Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 – «Информатика и вычислительная техника»

**ОТЧЁТ**

**Лабораторная работа №4**

**«Наследование»**

**Вариант №8**

Выполнила работу:

студентка гр. ИВТ-24-2б

Малая Алина Александровна

Проверил:

Доцент кафедры ИТАС

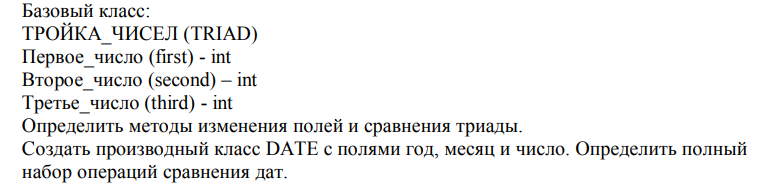
Полякова Ольга Андреевна

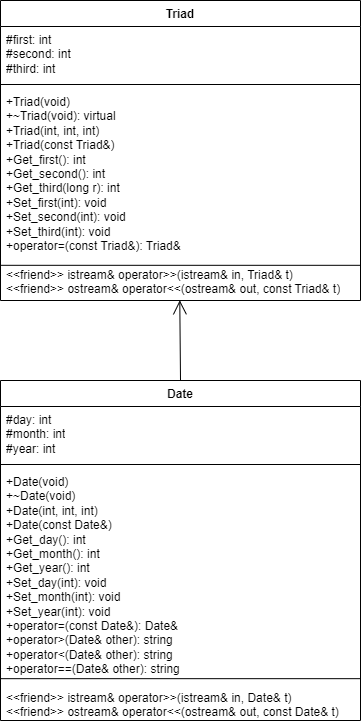
(оценка) (подпись)

(дата)

