Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.4**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Простое наследование

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

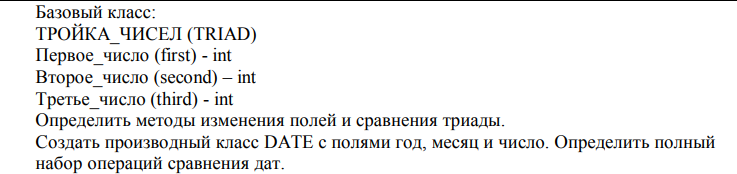
# **Цель работы**

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

# **Постановка задачи**

1. Определитель пользовательский класс.
2. Определить в классе следующие конструкторы:
3. Без параметров;
4. С параметрами;
5. Копирования;
6. Определить в классе деструктор.
7. Определить в классе компоненты-функции для просмотра и установки полей.
8. Перегрузить операции присваивания.
9. Перегрузить операции ввода и вывода объектов.
10. Определить производный класс.
11. Написать демонстрационную программу, в которой продемонстрировать создание объектов и работу всех перегруженных функций.
12. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать класс TRIAD с полями first, second, third.
* Разработать класс DATE с полями дата, месяц, год, который является производным класса TRIAD.
* Организовать методы для ввода данных, селекторы и модификатор, конструкторы и деструкторы, перегрузки операторов для класса TRIAD и DATE.

class Triad {

protected:

int first;

int second;

int third;

public:

//Конструктор без параметров

Triad(void);

//Конструктор с параметрами

Triad(int, int, int);

//Конструктор клпирования

Triad(const Triad&);

//Деструктор через virtual

virtual ~Triad(void);

//Селекторы

int get\_first() {

return first;

}

int get\_second() {

return second;

}

int get\_third() {

return third;

}

//Модификаторы

void set\_first(int);

void set\_second(int);

void set\_third(int);

//Перегруженные операции сравнения

bool operator == (const Triad&);

bool operator > (const Triad&);

bool operator < (const Triad&);

//Перегруженные операции ввода/вывода

friend istream& operator >> (istream&, Triad&);

friend ostream& operator << (ostream&, const Triad&);

};

class Date : public Triad {

public:

//Конструктор без параметров

Date(void);

//Конструктор с параметрами

Date(int, int, int);

//Конструктор клпирования

Date(const Date&);

//Деструктор

~Date(void);

//Перегруженные операции сравнения

bool operator == (const Date&);

bool operator > (const Date&);

bool operator < (const Date&);

//Перегруженные операции ввода/вывода

friend istream& operator >> (istream&, Date&);

friend ostream& operator << (ostream&, const Date&);

};

* Протестировать работу метода подстановки в функции main.

void f1(Triad& temp) {

temp.set\_first(2);

temp.set\_second(5);

cout << temp;

}

Triad f2() {

Date d(3, 5, 2002);

return d;

}

void main() {

system("chcp 1251");

Triad A, B(3, 5, 20);

cin >> A; cout << "\nТриада A " << A << "\nТриада B " << B;

if (A == B) {

cout << "\nТриады равны\n";

}

if (A > B) {

cout << "\nТриада A > Триады B\n" ;

}

if (A < B) {

cout << "\nТриада A < Триады B\n";

}

Date C;

cin >> C; cout << "\nДата 1: " << C;

Date D(12, 10, 2001);

cout << "\nДата 2: " << D;

if (C == D) {

cout << "\nДаты равны\n";

}

if (C > D) {

cout << "\nДата 2 следует после Даты 1\n";

}

if (C < D) {

cout << "\nДата 1 следует после Даты 2\n";

}

cout << "\nПодстановка";

f1(C);

A = f2();

cout << A;

system("pause");

}

1. C какими типами данных предстояло работать:

* Для хранения номера дня реализована переменная first типа int.

int first;

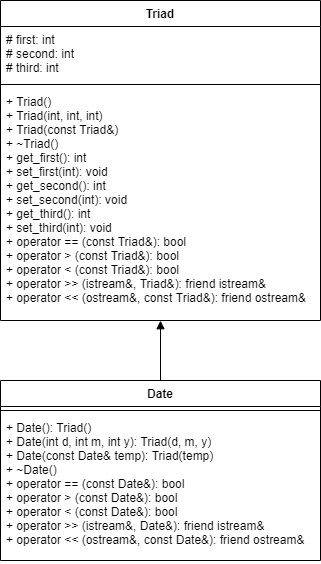
* Для хранения номера месяца реализована переменная second типа int.

