Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

специальность РИС

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Вариант №2

Выполнил студент группы РИС-13

Балахонцев Е.Н.

Руководитель работы:

ассистент кафедры ИТАС

Шилов В.С.

Пермь 2014 г.

**Аннотация**

Темой курсовой работы является разработка базы данных с использованием технологий объектно-ориентированного программирования. Необходимо, используя технологию объектно-ориентированного программирования разработать программу для создания объектов пользовательского класса, сохранения их в контейнере и в файле с возможностью последующего восстановления. Использовать алгоритмы сортировки и поиска для работы с данными, хранящимися в контейнере. Программа разрабатывается на языке С++ в среде Microsoft Visual Studio как консольное приложение. В моем случае нужно реализовать следующий вид классов: персона, рабочий, инженер.

**2. Оглавление**

Аннотация…………………………………………………………………………1

Оглавление………………………………………………………………………...2

Цель курсовой работы…………………………………………………………….3

1. Структура программы………………………………………………………….5

2. Описание программы……………………………………..……………………6

3. Описание основных алгоритмов………………………………………………7

4. Тестирование программы…………………………………………………….13

5.Заключение……………………………………………………………………..16

6. Список литературы……………………………………………………………17

Приложение………………………………………………………………………18

**Цель курсовой работы**

Целью курсовой работы является получение новых навыков, закрепление и углубление знаний, полученных при изучении дисциплины в ходе лекционных и практических занятий.

Выполнение курсовой работы предусматривает:

1.Создание классов, методов класса, конструкторов и деструкторов, объектов класса.

2.Создание иерархии классов путем наследования.

3.Использование виртуальных функций, абстрактных классов и полиморфных объектов.

4.Использование классов-контейнеров для хранения объектов

5.Использование стандартной библиотеки C++.

6.Использование потоковых классов стандартной библиотеки C++ для сохранения объектов в файле их загрузки из файла

7.Использование стандартной библиотеки шаблонов С++.

8.Получение практических навыков программирования алгоритмов обработки данных.

9.Получение новых знаний об объектно – ориентированном программировании.

1. **Структура программы**

Иерархия классов

Класс person подразделяется на: 1. worker

2. engineer

Были также использованы следующие библиотеки:

list – динамический массив

fstream – библиотека файлового ввода-вывода

algorithm – алгоритмы STL

iostream – библиотека ввода-вывода

Windows.h – библиотека WinAPI

string – строковый тип данных

Физическая структура проекта

Физическая структура проекта изображена на рисунке 1.

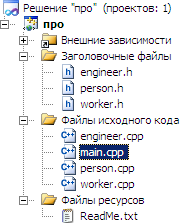


Рис. №1 Физическая структура программы

person.h - описание абстрактоного класса, прототипы чисто виртуальных функций

worker.h – описание класса рабочий

worker.cpp – реализация функций класса рабочий

engineer.h – описание класса инженер

engineer.cpp – реализация функций класса инженер

main.cpp – основная программа

**2. Описание программы**

В программе описаны три класса:

1. person

2. worker

3. engineer

person – абстрактный, содержит поле char nm[30], поля int age, bool sx и описание чисто виртуальных функций.

worker – наследуется от person, добавляется поле wqp

обладает: конструктором с параметрами функцией show(), которая осуществляет вывод на консоль

оператор ‘<’

оператор’==’

engineer – наследуется от person, обладает: конструктором с параметрами

функцией show(), которая осуществляет вывод на консоль

оператор ‘<’

оператор ‘==’

В программе описаны следующие функции:

void cwrker(); - создание рабочего

void cenger(); - создание инженера

void shfact(); - показ контейнера

void clfact(); - очистка контейнера

void save(); - сохранение в файл

void load(); - загрузка из файла

void sort(); - сортировка

void find(); - поиск

void input(person\* p2); - получение информации при вводе

void WritePersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f); - запись в файл

void ReadPersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f); - чтение из файла

void WriteW(person& ob, fstream& t\_f2); - запись рабочего в файл

void WriteE(person& ob, fstream& t\_f2); - запись инженера в файл

void Sortt(bool fs); - сортировка

void Findd(bool fs); - поиск

void inquiry(); - выполнение запроса

**3. Описание основных алгоритмов**

Запись и чтение из файла:

Основная сложность заключается в том, что, поскольку в контейнере хранятся полиморфные объекты разных классов, то и в файл должны записываться объекты разных классов. Следовательно, файл должен быть полиморфным. Основная трудность при работе с таким файлом связана с тем, что объекты разных классов имеют разный размер. Мы должны иметь механизм для записи в файл и чтении из файла количества байт в зависимости от типа записываемого (читаемого) объекта. Одним из способов решения этой задачи является запись перед полями объекта идентификатора, определяющего тип объекта. Например, это может быть целое число. Собственно именно этот способ и был использован при реализации программы.