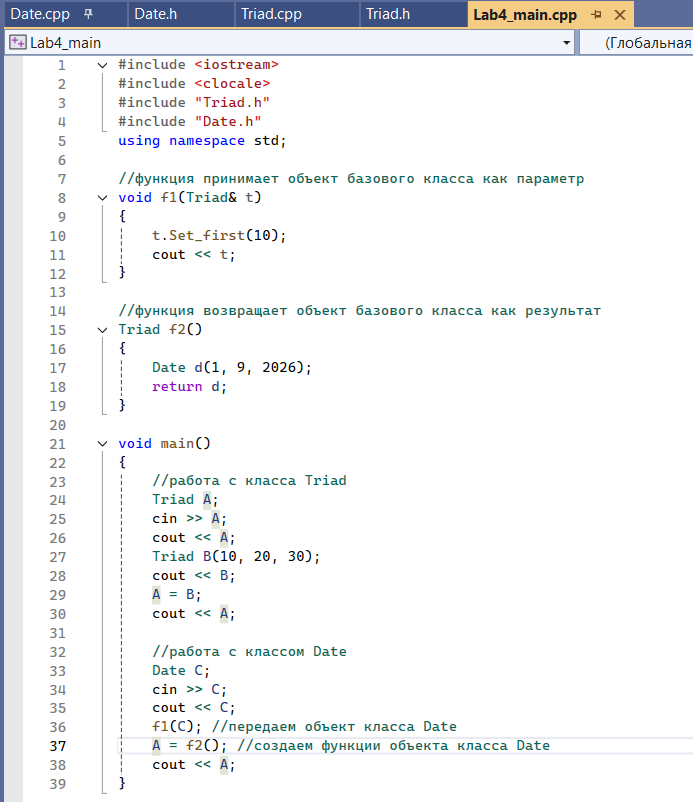
г. Пермь, 2025

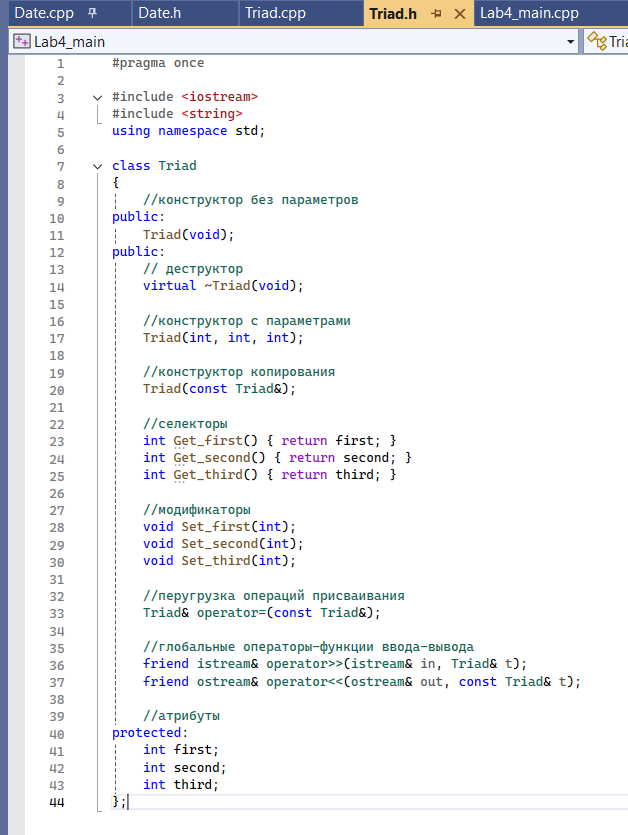
**Постановка задачи**

****

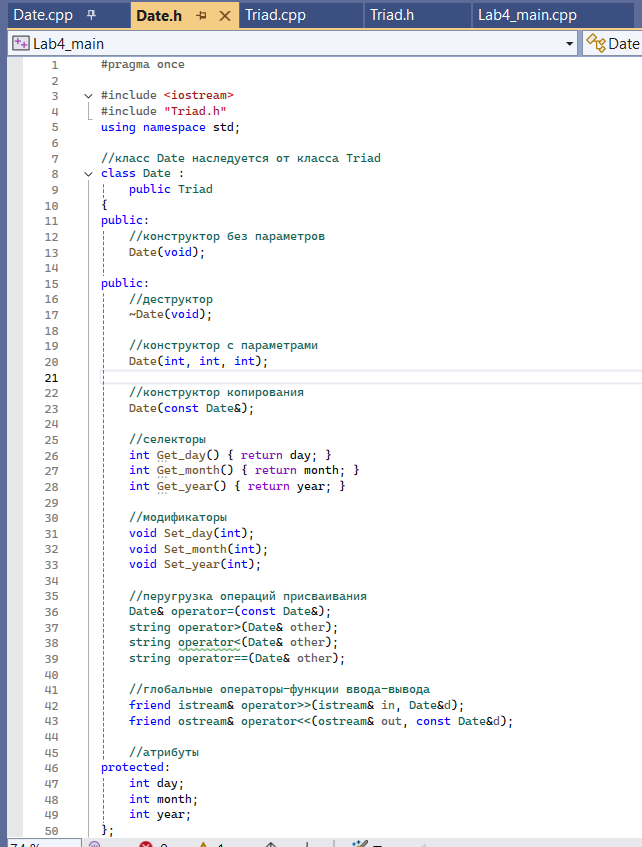
**UML Диаграмма**

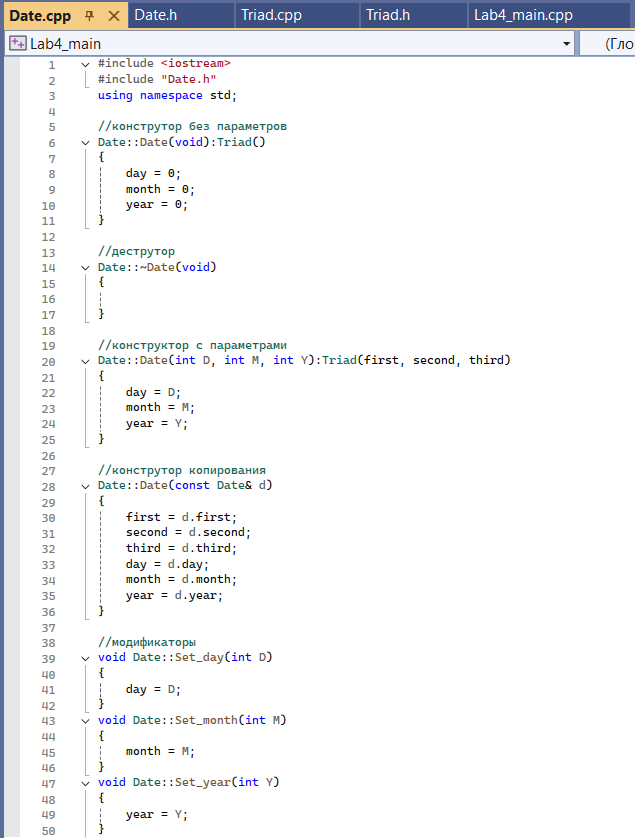
**Код программы**

****

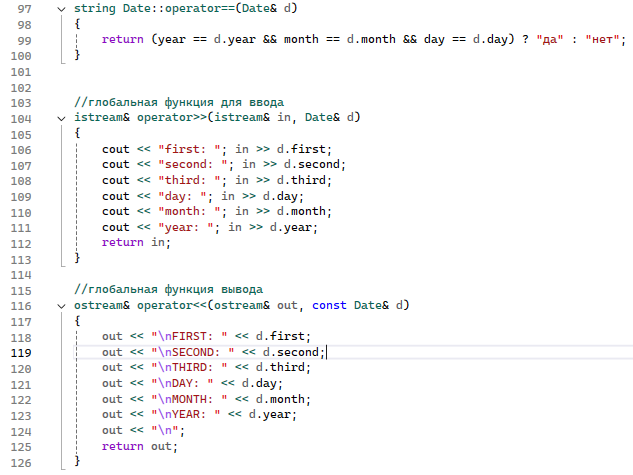
****

****

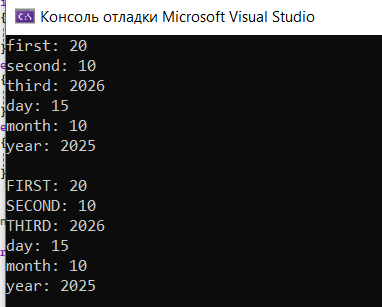
****







**Вывод**

****

**Контрольные вопросы**

1. Для чего используется механизм наследования?

— Для повторного использования кода, расширения функциональности существующих классов и организации иерархии объектов.

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

— Наследуются как открытые и защищённые члены базового класса, то есть доступны в производных классах.

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

— Не наследуются напрямую. Они доступны внутри базового класса, но не передаются в производные классы.

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

— Наследуются как открытые и защищённые члены, то есть доступны внутри производных классов.

1. Как описывается производный класс?

— Производный класс — это класс, который наследует свойства и методы другого класса (базового).

1. Наследуются ли конструкторы?

— Нет, конструкторы не наследуются, но производные классы могут вызывать конструкторы базового.

1. Наследуются ли деструкторы?

— Нет, деструкторы не наследуются, но их можно переопределить.

1. В каком порядке конструируются объекты производных классов?

— Вначале вызывается конструктор базового класса, затем — производного.

1. В каком порядке уничтожаются объекты производных классов?

— В обратном порядке: сначала вызывается деструктор производного класса, затем — базового.

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

— Виртуальные функции позволяют реализовать динамическое связывание вызова методов в иерархии классов, что обеспечивает полиморфизм.

1. Могут ли быть виртуальными конструкторы?

— Нет, конструкторы не могут быть виртуальными.

1. Наследуются ли спецификаторы virtual?

— Спецификатор virtual применяется к функциям, чтобы объявить виртуальную функцию, и наследуется по цепочке.

1. Какое отношение устанавливает между классами открытое наследование?

— Создает отношение "является" (например, Student — это Person), то есть наследование является "прямым".

1. Какое отношение устанавливает между классами закрытое наследование?

— Оно скрывает наследование, делая его недоступным извне, что редко используется.

