Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**по лабораторной работе № 18.4**

Дисциплина: Информатика

Тема: «Объектно-ориентированное программирование. Простое наследование»

Вариант 1

Выполнил работу:

студент группы РИС-20-1Б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь

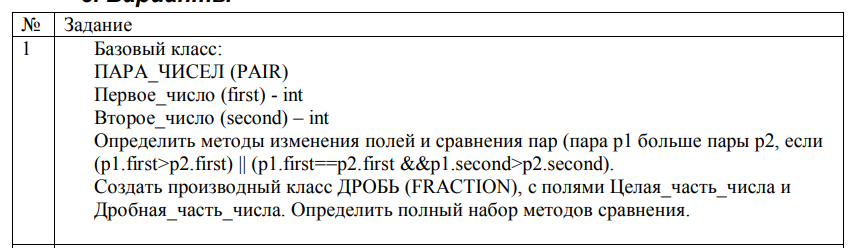
2021

**Цель работы**

* 1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
  2. Создание иерархии классов с использованием простого наследования.
  3. Изучение принципа подстановки.

**Постановка задачи**

* + 1. Определить пользовательский класс.
    2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
    3. Определить в классе деструктор
    4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
    5. Перегрузить операцию присваивания.
    6. Перегрузить операции ввода и вывода объектов с помощью потоков.
    7. Определить производный класс.
    8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.
    9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

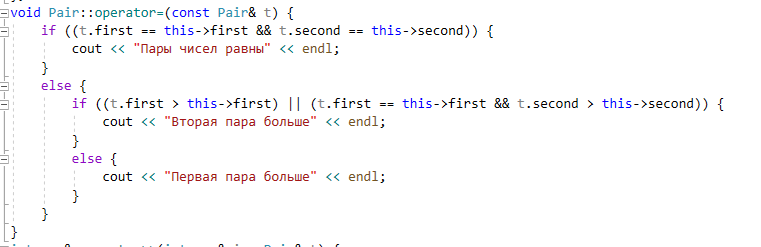


**Анализ задачи**

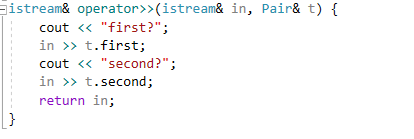
1. Необходимые действия

Работа с классом Pair

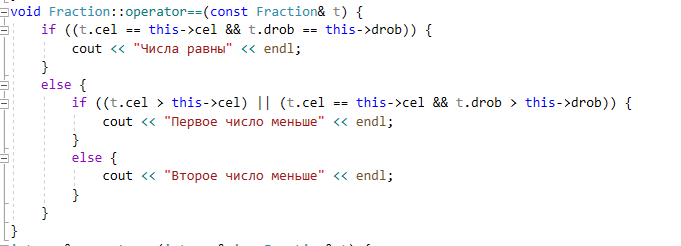
1)



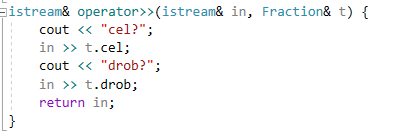
2) Создать Функцию по приему пар чисел



Работа с классом Fraction

1. Создать функцию по сравнению двух пар чисел 

2) Создать Функцию по приему пар чисел



Программный код

#include <iostream>

using namespace std;

class Pair {

public:

int first, second;

Pair() { first = 0; second = 0; }

Pair(int f, int s) { first = f; second = s; };

Pair(const Pair& t1) { first = t1.first; second = t1.second; }

~Pair() {};

//bool get\_Pair(const Pair& t1);

void operator=(const Pair&);

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

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

};

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

if ((t.first == this->first && t.second == this->second)) {

cout << "Пары чисел равны" << endl;

}

else {

if ((t.first > this->first) || (t.first == this->first && t.second > this->second)) {

cout << "Вторая пара больше" << endl;

}

else {

cout << "Первая пара больше" << endl;

}

}

}

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

cout << "first?";

in >> t.first;

cout << "second?";

in >> t.second;

return in;

}

/\*bool get\_Pair(const Pair& t1,const Pair&t2) {

return(t1.first>t2.first)||(t1.first==t2.first&&t1.second>t2.second);

}\*/

class Fraction :public Pair {

protected:

int cel, drob;

public:

Fraction() { cel = 0, drob = 0; };

Fraction(int f, int s) { cel = f, drob = s; }

Fraction(const Fraction& t1) { first = t1.cel; second = t1.drob; }

~Fraction() {};

void operator==(const Fraction&);

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

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

};

void Fraction::operator==(const Fraction& t) {

if ((t.cel == this->cel && t.drob == this->drob)) {

cout << "Числа равны" << endl;

}

else {

if ((t.cel > this->cel) || (t.cel == this->cel && t.drob > this->drob)) {

cout << "Первое число меньше" << endl;

}

else {

cout << "Второе число меньше" << endl;

}

}

}

istream& operator>>(istream& in, Fraction& t) {

cout << "cel?";

in >> t.cel;

cout << "drob?";

in >> t.drob;

return in;

}

int main() {

setlocale(LC\_ALL, "rus");

Pair a;

Pair b;

cout << "Первая пара чисел" << endl;

cin >> a;

cout << "Вторая пара чисел" << endl;

cin >> b;

a = b;

Fraction c;

