Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе 18.5**

Тема: Классы и объекты. Наследование. Виртуальные функции. Полиморфизм.

Выполнил работу

студент группы РИС-20-1б

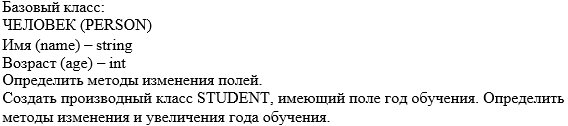
Томилов В.А.

Проверила

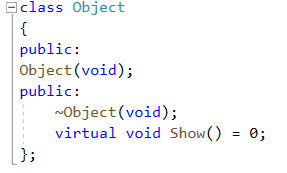
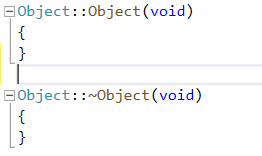
Доцент кафедры ИТАС

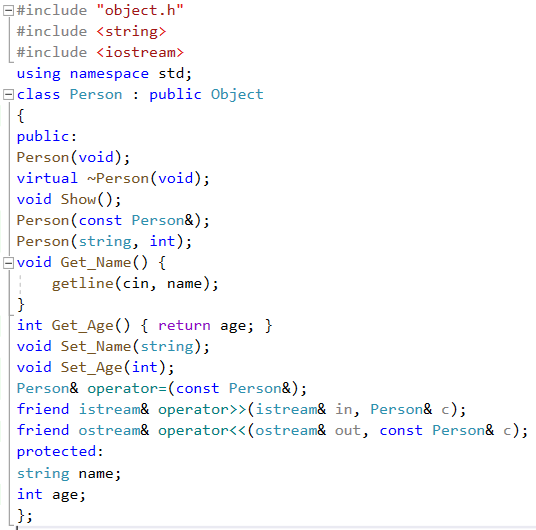
к.т.н ПоляковаО.А.

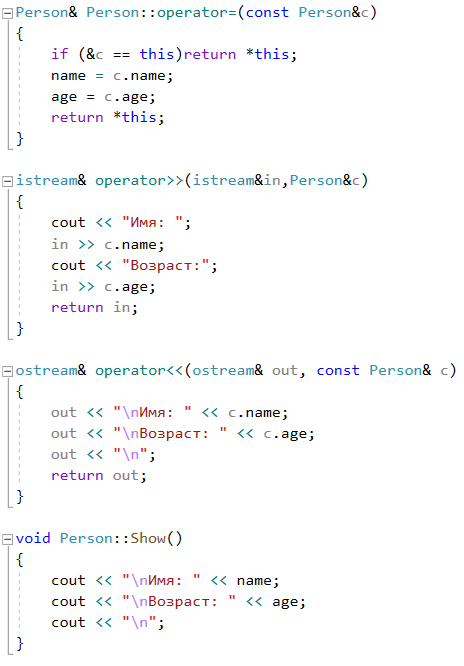
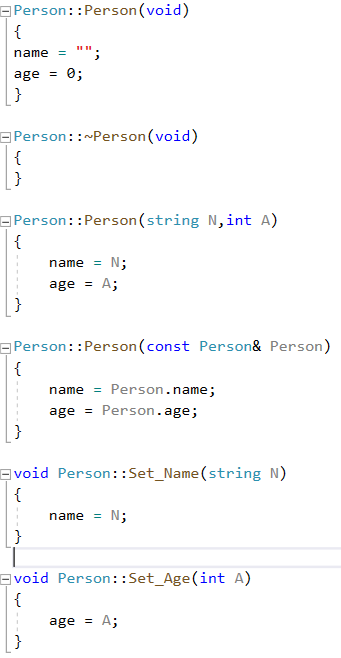
Пермь, 2021

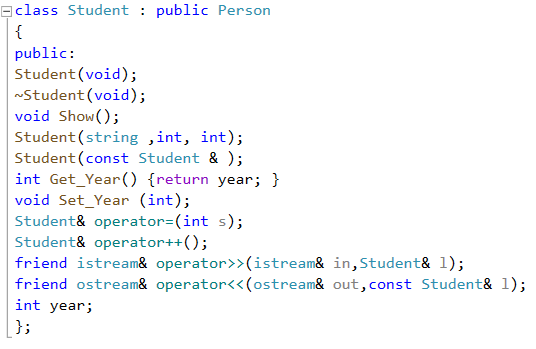
Постановка задачи

Анализ задачи

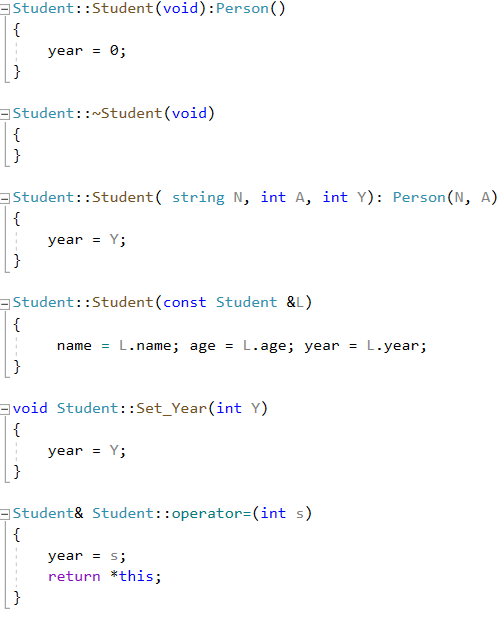
* 1. Для решения задачи необходимо
  2. Добавить в проект файл object.h, содержащий описание класса.
  3.  Добавим в проект файл object.cpp, содержащий описание методов класса object.
  4. Добавить в проект файл person.h, содержащий описание класса.

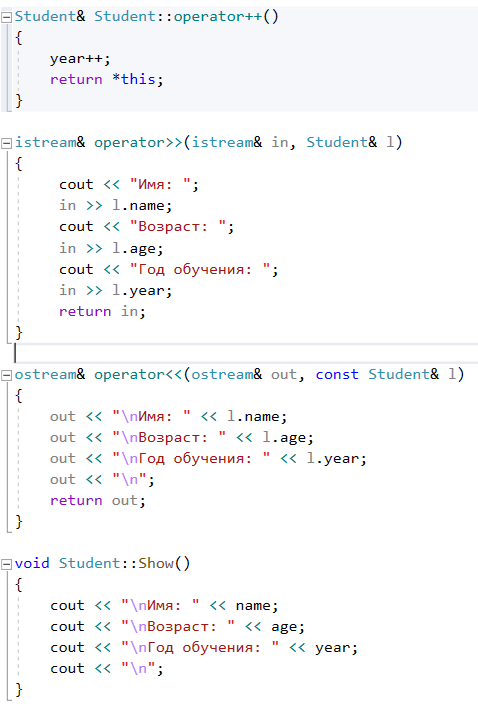


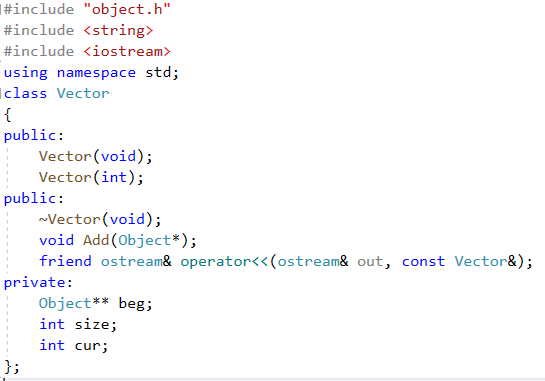
* 1. Добавим в проект файл person.cpp, содержащий описание методов класса person.
  2. Добавить в проект файл student.h, содержащий описание класса.

