Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.4**

Дисциплина: основы алгоритмизации и программирования

Тема: “Классы и объекты. Принцип подстановки.”

Вариант 19

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

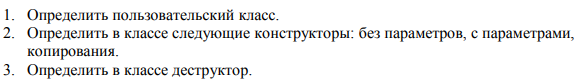
Проверила:

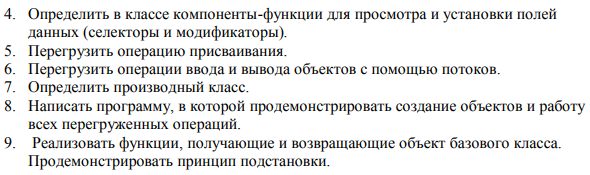
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Pair для хранения первого числа пары (first), второго числа пары (second).

**1.2.** Организовать класс Complex с полями типа int real, imaginary, который является производным класса Pair.

**1.3.** Организовать необходимые методы для ввода данных в поля first и second: гетторы, сетторы, конструкторы, деструктор.

**1.4.** Организовать перегрузку оператора >> дружественной классу Pair.

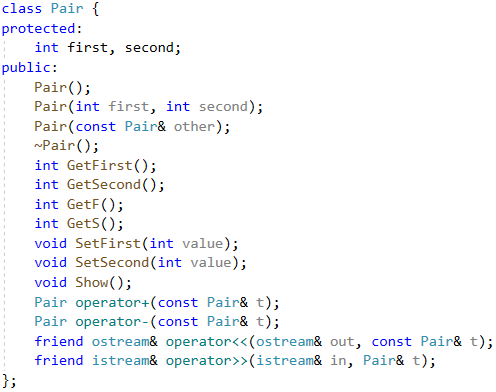
**1.5.** Организовать перегрузку оператора << дружественной классу Pair.

**1.6.** Организовать перегрузку оператора - дружественной классу Pair.

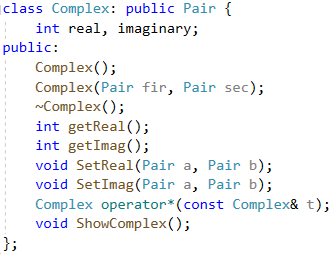
**1.7.** Организовать перегрузку оператора + дружественной классу Pair.

**2.** В ходе работы были использованы следующие типы данных:

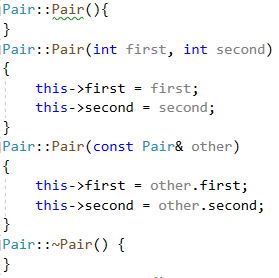
**2.1.** Организовать класс Pair для хранения первого числа пары (first), второго числа пары (second). в заголовочном файле Pair.h.

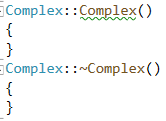


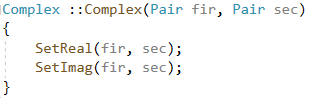
3



**2.2.** Организовать соответствующий файл Pair.cpp, в котором будут описаны методы, конструкторы и дружественные функции класса Pair.







**2.3.** Переменные типа int: tmp, res1, res2, где tmp – введённое число, res1– значение поля real класса Complex (производный класса Pair), res2 – значение поля imaginary класса Complex.

int res1=p.getReal(), res2=p.getImag();

int tmp;

4

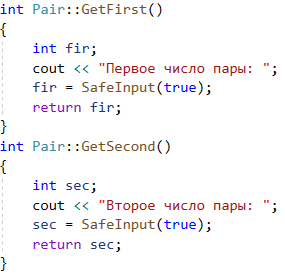
**2.4.** Объекты a, b, tmp класса Pair. Объект p принадлежит классу Complex.

Pair a, b;

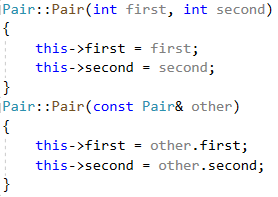
Complex p(a, b);

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через объекты класса Pair через гетторы.

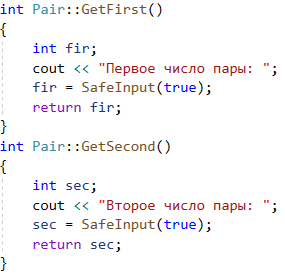


**3.2.** Данные также вводятся через объекты класса Pair через конструктор с параметрами.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода пар чисел используются гетторы (методы, описанные в классе для ввода), в которых используется функция cin.



