/\*main.cpp

#include "Start.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "RUS");

system("color 2");

Fut\_Team obj;

obj.Start();

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Start.h

#pragma once

#include "Defender.h"

#include "StackOtmena.h"

#include "Forward.h"

#include "Fis\_Trener.h"

#include "Vera\_Trener.h"

#include "String.h"

#include "Shablon.cpp"

#include "Iskluch.h"

#include "TXTFILE.h"

#include <list>

#include <exception>

#include <vector>

#include "Algoritm.h"

#define n1 "Chlen\_komandi"

#define n2 "Footballist"

#define n3 "Defender"

#define n4 "Forward"

#define n5 "Trener"

#define n6 "Fis\_Trener"

#define n7 "Vera\_Trener"

class Fut\_Team

{

public:

int Start();

string print\_table(int \_b);

template<typename TYPE>

void view(TYPE \_obj, int b, list<TYPE> \_c);

template<typename TYPE>

void stl(list<TYPE> \_c, int \_b);

};

string Fut\_Team:: print\_table(int \_b)

{

rewind(stdin);

if (\_b == 1)

{

cout << "----------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< endl;

cout << "----------------------------------------------------";

cout << endl;

return n1;

}

if (\_b == 2)

{

cout << "-------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "NOMER"

<< endl;

cout << "-------------------------------------------------------------";

cout << endl;

return n2;

}

if (\_b == 3)

{

cout << "----------------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "NOMER"

<< setw(10) << "Yellow card"

<< endl;

cout << "----------------------------------------------------------------------";

cout << endl;

return n3;

}

if (\_b == 4)

{

cout << "--------------------------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "NOMER"

<< setw(10) << "GOALS"

<< setw(10) << "ASSISTS"

<< endl;

cout << "--------------------------------------------------------------------------------";

cout << endl;

return n4;

}

if (\_b == 5)

{

cout << "-------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "Staj"

<< endl;

cout << "-------------------------------------------------------------";

cout << endl;

return n5;

}

if (\_b == 6)

{

cout << "--------------------------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "Staj"

<< setw(20) << " Vid trenirovki"

<< endl;

cout << "--------------------------------------------------------------------------------";

cout << endl;

return n6;

}

if (\_b == 7)

{

cout << "----------------------------------------------------------------------" << endl;

cout << setiosflags(ios::left)

<< setw(20) << " NAME"

<< setw(20) << " SURNAME"

<< setw(10) << "YEAR"

<< setw(10) << "Staj"

<< setw(10) << "Vremia trenirovki"

<< endl;

cout << "----------------------------------------------------------------------";

cout << endl;

return n7;

}

}

template<typename TYPE>

void Fut\_Team:: view(TYPE \_obj, int b, list<TYPE> \_c)

{

cout << "Количество?" << endl;

int z;

z = enter\_int();

linklist <TYPE> a;

Stack<TYPE> otm;

TYPE\* tmp1;

TYPE s, l;

tmp1 = new TYPE[z];

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

{

cout << "---------" << endl;

rewind(stdin);

cin >> tmp1[i];

a.sozd(tmp1[i]);

}

string k;

k = print\_table(b);

a.pop\_data();

char w;

int cc;

char A[80];

TYPE ss;

Algorithm<TYPE> alg;

while (1)

{

cout << endl;

cout << "Выберите тип: 1-Добавить в шаблон(в любую позицию)\n";

cout << " 2-Вывести шаблон\n";

cout << " 3-Поиск по названию\n";

cout << " 4-Поиск по индексу\n";

cout << " 5-Удаление объекта\n";

cout << " 6-Отмена удаления\n";

cout << " 7-Отмена добавления\n";

cout << " 8-Прочитать из текстового файла\n";

cout << " 9-Записать в текстовый файл\n";

cout << " 10-Очистить текстовый файл\n";

cout << " 11-показать в обратном порядке(с итератором)\n";

cout << " 12-показать(с итератором)\n";

cout << " 13-поиск по названию(с итератором)\n";

cout << " 14-сортировка\n";

cout << " 15-STL\n";

cout << " 16-выход\n";

rewind(stdin);

cc = enter\_int();

switch (cc)

{

case 1:

{

system("cls");

k = print\_table(b);

a.pop\_data();

cout << endl;

cin >> s;

cout << endl << "Введите позицию для добавления" << endl;

cc = enter\_int();

a.add\_element(s, cc);

otm.add\_data(s);

a.pop\_data();

break;

}

case 2:

