Наследование классов в C++

Наследование — это определение производного класса, который может обращаться ко всем элементам и методам базового класса за исключением тех, которые находятся в поле private;

Цель объектно-ориентированного программирования состоит в **повторном использовании** созданных вами классов, что экономит время и силы.

Если вы уже создали некоторый класс, то возможны ситуации, когда новому классу нужны многие или даже все особенности уже существующего класса, и необходимо добавить один или несколько элементов данных или функций.

В таких случаях C++ позволяет вам строить новый объект, используя характеристики уже существующего объекта. Т.е. новый объект будет наследовать элементы существующего класса (называемого **базовым классом**). Когда вы строите новый класс из существующего, этот новый класс часто называется **производным классом**.

Наследование в C++

```
class A{ //базовый класс
};
class B : public A{ //public наследование
};
class C : protected A{ //protected наследование
};
class Z : private A{ //private наследование
};
```

Синтаксис определения производного класса:

class Имя_Производного_Класса : спецификатор доступа Имя_Базового_Класса

```
{
 /*код*/

};
```

Отличия спецификаторов доступа

	private	**protected**	**public**
Доступ из тела класса	открыт	открыт	открыт
Доступ из производных классов	закрыт	открыт	открыт
Доступ из внешних функций и классов	закрыт	закрыт	открыт

Спецификаторы доступа членов базового класса меняются в потомках следующим образом:

Если класс объявлен как базовый для другого класса со спецификатором доступа **public**, тогда **public** члены базового класса доступны как **public** члены производного класса, **protected** члены базового класса доступны как **protected** члены производного класса.

Если класс объявлен как базовый для другого класса со спецификатором доступа **protected**, тогда **public** и **protected** члены базового класса доступны как **protected** члены производного класса.

Если класс объявлен как базовый для другого класса со спецификатором доступа **private**, тогда **public** и **protected** члены базового класса доступны как **private** члены производного класса.

```cpp
#include <iostream>
#include <conio.h>
using namespace std;
class ClassA{// базовый класс
protected: // спецификатор доступа к элементу value
    int value;
public:
    ClassA(int input){//Конструктур с параметром
        value = input;
    }
    void show_value(){
        cout << value << endl;
    }
};
class ClassB : public ClassA{   // производный класс
public:
// inputS передается в конструктор с параметром класса ClassA
    ClassB(int inputS) : ClassA (inputS){
    }
// Ввозводим value в квадрат.
// Без спецификатора доступа protected
// эта функция не могла бы изменить значение value
    void ValueSqr(){
        value *= value;
    }
};
```

```
int main(void){
    setlocale(LC_ALL, "rus");

    ClassA object_A(3);     // объект базового класса
    cout << "значение object_A = ";
    object_A.show_value();

    ClassB object_B(4);    // объект производного класса
    cout << "значение object_B = ";
    object_B.show_value();  // вызов метода базового класса

    object_B.ValueSqr();        // возводим value в квадрат
    cout << "квадрат значения object_B = ";
    object_B.show_value();

 // базовый класс не имеет доступа к методам производного класса
    //object_A.ValueSqr();
    cout << endl;
    return 0;
}
```

Задача.

Пусть в базе данных ВУЗа должна храниться информация о всех студентах и преподавателях. Представлять все данные в одном классе не получится, поскольку для преподавателей понадобится хранить данные, которые для студента не применимы, и наоборот.

Создание базового класса. базовый класс human, который будет описывать модель человека. В нем будут храниться имя, фамилия и отчество. human.h:

```
#ifndef HUMAN_H_INCLUDED
#define HUMAN_H_INCLUDED
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
class human {
    public:
        // Конструктор класса human
        human(string last_name, string name, string second_name){
            this->last_name = last_name;
            this->name = name;
            this->second_name = second_name;
        }
        // Получение ФИО человека
        string get_full_name(){
            ostringstream full_name;
            full_name << this->last_name << " "
                << this->name << " "
                << this->second_name;
            return full_name.str();
        }
    private:
        string name; // имя
        string last_name; // фамилия
        string second_name; // отчество
};
#endif // HUMAN_H_INCLUDED
```

Наследование от базового класса

```cpp
#include "human.h"
#include <string>
#include <vector>
using namespace std;
class student : public human {
    public:
        // Конструктор класса Student
        student(string last_name,string name,string second_name,vector<int> scores) : human(
            last_name,name,second_name) {
            this->scores = scores;
        }
        // Получение среднего балла студента
        float get_average_score()
        {
            // Общее количество оценок
            unsigned int count_scores = this->scores.size();
            // Сумма всех оценок студента
            unsigned int sum_scores = 0;
            // Средний балл
            float average_score;

            for (unsigned int i = 0; i < count_scores; ++i) {
                sum_scores += this->scores[i];
            }

            average_score = (float) sum_scores / (float) count_scores;
            return average_score;
        }

    private:
        // Оценки студента
        vector<int> scores;
};
```

```cpp
#include <iostream>
#include <vector>
#include <locale.h>
#include "human.h"
#include "student.h"
using namespace std;
int main(int argc, char* argv[]){
    // Оценки студента
    vector<int> scores;
    setlocale(LC_ALL,"rus");//задаем локализацию (русскую)

    // Добавление оценок студента в вектор
    scores.push_back(5);
    scores.push_back(3);
    scores.push_back(2);
    scores.push_back(2);
    scores.push_back(5);
    scores.push_back(3);
    scores.push_back(3);
    scores.push_back(3);
    scores.push_back(3);

    // Создание объекта класса student
    student *stud = new student("Петров", "Иван", "Алексеевич", scores);

    // Вывод полного имени студента (используется унаследованный метод класса human)
    cout << stud->get_full_name() << endl;
    // Вывод среднего балла студента
    cout << "Средний балл: " << stud->get_average_score() << endl;

    system("pause");//Приостанавливаем выполнение чтобы посмотреть результат
    return 0;
}
```