

Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт
Высшая школа фундаментальных физических исследований

Лабораторная работа

Наследование в C++
По дисциплине "Специальный практикум"

Выполнил
студент гр. 5040302/10301

С. А. Буланова

Научный руководитель:
д.ф.-м.н.

Я. А. Бердников

«_____» _____ 2021 г.

Санкт-Петербург
2021

Задание

Реализация наследования в классах на C++

Создать родительский класс `Person`, которому будут наследовать классы `Lecturer` и `Student`. Класс `Person` хранит имя, фамилию и ID в базе данных университета. Класс `Student` наследует их от класса `Person`, при этом хранит еще и номер группы студента. Класс `Lecturer` наследует от класса `Person` и хранит должность преподавателя.

Приведем полученный код:

```
1 //lab5
2 #include <iostream>
3 using namespace std;
4
5 class Person {
6 protected:
7     std::string fName;
8     std::string fSurname;
9     unsigned int fId;
10
11 public:
12     Person(
13         const std::string & name,
14         const std::string & surname,
15         unsigned int id): fName(name), fSurname(surname), fId(id){
16         std::cout << "New person registered: " << std::endl;
17     }
18     virtual void Print() const{
19         std::cout << "\tName: " << fName << std::endl;
20         std::cout << "\tSurname: " << fSurname << std::endl;
21         std::cout << "\tID: " << fId << std::endl;
22     }
23
24     virtual ~Person(){
25         std::cout << "Person is deleted" << std::endl;
26     };
27 };
28
29 class Student: public Person{
30 protected:
31     std::string fGroupNo;
32 public:
33     Student(
34         std::string name,
35         std::string surname,
36         std::string groupNo,
37         unsigned int id) : Person(name, surname, id), fGroupNo(groupNo)
38     {
```

```

39     std::cout << "New Student registered" << std::endl;
40 }
41 void Print() const{
42     std::cout << "\tName: " << fName << std::endl;
43     std::cout << "\tSurname: " << fSurname << std::endl;
44     std::cout << "\tID: " << fId << std::endl;
45     std::cout << "\tGroup ID: " << fGroupNo << std::endl;
46 }
47
48 ~Student(){
49     std::cout << "Student deleted" << std::endl;
50 }
51 };
52
53 class Lecturer: public Person{
54 protected:
55     std::string fDegree;
56 public:
57     Lecturer(
58         std::string name,
59         std::string surname,
60         std::string degree,
61         unsigned int id) : Person(name, surname, id), fDegree(degree)
62     {
63         std::cout << "New Lecturer registered" << std::endl;
64     }
65
66     void Print() const{
67         std::cout << "\tName: " << fName << std::endl;
68         std::cout << "\tSurname: " << fSurname << std::endl;
69         std::cout << "\tID: " << fId << std::endl;
70         std::cout << "\tDegree: " << fDegree << std::endl;
71     }
72
73     ~Lecturer(){
74         std::cout << "Lecturer deleted" << std::endl;
75     }
76 };
77
78 int main(){

```

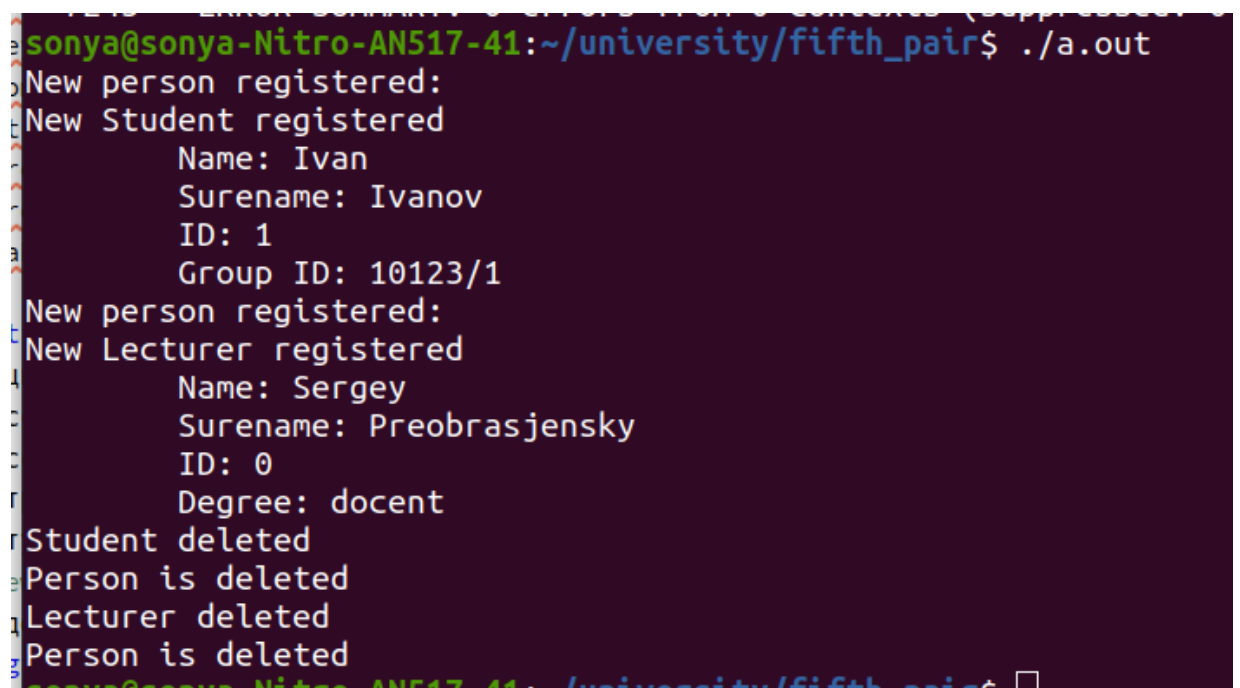
```

79  Person * student = new Student("Ivan", "Ivanov", "10123/1", 1);
80  student -> Print();
81  Person * lecturer = new Lecturer("Sergey", "Preobrasjensky", "docent", 0);
82  lecturer -> Print();
83
84  delete student;
85  delete lecturer;
86  return 0;
87  }

```

Спецификатор доступа `protected` разрешает доступ дочерним классам к членам класса-родителя. Чтобы родительский указатель мог вызвать методы дочернего класса (строки 79, 81), необходимо сделать методы `virtual`. Аналогично, чтобы деструктор удалял дочерний объект, а не родительский, деструктор делается виртуальным.

Продemonстрируем работу программы:



```

sonya@sonya-Nitro-AN517-41:~/university/fifth_pair$ ./a.out
New person registered:
New Student registered
    Name: Ivan
    Surname: Ivanov
    ID: 1
    Group ID: 10123/1
New person registered:
New Lecturer registered
    Name: Sergey
    Surname: Preobrasjensky
    ID: 0
    Degree: docent
Student deleted
Person is deleted
Lecturer deleted
Person is deleted
sonya@sonya-Nitro-AN517-41:~/university/fifth_pair$

```