Сортировка контейнера

В качестве контейнера в варианте попался список <list. Из алгоритмов STL для сортировки был выбран sort(), Для программной сортировки в варианте был указан метод сортировки простого обмена.

Описание алгоритма сортировки простого обмена:

На каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированном списке, до тех пор, пока набор входных данных не будет исчерпан. Метод выбора очередного элемента из исходного массива произволен; может использоваться практически любой алгоритм выбора. Обычно (и с целью получения устойчивого алгоритма сортировки), элементы вставляются по порядку их появления во входном массиве. Блок-схема алгоритма сортировки простыми включениями изображена на рисунке 2.

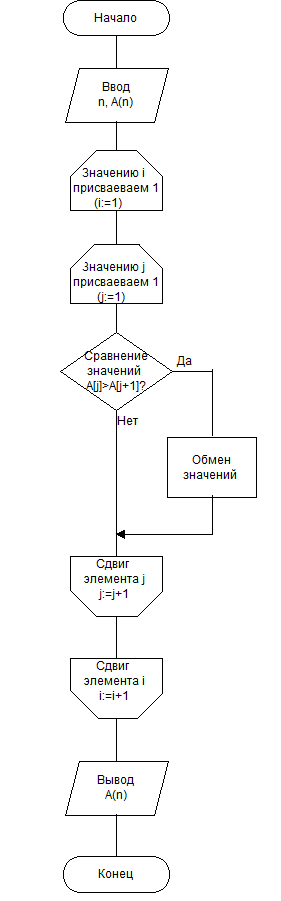


Рис. №2. Блок-схема алгоритма сортировки простыми включениями

Поиск в отсортированной последовательности

В качестве алгоритма поиска STL был использован find()

Для программной сортировки был реализован алгоритм бинарного поиска. Сначала образец сравнивается со средним (по номеру) элементом массива. Если образец равен среднему элементу, то задача решена. Если образец больше среднего элемента, то это значит, что искомый элемент расположен ниже среднего элемента (между элементами с номерами sred+1 и niz), и за новое значение verb принимается sred+i, а значение niz не меняется Если образец меньше среднего элемента, то это значит, что искомый элемент расположен выше среднего элемента (между элементами с номерами verh и sred-1), и за новое значение niz принимается sred-1, а значение verh не меняется. После того как определена часть массива, в которой может находиться искомый элемент, по формуле (niz-verh) /2+verh вычисляется новое значение sred и поиск продолжается. Блок-схема алгоритма представлена на рисунке 3.

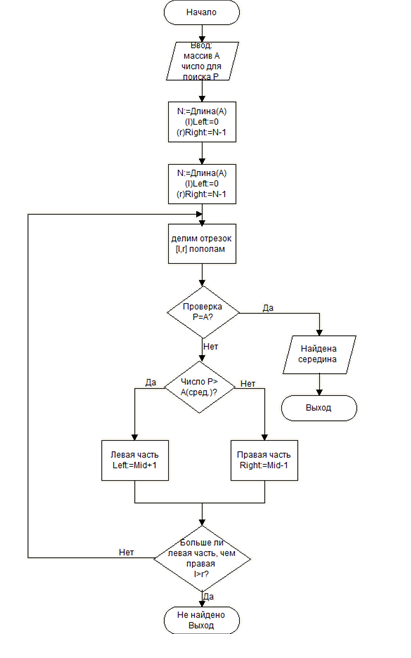


Рис. №3. Блок-схема алгоритма бинарного поиска

Так же в программе будет реализован запрос

Вывести названия всех деталей с весом не больше заданного

Для этого нам потребуется обратиться конкретно к классу izdelie. С помощью функции typeid мы конкретно указываем, что нам нужно только вес детали.

**4. Тестирование программы**

При запуске программы появляется главное меню (Рис. №4)

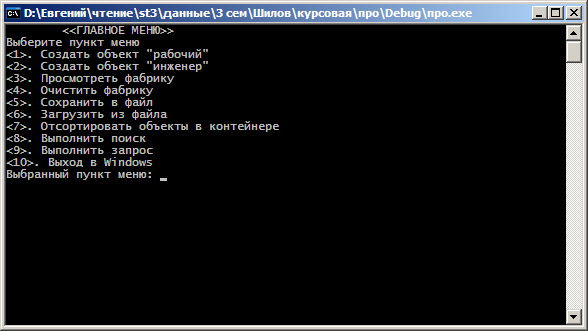


Рис. №4. Меню программы

При выборе пункта №1 появляется предложение заполнить соответствующие поля (Рис. №5)

