**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.

**Код программы**

#include "pch.h"

#include <iostream>

#include <string.h>

using namespace std;

template <class T=int, class F=float>

class send

{

public:

send<T, F>(const char\* c, T b, F d, const char\* a);

send<T, F>(const send<T, F> &a);

send<T, F>();

~send<T, F>();

void print();

bool operator==(send<T, F> a);

send operator+(send<T, F> &a);

send &operator=(const send<T, F> &a);

protected:

char\* type;

T weight;

F price;

char\* marka;

};

template <class TT=int>

class send2 {

public:

send2<TT>(const char\* c, TT b, double d, const char\* a);

send2<TT>(const send2<TT> &a);

send2<TT>();

~send2<TT>();

void print();

protected:

char\* type;

TT weight;

double price;

char\* marka;

};

template <class T, class F>

send<T, F>::send() {

type = new char[10];

strcpy(type, "-");

weight = 0;

price = 0;

marka = new char[10];

strcpy(marka, "-");

}

template <class T, class F>

send<T, F>::send(const send<T, F> &a) {

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

strcpy(type, a.type);

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

strcpy(marka, a.marka);

weight = a.weight;

price = a.price;

}

template <class T, class F>

send<T, F>::send(const char\* c, T b, F d, const char\* a)

{

type = new char[strlen(c) + 1];

strcpy(type, c);

weight = b;

price = d;

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

strcpy(marka, a);

}

template <class T, class F>

send<T, F>::~send() {

type = NULL; marka = NULL;

}

template <class T, class F>

void send<T, F>::print()

{

cout << "Тип:" << type << endl;

cout << "Вес:" << weight << endl;

cout << "Стоимость:" << price << endl;

cout << "Марка:" << marka << endl;

}

template <class T, class F>

bool send<T, F>::operator==(send<T, F> a) {

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

weight == a.weight && price == a.price &&

(!strcmp(type, a.type)));

}

template <class T, class F>

send<T, F> send<T, F>::operator+(send<T, F> &a) {

send s;

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

strcpy(s.type, type);

strcat(s.type, a.type);

s.weight = weight + a.weight;

s.price = price + a.price;

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

strcpy(s.marka, marka);

strcat(s.marka, a.marka);

return s;

}

template <class T, class F>

send<T, F> &send<T, F>::operator=(const send<T, F> &a)

{

send e;

type = NULL;

type = a.type;

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

strcpy(type, a.type);

weight = a.weight;

price = a.price;

marka = NULL;

marka = a.marka;

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

strcpy(marka, a.marka);

return e;

}

template <class TT>

send2<TT>::send2() {

type = new char[10];

strcpy(type, "-");

weight = 0;

price = 0;

marka = new char[10];

strcpy(marka, "-");

}

template <class TT>

send2<TT>::send2(const send2<TT> &a) {

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

strcpy(type, a.type);

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

strcpy(marka, a.marka);

weight = a.weight;

price = a.price;

}

template <class TT>

send2<TT>::send2(const char\* c, TT b, double d, const char\* a)

{

type = new char[strlen(c) + 1];

strcpy(type, c);

weight = b;

price = d;

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

strcpy(marka, a);

}

template <class TT>

send2<TT>::~send2() {

type = NULL; marka = NULL;

}

template <class TT>

void send2<TT>::print()

{

cout << "Тип:" << type << endl;

cout << "Вес:" << weight << endl;

cout << "Стоимость:" << price << endl;

cout << "Марка:" << marka << endl;

}

// ПК

template <class T, class F>

class sendrate : public send<int, double > {

public:

sendrate(const char\* c, T b, F d, const char\* a, int r);

sendrate<T, F>(const sendrate<T, F> &a);

sendrate<T, F>();

void print();

protected:

int R;

};

template <class T, class F>

sendrate<T, F>::sendrate()

{

type = new char[10];

strcpy(type, "-");

weight = 0;

price = 0;

marka = new char[10];

strcpy(marka, "-");

R = 0;

}

template <class T, class F>

sendrate<T, F>::sendrate(const sendrate<T, F> &a) {

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

strcpy(type, a.type);

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

strcpy(marka, a.marka);

weight = a.weight;

price = a.price;

R = a.R;

}

template <class T, class F>

sendrate<T, F>::sendrate(const char\* c, T b, F d, const char\* a, int r)

{

type = new char[strlen(c) + 1];

strcpy(type, c);

weight = b;

price = d;

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

strcpy(marka, a);

R = r;

}

template <class T, class F>

void sendrate<T,F>::print()

{

send::print();

cout <<"Рейтинг:" << R;

}

template <class TT>

class sendBC :public send2<int> {

public:

sendBC(const char\* c, TT b, double d, const char\* a, int r);

sendBC<TT>(const sendBC<TT> &a);

sendBC<TT>();

void print();

protected:

int R;

};

template <class TT>

sendBC<TT>::sendBC(const char\* c, TT b, double d, const char\* a, int r)

{

type = new char[strlen(c) + 1];

strcpy(type, c);

weight = b;

price = d;

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

strcpy(marka, a);

R = r;

}

template <class TT>

sendBC<TT>::sendBC()

{

type = new char[10];

strcpy(type, "-");

weight = 0;

price = 0;

marka = new char[10];

strcpy(marka, "-");

R = 0;

}

template <class TT>

sendBC<TT>::sendBC(const sendBC<TT> &a) {

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

strcpy(type, a.type);

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

strcpy(marka, a.marka);

weight = a.weight;

price = a.price;

R = a.R;

}

template <class TT>

void sendBC<TT>::print()

{

send2::print();

cout << "Рейтинг:" << R;

}

int main()

{

setlocale(LC\_ALL, "Russian");

char\* ty = new char[10];

char\* ty2 = new char[10];

int w, w2, r, r2;

double f, f2;

char\* mark = new char[10];

char\* mark2 = new char[10];

cout << '\t' << "Базовый шаблонный класс" << endl;

cout <<"Введите тип модема: ";

cin >> ty;

cout << endl << "Введите вес: ";

cin >> w;

cout << endl << "Введите стоимость(руб.): ";

cin >> f;

cout << endl << "Введите марку модема: ";

cin >> mark;

cout << endl << "Введите рейтинг модема: ";

cin >> r;

cout << endl;

sendrate<double, int> a(ty, w, f, mark, r);

a.print();

cout << endl;

sendrate<double, int> b(a);

b.print();

cout << endl;

cout << '\t' << "Производный шаблонный класс" << endl;

cout << "Введите тип модема: ";

cin >> ty2;

cout << endl << "Введите вес: ";

cin >> w2;

cout << endl << "Введите стоимость(руб.): ";

cin >> f2;

cout << endl << "Введите марку модема: ";

cin >> mark2;

cout << endl << "Введите рейтинг модема: ";

cin >> r2;

sendBC<char> c(ty2, w2, f2, mark2, r2);

c.print();

cout << endl;

sendBC<char> d(c);

d.print();

cout << endl;

return 0;

}

**Тестовые данные**



