Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Класс №4»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**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

**Исходные данные**

**Вариант 10**  
Базовый класс:  
ТРОЙКА\_ЧИСЕЛ(TRIAD)  
Первое\_число(first) – int  
Второе число(second) - int  
Третье число(third) - int  
Определить методы изменения полей и увелечения полей на один.  
Создать производный класс DATE с полями год, месяц и число. Переопределить методы увеличения полей на 1 и определить метод увеличения даты на n дней.

#include <iostream>

using namespace std;

class Triad

{

protected:

int first;

int second;

int third;

public:

Triad()

{

first = 0;

second = 0;

third = 0;

}

Triad(int f, int s, int t)

{

first = f;

second = s;

third = t;

}

Triad(const Triad& tmp)

{

first = tmp.first;

second = tmp.second;

third = tmp.third;

}

//сеттеры

void setFirst(int f)

{

first = f;

}

void setSecond(int s)

{

second = s;

}

void setThird(int t)

{

third = t;

}

//геттеры

int getFirst() { return first; }

int getSecond() { return second; }

int getThird() { return third; }

//Добавление единицы к объектам

void IncrementFirst()

{

first = first + 1;

}

void IncrementSecond()

{

second = second + 1;

}

void IncrementThird()

{

third = third + 1;

}

//Дружественные функции ввода и вывода

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

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

//Виртуальный диструктор

virtual ~Triad()

{

cout << "Delete Triad..." << endl << endl;

}

};

istream& operator>>(istream& in, Triad& tmp)

{

cout << "Введите первое число: "; in >> tmp.first;

cout << "Введите второе число: "; in >> tmp.second;

cout << "Введите третье число(год): "; in >> tmp.second;

return in;

}

ostream& operator<<(ostream& out, const Triad& tmp)

{

return (out << "First NUM: " << tmp.first << ", Second NUM: " << tmp.second << ", Third NUM: " << tmp.third);

}

//Класс наследуемый от Триады

class Date : public Triad

{

public:

Date()

{

first = 0;

second = 0;

third = 0;

}

Date(int f, int s, int t)

{

first = f;

second = s;

third = t;

}

Date(const Date& tmp)

{

first = tmp.first;

second = tmp.second;

third = tmp.third;

}

void setFirst(int f)

{

first = f;

}

void setSecond(int s)

{

second = s;

}

void setThird(int t)

{

third = t;

}

int getFirst() { return first; }

int getSecond() { return second; }

int getThird() { return third; }

void PlusDays(int n)

{

first = first + n;

while (first > 31)

{

first -= 31;

second += 1;

}

while (second > 12)

{

second -= 12;

third += 1;

}

}

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

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

~Date()

{

cout << endl << "Delete Date..." << endl << endl;

}

};

istream& operator>>(istream& in, Date& tmp)

{

cout << endl << "Сформируем дату: " << endl;

cout << "Введите день: "; in >> tmp.first;

cout << "Введите месяц: "; in >> tmp.second;

cout << "Введите год: "; in >> tmp.third;

return in;

}

ostream& operator<<(ostream& out, const Date& tmp)

{

return (out << endl << "Days: " << tmp.first << ", Month: " << tmp.second << ", Year: " << tmp.third << endl);

}

int main()

{

setlocale(LC\_ALL, "RUS");

//cin >> a;

//cout << a << endl;

Triad a(12, 11, 1254);

cout << "Вывод трёх заданных чисел класса Triad: " << endl << a << endl;

cout << endl << "Добавляем по единице к каждому части: " << endl;

a.IncrementFirst();

a.IncrementSecond();

a.IncrementThird();

cout << a << endl;

Date b;

cin >> b;

cout << "Вывод даты из класса Date: " << endl << b << endl;

int n;