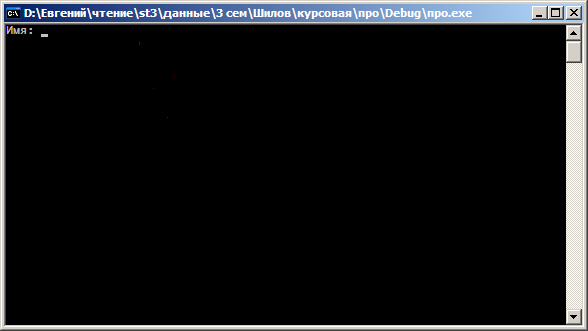


Рис. №5. Создание объекта рабочий

При выборе пункта №2 появляется аналогичная форма с полями для класса «инженер» (Рис. №6)

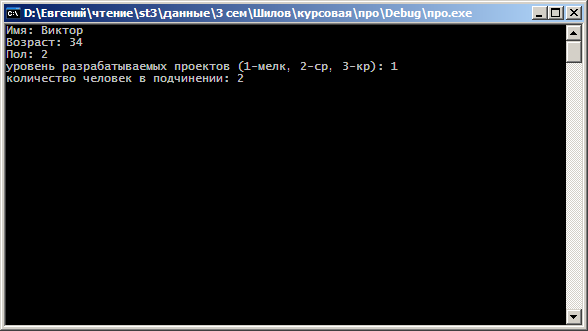
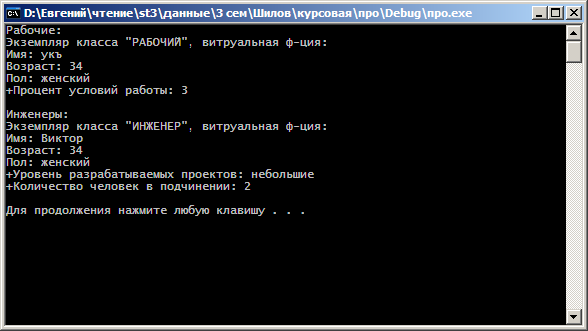


Рис. №6. создание объекта Деталь

При запуске функции просмотра контейнера появляется форма, изображённая на рисунке 7.

Рис. №7. просмотр контейнера

В программе предусмотрены два типа сортировок, выбрать один из них можно в соответствующем меню (рисунок 8)

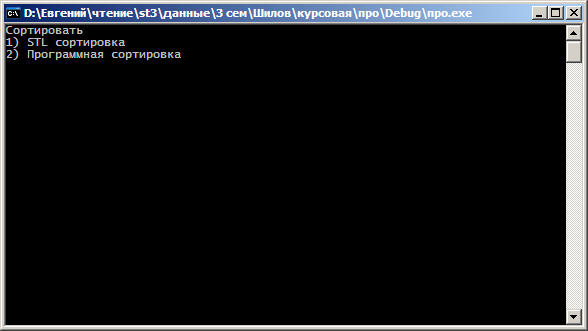


Рис. №8. выбор варианта сортировки

Принцип работы сортировки 2 (рисунок 9)

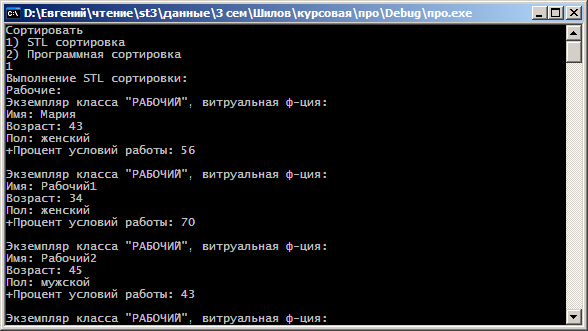


Рис. №9 Сортировка программная

Также в программе предусмотрены два типа поиска, выбор также осуществляется через меню (рисунок 10)

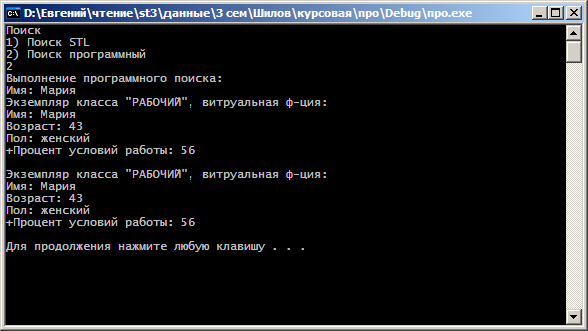


Рис. №10. Выбор варианта поиска и демонстрация поиска STL

**5. Заключение**

Все цели и задачи, поставленные перед разработкой программы, выполнены. Программа выполняет функции по созданию объектов неабстрактных классов иерархии, умеет просматривать контейнер <map> заданный вариантом и очищать его. Также, реализованы функции, позволяющие записывать объекты в файл и в дальнейшем загружать эти же объекты из файла, в который они были записаны ранее. Реализована возможность сортировки объектов в контейнере <vector> двумя способами: с помощью алгоритмов STL и программно-реализованной сортировки простыми включениями. Ключевое поле – имя(name). Программа умеет находить объекты по заданному ключевому полю. Как и для сортировки, реализованы 2 способа, которые можно выбрать в меню: с помощью алгоритмов STL и программно-реализованного бинарного поиска. Ключевым полем для методов поиска объекта является имя (name).