5

**4.2.** Для вывода на консоль суммы и разности чисел пар чисел используется функци cout.

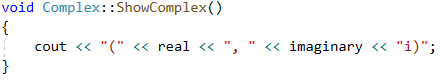
cout << endl<<"Сумма: " << a + b << endl;

cout << "\nРазность: " << a-b << endl;

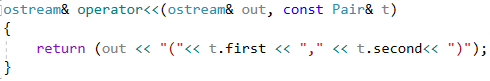
**4.3.** Для вывода на консоль произведения и разности чисел пар чисел используется метод ShowComplex() класса Complex, в котором используется функция cout.

cout << "Комплексное число: ";

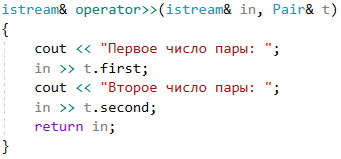
p.ShowComplex();



**4.4.** Так же вывод на консоль может быть осуществлён с помощью перегрузки оператора <<.

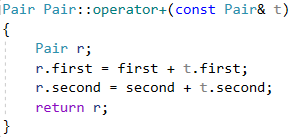


**4.5.** Так же ввод пар чисел может быть осуществлён с помощью перегрузки оператора >>.



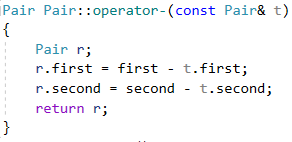
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В перегрузке оператора + выполняется сложение полей first двух объектов класса Pair, затем сложение полей second этих объектов.

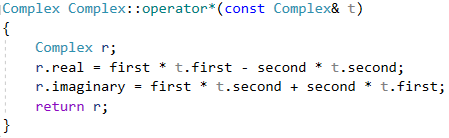


6

**5.2.** В перегрузке оператора - выполняется вычитание полей first двух объектов класса Pair, затем вычитание полей second этих объектов.