int second;

* Для хранения номера года реализована переменная third типа int.

int third;

# **UML диаграмма**



# **Код программы на языке C++**

Заголовочный файл Triad.h

#pragma once

#include <iostream>

using namespace std;

class Triad {

protected:

int first;

int second;

int third;

public:

//Конструктор без параметров

Triad(void);

//Конструктор с параметрами

Triad(int, int, int);

//Конструктор клпирования

Triad(const Triad&);

//Деструктор через virtual

virtual ~Triad(void);

//Селекторы

int get\_first() {

return first;

}

int get\_second() {

return second;

}

int get\_third() {

return third;

}

//Модификаторы

void set\_first(int);

void set\_second(int);

void set\_third(int);

//Перегруженные операции сравнения

bool operator == (const Triad&);

bool operator > (const Triad&);

bool operator < (const Triad&);

//Перегруженные операции ввода/вывода

friend istream& operator >> (istream&, Triad&);

friend ostream& operator << (ostream&, const Triad&);

};

Обьявление класса в Triad.cpp

#include <iostream>

#include "Triad.h"

using namespace std;

Triad::Triad(void) {

first = 1;

second = 1;

third = 1;

}

Triad::Triad(int f, int s, int t) {

first = f;

second = s;

third = t;

}

Triad::Triad(const Triad& temp) {

first = temp.first;

second = temp.second;

third = temp.third;

}

Triad::~Triad(void){

}

void Triad::set\_first(int f) {

first = f;

}

void Triad::set\_second(int s) {

second = s;

}

void Triad::set\_third(int t) {

third = t;

}

bool Triad::operator == (const Triad& temp) {

if (first == temp.first && second == temp.second && third == temp.third) {

return true;

}

return false;

}

bool Triad::operator > (const Triad& temp) {

long int triad1 = first + second \* 100 + third \* 10000;

long int triad2 = temp.first + temp.second \* 100 + temp.third \* 10000;;

if (triad1 > triad2) {

return true;

}

return false;

}

bool Triad::operator < (const Triad& temp) {

long int triad1 = first + second \* 100 + third \* 10000;

long int triad2 = temp.first + temp.second \* 100 + temp.third \* 10000;;

if (triad1 < triad2) {

return true;

}

return false;

}

istream& operator >> (istream& in, Triad& temp) {

cout << "\n";

do {

cout << "1 - "; in >> temp.first;

} while (temp.first < 1);

do {

cout << "2 - "; in >> temp.second;

} while (temp.second < 1);

do {

cout << "3 - "; in >> temp.third;

} while (temp.third < 1);

return in;

}

ostream& operator << (ostream& out, const Triad& temp) {

out << "\n" << temp.first << " " << temp.second << " " << temp.third << "\n";

return out;

}

Заголовочный файл Date.h

#pragma once

#include "Triad.h"

class Date : public Triad {

public:

//Конструктор без параметров

Date(void);

//Конструктор с параметрами

Date(int, int, int);

//Конструктор клпирования

Date(const Date&);

//Деструктор

~Date(void);

//Перегруженные операции сравнения

bool operator == (const Date&);

bool operator > (const Date&);

bool operator < (const Date&);

//Перегруженные операции ввода/вывода

friend istream& operator >> (istream&, Date&);

friend ostream& operator << (ostream&, const Date&);

};

Обьявление класса в Date.cpp

#include "Date.h"

#include <iostream>

using namespace std;

Date::Date(void):Triad() {

}

Date::~Date(void) {

}

Date::Date(int d, int m, int y): Triad(d, m, y) {

}

Date::Date(const Date& temp):Triad(temp) {

}

bool Date::operator==(const Date& temp) {

if (first == temp.first && second == temp.second && third == temp.third) {

return true;

}

return false;