**6. Список использованной литературы**

1. Шилдт Г. - Самоучитель C++ (3-е издание) / СПб.: БХВ-Петербург 2003г. 687с.
2. Страуструп Б. - Язык программирования С++ 2011г. / Бином, Невский Диалект, 2004 г. 1054с.
3. Вирт Н. - Алгоритмы и структуры данных. 2010г. / ДМК Пресс 274с.
4. Лафоре Р. - Объектно-ориентированное программирование в С++ (4-е издание). / ПИТЕР 2004г 924c.
5. Ноткин А.М. Объектно-ориентированное программирование на языке C++: учебное пособие / Перм. гос. техн. ун-т. Пермь, 2001. 93 с.

**Приложение**

Листинг программы

main.cpp

#include <list>

#include <algorithm>

#include <iterator>

#include <functional>

#include "person.h"

#include "worker.h"

#include "engineer.h"

#include <iostream>

#include <fstream>

#include <string>

#include <stdlib.h> //для atoi()

#include <Windows.h>

#include "typeinfo.h" //для определения типа при вводе данных

using namespace std;

person\* p;

worker\* wp;

engineer\* ep;

//vector <person\*> vp;

//vector <person\*>::iterator it;

list <worker> vw;

list <engineer> ve;

list <worker>::iterator itw;

list <engineer>::iterator ite;

fstream f1;

void cwrker();

void cenger();

void shfact();

void clfact();

void save();

void load();

void sort();

void find();

void input(person\* p2);

void WritePersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f);

void ReadPersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f);

void WriteW(person& ob, fstream& t\_f2);

void WriteE(person& ob, fstream& t\_f2);

void Sortt(bool fs);

void Findd(bool fs);

void inquiry();

bool comparre(person& p1, person& p2)

{

if (strcmp(p1.GetNm(), p2.GetNm())>0)

return true;

else

return false;

}

struct str\_w

{

char m\_nm[30];

int m\_age;

bool m\_sx;

int m\_wqp;

};

struct str\_e

{

char m\_nm[30];

int m\_age;

bool m\_sx;

int m\_pl;

int m\_px;

};

void main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int i=0;

do

{

system("cls");

cout<<" <<ГЛАВНОЕ МЕНЮ>> \n";

cout<<"Выберите пункт меню\n";

cout<<"<1>. Создать объект \"рабочий\"\n";

cout<<"<2>. Создать объект \"инженер\"\n";

cout<<"<3>. Просмотреть фабрику\n";

cout<<"<4>. Очистить фабрику\n";

cout<<"<5>. Сохранить в файл\n";

cout<<"<6>. Загрузить из файла\n";

cout<<"<7>. Отсортировать объекты в контейнере\n";

cout<<"<8>. Выполнить поиск\n";

cout<<"<9>. Выполнить запрос\n";

cout<<"<10>. Выход в Windows\n";

cout<<"Выбранный пункт меню: ";

cin>>i;

switch(i)

{

case 1:

{

system("cls");

cwrker();

system("pause");

break;

}

case 2:

{

system("cls");

cenger();

system("pause");

break;

}

case 3:

{

system("cls");

shfact();

system("pause");

break;

}

case 4:

{

system("cls");

clfact();

system("pause");

break;

}

case 5:

{

system("cls");

save();

system("pause");

break;

}

case 6:

{

system("cls");

load();

system("pause");

break;

}

case 7:

{

system("cls");

sort();

system("pause");

break;

}

case 8:

{

system("cls");

find();

system("pause");

break;

}

case 9:

{

system("cls");

inquiry();

system("pause");

break;

}

default:

{

if(i!=10)

{

cout<<"Вводите значения от 1 до 9\n";

system("pause");

break;

}

}

}

}

while(i!=10);

system("pause");

}

void cwrker()

{

/\*cout<<"Создание рабочего\n";\*/

char m\_nm[30];

int m\_age=-1;

bool m\_sx=0;

int m\_wqp=-1;

wp = new worker(m\_nm, m\_age, m\_sx, m\_wqp);

input(wp);

vw.push\_back(\*wp);

}

void cenger()

{

/\*cout<<"Создание инженера\n";\*/

char m\_nm[30];

int m\_age=-1;

bool m\_sx=0;

int m\_pl=-1;

int m\_px=-1;

ep = new engineer(m\_nm, m\_age, m\_sx, m\_pl, m\_px);

input(ep);

ve.push\_back(\*ep);

}

void shfact()

{

/\* cout<<"Показать фабрику\n";

int k = vp.size();

if (k==0) cout<<"Фабрика пуста\n";

else

for(int i=0; i<k; i++)

vp[i]->show();\*/

//it = vp.begin();

//while(it != vp.end())

// {

// /\*person \*pp = vp.front();

// pp->show(); \*/

// //vp.front()->show();

