**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.

#include <string.h>

#include <iostream>

using namespace std;

template <class T1 = int, class T2 = float>

class MemoryDevice {

public:

MemoryDevice<T1, T2>(const char\* programa, T1 v, T2 p);

MemoryDevice<T1, T2>();

void print();

void input();

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

bool operator==(const MemoryDevice& a);

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

MemoryDevice<T1, T2>& operator++();

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

~MemoryDevice<T1, T2>();

private:

char\* programm; T1 cost; T2 size;

};

template <class T>

class MemoryDevice1 {

public:

MemoryDevice1<T>(const char\* programa, T v, T p);

void print();

~MemoryDevice1<T>();

private:

char\* programm; T cost; T size;

};

template <class T>

MemoryDevice1<T>::MemoryDevice1<T>(const char\* programa, T v, T p) {

programm = new char[strlen(programa) + 1];

strcpy(programm, programa);

cost = v;

size = p;

};

template <class T>

void MemoryDevice1<T>::print() {

cout << "Программа: " << programm << endl;

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

cout << "Размер: " << size << endl;

};

template <class T>

MemoryDevice1<T>::~MemoryDevice1<T>() {

delete[] programm;

};

template <class T1, class T2>

MemoryDevice<T1, T2>::MemoryDevice<T1, T2>() {

programm = new char[strlen("EVA 255") + 1];

strcpy(programm, "EVA 255");

cost = 200;

size = 300;

};

template <class T1, class T2>

MemoryDevice<T1, T2>::MemoryDevice<T1, T2>(const char\* programa, T1 v, T2 p) {

programm = new char[strlen(programa) + 1];

strcpy(programm, programa);

cost = v;

size = p;

};

template <class T1, class T2>

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

cout << "Название: " << programm << endl;

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

cout << "Размер: " << size << endl;

};

template <class T1, class T2>

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

{

cout << "Введите название запоминающего устройства: ";

cin >> programm;

cout << "Введите стоимость запоминающего устройства: ";

cin >> cost;

cout << "Введите размер запоминающего устройства: ";

cin >> size;

}

template <class T1, class T2>

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

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

strcpy(programm, a.programm);

cost = a.cost;

size = a.size;

};

template <class T1, class T2>

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

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

cost == a.cost &&

size == a.size);

}

template <class T1, class T2>

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

delete[] programm;

};

template <class T1, class T2>

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

MemoryDevice <T1, T2> v;

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

strcpy(v.programm, programm);

strcat(v.programm, ",");

strcat(v.programm, a.programm);

v.cost = cost + a.cost;

v.size = size + a.size;

return v;

};

template <class T1, class T2>

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

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

strcpy(programm, a.programm);

cost = a.cost;

size = a.size;

return(\*this);

};

template <class T1, class T2>

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

cost += 1;

size += 1;

return(\*this);

};

template <class T1, class T2>

class korpys : public MemoryDevice<int, double> {

public:

korpys();

korpys(const char\* programa, T1 v, T2 p, const int seria);

void input();

void print();

protected:

int seria;

};

template <class T1, class T2>

korpys<T1, T2>::korpys() : MemoryDevice()

{

seria = 3;

}

template <class T1, class T2>

korpys<T1, T2>::korpys(const char\* programa, T1 v, T2 p, const int seria)

{

this->seria = seria;

}

template <class T1, class T2>

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

{

MemoryDevice::input();

cout << "Введите серию запоминающего устройства: ";

cin >> seria;

}

template <class T1, class T2>

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

{

MemoryDevice::print();

cout << "Серия запоминающего устройства: " << seria << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

korpys<int, double> MemoryPDevice1, MemoryPDevice2;

cout << "Введите информацию о 1 запоминающем устройствe:" << endl;

MemoryPDevice1.input();

cout << endl << "Введите информацию о 2 запоминающем устройствe:" << endl;

MemoryPDevice2.input();

cout << endl << "Информация о 1 запоминающем устройствe: " << endl;

MemoryPDevice1.print();

cout << endl << "Информация о 2 запоминающем устройствe: " << endl;

MemoryPDevice2.print();

cout << endl;

korpys<double, int> MemoryPDevice3, MemoryPDevice4;

cout << "Введите информацию о 3 запоминающем устройствe:" << endl;

MemoryPDevice3.input();

cout << endl << "Введите информацию о 4 запоминающем устройствe:" << endl;

MemoryPDevice4.input();

cout << endl << "Информация о 3 запоминающем устройствe: " << endl;

MemoryPDevice3.print();

cout << endl << "Информация о 4 запоминающем устройствe: " << endl;

MemoryPDevice4.print();

cout << endl;

korpys<double, float> MemoryPDevice5, MemoryPDevice6;

cout << "Введите информацию о 5 запоминающем устройствe:" << endl;

MemoryPDevice5.input();

cout << endl << "Введите информацию о 6 запоминающем устройствe:" << endl;

MemoryPDevice6.input();

cout << endl << "Информация о 5 запоминающем устройствe: " << endl;

MemoryPDevice5.print();

cout << endl << "Информация о 6 запоминающем устройствe: " << endl;

MemoryPDevice6.print();

cout << endl;

korpys<int, float> MemoryPDevice7, MemoryPDevice8;

cout << "Введите информацию о 7 запоминающем устройствe:" << endl;

MemoryPDevice7.input();

cout << endl << "Введите информацию о 8 запоминающем устройствe:" << endl;

MemoryPDevice8.input();

cout << endl << "Информация о 7 запоминающем устройствe: " << endl;

MemoryPDevice7.print();

cout << endl << "Информация о 8 запоминающем устройствe: " << endl;

MemoryPDevice8.print();

korpys<int, int> MemoryPDevice9, MemoryPDevice10;

cout << "Введите информацию о 9 запоминающем устройствe:" << endl;

MemoryPDevice9.input();

cout << endl << "Введите информацию о 10 запоминающем устройствe:" << endl;

MemoryPDevice10.input();

cout << endl << "Информация о 9 запоминающем устройствe: " << endl;

MemoryPDevice9.print();

cout << endl << "Информация о 10 запоминающем устройствe: " << endl;

MemoryPDevice10.print();

system("pause");

return 0;

}

**Результат работы программы:**









