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 18.4**

Тема: Классы и объекты. Простое наследование. Принцип подстановки.

Выполнил работу

студент группы РИС-20-1б

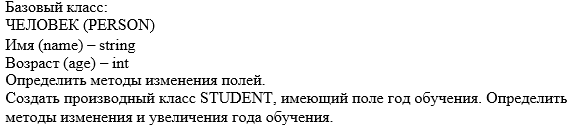
Томилов В.А.

Проверила

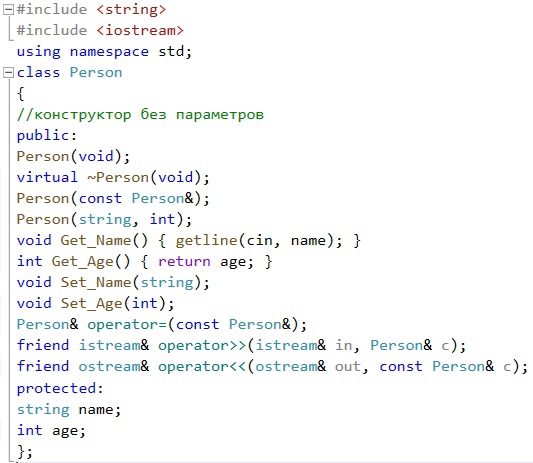
Доцент кафедры ИТАС

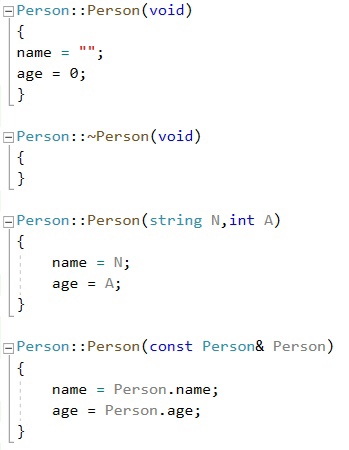
к.т.н ПоляковаО.А.

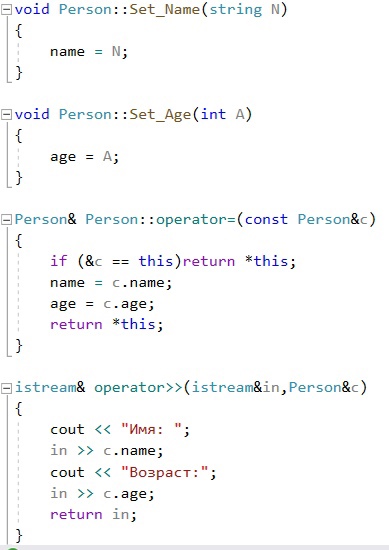
Пермь, 2021

Постановка задачи

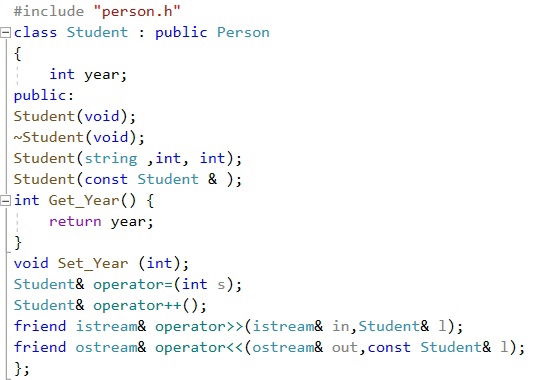
Анализ задачи

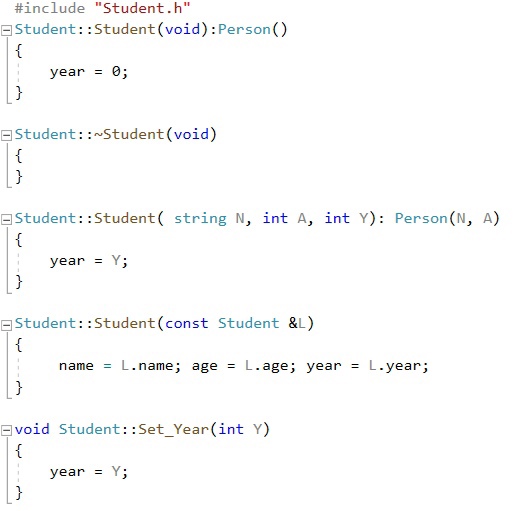
* 1. Для решения задачи необходимо
  2. Добавить в проект файл Person.h, содержащий описание класса.
  3. Добавим в проект файл Person.cpp, содержащий описание методов класса person.

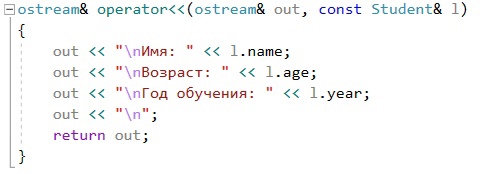


* 1. Добавить в проект файл student.h, содержащий описание класса.

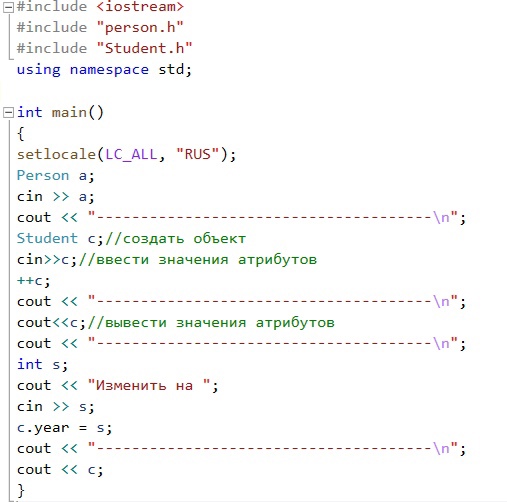


* 1. Добавим в проект файл student.cpp, содержащий описание методов класса student.