{

system("cls");

a.pop\_data();

break;

}

case 3:

{

system("cls");

a.pop\_data();

cin >> ss;

alg.search2(a.Begin(), ss);

break;

}

case 4:

{

system("cls");

a.pop\_data();

cout << "Введи индекс" << endl;

cc = enter\_int();

if (a.getrazmer() >= cc)

alg.search1(a.Begin(), cc);

break;

}

case 5:

{

system("cls");

a.pop\_data();

cout << endl << "Введите что хотите удалить" << endl;

cin >> ss;

a.udalenie(ss);

otm.add\_data(ss);

a.pop\_data();

break;

}

case 6:

{

s = otm.del();

a.add\_data(s);

cout << "Отмена удаления прошла успешно." << endl;

break;

}

case 7:

{

s = otm.del();

a.udalenie(s);

cout << "Отмена добавления прошла успешно." << endl;

break;

}

case 8:

{

system("cls");

TXTFILE \_b;

linklist <TYPE> c;

TYPE\* tmp2;

string q;

q = k + ".txt";

int count;

count = \_b.checkCount(q);

tmp2 = new TYPE[count];

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

{

\_b.fromFile(tmp2[i], q, i);

c.add\_data(tmp2[i]);

}

cout << "Данные успешно считаны из файла." << endl;

print\_table(b);

c.pop\_data();

break;

}

case 9:

{

system("cls");

string q;

q = k + ".txt";

vector<TYPE> ww;

ww = a.get();

TXTFILE b;

for (int i = 0; i < a.getrazmer(); i++)

{

b.toFile(ww[i], q);

}

cout << "Данные успешно записаны в файл." << endl;

break;

}

case 10:

{

system("cls");

string q;

q = k + ".txt";

TXTFILE b;

b.clear(q);

cout << "Файл очищен." << endl;

break;

}

case 11:

{

system("cls");

a.pop\_data();

cout << endl;

a.printback();

break;

}

case 12:

{

system("cls");

a.pop\_data();

cout << endl;

a.printIterator();

break;

}

case 13:

{

system("cls");

a.pop\_data();

cin >> ss;

alg.search2Iterator(a.Begin(), a.End(), ss);

break;

}

case 14:

{

system("cls");

a.pop\_data();

alg.sort(a);

cout << endl << "После сортировки" << endl;

a.pop\_data();

break;

}

case 15:

{

stl(\_c, b);

break;

}

case 16:

{

return;

}

}

cout << endl;

system("pause");

}

}

template<typename TYPE>

void Fut\_Team::stl(list<TYPE> \_c, int \_b)

{

system("cls");

list<TYPE> a;

char w;

TYPE d;

while (1)

{

cout << endl << "Выбери операцию" << endl;

cout << "1-Добавить в конец list" << endl << "2-Добавить в начало list" << endl << "3-Показать list" << endl << "4-Удалить элемент" << endl

<< "5-Очистить list" << endl << "6-Реверс элементов" << endl << "7-Узнать размер list"

<< endl << "8-Убрать совпадающие элементы list" << endl << "9-Вставить элемент в list" << endl << "10-Выход" << endl;

w = enter\_int();

switch (w)

{

case 1:

{

system("cls");

cin >> d;

a.push\_back(d);

system("cls");

cout << "Объект успешно добавлен" << endl;

break;

}

case 2:

{

system("cls");

cin >> d;

a.push\_front(d);

system("cls");

cout << "Объект успешно добавлен" << endl;

break;

}

case 3:

{

system("cls");

print\_table(\_b);

list<TYPE>::template iterator at;

for (at = a.begin(); at != a.end(); at++)

{

cout << (\*at);

cout << endl;

}

break;

}

case 4:

{

system("cls");

print\_table(\_b);

list<TYPE>::template iterator at;

for (at = a.begin(); at != a.end(); at++)

{

cout << (\*at);

cout << endl;

}

cout << "Какой элемент удалить?(Название)" << endl;

string t;

cin >> t;

for (at = a.begin(); at != a.end(); at++)

{

if ((at->GetName()) == t)

{

a.erase(at);

system("cls");

cout << "Объект удалён" << endl;

break;

}

}

break;

}

case 5:

{

system("cls");

a.clear();

cout << "list очищен" << endl;

break;

}

case 6:

{

system("cls");

a.reverse();

cout << "Реверс list" << endl;

break;

}

case 7:

{

system("cls");

int i;

i = a.size();

cout << "Размер list = " << i << endl;

break;

}

case 8:

{

system("cls");

a.unique();

cout << "Совпадающие элементы удалены" << endl;

break;

}

case 9:

{

system("cls");

cin >> d;

cout << "Введи позицию для вставки элемента" << endl;

int i, kl = 0;

cin >> i;

list<TYPE>::template iterator at;

for (at = a.begin(); at != a.end(); at++)

{

kl++;

if (kl == i)

{

a.insert(at, d);

break;

}

}

cout << "Элемент добавлен" << endl;

break;

}

case 10:

{

return;

}

}

}

}

int Fut\_Team::Start()

{

int zz;

cout << "\n\t\t\t\*\*\*\*\*\*\*\*\*\*\* СВЕДЕНИЯ ОБ ИГРОКАХ ФУТБОЛЬНОЙ КОМАНДЫ \*\*\*\*\*\*\*\*\*\*\*\n\n\n\n";

while (1)

{

cout << "Выберите команду: 1-Добавить и показать информацию Chlen\_komandi\n";

cout << " 2-Добавить и показать информацию Footballist\n";

cout << " 3-Добавить и показать информацию Defender\n";

cout << " 4-Добавить и показать информацию Forward\n";

cout << " 5-Добавить и показать информацию Trener\n";

cout << " 6-Добавить и показать информацию Fis\_Trener\n";

cout << " 7-Добавить и показать информацию Vera\_Trener\n";

cout << " 0-Выйти\n";

rewind(stdin);

zz = enter\_int();

switch (zz)

{

case 1:

{

list<Chlen\_komandi> c;

Chlen\_komandi t1;

int b = 1;

view(t1, b, c);

break;

}

case 2:

{

list<Footballist> c;

Footballist m1;

int b = 2;

view(m1, b, c);

break;

}

case 3:

{

list<Defender> c;

Defender l1;

int b = 3;

view(l1, b, c);

break;

}

case 4:

{

list<Forward> c;

Forward me1;

int b = 4;

view(me1, b, c);

break;

}

case 5:

{

list<Trener> c;

Trener a1;

int b = 5;

view(a1, b, c);

break;

}

case 6:

{

list<Fis\_Trener> c;

Fis\_Trener ma1;

int b = 6;

view(ma1, b, c);

break;

}

case 7:

{

list<Vera\_Trener> c;

Vera\_Trener ma1;

int b = 7;

view(ma1, b, c);

break;

}

case 0: return 0;

default: cout << "Ошибка, повторите \n";

}

cout << endl;

system("pause");

}

system("pause");

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Chlen\_comandi.h

#pragma once

#include "String.h"

using namespace std;

class Chlen\_komandi

{

protected:

String name;

String surname;

int year;

public:

Chlen\_komandi()

{

name = "";

surname = "";

year = 0;

}

Chlen\_komandi(String str,String str1, int value = 0)

{

name = str;

surname = str1;

year = value;

}

friend istream& operator>>(istream& in, Chlen\_komandi& ob)

{

cout << "Введите имя члена команды";

char\* pr\_str(istream & in);

ob.name = pr\_str(in);

cout << "Введите фамилию члена команды";

char\* pr\_str(istream & in);

ob.surname = pr\_str(in);

cout << "Введите возраст игрока:";

int enter\_int(istream & in);

ob.year=enter\_int(in);

return in;

}

friend ostream& operator<<(ostream& on, Chlen\_komandi& ob)

{

on << setw(20) << ob.name << setw(20) << ob.surname << setw(10) << ob.year;

return on;

}

friend void operator <<= (std::ostream& stream, Chlen\_komandi& tmp)

{

stream << tmp.name << "|" << tmp.surname << "|" << tmp.year << "|";

}

friend void operator >>= (std::istream& stream, Chlen\_komandi& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Chlen\_komandi& tmp);

friend void operator >= (std::istream& is, Chlen\_komandi& tmp);

bool operator==(Chlen\_komandi& obj)

{

bool m;

if (name == "" && year != 0 && surname == "")

{

return m = (year == obj.year);

}

if (name != "" && year == 0 && surname == "")

{

return m = (name == obj.name);

}

if (name == "" && year == 0 && surname != "")

{

return m = (surname == obj.surname);

}

if (name != "" && year != 0 && surname == "")

{

return m = ((year == obj.year)||(name == obj.name));

}

if (name == "" && year != 0 && surname != "")

{

return m = ((year == obj.year) || (surname == obj.surname));

}

if (name != "" && year == 0 && surname != "")

