.set\_first(p.get\_first() / l.size());

p.set\_second(p.get\_second() / l.size());

l.insert(l.end(), avg);

return l;

}

//Удаляем числа из диапазона(10 - 50)

TList DeleteRange(TList l) {

auto i = l.begin();

while (i != l.end()) {

Pair value = \*i;

if ((value.get\_first() >= 10) && (value.get\_first() <= 50) ||

(value.get\_second() >= 10) && (value.get\_second() <= 50)) {

l.erase(i++);

}

else {

++i;

}

}

return l;

}

// Поиск минимального значения

Pair Min(TList l) {

Pair min;

Pair& p = min;

p.set\_first(0);

p.set\_second(0.0);

for (auto i = l.begin(); i != l.end(); ++i) {

if (min > \*i || (p.get\_first() == 0 && p.get\_second() == 0.0)) { min = \*i; }

}

return min;

}

// Поиск максимального значения

Pair Max(TList l) {

Pair max;

for (auto i = l.begin(); i != l.end(); ++i) {

if (max < \*i) { max = \*i; }

}

return max;

}

// Суммируем каждый элемент с мин. и макс. значениями

TList Sum(TList l) {

Pair min = Min(l);

Pair max = Max(l);

for (auto i = l.begin(); i != l.end(); ++i) {

\*i = \*i + min + max;

}

return l;

}

int main()

{

setlocale(LC\_ALL, "Rus");

int n;

cout << "Введите длину списка:\n";

cin >> n;

cout << "\n";

TList l;

l = MakeList(n);

cout << "\nНаш список:\n";

PrintList(l);

cout << "Находим среднее арифметическое и добавляем в конец списка:\n";

l = Average(l);

PrintList(l);

cout << "Удаляем числа из диапазона (10-50):\n";

l = DeleteRange(l);

PrintList(l);

cout << "К каждому элементу добавляем сумму макс. и мин. элементов:\n";

l = Sum(l);

PrintList(l);

return 0;

}

Файл Pair.cpp

#include "Pair.h"

// Перегрузка операции присваивания

Pair& Pair::operator = (const Pair& p) {

if (&p == this) {

return \*this;

}

first = p.first;

second = p.second;

return \*this;

}

// Перегрузка глобальной функции-операции ввода

istream& operator >> (istream& in, Pair& p) {

cout << "Введите первое число (int): ";

in >> p.first;

cout << "Введите второе число (double): ";

in >> p.second;

return in;

}

// Перегрузка глобальной функции-операции вывода

ostream& operator << (ostream& out, const Pair& p) {

return (out << p.first << " : " << p.second);

}

bool Pair::operator < (const Pair& p) {

if (first < p.first) return true;

if (first == p.first && second < p.second) return true;

return false;

}

bool Pair::operator > (const Pair& p) {

if (first > p.first) return true;

if (first == p.first && second > p.second) return true;

return false;

}

// Перегрузка вычитания пар чисел

Pair Pair::operator-(const Pair& t) {

int temp = first;

int temp2 = t.first;

double tempDouble = second;

double tempDouble2 = t.second;

Pair p;

p.first = temp - temp2;

p.second = tempDouble - tempDouble2;

return p;

}

// Перегрузка сложения пар чисел

Pair Pair::operator+(const Pair& t) {

int temp = first;

int temp2 = t.first;

double tempDouble = second;

double tempDouble2 = t.second;

Pair p;

p.first = temp + temp2;

p.second = tempDouble + tempDouble2;

return p;

}

// Перегрузка деления пар чисел

Pair Pair::operator/(const Pair& t) {

int temp = first;

int temp2 = t.first;

double tempDouble = second;

double tempDouble2 = t.second;

Pair p;

p.first = temp / temp2;

p.second = tempDouble / tempDouble2;

return p;

}

// Перегрузка операции добавления константы к паре

Pair Pair::operator\*(const Pair& a) {

const double number = 10.5;

int temp = first;

double temp2 = second;

Pair p;

if ((int)number == number) {

p.first = temp + number;

p.second = temp2;

}

else {

p.second = temp2 + number;

p.first = temp;

}

return p;

}

bool Pair::operator==(const Pair& a) {

if (first == a.first && second == a.second) return true;

return false;

}

// Перегрузка оператора индексирования

/\*Pair Pair::operator[](const Pair& t) {

Pair\* p;

p->first = 0;

p->second = 0;

return p[10];

}\*/

Файл Pair.h

#pragma once

#include <iostream>

using namespace std;

class Pair {

int first;

double second;

// Глобальные функции ввода-вывода

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

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

public:

Pair() {

first = 0;

second = 0;

}

Pair(int f, double s) {

first = f;

second = s;

}

Pair(const Pair& p) {

first = p.first;

second = p.second;

}

~Pair() {}

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int f) { first = f; }

void set\_second(double s) { second = s; }

// Перегруженные операции

Pair& operator=(const Pair&);

Pair operator-(const Pair&);

Pair operator/(const Pair&);

Pair operator\*(const Pair&);

Pair operator+(const Pair&);

//Pair operator[](const Pair&);

bool operator>(const Pair&);

bool operator<(const Pair&);

bool operator==(const Pair&);

};