}

bool Date::operator>(const Date& temp) {

long int date1 = first + second \* 100 + third \* 10000;

long int date2 = temp.first + temp.second \* 100 + temp.third \* 10000;;

if (date1 > date2) {

return true;

}

return false;

}

bool Date::operator<(const Date& temp) {

long int date1 = first + second \* 100 + third \* 10000;

long int date2 = temp.first + temp.second \* 100 + temp.third \* 10000;;

if (date1 < date2) {

return true;

}

return false;

}

istream& operator >> (istream& in, Date& temp) {

cout << "\n";

do {

cout << "Число - "; in >> temp.first;

} while (temp.first < 1 || temp.first>31);

do {

cout << "Месяц - "; in >> temp.second;

} while (temp.second < 1 || temp.second>12);

do {

cout << "Год - "; in >> temp.third;

} while (temp.third < 1 || temp.third>2021);

return in;

}

ostream& operator << (ostream& out, const Date& temp) {

if (temp.first < 10) {

out << "\n0" << temp.first << ".";

}

else {

out << "\n" << temp.first << ".";

}

if (temp.second < 10) {

out << "0" << temp.second << ".";

}

else {

out << temp.second << ".";

}

out << temp.third << "\n";

return out;

}

Main файл

#include <iostream>

#include "Triad.h"

#include "Date.h"

using namespace std;

void f1(Triad& temp) {

temp.set\_first(2);

temp.set\_second(5);

cout << temp;

}

Triad f2() {

Date d(3, 5, 2002);

return d;

}

void main() {

system("chcp 1251");

Triad A, B(3, 5, 20);

cin >> A; cout << "\nТриада A " << A << "\nТриада B " << B;

if (A == B) {

cout << "\nТриады равны\n";

}

if (A > B) {

cout << "\nТриада A > Триады B\n" ;

}

if (A < B) {

cout << "\nТриада A < Триады B\n";

}

Date C;

cin >> C; cout << "\nДата 1: " << C;

Date D(12, 10, 2001);

cout << "\nДата 2: " << D;

if (C == D) {

cout << "\nДаты равны\n";

}

if (C > D) {

cout << "\nДата 2 следует после Даты 1\n";

}

if (C < D) {

cout << "\nДата 1 следует после Даты 2\n";

}

cout << "\nПодстановка";

f1(C);

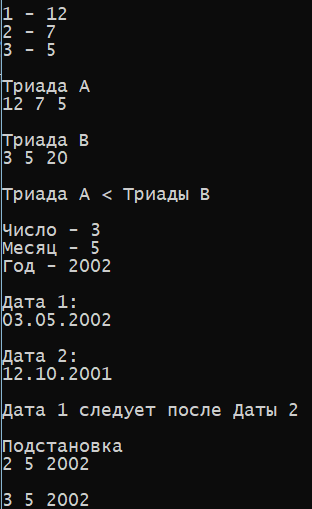
A = f2();

cout << A;

system("pause");

}

# **Скриншоты тестов**



**Контрольные вопросы**



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.



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



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



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







Конструкторы не наследуются.



Деструкторы не наследуются.









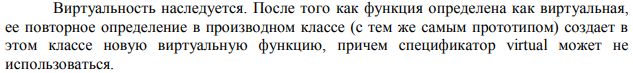




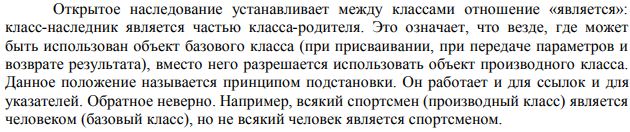










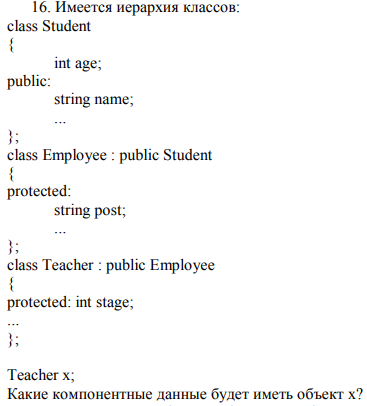








Пункт 13 и 14 – принцип подстановки.



age, name, post, stage.



Student();

Employee() : Student();

Teacher() : Employee();