1. В чем заключается принцип подстановки?

— Объекты производных классов могут заменять объекты базовых классов без нарушения правильности работы программы.

1. Иерархия классов

class Student

{

int age;

public:

string name;

...

};

class Employee : public Student

{

protected:

string post;

...

};

class Teacher : public Employee

{

protected:

int stage;

...

};

Объект x будет содержать компоненты данных:

- age (private, внутри класса)

- name (public)

- post (protected)

- stage (protected)

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

class Student

{

public:

int age;

string name;

Student() : age(0), name("") {}

...

};

class Employee : public Student {

public:

string post;

Employee() : Student(), post("") {}

// ...

};

class Teacher : public Employee {

public:

int stage;

Teacher() : Employee(), stage(0) {}

// ...

};

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

class Student

{

public:

int age;

string name;

Student(int a, const string& n) : age(a), name(n) {}

...

};

class Employee : public Student

{

public:

string post;

Employee(int a, const string& n, const string& p) : Student(a, n), post(p) {}

...

};

class Teacher : public Employee

{

public:

int stage;

Teacher(int a, const string& n, const string& p, int s) : Employee(a, n, p), stage(s) {}

...

};

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

class Student

{

public:

int age;

string name;

Student(const Student& s) : age(s.age), name(s.name) {}

...

};

class Employee : public Student

{

public:

string post;

Employee(const Employee& e) : Student(e), post(e.post) {}

...

};

class Teacher : public Employee

{

public:

int stage;

Teacher(const Teacher& t) : Employee(t), stage(t.stage) {}

...

};

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

class Student

{

public:

int age;

string name;

Student& operator=(const Student& s)

{

if (this != &s)

{

age = s.age;

name = s.name;

}

return \*this;

}

...

};

class Employee : public Student

{

public:

string post;

Employee& operator=(const Employee& e)

{

if (this != &e)

{

Student::operator=(e);

post = e.post;

}

return \*this;

}

...

};

class Teacher : public Employee

{

public:

int stage;

Teacher& operator=(const Teacher& t)

{

if (this != &t)

{

Employee::operator=(t);

stage = t.stage;

}

return \*this;

}

// ...

};

**GitHub**

<https://github.com/amalayaa>