{

return m = ((name == obj.name) || (surname == obj.surname));

}

if (name != "" && year != 0 && surname != "")

{

return m = ((year == obj.year) || (surname == obj.surname)|| (name == obj.name));

}

}

void operator = (const Chlen\_komandi& tmp);

bool operator > (const Chlen\_komandi& tmp);

//Методы доступа к полям класса

String GetName();

void SetName(const char\* str);

void SetName(String str);

String GetSurname();

void SetSurname(const char\* str);

void SetSurname(String str);

int GetYear();

void SetYear(int value1);

~Chlen\_komandi() {};

};

/\*Chlen\_komandi.cpp

#include "Chlen\_komandi.h"

String Chlen\_komandi:: GetName()

{

return name;

};

void Chlen\_komandi:: SetName(const char\* str)

{

name = str;

}

void Chlen\_komandi::SetName(String str)

{

name = str;

}

String Chlen\_komandi::GetSurname()

{

return surname;

};

void Chlen\_komandi::SetSurname(const char\* str)

{

surname = str;

}

void Chlen\_komandi::SetSurname(String str)

{

surname = str;

}

int Chlen\_komandi::GetYear()

{

return year;

};

void Chlen\_komandi::SetYear(int value1)

{

year = value1;

}

void operator <= (std::ostream& os, Chlen\_komandi& tmp)

{

os.write(reinterpret\_cast<const char\*>(&tmp), sizeof(Chlen\_komandi));

os.write(reinterpret\_cast<const char\*>(&tmp.year), sizeof(int));

}

void operator >= (std::istream& is, Chlen\_komandi& tmp)

{

int n;

is.read(reinterpret\_cast<char\*>(&tmp), sizeof(Chlen\_komandi));

tmp.SetName(tmp.name);

is.read(reinterpret\_cast<char\*>(&tmp), sizeof(Chlen\_komandi));

tmp.SetSurname(tmp.surname);

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetYear(n);

}

void Chlen\_komandi::operator = (const Chlen\_komandi& tmp)

{

/\*this->god\_proizvodstva = tmp.god\_proizvodstva;\*/

if (this->name!="" && (this->name == tmp.name))

return;

name = tmp.name;

surname = tmp.surname;

year = tmp.year;

}

bool Chlen\_komandi::operator > (const Chlen\_komandi& tmp)

{

if (this->name>tmp.name)

return true;

if (this->name<tmp.name)

return false;

if (this->name == tmp.name)

return false;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Footballist.h

#pragma once

#include "Footballist.h"

class Forward :public Footballist

{

protected:

int goals;

int assists;

public:

Forward() :Footballist()

{

goals = 0;

assists = 0;

}

Forward(String ptr,int per = 0, int per1 = 0, int value1 = 0) :Footballist(ptr,value1)

{

goals = per;

assists = per1;

}

friend istream& operator>>(istream& in, Forward& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Footballist&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите количество голов:";

int enter\_int(istream & in);

obj.goals=enter\_int(in);

cout << "Введите количество голевых передач:";

int enter\_int(istream & in);

obj.assists=enter\_int(in);

return in;

}

friend void operator <<= (std::ostream& stream, Forward& tmp)

{

stream <<= dynamic\_cast <Footballist&>(tmp);

stream << tmp.goals << "|" << tmp.assists << "|";

}

friend void operator >>= (std::istream& stream, Forward& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetNumber(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetGoals(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetAssists(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Forward& tmp);

friend void operator >= (std::istream& is, Forward& tmp);

friend ostream& operator << (ostream& out, Forward& obj)

{

out << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.number

<< setw(10) << obj.goals

<< setw(10) << obj.assists;

return out;

}

bool operator==(Forward& obj)

{

bool m;

if (goals == 0 && assists != 0)

return m = (assists != assists);

if (goals != 0 && assists == 0)

return m = (goals != goals);

if (goals != 0 && assists != 0)

return m = (goals != goals||assists!=assists);

}

int GetGoals();

void SetGoals(int per);

int GetAssists();

void SetAssists(int per1);

~Forward() {};

};

/\*Footballist.cpp

#include "Footballist.h"

int Footballist::GetNumber()

{

return number;

};

void Footballist::SetNumber(int sss)

{

number = sss;

}

void operator <= (std::ostream& os, Footballist& tmp)

{

os <= dynamic\_cast <Chlen\_komandi&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp.number), sizeof(int));

}

void operator >= (std::istream& is, Footballist& tmp)