Fraction d;

cout << "Первое число" << endl;

cin >> c;

cout << "Второе число" << endl;

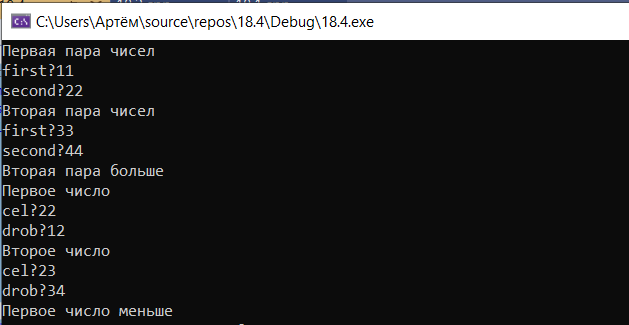
cin >> d;

c == d;

system("pause");

}

1. Демонстрация работы программы



1. Контрольные вопросы
   1. Для чего используется механизм наследования?

- Наследование - это механизм получения нового класса на основе уже существующего. Существующий класс может быть дополнен или изменен для создания нового класса.

* 1. Каким образом наследуются компоненты класса, описанные со спецификатором public?

- член класса может использоваться любой функцией, которая является членом данного или производного класса, а также к public - членам возможен доступ извне через имя объекта

* 1. Каким образом наследуются компоненты класса, описанные со спецификатором private?

- член класса может использоваться только функциями – членами данного класса и функциями – “друзьями” своего класса. В производном классе он недоступен.

* 1. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

-protected– то же, что и private***,*** но дополнительно член класса с данным атрибутом доступа может использоваться функциями-членами и функциями – “друзьями” классов, производных от данного.

* 1. Каким образом описывается производный класс?

class имя\_класса : список\_базовых\_классов

{список\_компонентов\_класса};

* 1. Наследуются ли конструкторы?

- не наследуются

* 1. Наследуются ли деструкторы?

- не наследуются

* 1. В каком порядке конструируются объекты производных классов?

- Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты- объекты (если они имеются), а потом сам производный класс. Таким образом, объект производного класса содержит в качестве подобъекта объект базового класса.

* 1. В каком порядке уничтожаются объекты производных классов?

- Уничтожаются объекты в обратном порядке: сначала производный, потом его компоненты-объекты, а потом базовый объект.

* 1. Что представляют собой виртуальные функции и механизм позднего связывания?

- К механизму виртуальных функций обращаются в тех случаях, когда в каждом производном классе требуется свой вариант некоторой компонентной функции. Классы, включающие такие функции, называются полиморфнымии играют особую роль в ООП.

Виртуальные функции предоставляют механизм позднего(отложенного) или динамическогосвязывания**.** Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используется ключевое слово virtual.

* 1. Могут ли быть виртуальными конструкторы? Деструкторы?

- конструкторы не могут быть виртуальными, в отличии от деструкторов

* 1. Наследуется ли спецификатор virtual?

- виртуальность наследуется

* 1. Какое отношение устанавливает между классами открытое наследование?

- Открытое наследование устанавливает между классами отношение «является»: класс-наследник является частью класса-родителя.

* 1. Какое отношение устанавливает между классами закрытое наследование?

- Закрытое наследование – это наследование реализации

* 1. В чем заключается принцип подстановки?

- Принцип означает, что везде, где может быть использован объект базового класса (при присваивании, при передаче параметров и возврате результата), вместо него разрешается использовать объект производного класса. Данное положение называется принципом подстановки. Он работает и для ссылок и для указателей.

* 1. Имеется иерархия классов:

class Student

{

public:

};

Class Employee : public Student

{

protected:

string post;

...

};

int age;

string name;

...

class Teacher : public Employee

{

protected: int stage;

...

};

Teacher x;

Какие компонентные данные будет иметь объект х?

- объект будет иметь следующие компонентные данные:

Name, post, stage

* 1. Для классов Student, Employee и Teacher написать конструкторы без параметров.

Student(){

age = 0;

name = “”;

}

Employee(){

name = “”;

post = “”;

}

Teacher(){

name = “”;

post = “”;

stage = “”;

}

* 1. Для классов Student, Employee и Teacher написать конструкторы с параметрами.

Student(const int Age, const string Name){

age = Age;

name = Name;

}

Employee(const string Name, const string Post){

name = Name;

post = Post;

}

Teacher(const string Name, const string Post, const string Stage){

name = Name;

post = Post;

stage = Stage;

}

* 1. Для классов Student, Employee и Teacher написать конструкторы копирования.

Student(const Student& student){

age = student.age;

name = student.name;

}

Employee(const Employee& employee){

name = employee.name;

post = employee.post;

}

Teacher(const Teacher& teacher){

name = teacher.name;

post = teacher.post;

stage = teacher.stage;

}

* 1. Для классов Student, Employee и Teacher определить операцию присваивания.

Student& operator = (const Student& student){

If (&student == this) return \*this;

age = student.age;

name = student.name;

return \*this;

}

Employee& operator = (const Employee& employee){

If (&employee == this) return \*this;

name = employee.name;

post = employee.post;

return \*this;

}

Teacher& operator = (const Teacher& teacher){

If (&teacher == this) return \*this;

name = teacher.name;

post = teacher.post;

stage = teacher.stage;

return \*this;

}