Лабораторная работа №5

Программная реализация отношения простого наследования

Цель работы:

Изучить способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса; приобрести практические навыки наследования.

Задание:

Создайте производный класс для АТД, реализованного по заданию лабораторной работы 4, используя одиночное наследование.

Проверьте работоспособность АТД на тестовом наборе данных.

Ход работы:

#include <iostream>

#include <stdlib.h>

#include "stdio.h"

#include <iostream>

using namespace std;

// Базовый класс

template <class T1, class T2>

class Agr {

public:

Agr<T1, T2>();

Agr<T1, T2>(const char\* name, T1 n, T2 t);

void input();

void print();

Agr<T1, T2>(const Agr<T1, T2>& a);

bool operator==(const Agr<T1, T2>& a);

~Agr<T1, T2>();

Agr <T1, T2>operator+(Agr<T1, T2>& a);

Agr <T1, T2>& operator=(const Agr<T1, T2>& a);

protected:

char\* tractor; T1 mass; T2 cost;

};

template <class T1, class T2>

Agr<T1, T2>::Agr()

{

tractor = new char[30];

strcpy(tractor, "Tractor");

mass = 200;

cost = 600;

}

template <class T1, class T2>

Agr<T1, T2>::Agr<T1, T2>(const char\* name, T1 n, T2 t) {

tractor = NULL;

tractor = new char[strlen(name) + 1];

strcpy(tractor, name);

cost = t;

mass = n;

};

template <class T1, class T2>

void Agr<T1, T2>::print() {

cout << endl;

cout << "Трактор: " << tractor << endl;

cout << "Масса: " << mass << " кг" << endl;

cout << "Стоимость: " << cost << " руб" << endl;

};

template <class T1, class T2>

void Agr<T1, T2>::input()

{

cout << "Введите название трактора: ";

cin >> tractor;

cout << "Введите массу трактора (кг): ";

cin >> mass;

cout << "Введите стоимость трактора (руб): ";

cin >> cost;

}

template <class T1, class T2>

Agr<T1, T2>::Agr<T1, T2>(const Agr& a) {

tractor = new char[strlen(a.tractor) + 1];

strcpy(tractor, a.tractor);

cost = a.cost;

mass = a.mass;

};

template <class T1, class T2>

bool Agr<T1, T2>::operator==(const Agr& a) {

return ((!strcmp(tractor, a.tractor)) &&

cost == a.cost &&

mass == a.mass);

};

template <class T1, class T2>

Agr<T1, T2>::~Agr<T1, T2>() {

delete[]tractor;

};

template <class T1, class T2>

Agr<T1, T2> Agr<T1, T2>::operator+(Agr& a) {

Agr v;

v.tractor = new char[strlen(a.tractor) + strlen(tractor) + 2];

strcpy(v.tractor, tractor);

strcat(v.tractor, ",");

strcat(v.tractor, a.tractor);

v.cost = cost + a.cost;

v.mass = mass + a.mass;

return v;

};

template <class T1, class T2>

Agr<T1, T2>& Agr<T1, T2>::operator=(const Agr& a) {

tractor = new char[strlen(a.tractor) + 1];

strcpy(tractor, a.tractor);

cost = a.cost;

mass = a.mass;

return(\*this);

};

// Производный класс

template <class T1, class T2>

class Engine : public Agr<int, double> {

public:

Engine();

Engine(const char\* name, T1 n, T2 t, const int pow);

~Engine();

void input();

void print();

Engine<T1, T2>(const Engine& a);

Engine& operator = (const Engine<T1, T2>& a);

protected:

double pow;

};

template <class T1, class T2>

Engine<T1, T2>::Engine() : Agr()

{

pow = 240;

}

template <class T1, class T2>

Engine<T1, T2>::Engine(const char\* name, T1 n, T2 t, const int pow)

{

this->pow = pow;

}

template <class T1, class T2>

Engine<T1, T2>::~Engine()

{

Agr::~Agr<T1, T2>();

}

template <class T1, class T2>

void Engine<T1, T2>::input()

{

Agr::input();

cout << "Введите мощность двигателя (л.с.): ";

cin >> pow;

}

template <class T1, class T2>

void Engine<T1, T2>::print()

{

Agr::print();

cout << "Мощность двигателя: " << pow << " л.с." << endl;

}

template <class T1, class T2>

Engine<T1, T2>::Engine<T1, T2>(const Engine& a) {

pow = a.pow;

this->tractor = new char[strlen(a.tractor) + 1];

strcpy(this->tractor, a.tractor);

this->cost = a.cost;

this->mass = a.mass;

