Лабораторная работа №5

*Простая 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 с 2 шаблонами

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);

private:

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) {

Agr <int> e;

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

strcpy(tractor, a.tractor);

cost = a.cost;

mass = a.mass;

return e;

};

// Базовый класс с одним шаблоном

template <class T>

class Agr1 {

public:

Agr1<T>();

Agr1<T>(const char\* name, T n, T t);

void input();

void print();

Agr1<T>(const Agr1<T>& a);

bool operator==(const Agr1<T>& a);

~Agr1<T>();

Agr1 <T>operator+(Agr1<T>& a);

Agr1 <T>& operator=(const Agr1<T>& a);

private:

char\* tractor; T mass; T cost;

};

template <class T>

Agr1<T>::Agr1()

{

tractor = new char[30];

strcpy(tractor, "Tractor");

mass = 200;

cost = 600;

}

template <class T>

Agr1<T>::Agr1<T>(const char\* name, T n, T t) {

tractor = NULL;

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

strcpy(tractor, name);

cost = t;

mass = n;

};

template <class T>

void Agr1<T>::print() {

cout << endl;

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

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

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

};

template <class T>

void Agr1<T>::input()

{

cout << "Введите название трактора: ";

cin >> tractor;

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

cin >> mass;

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

cin >> cost;

}

template <class T>

Agr1<T>::Agr1<T>(const Agr1& a) {

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

strcpy(tractor, a.tractor);

cost = a.cost;

mass = a.mass;

};

template <class T>

bool Agr1<T>::operator==(const Agr1& a) {

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

cost == a.cost &&

mass == a.mass);

};

template <class T>

Agr1<T>::~Agr1<T>() {

delete[]tractor;

};

template <class T>

Agr1<T> Agr1<T>::operator+(Agr1& a) {

Agr1 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 T>

Agr1<T>& Agr1<T>::operator=(const Agr1& a) {

Agr1 e;

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

strcpy(tractor, a.tractor);

cost = a.cost;

mass = a.mass;

return e;

};

// Производный класс двигатель трактора

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();

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()

{

}

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;

}

int main()

{

setlocale(LC\_ALL, "Russian");

Engine<int, double> Tractor1,Tractor2;

cout << "Введите информацию о 1 тракторе:" << endl;

Tractor1.input();

cout << endl << "Введите информацию о 2 тракторе:" << endl;

Tractor2.input();

cout << endl << "Информация о 1 тракторе: " << endl;

Tractor1.print();

cout << endl << "Информация о 2 тракторе: " << endl;

Tractor2.print();

cout << endl;

Engine<double, int> Tractor3, Tractor4;

cout << "Введите информацию о 3 тракторе:" << endl;

Tractor3.input();

cout << endl << "Введите информацию о 4 тракторе:" << endl;

Tractor4.input();

cout << endl << "Информация о 3 тракторе: " << endl;

Tractor3.print();

cout << endl << "Информация о 4 тракторе: " << endl;

Tractor4.print();

cout << endl;

Engine<double, float> Tractor5, Tractor6;

cout << "Введите информацию о 5 тракторе:" << endl;

Tractor5.input();

cout << endl << "Введите информацию о 6 тракторе:" << endl;

Tractor6.input();

cout << endl << "Информация о 5 тракторе: " << endl;

Tractor5.print();

cout << endl << "Информация о 6 тракторе: " << endl;

Tractor6.print();

cout << endl;

Engine<int, float> Tractor7, Tractor8;

cout << "Введите информацию о 7 тракторе:" << endl;

Tractor7.input();

cout << endl << "Введите информацию о 8 тракторе:" << endl;

Tractor8.input();

cout << endl << "Информация о 7 тракторе: " << endl;

Tractor7.print();

cout << endl << "Информация о 8 тракторе: " << endl;

Tractor8.print();

Engine<int, int> Tractor9, Tractor10;

cout << "Введите информацию о 9 тракторе:" << endl;

Tractor9.input();

cout << endl << "Введите информацию о 10 тракторе:" << endl;

Tractor10.input();

cout << endl << "Информация о 9 тракторе: " << endl;

Tractor9.print();

cout << endl << "Информация о 10 тракторе: " << endl;

Tractor10.print();

system("pause");

return 0;

}

Результат работы программы:









