***Лабораторная работа* *8***

## 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.

**Задания:**

1. Создайте производный класс для АТД, реализованного по заданию лабораторной работы 7, используя одиночное наследование.

2. Проверьте работоспособность АТД и производного класса на тестовом наборе данных.

Код:

#include <iostream>

#include <string>

using namespace std;

void PrintLine() {

cout << "-================================-\n";

}

template <class T>

class SoundRecord

{

private:

char\* name;

int kolvo;

int price;

char\* power;

public:

SoundRecord();

SoundRecord(char\* a, int b, int c, char\* d);

SoundRecord(const SoundRecord& a);

~SoundRecord();

SoundRecord Sum(const SoundRecord& a);

bool Compair(SoundRecord& a);

void destroy();

void SetName(char\* a);

void GetName();

void SetKolvo(int a);

int GetKolvo();

void SetPrice(int a);

void GetPrice();

void SetPower(char\* a);

void GetPower();

SoundRecord& operator+(const SoundRecord& a);

bool operator>(const SoundRecord& r);

SoundRecord& operator= (const SoundRecord& r);

bool operator==(const SoundRecord& a);

};

template <class T> SoundRecord<T>::SoundRecord()

{

name = new char[11];

power = new char[11];

strcpy(name, "none");

strcpy(power, "none");

price = 0;

kolvo = 0;

}

template <class T> SoundRecord<T>::SoundRecord<T>(char\* a, int b, int c, char\* d)

{

name = a;

kolvo = b;

price = c;

power = d;

}

template <class T> SoundRecord<T>::SoundRecord<T>(const SoundRecord<T>& a)

{// констр копирования

// тело конструктора

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

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

strcpy(name, a.name);

strcpy(power, a.power);

price = a.price;

kolvo = a.kolvo;

}

template <class T>SoundRecord<T>::~SoundRecord()

{

//delete[] name;

//delete[] pr;

cout << "Memory has been succesfully cleaned" << endl;

};

template <class T>void SoundRecord<T>::GetName() {

cout.width(5);

cout << "Наименование: " << name << "\n";

}

template <class T>void SoundRecord<T>::SetKolvo(int a) {

kolvo = a;

}

template <class T>int SoundRecord<T>::GetKolvo() {

cout.width(5);

cout << "Количество: " << kolvo << "\n";

return kolvo;

}

template <class T>void SoundRecord<T>::SetName(char\* a) {

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

strcpy(name, a);

}

template <class T>void SoundRecord<T>::SetPrice(int a) {

price = a;

}

template <class T>void SoundRecord<T>::GetPrice() {

cout.width(5);

cout << "Стоимость: " << price << "\n";

}

template <class T>void SoundRecord<T>::SetPower(char\* a) {

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

strcpy(power, a);

}

template <class T>void SoundRecord<T>::GetPower() {

cout.width(5);

cout << "Мощность: " << power << "\n";

}

template <class T>SoundRecord<T>& SoundRecord<T>::operator+(const SoundRecord<T>& a)

{

SoundRecord v;

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

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

strcpy(v.name, name);

strcat(v.name, ",");

strcat(v.name, a.name);

strcpy(v.power, power);

strcat(v.power, ",");

strcat(v.power, a.power);

v.price = price + a.price;

v.kolvo = kolvo + a.kolvo;

v.GetName();

v.GetPower();

v.GetPrice();

v.GetKolvo();

return v;

}

template <class T>bool SoundRecord<T>::operator>(const SoundRecord<T>& a)

{

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

price > a.price &&

kolvo > a.kolvo);

}

template <class T>SoundRecord<T>& SoundRecord<T>::operator=(const SoundRecord<T>& r)

{

price = r.price;

kolvo = r.kolvo;

power = r.power;

name = r.name;

return \*this;

}

template <class T>bool SoundRecord<T>::operator==(const SoundRecord<T>& a)

{

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

price == a.price &&

kolvo == a.kolvo);

}

template <class T>void SoundRecord<T>::destroy() {

delete[] name;

delete[] power;

cout << "Memory has been succesfully cleaned" << endl;

};

template <class T>SoundRecord<T> SoundRecord<T>::Sum(const SoundRecord<T>& a)

{

SoundRecord v;

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

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

strcpy(v.name, name);

strcat(v.name, ",");

strcat(v.name, a.name);

strcpy(v.power, power);

strcat(v.power, ",");

strcat(v.power, a.power);

v.price = price + a.price;

v.kolvo = kolvo + a.kolvo;

v.GetMark();

v.Getpr();

v.GetPrice();

v.GetReleaseY();

return v;

}

template <class T>bool SoundRecord<T>::Compair(SoundRecord<T>& a)

{

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

price == a.price &&

kolvo == a.kolvo);

};

//производный класс

template <class T>

class Sound : public SoundRecord<T> // производный класс

{

private:

bool disk;

char\* td;// type of dynamics

public:

void SetTd(char\* e);

void GetTd();

void SetDisk(bool d);

void Disk();

};

template <class T>void Sound<T>::SetTd(char\* e) {

td = new char[strlen(e) + 1];

strcpy(td, e);

}

template <class T>void Sound<T>::GetTd() {

cout.width(5);

cout << "Тип динамика: " << td << "\n";

}

template <class T>void Sound<T>::SetDisk(bool d)

{

disk = d;

}

template <class T>void Sound<T>::Disk() {

if (disk ) {

cout << "Есть CD-проигрыватель: " << "\n";

}

else {

cout << "Нет CD-проигрывателя: " << "\n";

};

}

int main()

{

setlocale(LC\_ALL, "Russian");

SoundRecord<int> G5;

G5.SetName((char\*)"Marshall");

G5.GetName();

G5.SetKolvo(20);

G5.GetKolvo();

G5.SetPrice(300);

G5.GetPrice();

G5.SetPower((char\*)"200");

G5.GetPower();

PrintLine();

cout << "объект созданный с помощью конструктора по умолчанию\n";

SoundRecord<int> Pixel;

Pixel.GetName();

Pixel.GetKolvo();

Pixel.GetPrice();

Pixel.GetPower();

PrintLine();

SoundRecord<int> M5s\_lite((char\*)"VOX", 12, 200, (char\*)"SONY");

SoundRecord<int> M5s = SoundRecord<int>(M5s\_lite);

cout << "объект созданный с помощью констр копирования" << endl;

M5s.GetName();

M5s.GetKolvo();

M5s.GetPrice();

M5s.GetPower();

PrintLine();

M5s\_lite.GetName();

M5s\_lite.GetKolvo();

M5s\_lite.GetPrice();

M5s\_lite.GetPower();

PrintLine();

if (Pixel.Compair(M5s)) {

cout << "equal" << endl;

}

else {

cout << "not equal" << endl;

}

PrintLine();

//Pixel.Sum(G5);

//PrintLine();

SoundRecord<int> a;

a.operator=(M5s);

cout << "объект созданный из пергруженного оператора присваивания" << endl;

a.GetName();

a.GetKolvo();

a.GetPrice();

a.GetPower();

PrintLine();

M5s.GetName();

M5s.GetKolvo();

M5s.GetPrice();

M5s.GetPower();

PrintLine();

if (a.operator==(M5s)) {

cout << "1й объект равен второму" << endl;

}

else {

cout << "1й объект не равен второму" << endl;

}

PrintLine();

SoundRecord<int> b;

cout << "объект созданный из пергруженного оператора сложения" << endl;

b = a.operator+(G5);

PrintLine();

if (Pixel.operator>(M5s)) {

cout << "1й объект больше второго" << endl;

}

else {

cout << "1й объект не больше второго" << endl;

}

PrintLine();

PrintLine();

PrintLine();

PrintLine();

////////////////////////////////////////////////////////

Sound<int> G6;

G6.SetName((char\*)"Orange");

G6.GetName();

G6.SetKolvo(20);

G6.GetKolvo();

G6.SetPrice(300);

G6.GetPrice();

G6.SetPower((char\*)"200");

G6.GetPower();

G6.SetDisk(0);

G6.Disk();

G6.SetTd((char\*)"AUX");

G6.GetTd();

PrintLine();

cout << "объект созданный производным классом\n";

system("pause");

return 0;

};

**Тестирование:**