* 1. Добавить в проект файл Lab4\_main.cpp, содержащий основную программу.

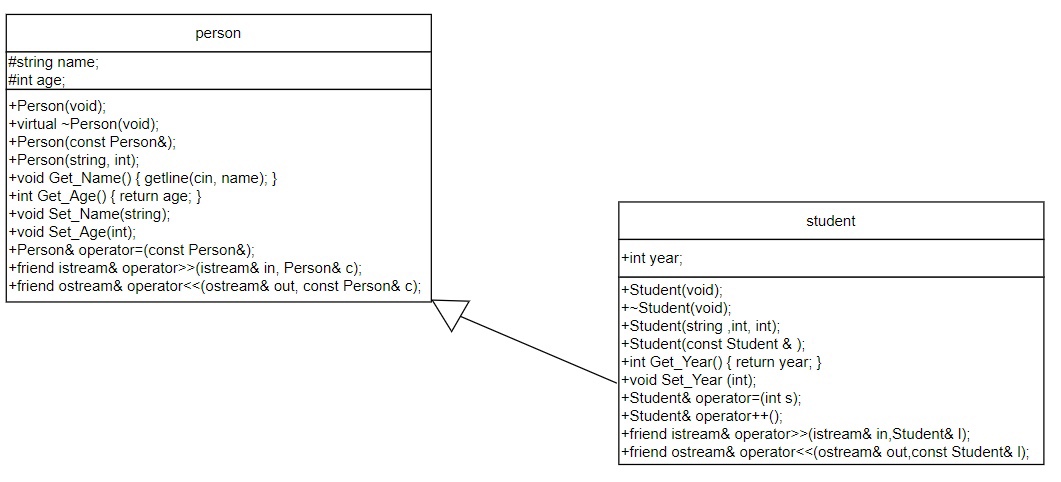


* 1. В ходе работы были использованы следующие типы данных:

2.1 (int) age, year – возраст и год обучения Студента

2.2 (strinfg) name – ФИО студента

UML – диаграмма



Код программы

Main.cpp

[#include](https://vk.com/im?sel=223733620&st=%23include) <iostream>  
[#include](https://vk.com/im?sel=223733620&st=%23include) "person.h"  
[#include](https://vk.com/im?sel=223733620&st=%23include) "Student.h"  
using namespace std;  
  
int main()  
{  
setlocale(LC\_ALL, "RUS");  
Person a;  
cin » a;  
cout « "--------------------------------------\n";  
Student c;  
cin»c;  
++c;  
cout « "--------------------------------------\n";  
cout«c;  
cout « "--------------------------------------\n";  
int s;  
cout « "Изменить на ";  
cin » s;  
c.year = s;  
cout « "--------------------------------------\n";  
cout « c;  
}

Person.h

[#pragma](https://vk.com/im?sel=223733620&st=%23pragma) once  
[#include](https://vk.com/im?sel=223733620&st=%23include) <string>  
[#include](https://vk.com/im?sel=223733620&st=%23include) <iostream>  
using namespace std;  
class Person  
{  
public:  
Person(void);  
virtual ~Person(void);  
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](https://vk.com/im?sel=223733620&st=%23include) "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](https://vk.com/away.php?utf=1&to=http%3A%2F%2FPerson.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](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fc.name);  
age = c.age;  
return \*this;  
}  
  
istream& operator»(istream&in,Person&c)  
{  
cout « "Имя: ";  
in » [c.name](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fc.name);  
cout « "Возраст:";  
in » c.age;  
return in;  
}  
  
ostream& operator«(ostream& out, const Person& c)  
{  
out « "\nИмя: " « [c.name](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fc.name);  
out « "\nВозраст: " « c.age;  
out « "\n";  
return out;  
}

Student.h

[#pragma](https://vk.com/im?sel=223733620&st=%23pragma) once  
[#include](https://vk.com/im?sel=223733620&st=%23include) "person.h"  
class Student : public Person  
{  
  
public:  
int year;  
Student(void);  
~Student(void);  
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);  
};

Student.cpp

[#include](https://vk.com/im?sel=223733620&st=%23include) "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](https://vk.com/away.php?utf=1&to=http%3A%2F%2FL.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](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fl.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](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fl.name);  
out « "\nВозраст: " « l.age;  
out « "\nГод обучения: " « l.year;  
out « "\n";  
return out;  
}

Ответы на вопросы

1. Наследование — концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.
2. public

public-члены базового класса — доступны как public-члены производного класса;

protected-члены базового класса — доступны как protected-члены производного класса;

1. private

public- и protected-члены базового класса — доступны как private-члены производного класса.

1. protected

public- и protected-члены базового класса — доступны как protected-члены производного класса;

1. сlass Имя : МодификаторДоступа ИмяБазовогоКласса

{ объявление\_членов;};

1. нет
2. нет
3. Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты- объекты (если они имеются), а потом сам производный класс.
4. Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты- объекты (если они имеются), а потом сам производный класс.
5. Виртуальные функции предоставляют механизм позднего (отложенного) или динамического связывания. Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используется ключевое слово virtual.
6. Нет. Да
7. Да
8. Открытое наследование устанавливает между классами отношение «является»: класс-наследник является частью класса-родителя
9. Закрытое наследование – это наследование реализации, в этом случае принцип подстановки не соблюдается.
10. 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.