// it.\_Mynode()->\_Myval->show();

// //it.\_Mynode()->\_Myval.show();

// it++;

// }

cout<<"Рабочие:\n";

itw = vw.begin();

while(itw!=vw.end())

{

itw.\_Mynode()->\_Myval.show();

itw++;

}

cout<<"Инженеры:\n";

ite = ve.begin();

while(ite!=ve.end())

{

ite.\_Mynode()->\_Myval.show();

ite++;

}

}

void clfact()

{

/\*cout<<"Очистить фабрику\n";

int k = vp.size();

if (k==0) cout<<"Фабрика пуста\n";

else

for(int i=0; i<k; i++) delete vp[i];

vp.clear(); \*/

//vp.clear();

vw.clear();

ve.clear();

}

void save()

{

//cout<<"Сохранить в файл\n";

WritePersons(vw, ve, f1);

}

void load()

{

//cout<<"Зайгрузить из файла\n";

ReadPersons(vw, ve, f1);

}

void sort()

{

cout<<"Сортировать\n";

int choose;

cout<<"1) STL сортировка\n";

cout<<"2) Программная cортировка\n";

cin>>choose;

if(choose==1)

Sortt(true);

else

Sortt(false);

}

void find()

{

cout<<"Поиск\n";

int choose2;

cout<<"1) Поиск STL\n";

cout<<"2) Поиск программный\n";

cin>>choose2;

if(choose2==1)

Findd(true);

else

Findd(false);

}

void input(person\* p2)

{

char t\_str[30];

char t\_chr[10];

//cin.clear();

//while(cin.get()!='\n');

cout<<"Имя: ";

cin>>t\_str;

p2->SetNm(t\_str);

cin.clear();

while(cin.get()!='\n');

cout<<"Возраст: ";

cin.getline(t\_chr, '\n');

p2->SetAge(atoi(t\_chr));

cout<<"Пол: ";

cin.getline(t\_chr, '\n');

if(\*t\_chr=='m'||\*t\_chr=='м'||\*t\_chr=='1'||\*t\_chr=='M'||\*t\_chr=='М')

p2->SetSx(true);

else

p2->SetSx(false);

if(typeid(\*p2)==typeid(worker))

{

worker\* t\_w = (worker\*) p2;

cout<<"% удобств условий работы: ";

cin.getline(t\_chr, '\n');

t\_w->SetWqp(atoi(t\_chr));

}

if(typeid(\*p2)==typeid(engineer))

{

engineer\* t\_e = (engineer\*) p2;

do

{

cout<<"уровень разрабатываемых проектов (1-мелк, 2-ср, 3-кр): ";

cin.getline(t\_chr, '\n');

}

while(\*t\_chr!='1'&&\*t\_chr!='2'&&\*t\_chr!='3');

t\_e->SetPl(atoi(t\_chr));

cout<<"количество человек в подчинении: ";

cin.getline(t\_chr, '\n');

t\_e->SetPx(atoi(t\_chr));

}

}

//void WritePersons(vector<person\*> &t\_vp,fstream& t\_f)

void WritePersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f)

{

t\_f.open("my.txt", ios::binary|ios::out); //

t\_f.seekp(0);

//int n=t\_vw.size();

itw = vw.begin();

while(itw != vw.end())

{

WriteW((itw.\_Mynode()->\_Myval), t\_f);

itw++;

}

ite = ve.begin();

while(ite != ve.end())

{

WriteE((ite.\_Mynode()->\_Myval), t\_f);

ite++;

}

/\* for(int i=0; i<n; i++)

{

if(typeid(\*t\_vp[i])==typeid(worker))

WriteW(\*t\_vp[i], t\_f);

if(typeid(\*t\_vp[i])==typeid(engineer))

WriteE(\*t\_vp[i], t\_f);

}\*/

t\_f.close();

}

//void ReadPersons(vector<person\*> &t\_vp,fstream& t\_f)

void ReadPersons(list<worker> &t\_vw, list<engineer> &t\_ve, fstream& t\_f)

{

int k;

str\_e en1;

str\_w wrk1;

//cout<<t\_f.tellp()<<endl;

t\_f.open("my.txt", ios::binary|ios::in|ios::out); //

//cout<<t\_f.tellp()<<endl;

t\_f.clear();

//cout<<t\_f.tellp()<<endl;

t\_f.seekp(0); //!