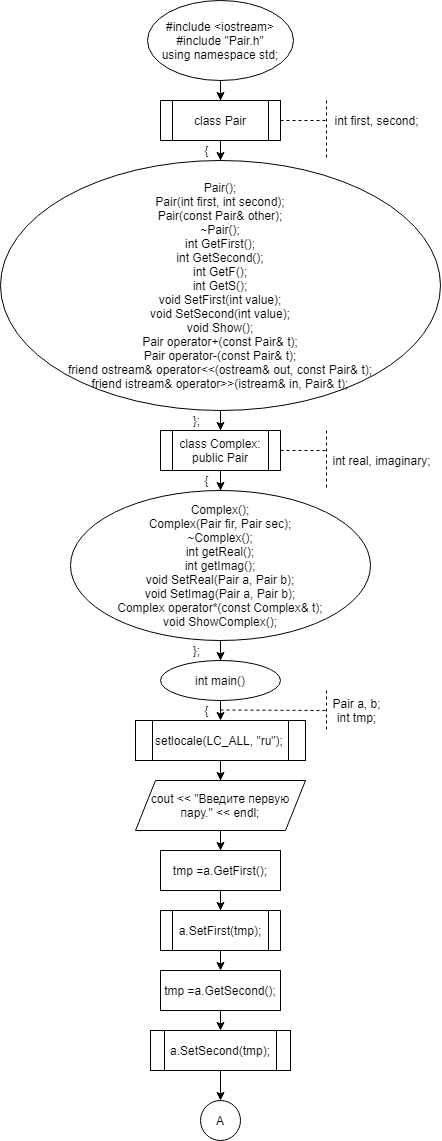


**5.3.** В перегрузке оператора \* выполняется преобразование полей real и imaginary по заданию.

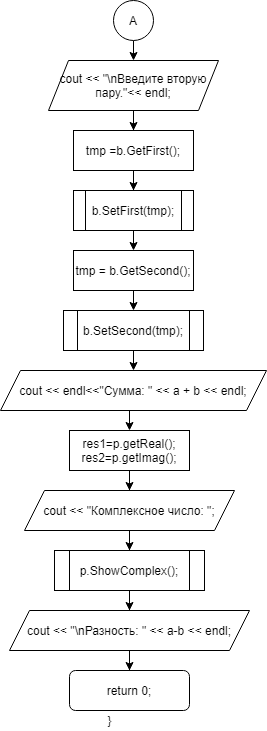


7

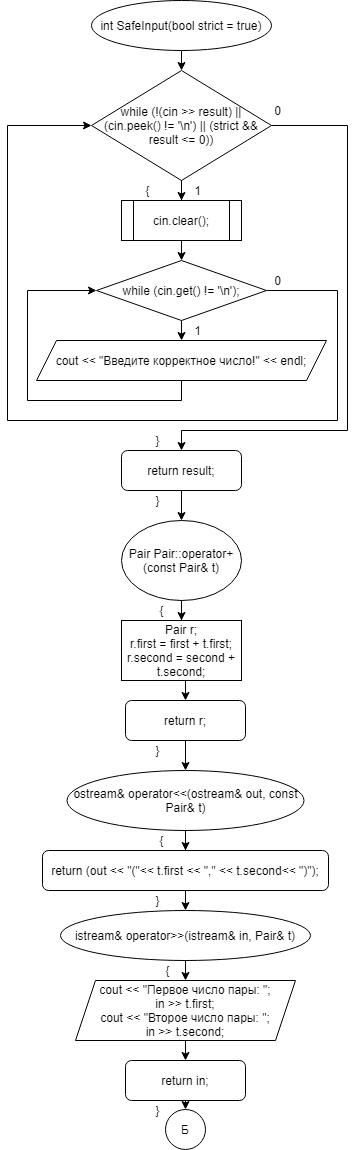
**Блок-схема**



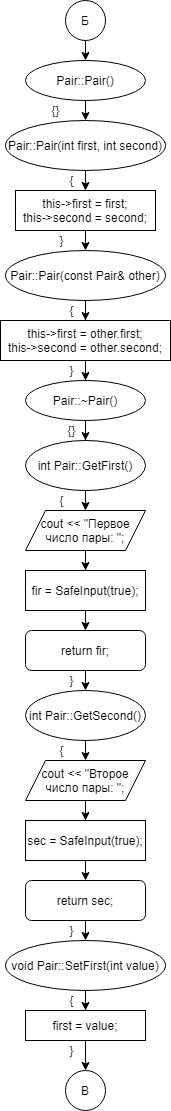
8



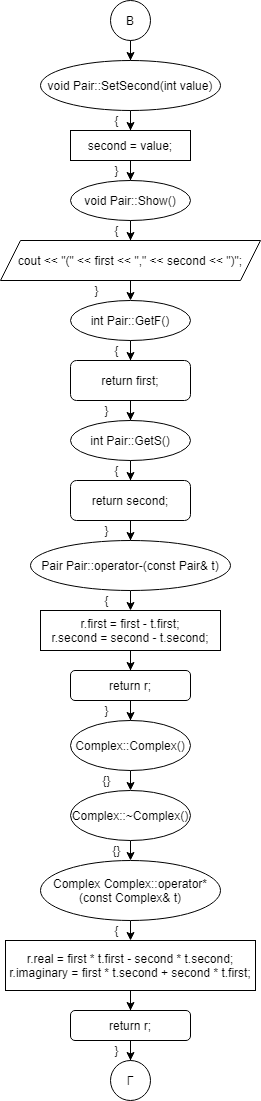
9



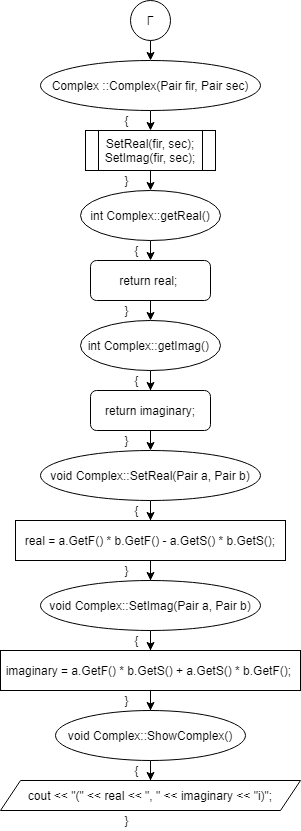
10



11



12



13

**Код**

#include <iostream>

#include "Pair.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

Pair a, b;

int tmp;

cout << "Введите первую пару." << endl;

tmp =a.GetFirst();

a.SetFirst(tmp);

tmp =a.GetSecond();

a.SetSecond(tmp);

cout << "\nВведите вторую пару."<< endl;

tmp =b.GetFirst();

b.SetFirst(tmp);

tmp = b.GetSecond();

b.SetSecond(tmp);

cout << endl<<"Сумма: " << a + b << endl;

Complex p(a, b);

int res1=p.getReal(), res2=p.getImag();

cout << "Комплексное число: ";

p.ShowComplex();

cout << "\nРазность: " << a-b << endl;

}

14

#pragma once

#include <iostream>

using namespace std;

class Pair {

protected:

int first, second;

public:

Pair();

Pair(int first, int second);

Pair(const Pair& other);

~Pair();

int GetFirst();

int GetSecond();

int GetF();

int GetS();

void SetFirst(int value);

void SetSecond(int value);

void Show();

Pair operator+(const Pair& t);

Pair operator-(const Pair& t);

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

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

};

class Complex: public Pair {

int real, imaginary;

public:

Complex();

Complex(Pair fir, Pair sec);