{

is >= dynamic\_cast <Chlen\_komandi&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetNumber(n);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Trener.h

#pragma once

#include "Chlen\_komandi.h"

class Trener :public Chlen\_komandi

{

protected:

int staj\_work;

public:

Trener() :Chlen\_komandi()

{

staj\_work = 0;

}

Trener(String \_name,String name1,int \_staj\_work =0,int \_year = 0) :Chlen\_komandi(\_name,name1,\_year)

{

this->staj\_work = \_staj\_work;

}

friend ostream& operator<<(ostream& on, Trener& obj)

{

on << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.staj\_work;

return on;

}

friend istream& operator>>(istream& in, Trener& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Chlen\_komandi&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите стаж работы";

int enter\_int(istream & in);

obj.staj\_work=enter\_int(in);

return in;

}

friend void operator <<= (std::ostream& stream, Trener& tmp)

{

stream <<= dynamic\_cast <Chlen\_komandi&>(tmp);

stream << tmp.staj\_work << "|";

}

friend void operator >>= (std::istream& stream, Trener& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetStaj\_Work(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Trener& tmp);

friend void operator >= (std::istream& is, Trener& tmp);

bool operator==(Trener& obj)

{

if (obj.staj\_work == this->staj\_work)

return true;

else

return false;

}

int GetStaj\_Work();

void SetStaj\_Work(int value1);

~Trener() {}

};

/\*Trener.cpp

#include "Trener.h"

int Trener::GetStaj\_Work()

{

return this->staj\_work;

}

void Trener::SetStaj\_Work(int value1)

{

this->staj\_work = value1;

}

void operator <= (std::ostream& os, Trener& tmp)

{

os <= dynamic\_cast <Chlen\_komandi&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp.staj\_work), sizeof(int));

}

void operator >= (std::istream& is, Trener& tmp)

{

is >= dynamic\_cast <Chlen\_komandi&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetStaj\_Work(n);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Forward.h

#pragma once

#include "Footballist.h"

class Forward :public Footballist

{

protected:

int goals;

int assists;

public:

Forward() :Footballist()

{

goals = 0;

assists = 0;

}

Forward(String ptr,int per = 0, int per1 = 0, int value1 = 0) :Footballist(ptr,value1)

{

goals = per;

assists = per1;

}

friend istream& operator>>(istream& in, Forward& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Footballist&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите количество голов:";

int enter\_int(istream & in);

obj.goals=enter\_int(in);

cout << "Введите количество голевых передач:";

int enter\_int(istream & in);

obj.assists=enter\_int(in);

return in;

}

friend void operator <<= (std::ostream& stream, Forward& tmp)

{

stream <<= dynamic\_cast <Footballist&>(tmp);

stream << tmp.goals << "|" << tmp.assists << "|";

}

friend void operator >>= (std::istream& stream, Forward& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetNumber(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetGoals(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetAssists(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Forward& tmp);

friend void operator >= (std::istream& is, Forward& tmp);

friend ostream& operator << (ostream& out, Forward& obj)

{

out << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.number

<< setw(10) << obj.goals

<< setw(10) << obj.assists;

return out;

}

bool operator==(Forward& obj)

{

bool m;

if (goals == 0 && assists != 0)

return m = (assists != assists);

if (goals != 0 && assists == 0)

return m = (goals != goals);

if (goals != 0 && assists != 0)

return m = (goals != goals||assists!=assists);

}

int GetGoals();

void SetGoals(int per);

int GetAssists();

void SetAssists(int per1);

~Forward() {};

};

/\*Forward.cpp

#include "Forward.h"

int Forward::GetGoals()

{

return goals;

};

void Forward::SetGoals(int per)

{

goals = per;

}

int Forward::GetAssists()

{

return assists;

};

void Forward::SetAssists(int per1)

{

assists = per1;

}

void operator <= (std::ostream& os, Forward& tmp)

{

os <= dynamic\_cast <Footballist&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp.goals), sizeof(int));

os.write(reinterpret\_cast<const char\*>(&tmp.assists), sizeof(int));

}

void operator >= (std::istream& is, Forward& tmp)

{

is >= dynamic\_cast <Footballist&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetGoals(n);

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetAssists(n);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Defender.h

#pragma once

#include "Footballist.h"

class Defender :public Footballist

{

protected:

int yellow\_card;

public:

Defender() :Footballist()

{

yellow\_card = 0;

}

Defender(String ptr,int per = 0, int value1 = 0) :Footballist(ptr,value1)