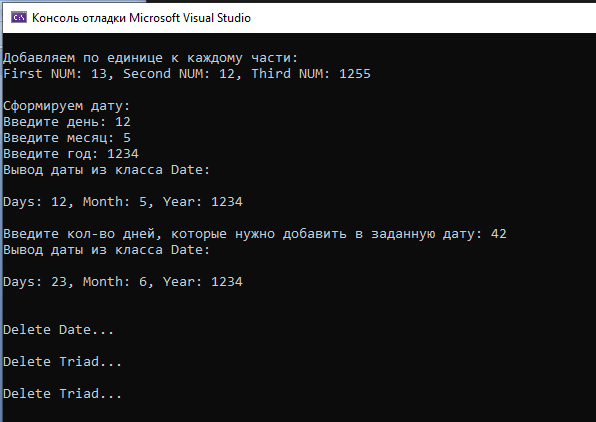
cout << "Введите кол-во дней, которые нужно добавить в заданную дату: ";

cin >> n;

b.PlusDays(n);

cout << "Вывод даты из класса Date: " << endl << b << endl;

}

**Результат**  


**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?**

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?**

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?**

Производный класс наследует описание базового класса; затем он может быть изменен добавлением новых членов, изменением существующих функций-членов и изменением прав доступа. С помощью наследования может быть создана иерархия классов, которые совместно используют код и интерфейсы.

* 1. **Наследуются ли конструкторы?**

Нет.

* 1. **Наследуются ли деструкторы?**

Нет.

* 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?**

Уничтожаютсяобъектывобратномпорядке:сначалапроизводный,потомегокомпоненты-объекты,апотомбазовыйобъект.

* 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 полиморфными и играют особую роль в ООП.

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

* 1. **Могут ли быть виртуальными конструкторы? Деструкторы?**

Конструкторы нет. Деструкторы да.

* 1. **Наследуется ли спецификато рvirtual?**

Да.

* 1. **Какое отношение устанавливает между классами открытое наследование?**

Private – не наследуется, protected – protected, public – public.

* 1. **Какое отношение устанавливает между классами закрытое наследование?**

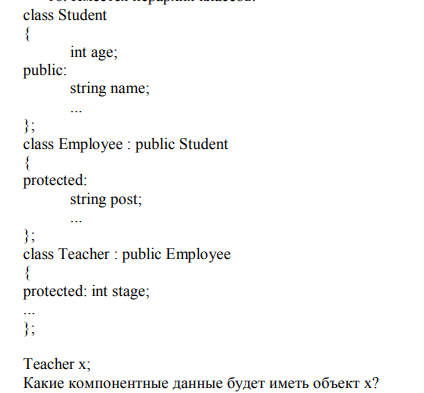
Private – не наследуется, protected – private, public – private.

* 1. **В чем заключается принцип подстановки?**

Открытое наследование устанавливает между классами отношение «является»: класс-наследник является частью класса-родителя. Это 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. Обратное неверно. Например, всякий спортсмен (производный класс) является человеком (базовый класс), но не всякий человек является спортсменом.

Закрытое наследование – это наследование реализации, в этом случае принцип подстановки не соблюдается.

* 1. **Имеется иерархия классов:**

****

Protected: int stage, string post;

Public: string name;

**17. Для классов Students, Employee и Teacher написать конструкторы** **без параметров**  
  
Students(){age = 0; name=””}

Employee(){name=””; post =””}

Teacher(){name=””;post =””;stage=0;}

**18.Для классов Student, Employee и Teacher написать конструкторы с параметрами**  
  
Students(int a, string n){age = a; name = n;}  
Employee(string n, string p){name=n; post =p;}

Teacher(string n, string p, int s){name=n; post =p; stage=s;}

**19. Для классов Student, Employee и Teacher написать конструкторы** **копирования**  
Students(const Students& s){age = s.age; name = s.name;}  
Employee(const Employee& e){name=e.name; post =e.post;}

Teacher(const Teacher& t){name=t.name; post =t.post; stage=t.stage;}

**20. Для классов Student, Employee и Teacher написать операцию присваивания**Students& operator =(const Students& s)  
{

age = s.age;

name = s.name;

return \*this;

}  
Employee& operator =(const Employee& e)

{

name=e.name;

post =e.post;

return \*this;

}

Teacher(const Teacher& t)

{

name=t.name;

post =t.post;

stage=t.stage;

return \*this;

}