//cout<<t\_f.tellp()<<endl;

t\_f.read((char\*)&k,sizeof(int));

//cout<<t\_f.tellp()<<endl;

if(k==1)

t\_f.read((char\*)&wrk1,sizeof(wrk1));

//cout<<t\_f.tellp()<<endl;

if(k==2)

t\_f.read((char\*)&en1,sizeof(en1));

//cout<<t\_f.tellp()<<endl;

while(!t\_f.eof())

{

if(k==1)

{

//cout<<t\_f.tellp()<<endl;

wp = new worker(wrk1.m\_nm, wrk1.m\_age, wrk1.m\_sx, wrk1.m\_wqp);

//cout<<t\_f.tellp()<<endl;

t\_vw.push\_back(\*wp);

}

if(k==2)

{

//cout<<t\_f.tellp()<<endl;

ep = new engineer(en1.m\_nm, en1.m\_age, en1.m\_sx, en1.m\_pl, en1.m\_px);

//cout<<t\_f.tellp()<<endl;

t\_ve.push\_back(\*ep);

}

//cout<<t\_f.tellp()<<endl;

t\_f.read((char\*)&k,sizeof(int));

//cout<<t\_f.tellp()<<endl;

if(k==1)

{

t\_f.read((char\*)&wrk1,sizeof(wrk1));

}

//cout<<t\_f.tellp()<<endl;

if(k==2)

{

t\_f.read((char\*)&en1,sizeof(en1));

}

}

t\_f.close();

}

void WriteW(person& ob, fstream& t\_f2)

{

int k=1; // рабочий

char\* t\_chW;

str\_w s\_w;

//cout<<sizeof(s\_w)<<endl;

//w=dynamic\_cast<worker\*>(&ob);

wp = (worker\*) &ob;

t\_chW = wp->GetNm();

for(int i=0; i<30; i++)

s\_w.m\_nm[i] = t\_chW[i];

s\_w.m\_age=wp->GetAge();

s\_w.m\_sx=wp->GetSx();

s\_w.m\_wqp=wp->GetWqp();

//cout<<sizeof(s\_w)<<endl;

//cout<<t\_f2.tellp()<<endl;

t\_f2.write((char\*)&k, sizeof(int));

//cout<<t\_f2.tellp()<<endl;

t\_f2.write((char\*)&s\_w, sizeof(s\_w));

//cout<<t\_f2.tellp()<<endl;

}

void WriteE(person& ob, fstream& t\_f2)

{

int k=2; // инженер

char\* t\_chE;

str\_e s\_e;

//e=dynamic\_cast<engineer\*>(&ob);

ep = (engineer\*) &ob;

t\_chE = ep->GetNm();

for(int i=0; i<30; i++)

s\_e.m\_nm[i] = t\_chE[i];

s\_e.m\_age=ep->GetAge();

s\_e.m\_sx=ep->GetSx();

s\_e.m\_pl=ep->GetPl();

s\_e.m\_px=ep->GetPx();

//cout<<t\_f2.tellp()<<endl;

t\_f2.write((char\*)&k, sizeof(int));

//cout<<t\_f2.tellp()<<endl;

t\_f2.write((char\*)&s\_e, sizeof(s\_e));

//cout<<t\_f2.tellp()<<endl;

}

void Sortt(bool fs)

{

if(fs)

{

cout<<"Выполнение STL сортировки:\n";

vw.sort();

ve.sort();

shfact();

}

else

{

cout<<"Выполнение программной сортировки:\n";

/\* sort(vw.begin(), vw.end(), srtw());

sort(ve.begin(), ve.end(), srte());\*/

int nw = vw.size();

worker cwrk[100];

itw = vw.begin();

for (int i=0; i<nw; i++)

{

cwrk[i] = itw.\_Mynode()->\_Myval;

itw++;

}

vw.clear();

bool bw=true;

while (bw)

{

bw = false;

for(int i=0; i<nw-1; i++)

{

if (comparre(cwrk[i],cwrk[i+1]))

{

bw = true;

swap(cwrk[i], cwrk[i+1]);

}

}

}

for (int i=0; i<nw; i++)

vw.push\_back(cwrk[i]);

int ne = ve.size();

engineer ceng[100];

ite = ve.begin();

for (int i=0; i<ne; i++)

{

ceng[i] = ite.\_Mynode()->\_Myval;

ite++;

}

ve.clear();

bool be=true;

while (be)

{

be = false;

for(int i=0; i<ne-1; i++)

{

if (comparre(ceng[i],ceng[i+1]))

{

be = true;

swap(ceng[i], ceng[i+1]);

}

}

}

for (int i=0; i<ne; i++)

ve.push\_back(ceng[i]);

shfact();

}

}

void Findd(bool fs)

{

if(fs)

{

cout<<"Выполнение STL-поиска:\n";

cout<<"Имя: ";

char chre [30];

cin>>chre;

worker t\_5w = worker(chre, 0, false, 0);

itw = find(vw.begin(), vw.end(), t\_5w);

if (itw==vw.end())

{

cout<<"Среди рабочих не найдено объектов\n";

}

else

itw.\_Mynode()->\_Myval.show();

engineer t\_5e = engineer(chre, 0, false, 0, 0);

ite = find(ve.begin(), ve.end(), t\_5e);

if (ite==ve.end())

{

cout<<"Среди инженеров не найдено объектов\n";

}

else

ite.\_Mynode()->\_Myval.show();