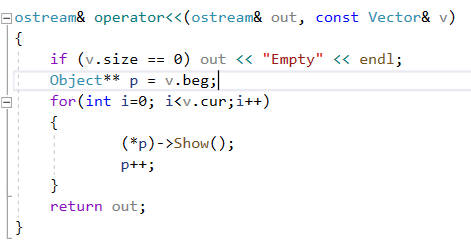


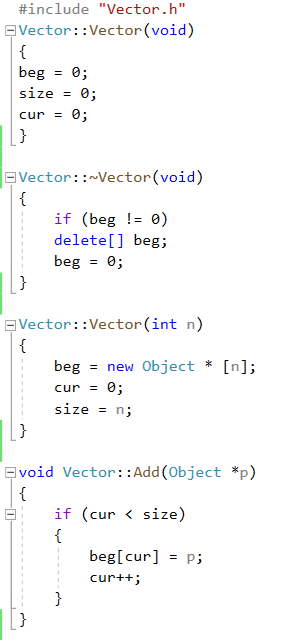
* 1. Добавим в проект файл student.cpp, содержащий описание методов класса student.

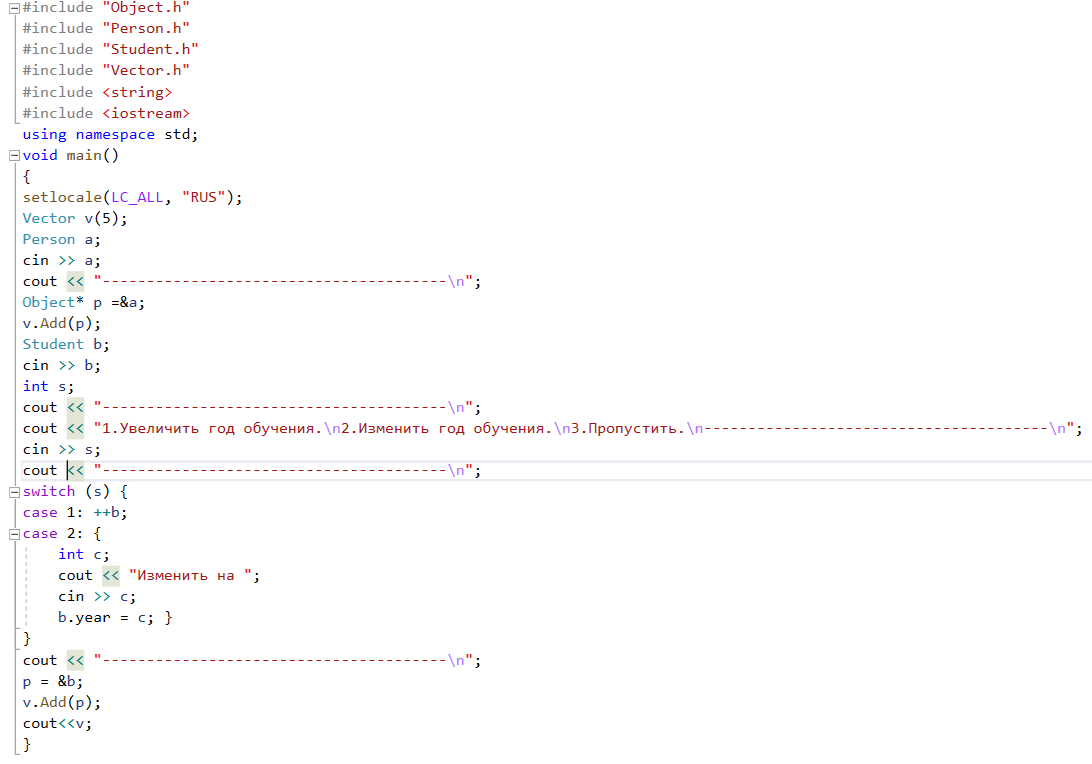




* 1. Добавить в проект файл vector.h, содержащий описание класса
  2. Добавим в проект файл vector.cpp, содержащий описание методов класса vector.



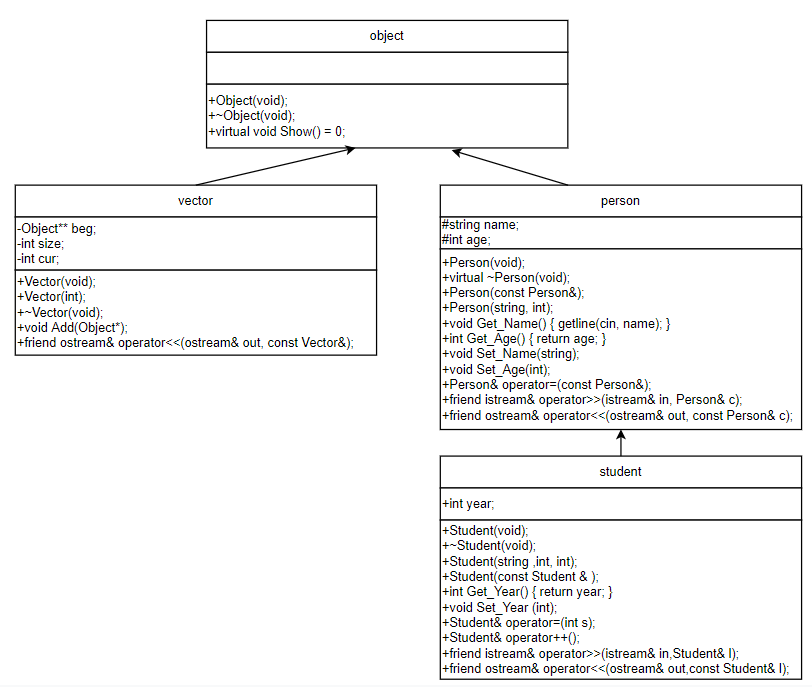


* 1. Добавить в проект файл Lab5\_main.cpp, содержащий основную программу.
  2. В ходе работы были использованы следующие типы данных:

2.1 (int) age, year, size, cur – возраст и год обучения студента, размер и текущая позиция вектора

2.2 (string) name – ФИО студента

2.3 (Object\*\*) beg – указатель на начальный элемент вектора

UML – диаграмма

Код программы

Main.cpp

#include "Object.h"

#include "Person.h"

#include "Student.h"

#include "Vector.h"

#include <string>

#include <iostream>

using namespace std;

void main()

{

setlocale(LC\_ALL, "RUS");

Vector v(5);

Person a;

cin >> a;

cout << "---------------------------------------\n";

Object\* p =&a;

v.Add(p);

Student b;

cin >> b;

int s;

cout << "---------------------------------------\n";

cout << "1.Увеличить год обучения.\n2.Изменить год обучения.\n3.Пропустить.\n---------------------------------------\n";

cin >> s;

cout << "---------------------------------------\n";

switch (s) {

case 1: ++b;

case 2: {

int c;

cout << "Изменить на ";

cin >> c;

b.year = c; }

}

cout << "---------------------------------------\n";

p = &b;

v.Add(p);

cout<<v;

}

Object.h

#pragma once

class Object

{

public:

Object(void);

public:

~Object(void);

virtual void Show() = 0;

};

Object.cpp

#include "object.h"

Object::Object(void)

{

}

Object::~Object(void)

{

}

Person.h

#pragma once

#include "object.h"

#include <string>

#include <iostream>

using namespace std;

class Person : public Object

{

public:

Person(void);

virtual ~Person(void);

void Show();

Person(const Person&);

Person(string, int);

void Get\_Name() {

getline(cin, name);

}

int Get\_Age() { return age; }

void Set\_Name(string);

void Set\_Age(int);

Person& operator=(const Person&);

friend istream& operator>>(istream& in, Person& c);

friend ostream& operator<<(ostream& out, const Person& c);

protected:

string name;

int age;

};

Person.cpp

#include "person.h"

Person::Person(void)

{

name = "";

age = 0;

}

Person::~Person(void)

{

}

Person::Person(string N,int A)