};

template <class T1, class T2>

Engine<T1, T2>& Engine<T1, T2>::operator=(const Engine& a) {

pow = a.pow;

this->tractor = new char[strlen(a.tractor) + 1];

strcpy(this->tractor, a.tractor);

this->cost = a.cost;

this->mass = a.mass;

return(\*this);

};

int main()

{

setlocale(LC\_ALL, "Russian");

Agr<int, double> Tractor1;

Engine<int, double> Engine1;

//базовый класс

cout << "Введите информацию о тракторе:" << endl;

Tractor1.input();

cout << endl << "Базовый класс: " << endl;

Tractor1.print();

cout << endl;

//конструктор с параметрами

cout << "Введите информацию о тракторе с двигателем: " << endl;

Engine1.input();

cout << endl;

cout << "Производный класс: " << endl;

Engine1.print();

cout << endl;

//конструктор по умолчанию

cout << "Конструктор по умолчанию: " << endl;

Engine<int, double> Engine2;

Engine2.print();

cout << endl;

//копирующий конструктор

cout << "Копирующий конструктор: " << endl;

Engine<int, double> Engine3(Engine1);

Engine3.print();

cout << endl;

//Оператор =

cout << endl << "Оператор = " << endl;

Engine2 = Engine1;

Engine2.print();

//Тестирование функций базового класса через производный

//Оператор ==

cout << endl << "Оператор ==" << endl;

if (Engine1 == Engine2)cout << "Равны" << endl;

else cout << "Неравны" << endl;

//Оператор +

cout << endl << "Оператор +" << endl;

Tractor1 = Engine1 + Engine2;

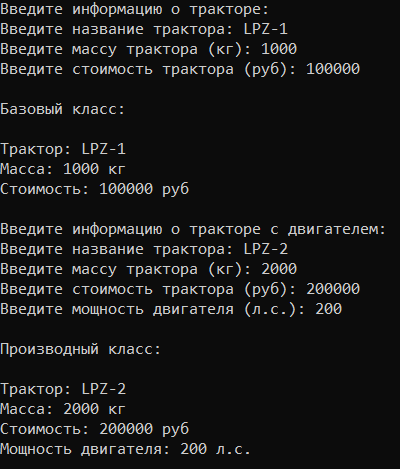
Tractor1.print();

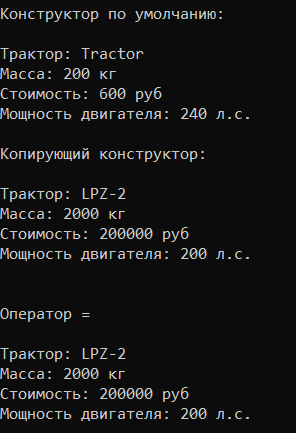
system("pause");

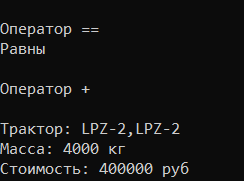
return 0;

}

Результат работы программы:







**Контрольные вопросы**

1. Дайте определение наследования.

Наследование – это механизм получения нового класса на основе существующего класса. Существующий класс может быть дополнен или изменен для создания производного класса. Наследование осуществляется с помощью конструкции:

class имя\_класса:

public|protected|private)opt имя\_базового\_класса

{

объявления членов

};

2. Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?

Модификаторы прав доступа к членам класса могут применяться в объявлении класса в любом порядке и сколько угодно раз. Открытый член доступен во всей области видимости, где виден класс. Закрытый член доступен другим функциям-членам своего класса. Защищенный член доступен не только другим функциям-членам своего класса, но и функциям-членам класса, унаследованного непосредственно от данного класса. Если все члены открыты, то ключевое слово class можно заменить ключевым словом struct.

3. Как выполняется конструктор при наследовании?

При наследовании и инициализации членов класса конструкторы выполняются в следующем порядке:

1). Базовые классы инициализируются в порядке объявления.

2). Члены инициализируются в порядке объявления.

4. Дайте определение чисто виртуальной функции.

Чисто виртуальная функция – это виртуальная функция-член, тело которой не определено. Она объявляется внутри класса:

virtual прототип\_функции = 0;

5. Какие операторы используются для разыменования указателя на член класса?

Для разыменования указателя на член класса используются два оператора: .\* и →\*.

Сначала разыменовывается указатель для получения переменной-члена, а затем происходит доступ к члену объекта.