}

else

{

cout<<"Выполнение программного поиска:\n";

cout<<"Имя: ";

char chre [30];

cin>>chre;

itw = vw.begin();

while(itw!=vw.end())

{

if(strcmp(itw.\_Mynode()->\_Myval.GetNm(), chre)==0)

{

itw.\_Mynode()->\_Myval.show();

break;

}

itw++;

}

itw = vw.begin();

while(itw!=vw.end())

{

if(strcmp(itw.\_Mynode()->\_Myval.GetNm(), chre)==0)

{

itw.\_Mynode()->\_Myval.show();

break;

}

itw++;

}

ite = ve.begin();

while(ite!=ve.end())

{

if(strcmp(ite.\_Mynode()->\_Myval.GetNm(), chre)==0)

{

ite.\_Mynode()->\_Myval.show();

break;

}

ite++;

}

}

}

void inquiry()

{

cout<<"1) Вывести имена всех рабочих\n";

cout<<"2) Вывести имена всех инженеров\n";

int chose;

cin>>chose;

if (chose==1)

{

itw = vw.begin();

while(itw!=vw.end())

{

itw.\_Mynode()->\_Myval.showname();

itw++;

}

}

else

{

ite = ve.begin();

while(ite!=ve.end())

{

ite.\_Mynode()->\_Myval.showname();

ite++;

}

}

}

person.h

// ПЕРСОНА

#pragma once

#include <string>

#include <iostream>

using namespace std;

class person

{

protected:

char nm[30]; // имя

int age; // возраст

bool sx; // пол (true - м.р., false - ж.р.)

public:

person();

person(char nm2[30], int, bool);

virtual ~person();

virtual void SetNm(char Nm3[30])=0;

virtual void SetAge(int)=0;

virtual void SetSx(bool)=0;

virtual char\* GetNm()=0;

virtual int GetAge()=0;

virtual bool GetSx()=0;

virtual void show() const;

bool operator<(const person& l1);

bool operator == (const person& f1);

};

person.cpp

// ПЕРСОНА

#include "person.h"

#include <string>

person::person()

{

for (int i=0; i<29; i++)

this->nm[i] = ' ';

this->nm[29] = '\0';

this->age = -1;

this->sx = false;

}

person::person(char nm2[30], int age2, bool sx2)

{

//cout<<"Конструктор с параметрами person\n";

for (int i=0; i<29; i++)

this->nm[i] = nm2[i];

this->nm[29] = '\0';

this->age = age2;

this->sx = sx2;

}

person::~person()

{

//cout<<"Деструктор person\n";

}

void person::show() const

{

cout<<"Экземпляр класса \"ПЕРСОНА\", витруальная ф-ция: \n";

cout<<"Имя: "<<this->nm<<endl;

cout<<"Возраст: "<<this->age<<endl;

cout<<"Пол: ";

if(this->sx) cout<<"мужской\n\n";

else cout<<"женский\n\n";

}

void person::SetNm(char Nm3[30])

{

for (int i=0; i<29; i++)

this->nm[i] = Nm3[i];

this->nm[29] = '\0';

}

void person::SetAge(int Age3)

{

this->age=Age3;

}

void person::SetSx(bool Sx3)

{

this->sx=Sx3;

}

char\* person::GetNm()

{

return this->nm;

}

int person::GetAge()

{

return this->age;

}

bool person::GetSx()

{

return this->sx;

}

bool person::operator<(const person& l1)

{

if(strcmp(this->nm, l1.nm)<0)

return 1;

else

return 0;

}

bool person::operator == (const person& f1)

{

if (this->nm == f1.nm)

return true;

else

return false;

}

worker.h

// РАБОЧИЙ

#pragma once

#include <string>

#include <iostream>

#include "person.h"

using namespace std;

class worker:public person

{

protected:

int wqp; // % удобств в условиях работы

public:

worker();

worker(char nm3[30], int age3, bool sx3, int wqp2);

~worker();

void SetNm(char Nm5[30]);

void SetAge(int);

void SetSx(bool);

void SetWqp(int);

char\* GetNm();

int GetAge();

bool GetSx();

int GetWqp();

void show() const;

bool operator<(const worker& l2);

bool operator == (const worker& f2);

void showname() const;

};

worker.cpp

// РАБОЧИЙ

#include "worker.h"

#include "person.h"

#include <string>

worker::worker()

{

for (int i=0; i<29; i++)

this->nm[i] = ' ';

this->nm[29] = '\0';

this->age = 0;

this->sx = false;

this->wqp = -1;

}

worker::worker(char nm3[30], int age3, bool sx3, int wqp2):person(nm3, age3, sx3)

{

//cout<<"Конструктор worker\n";

for (int i=0; i<29; i++)

this->nm[i] = nm3[i];

this->nm[29] = '\0';

this->age = age3;

this->sx = sx3;

this->wqp = wqp2;

}

worker::~worker()