{

yellow\_card = per;

}

friend istream& operator>>(istream& in, Defender& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Footballist&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите количество желтых карточек:";

int enter\_int(istream & in);

obj.yellow\_card=enter\_int(in);

return in;

}

friend ostream& operator<<(ostream& on, Defender& obj)

{

on << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.number

<< setw(10) << obj.yellow\_card;

return on;

}

friend void operator <<= (std::ostream& stream, Defender& tmp)

{

stream <<= dynamic\_cast <Footballist&>(tmp);

stream << tmp.yellow\_card << "|";

}

friend void operator >>= (std::istream& stream, Defender& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetYellow\_Card(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Defender& tmp);

friend void operator >= (std::istream& is, Defender& tmp);

bool operator==(Defender& obj)

{

if (obj.yellow\_card == this->yellow\_card)

return true;

else

return false;

}

int GetYellow\_Card();

void SetYellow\_Card(int per);

~Defender() {};

};

/\*Defender.cpp

#include "Defender.h"

int Defender::GetYellow\_Card()

{

return yellow\_card;

};

void Defender::SetYellow\_Card(int per)

{

yellow\_card = per;

}

void operator <= (std::ostream& os, Defender& tmp)

{

os <= dynamic\_cast <Footballist&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp.yellow\_card), sizeof(int));

}

void operator >= (std::istream& is, Defender& tmp)

{

is >= dynamic\_cast <Footballist&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetYellow\_Card(n);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Fis\_Trener

#pragma once

#include "Trener.h"

class Fis\_Trener:public Trener

{

protected:

String vid\_treni;

public:

Fis\_Trener():Trener()

{

vid\_treni = "";

}

Fis\_Trener(String \_vid\_treni,String ptr, int per = 0) :

Trener(ptr,per)

{

vid\_treni = \_vid\_treni;

}

friend ostream& operator<<(ostream& on, Fis\_Trener& obj)

{

on << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.staj\_work

<< setw(20) << obj.vid\_treni;

return on;

}

friend istream& operator>>(istream& in, Fis\_Trener& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Trener&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите вид тренировки";

char\* pr\_str(istream & in);

obj.vid\_treni = pr\_str(in);

return in;

}

friend void operator >>= (std::istream& stream, Fis\_Trener& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetStaj\_Work(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetVid\_Treni(s1.c\_str());

}

}

friend void operator <= (std::ostream& os, Fis\_Trener& tmp);

friend void operator >= (std::istream& is, Fis\_Trener& tmp);

friend void operator <<= (std::ostream& stream, Fis\_Trener& tmp)

{

stream <<= dynamic\_cast <Trener&>(tmp);

stream << tmp.vid\_treni << "|";

}

bool operator==(Fis\_Trener& obj)

{

if (obj.vid\_treni == this->vid\_treni)

return true;

else

return false;

}

String GetVid\_Treni();

void SetVid\_Treni(const char\* str);

void SetVid\_Treni(String str);

~Fis\_Trener() {}

};

/\*Fis\_Trener.cpp

#include "Fis\_Trener.h"

String Fis\_Trener::GetVid\_Treni()

{

return this->vid\_treni;

}

void Fis\_Trener::SetVid\_Treni(const char\* str)

{

vid\_treni = str;

}

void Fis\_Trener::SetVid\_Treni(String str)

{

vid\_treni = str;

}

void operator <= (std::ostream& os, Fis\_Trener& tmp)

{

os <= dynamic\_cast <Trener&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp), sizeof(Fis\_Trener));

}

void operator >= (std::istream& is, Fis\_Trener& tmp)

{

is >= dynamic\_cast <Trener&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&tmp), sizeof(Fis\_Trener));

tmp.SetVid\_Treni(tmp.vid\_treni);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Vera\_Trener.h

#pragma once

#include "Trener.h"

class Vera\_Trener:public Trener

{

protected:

int vremia\_trener;

public:

Vera\_Trener():Trener()

{

vremia\_trener = 0;

}

Vera\_Trener(String ptr,int \_vremia\_trener=0,int per=0) :Trener(ptr,per)

{

vremia\_trener = \_vremia\_trener;

}

friend ostream& operator<<(ostream& on, Vera\_Trener& obj)

{

on << setw(20) << obj.name

<< setw(20) << obj.surname

<< setw(10) << obj.year

<< setw(10) << obj.staj\_work

<< setw(10) << obj.vremia\_trener;

return on;

}

friend istream& operator>>(istream& in, Vera\_Trener& obj)

