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 №18.5**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Виртуальные функции. Полиморфизм

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

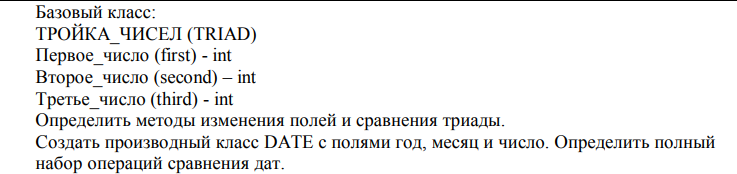
# **Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Создание иерархии классов с использованием простого наследия и абстрактного класса.
3. Изучение полиморфизма и виртуальных методов.

# **Постановка задачи**

1. Определитель абстрактный класс.
2. Определить иерархию классов в основе которой будет находится абстрактный класс.
3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
4. Перегрузить для класса Вектор операцию вывод объектов с помощью потоков.
5. Перегрузить операции ввода и вывода объектов.
6. Определить производный класс.
7. Написать демонстрационную программу, в которой продемонстрировать создание объектов и работу всех перегруженных функций.
8. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать класс Object, который будет содержать чисто виртуальную функцию Show() для вывода объектов разных классов в консоль.

class Object {

public:

Object(void) {};

~Object(void) {};

virtual void Show() = 0;

};

* 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, а также абстрактного метода Show() для класса TRIAD и DATE.

class Triad : public Object{

protected:

int first;

int second;

int third;

public:

Triad(void);

Triad(int, int, int);

Triad(const Triad&);

virtual ~Triad(void);

void Show();

int get\_first() { return first; }

int get\_second() { return second; }

int get\_third() { return third; }

void set\_first(int);

void set\_second(int);

void set\_third(int);

bool operator == (const Triad&);

bool operator > (const Triad&);

bool operator < (const Triad&);

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

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

};

class Date : public Triad {

public:

Date(void);

Date(int, int, int);

Date(const Date&);

~Date(void);

void Show();

bool operator == (const Date&);

bool operator > (const Date&);

bool operator < (const Date&);

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

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

};

* Разработать класс Vector с полями beg, size, CurrentPos,
* Организовать метод добавления указателей на объекты в Vector для класса Vector.

class Vector {

public:

Vector(void);

Vector(int);

~Vector(void);

void Add(Object\*);

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

private:

Object\*\* beg;

int size;

int CurrentPos;

};

* Протестировать работу полиморфизма Vector в функции main.

void main() {

system("chcp 1251");

Vector v(3);

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

Object\* p;

cin >> A;

p = &A;

v.Add(p);

cout << "\nТриада A";

p->Show();

p = &B;

v.Add(p);

cout << "\nТриада B";

p->Show();

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, D(12, 10, 2001);

cin >> C;

p = &C;

v.Add(p);

cout << "\nДата 1:";

p->Show();

p = &D;

v.Add(p);

cout << "\nДата 2:";

p->Show();

if (C == D) {

cout << "\nДаты равны\n";

}

if (C > D) {

cout << "\n Дата 1 следует Дате 2 \n";

}

if (C < D) {

cout << "\n Дата 2 следует Дате 1\n";

}

cout << "\nМножество укзателей:\n" << v;

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;

* Для хранения указателя на первый элемент вектора указатель реализована переменная beg типа Object\*\*.

Object\*\* beg;

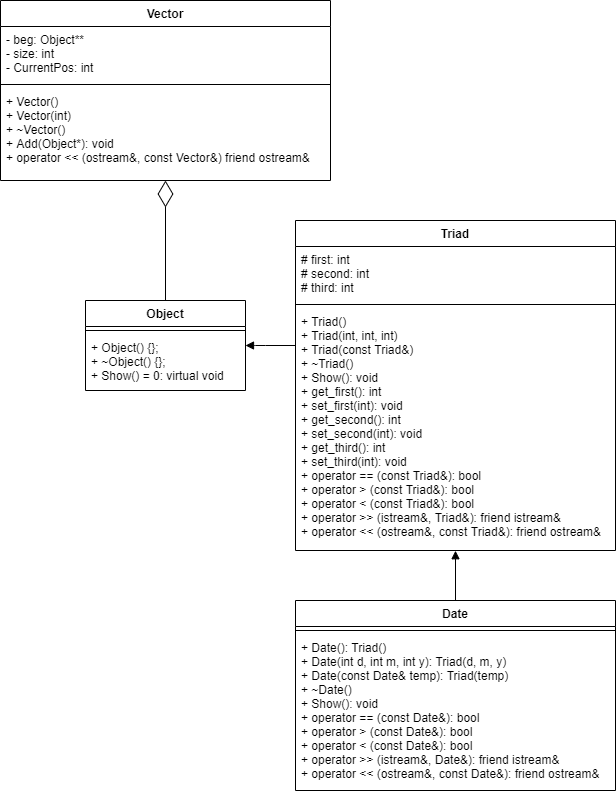
* Для хранения длины вектор реализована переменная size типа int.

int size;

* Для хранения текущей позиции в векторе реализована переменная CurrentPos типа int.

int CurrentPos;

# **UML диаграмма**



# **Код программы на языке C++**

Заголовочный файл Triad.h

#pragma once

#include <iostream>

#include "Object.h"

using namespace std;

class Triad : public Object{

protected:

int first;

int second;

int third;

public:

Triad(void);

Triad(int, int, int);

Triad(const Triad&);

virtual ~Triad(void);

void Show();

int get\_first() { return first; }

int get\_second() { return second; }

int get\_third() { return third; }

void set\_first(int);

void set\_second(int);

void set\_third(int);

bool operator == (const Triad&);

bool operator > (const Triad&);

bool operator < (const Triad&);

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

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

};

Обьявление класса в Triad.cpp

#include "Triad.h"

#include <iostream>

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::Show() {

cout << "\n" << first << " " << second << " " << third << "\n";

}

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:

Date(void);

Date(int, int, int);

Date(const Date&);

~Date(void);

void Show();

bool operator == (const Date&);

bool operator > (const Date&);

bool operator < (const Date&);

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

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

};

Обьявление класса в Date.cpp

#include "Date.h"

#include <iostream>

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

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

Date::Date(const Date& temp) {

first = temp.first;

second = temp.second;

third = temp.third;

}

Date::~Date(void) {}

void Date::Show() {

if (first < 10) {

cout << "\n0" << first << ".";

}

else {

cout << "\n" << first << ".";

}

if (second < 10) {

cout << "0" << second << ".";

}

else {

cout << second << ".";

}

cout << third << "\n";

}

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;

}

Заголовочный файл Object.h

#pragma once

#include <iostream>

class Object {

public:

Object(void) {};

~Object(void) {};

virtual void Show() = 0;

};

Заголовочный файл Vector.h

#pragma once

#include "Object.h"

#include <iostream>

using namespace std;

class Vector {

public:

Vector(void);

Vector(int);

~Vector(void);

void Add(Object\*);

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

private:

Object\*\* beg;

int size;

int CurrentPos;

};

Обьявление класса в Date.cpp

#include "Vector.h"

Vector::Vector(void){

beg = 0;

size = 0;

CurrentPos = 0;

}

Vector::Vector(int N) {

beg = new Object \* [N];

size = N;

CurrentPos = 0;

}

Vector::~Vector(void) {

if (beg != 0) { delete[]beg; }

beg = 0;

}

void Vector::Add(Object\* p) {

if (CurrentPos < size) {

beg[CurrentPos] = p;

CurrentPos++;

}

}

ostream& operator<<(ostream& out, const Vector& temp)

{

if (temp.size == 0) { out << "\nПусто\n"; }

Object\*\* p = temp.beg;

for (int i = 0; i < temp.CurrentPos; i++) {

(\*p)->Show();

p++;

}

return out; }

Main файл

#include <iostream>

#include "Triad.h"

#include "Object.h"

#include "Date.h"

#include "Vector.h"

using namespace std;

void main() {

system("chcp 1251");

Vector v(4);

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

Object\* p;

cin >> A;

p = &A;

v.Add(p);

cout << "\nТриада A";

p->Show();

p = &B;

v.Add(p);

cout << "\nТриада B";

p->Show();

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, D(12, 10, 2001);

cin >> C;

p = &C;

v.Add(p);

cout << "\nДата 1:";

p->Show();

p = &D;

v.Add(p);

cout << "\nДата 2:";

p->Show();

if (C == D) {

cout << "\nДаты равны\n";

}

if (C > D) {

cout << "\nДата 1 следует Дате 2\n";

}

if (C < D) {

cout << "\nДата 2 следует Дате 1\n";

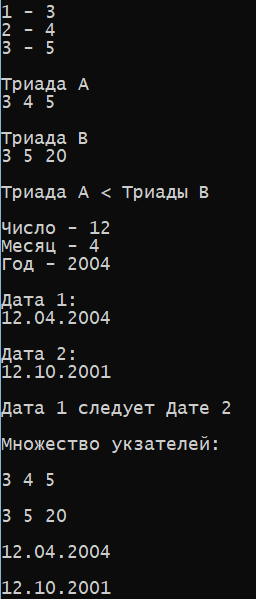
}

cout << "\nМножество укзателей:\n" << v;

system("pause");

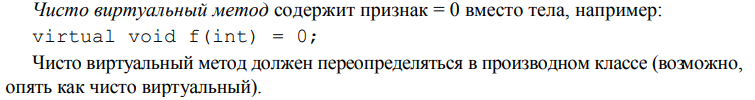
}

# **Скриншоты тестов**



**Контрольные вопросы**

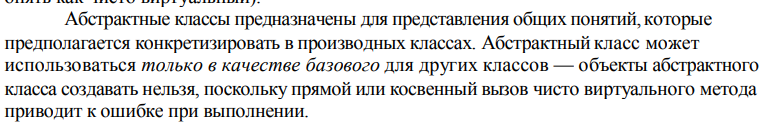














Полиморфные функции – это функции, которые работают с объектом любого типа в пределах одной иерархии.



При полиморфизме невозможно создать объект, так как базовым классом является абстрактный, когда в принципе подстановки используется наследование, что означает, что везде, где может быть использован объект базового класса, может быть и использован объект производного класса.



Пример: класс собака и класс кошка имеют одну из функций: говорить, но делают они это по-разному, так как собака гавкает, а кошка мяукает.



Метод Show, который наследуется от абстрактного класса, для вывода на консоль объектов разных классов.