**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 21**

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Основы алгоритмов и программирование»

|  |  |
| --- | --- |
| Преподаватель:  Яковлева Н.М.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_202\_\_г.  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил:  студент группы Y2231  Петров Н.В. |

Санкт-Петербург

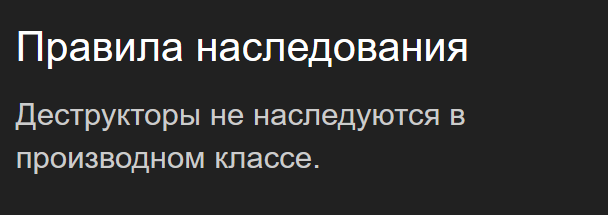
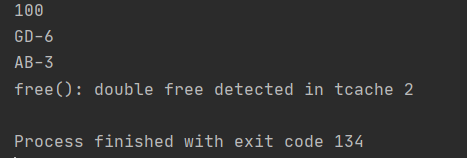
2019/2020

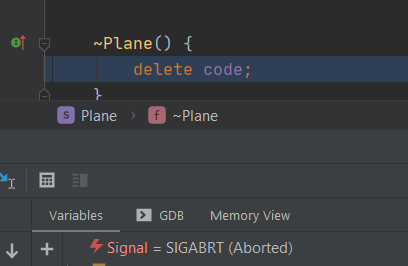
**Код:**

1. main.cpp

#include <iostream>  
#include <cstring>  
  
class Unit {  
protected:  
 int health;  
public:  
 Unit() {  
 health = 100;  
 }  
  
 ~Unit() = default;  
  
 Unit &operator=(const Unit &t) {  
 if (this == &t) return \*this;  
 health = t.health;  
 return \*this;  
 }  
  
 virtual int get\_health() = 0;  
};  
  
class Car : public Unit {  
protected:  
 int engine;  
 int wheel;  
 char \*id;  
public:  
 Car() {  
 engine = 1;  
 wheel = 4;  
 id = new char[5];  
 strcpy(id, "GD-6");  
 }  
  
 ~Car() {  
 delete id;  
 }  
  
 Car &operator=(const Car &t) {  
 if (this == &t) return \*this;  
 health = t.health;  
 engine = t.engine;  
 wheel = t.wheel;  
 id = new char[strlen(t.id) + 1];  
 strcpy(id, t.id);  
 return \*this;  
 }  
  
 int get\_health() {  
 return this->health;  
 }  
  
 int get\_engine() {  
 return this->engine;  
 }  
  
 int get\_wheel() {  
 return this->wheel;  
 }  
  
 char \*get\_id() {  
 return this->id;  
 }  
  
};  
  
class Plane : public Unit {  
protected:  
 int engine;  
 int prop;  
 char \*code;  
public:  
 Plane() {  
 engine = 2;  
 prop = 2;  
 code = new char[5];  
 strcpy(code, "AB-3");  
 }  
  
 ~Plane() {  
 delete code;  
 }  
  
 Plane &operator=(const Plane &t) {  
 if (this == &t) return \*this;  
 health = t.health;  
 engine = t.engine;  
 prop = t.prop;  
 code = new char[strlen(t.code) + 1];  
 strcpy(code, t.code);  
 return \*this;  
 }  
  
 int get\_health() {  
 return this->health;  
 }  
  
 int get\_engine() {  
 return this->engine;  
 }  
  
 int get\_prop() {  
 return this->prop;  
 }  
  
 char \*get\_code() {  
 return this->code;  
 }  
  
};  
  
class CarPlane : public Car, public Plane {  
 char \*wtfisthis;  
public:  
 explicit CarPlane(char \*str = "I don't know") {  
 this->wtfisthis = new char[strlen(str) + 1];  
 strcpy(this->wtfisthis, str);  
 }  
  
 ~CarPlane() {  
 delete this->id;  
 delete this->code;  
 delete this->wtfisthis;  
 }  
  
 CarPlane &operator=(const CarPlane &t) {  
 if (this == &t) return \*this;  
 Car::health = t.Car::health;  
 Car::engine = t.Car::engine;  
 wheel = t.wheel;  
 wtfisthis = new char[strlen(t.wtfisthis) + 1];  
 strcpy(wtfisthis, t.wtfisthis);  
 id = new char[strlen(t.id) + 1];  
 strcpy(id, t.id);  
 code = new char[strlen(t.code) + 1];  
 strcpy(code, t.code);  
 return \*this;  
 }  
};  
  
int main() {  
 CarPlane item = CarPlane();  
 std::cout << item.Car::get\_health() << std::endl;  
 std::cout << item.get\_id() << std::endl;  
 std::cout << item.get\_code() << std::endl;  
 return 0;  
}

Класс CarPlane имеет 2ух предков, класс Unit абстрактный, т.к. имеет виртуальный метод get\_health().





Деструктор кстати наследуется, во всяком случае вызывается после вызова деструктора наследника, вызывается деструктор родителя.