{

int flag;

do

{

try

{

flag = 0;

in >> dynamic\_cast<Trener&>(obj);

}

catch (const bad\_cast& ob)

{

flag = 1;

cout << "Error: " << ob.what();

}

} while (flag);

cout << "Введите время тренировки вратарей(в минутах)";

int enter\_int(istream & in);

obj.vremia\_trener=enter\_int(in);

return in;

}

friend void operator <<= (std::ostream& stream, Vera\_Trener& tmp)

{

stream <<= dynamic\_cast <Trener&>(tmp);

stream << tmp.vremia\_trener << "|";

}

friend void operator >>= (std::istream& stream, Vera\_Trener& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

tmp.SetName(s1.c\_str());

getline(ss, s1, '|');

tmp.SetSurname(s1.c\_str());

getline(ss, s1, '|');

tmp.SetYear(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetStaj\_Work(atoi(s1.c\_str()));

getline(ss, s1, '|');

tmp.SetVremia\_Trener(atoi(s1.c\_str()));

}

}

friend void operator <= (std::ostream& os, Vera\_Trener& tmp);

friend void operator >= (std::istream& is, Vera\_Trener& tmp);

bool operator==(Vera\_Trener& obj)

{

if (obj.vremia\_trener == this->vremia\_trener)

return true;

else

return false;

}

int GetVremia\_Trener();

void SetVremia\_Trener(int value);

~Vera\_Trener(){}

};

/\*Vera\_Trener.cpp

#include "Vera\_Trener.h"

int Vera\_Trener::GetVremia\_Trener()

{

return this->vremia\_trener;

}

void Vera\_Trener::SetVremia\_Trener(int per)

{

vremia\_trener = per;

}

void operator <= (std::ostream& os, Vera\_Trener& tmp)

{

os <= dynamic\_cast <Trener&>(tmp);

os.write(reinterpret\_cast<const char\*>(&tmp.vremia\_trener), sizeof(int));

}

void operator >= (std::istream& is, Vera\_Trener& tmp)

{

is >= dynamic\_cast <Trener&>(tmp);

int n;

is.read(reinterpret\_cast<char\*>(&n), sizeof(int));

tmp.SetVremia\_Trener(n);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Algoritm.h

#pragma once

#include "Shablon.h"

template <class TYPE>

class Algorithm

{

public:

Algorithm();

~Algorithm();

TYPE search1(Node<TYPE>\* beg, int c);

linklist<TYPE>& search2(Node<TYPE>\* beg, TYPE \_obj);

linklist<TYPE>& search2Iterator(Node<TYPE>\* beg, Node<TYPE>\* end, TYPE \_obj);

void sort(linklist<TYPE>& \_a);

};

template <class TYPE>

Algorithm<TYPE>::Algorithm()

{

}

template <class TYPE>

Algorithm<TYPE>::~Algorithm()

{

}

template<typename TYPE>

TYPE Algorithm<TYPE>::search1(Node<TYPE>\* beg, int c)

{

TYPE temp;

Node <TYPE>\* rab = beg;

while (c > 1)

{

rab = rab->next;

c--;

}

cout << rab->data;

temp = rab->data;

return temp;

}

template<class TYPE>

linklist<TYPE>& Algorithm<TYPE>::search2(Node<TYPE>\* beg, TYPE \_obj)

{

Node <TYPE>\* rab = beg;

linklist <TYPE> temp;

int fl = 0;

while (rab != NULL)

{

if (rab->data == \_obj)

{

cout << rab->data;

fl = 1;

temp.add\_element(rab->data, 1);

}

rab = rab->next;

}

if (fl == 0)

{

cout << "Ошибка" << endl;

}

cout << endl;

return temp;

}

template<class TYPE>

linklist<TYPE>& Algorithm<TYPE>::search2Iterator(Node<TYPE>\* beg, Node<TYPE>\* end, TYPE \_obj)

{

Node <TYPE>\* rab = beg;

linklist <TYPE> temp;

int fl = 0;

Iterator<TYPE> it;

for (it = beg; it != end->next; ++it)

{

if (\*it == \_obj)

{

cout << \*it;

fl = 1;

temp.add\_element((\*it), 1);

}

}

if (fl == 0)

{

cout << "Ошибка" << endl;

}

return temp;

}

template<class TYPE>

void Algorithm<TYPE>::sort(linklist<TYPE>& \_a)

{

Node<TYPE>\* tmp = new Node<TYPE>;