{

//cout<<"Деструктор worker\n";

}

void worker::show() const

{

cout<<"Экземпляр класса \"РАБОЧИЙ\", витруальная ф-ция: \n";

cout<<"Имя: "<<this->nm<<endl;

cout<<"Возраст: "<<this->age<<endl;

cout<<"Пол: ";

if (this->sx) cout<<"мужской\n";

else cout<<"женский\n";

cout<<"+Процент условий работы: "<<this->wqp<<"\n\n";

}

void worker::SetNm(char Nm5[30])

{

for (int i=0; i<29; i++)

this->nm[i] = Nm5[i];

this->nm[29] = '\0';

}

void worker::SetAge(int Age5)

{

this->age=Age5;

}

void worker::SetSx(bool Sx5)

{

this->sx=Sx5;

}

void worker::SetWqp(int Wqp5)

{

this->wqp = Wqp5;

}

char\* worker::GetNm()

{

return this->nm;

}

int worker::GetAge()

{

return this->age;

}

//

bool worker::GetSx()

{

return this->sx;

}

int worker::GetWqp()

{

return this->wqp;

}

bool worker::operator<(const worker& l2)

{

if(strcmp(this->nm, l2.nm)<0)

return 1;

else

return 0;

}

bool worker::operator == (const worker& f2)

{

if (strcmp(this->nm, f2.nm)==0)

return true;

else

return false;

}

void worker::showname() const

{

cout<<nm<<endl;

}

engineer.h

// ИНЖЕНЕР

#pragma once

#include <string>

#include <iostream>

#include "person.h"

using namespace std;

class engineer:public person

{

protected:

int pl; // уровень разрабатываемых проектов (1-мелк, 2-ср, 3-кр)

int px; // количество человек в подчинении

public:

engineer();

engineer(char nm4[30], int age4, bool sx4, int pl2, int px2);

~engineer();

void SetNm(char Nm4[30]);

void SetAge(int);

void SetSx(bool);

void SetPl(int);

void SetPx(int);

char\* GetNm();

int GetAge();

bool GetSx();

int GetPl();

int GetPx();

void show() const;

bool operator<(const engineer& l3);

bool operator == (const engineer& f3);

void showname() const;

};

engineer.cpp

// ИНЖЕНЕР

#include "engineer.h"

#include "person.h"

#include <string>

engineer::engineer()

{

for (int i=0; i<29; i++)

this->nm[i] = ' ';

this->nm[29] = '\0';

this->age = 0;

this->sx = false;

this->px = -1;

this->pl = -1;

}

engineer::engineer(char nm4[30], int age4, bool sx4, int pl2, int px2):person(nm4, age4, sx4)

{

//cout<<"Конструктор engineer\n";

for (int i=0; i<29; i++)

this->nm[i] = nm4[i];

this->nm[29] = '\0';

this->age = age4;

this->sx = sx4;

this->pl = pl2;

this->px = px2;

}

engineer::~engineer()

{

//cout<<"Деструктор engineer\n";

}

void engineer::show() const

{

cout<<"Экземпляр класса \"ИНЖЕНЕР\", витруальная ф-ция: \n";

cout<<"Имя: "<<this->nm<<endl;

cout<<"Возраст: "<<this->age<<endl;

cout<<"Пол: ";

if (this->sx) cout<<"мужской\n";

else cout<<"женский\n";

cout<<"+Уровень разрабатываемых проектов: ";

switch (this->pl)

{

case 1:

{

cout<<"небольшие\n";

break;

}

case 2:

{

cout<<"средние\n";

break;

}

case 3:

{

cout<<"крупные\n";

break;

}

default:

{

cout<<"некорретный параметр ( допустимые значения 1,2,3)\n";

break;

}

}

cout<<"+Количество человек в подчинении: "<<this->px<<"\n\n";

}

void engineer::SetNm(char Nm4[30])

{

for (int i=0; i<29; i++)

this->nm[i] = Nm4[i];

this->nm[29] = '\0';

}

void engineer::SetAge(int Age4)

{

this->age=Age4;

}

void engineer::SetSx(bool Sx4)

{

this->sx=Sx4;

}

void engineer::SetPl(int Pl4)

{

this->pl = Pl4;

}

void engineer::SetPx(int Px4)

{

this->px = Px4;

}

char\* engineer::GetNm()

{

return this->nm;

}

int engineer::GetAge()

{

return this->age;

}

bool engineer::GetSx()

{

return this->sx;

}

int engineer::GetPl()

{

return this->pl;

}

int engineer::GetPx()

{

return this->px;

}

bool engineer::operator<(const engineer& l3)

{

if(strcmp(this->nm, l3.nm)<0)

return 1;

else

return 0;

}

bool engineer::operator == (const engineer& f3)

{

if (strcmp(this->nm, f3.nm)==0)

return true;

else

return false;

}

void engineer::showname() const

{

cout<<nm<<endl;

}