~Complex();

int getReal();

int getImag();

void SetReal(Pair a, Pair b);

void SetImag(Pair a, Pair b);

Complex operator\*(const Complex& t);

void ShowComplex();

};

15

#include "Pair.h"

int SafeInput(bool strict = true) {

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0)) {

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число!" << endl;

}

return result;

}

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

{

Pair r;

r.first = first + t.first;

r.second = second + t.second;

return r;

}

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

{

return (out << "("<< t.first << "," << t.second<< ")");

}

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

{

cout << "Первое число пары: ";

in >> t.first;

cout << "Второе число пары: ";

in >> t.second;

return in;

}

Pair::Pair(){

}

Pair::Pair(int first, int second)

{

this->first = first;

this->second = second;

}

Pair::Pair(const Pair& other)

{

this->first = other.first;

this->second = other.second;

}

Pair::~Pair() {

}

int Pair::GetFirst()

{

int fir;

cout << "Первое число пары: ";

fir = SafeInput(true);

return fir;

}

int Pair::GetSecond()

{

int sec;

cout << "Второе число пары: ";

sec = SafeInput(true);

return sec;

}

void Pair::SetFirst(int value)

{

first = value;

}

void Pair::SetSecond(int value)

{

second = value;

16

}

void Pair::Show()

{

cout << "(" << first << "," << second << ")";

}

int Pair::GetF()

{

return first;

}

int Pair::GetS()

{

return second;

}

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

{

Pair r;

r.first = first - t.first;

r.second = second - t.second;

return r;

}

Complex::Complex()

{

}

Complex::~Complex()

{

}

Complex Complex::operator\*(const Complex& t)

{

Complex r;

r.real = first \* t.first - second \* t.second;

r.imaginary = first \* t.second + second \* t.first;

return r;

}

Complex ::Complex(Pair fir, Pair sec)

{

SetReal(fir, sec);

SetImag(fir, sec);

}

int Complex::getReal()

{

return real;

}

int Complex::getImag()

{

return imaginary;

}

void Complex::SetReal(Pair a, Pair b)

{

real = a.GetF() \* b.GetF() - a.GetS() \* b.GetS();

}

void Complex::SetImag(Pair a, Pair b)

{

imaginary = a.GetF() \* b.GetS() + a.GetS() \* b.GetF();

}

void Complex::ShowComplex()

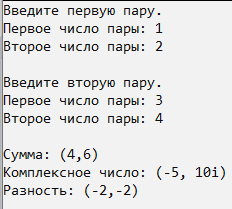
{

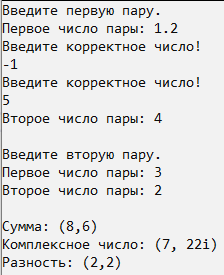
cout << "(" << real << ", " << imaginary << "i)";

}

17

**Скриншоты**





18

**Контрольные вопросы**



1. Наследование позволяет реализовать типизированное повторное использование объектов, создание иерархии «тип-подтип» с сохранением подтипом всех свойств своего предка. Можно рассматривать наследование для всех трех составляющих объекта.



2. Когда вы открыто наследуете родительский класс, то унаследованные public-члены остаются public.



3. Когда вы открыто наследуете родительский класс, то унаследованные private-члены остаются недоступными для дочернего класса.



4. К protected-членам родительского класса доступ открыт для членов дочернего класса.



5. 



6. Конструкторы не наследуются.



7. Деструкторы не наследуются.



8. 



9. 

19

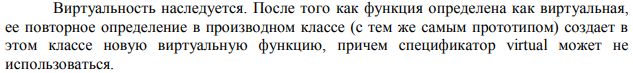


10. 

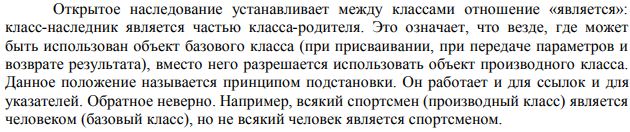


11. 



12. 



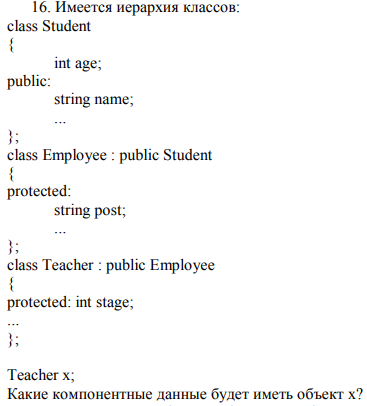
13. 



14. 



15. Пункт 13 и 14 – принцип подстановки.



20

16. age, name, post, stage.



17. Student(); Employee() : Student(); Teacher() : Employee();

21