{

name = N;

age = A;

}

Person::Person(const Person& Person)

{

name = Person.name;

age = Person.age;

}

void Person::Set\_Name(string N)

{

name = N;

}

void Person::Set\_Age(int A)

{

age = A;

}

Person& Person::operator=(const Person&c)

{

if (&c == this)return \*this;

name = c.name;

age = c.age;

return \*this;

}

istream& operator>>(istream&in,Person&c)

{

cout << "Имя: ";

in >> c.name;

cout << "Возраст:";

in >> c.age;

return in;

}

ostream& operator<<(ostream& out, const Person& c)

{

out << "\nИмя: " << c.name;

out << "\nВозраст: " << c.age;

out << "\n";

return out;

}

void Person::Show()

{

cout << "\nИмя: " << name;

cout << "\nВозраст: " << age;

cout << "\n";

}

Student.h

#pragma once

#include "person.h"

class Student : public Person

{

public:

Student(void);

~Student(void);

void Show();

Student(string ,int, int);

Student(const Student & );

int Get\_Year() {return year; }

void Set\_Year (int);

Student& operator=(int s);

Student& operator++();

friend istream& operator>>(istream& in,Student& l);

friend ostream& operator<<(ostream& out,const Student& l);

int year;

};

Student.cpp

#include "Student.h"

Student::Student(void):Person()

{

year = 0;

}

Student::~Student(void)

{

}

Student::Student( string N, int A, int Y): Person(N, A)

{

year = Y;

}

Student::Student(const Student &L)

{

name = L.name; age = L.age; year = L.year;

}

void Student::Set\_Year(int Y)

{

year = Y;

}

Student& Student::operator=(int s)

{

year = s;

return \*this;

}

Student& Student::operator++()

{

year++;

return \*this;

}

istream& operator>>(istream& in, Student& l)

{

cout << "Имя: ";

in >> l.name;

cout << "Возраст: ";

in >> l.age;

cout << "Год обучения: ";

in >> l.year;

return in;

}

ostream& operator<<(ostream& out, const Student& l)

{

out << "\nИмя: " << l.name;

out << "\nВозраст: " << l.age;

out << "\nГод обучения: " << l.year;

out << "\n";

return out;

}

void Student::Show()

{

cout << "\nИмя: " << name;

cout << "\nВозраст: " << age;

cout << "\nГод обучения: " << year;

cout << "\n";

}

Vector.h

#pragma once

#include "object.h"

#include <string>

#include <iostream>

using namespace std;

class Vector

{

public:

Vector(void);

Vector(int);

public:

~Vector(void);

void Add(Object\*);

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

private:

Object\*\* beg;

int size;

int cur;

};

Vector.cpp

#include "Vector.h"

Vector::Vector(void)

{

beg = 0;

size = 0;

cur = 0;

}

Vector::~Vector(void)

{

if (beg != 0)

delete[] beg;

beg = 0;

}

Vector::Vector(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

void Vector::Add(Object \*p)

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

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

{

if (v.size == 0) out << "Empty" << endl;

Object\*\* p = v.beg;

for(int i=0; i<v.cur;i++)

{

(\*p)->Show();

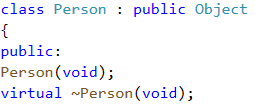
p++;

}

return out;

}

Ответы на вопросы

1. Чисто виртуальный метод содержит признак = 0 вместо тела.
2. Класс, содержащий хотя бы один чисто виртуальный метод, называется абстрактным
3. Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах
4. Таким образом, можно создать функцию, параметром которой является указатель на абстрактный класс. На место этого параметра при выполнении программы может передаваться указатель на объект любого производного класса
5. Это позволяет создавать полиморфные функции, работающие с объектом любого типа в пределах одной иерархии
6. 
7. void Add(Object\*);

void Vector::Add(Object \*p)

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

1. Позднее связывание означает, что объект связывается с вызовом функции только во время исполнения программы, а не раньше