Iterator<TYPE> it\_1 = \_a.Begin();

Iterator<TYPE> it\_2 = \_a.Begin()->next;

while (it\_1 != NULL)

{

it\_2 = it\_1;

++it\_2;

while (it\_2 != NULL)

{

if ((\*it\_1) > (\*it\_2))

{

tmp->data = \*it\_1;

\*it\_1 = \*it\_2;

\*it\_2 = tmp->data;

}

++it\_2;

}

++it\_1;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Exp\_vvod.h

#pragma once

#include <iostream>

using namespace std;

class Exp\_vvod

{

int number;

char ch[80];

public:

Exp\_vvod(const Exp\_vvod& temp)

{

number = temp.number;

strcpy\_s(ch, strlen(temp.ch) + 1, temp.ch);

}

Exp\_vvod(int \_number, const char\* str)

{

strcpy\_s(ch, strlen(str) + 1, str);

number = \_number;

}

void show()

{

for (int i = 0; ch[i]; i++)

cout << ch[i];

cout << endl;

}

~Exp\_vvod()

{

}

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Iskluch.h

#pragma once

#include "Exp\_vvod.h"

#include <iomanip>

#include <conio.h>

#include <process.h>

#include <windows.h>

#include <sstream>

#include <string>

char\* pr\_str(istream& in)

{

int fe;

char buf[80];

char\* str= new char[80];

do

{

rewind(stdin);

try

{

fe = 0;

in.getline(buf, 80);

strcpy\_s(str, strlen(buf) + 1, buf);

for (int i = 0; str[i]; i++)

{

if (str[i] == ' ')

i++;

else if (str[i] < 'A' || str[i]>'z')

throw Exp\_vvod(1, "Error: Написано не на английском языке");

}

}

catch (Exp\_vvod ob)

{

fe = 1;

ob.show();

rewind(stdin);

}

} while (fe);

return str;

}

int enter\_int()

{

int x;

bool flag;

do

{

rewind(stdin);

try

{

rewind(stdin);

flag = false;

cin >> x;

if (!cin.good() || cin.peek() != '\n'||x<0)

throw overflow\_error("Введено не целое число или отрицательное");

}

catch (overflow\_error ob)

{

cin.clear();

rewind(stdin);

flag = true;

cout << "Error: " << ob.what() << endl;

}

} while (flag);

return x;

}

int enter\_int(istream& in)

{

int x;

bool flag;

do

{

rewind(stdin);

try

{

rewind(stdin);

flag = false;

in >> x;

if (!in.good() || in.peek() != '\n' || x < 0)

throw overflow\_error("Введено не целое число или отрицательное");

}

catch (overflow\_error ob)

{

in.clear();

rewind(stdin);

flag = true;

cout << "Error: " << ob.what() << endl;

}

} while (flag);

return x;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Shablon.h

#pragma once

#include <iostream>

#include <process.h>

#include <string>

#include <conio.h>

#include <iomanip>

#include <algorithm>

#include <Windows.h>

#include <vector>

using namespace std;

template<class TYPE>

struct Node

{

TYPE data;//данный

Node\* next;//указатель на след элемент

Node\* pred;//указатель на предыдущий элемент

};

template<class TYPE>

class linklist

{

private:

Node<TYPE>\* first;

Node<TYPE>\* tail;

int size;

public:

linklist()

{

first = nullptr;

tail = nullptr;

size = 0;

}

linklist(const linklist<TYPE>& obj)

{

first = obj.first;

tail = obj.tail;

size = obj.size;

}

template <typename TYPE>

friend class Iterator;

void printIterator();

void printback();

void sozd(TYPE note);

TYPE del(TYPE value);

void add\_data(TYPE data);

void pop\_data();

void udalenie(TYPE value);

void ochistka();

void show\_size();

void add\_element(TYPE note, int poz);

Node<TYPE>\* Begin() { return first; }

Node<TYPE>\* End() { return tail; }

vector<TYPE> get();

int getrazmer();

~linklist()

{

ochistka();

}

};

/\*Shablon.cpp

#include "Shablon.h"

template<typename TYPE>

void linklist<TYPE>::add\_element(TYPE note, int poz)

{

Node<TYPE>\* rab = new Node <TYPE>;

Node<TYPE>\* tmp = new Node <TYPE>;

tmp = first;

if (size == 0)

{

rab->data = note;

rab->pred = NULL;

rab->next = NULL;

first = tail = rab;

size++;

return;

}

if (poz == 1)

{

rab->data = note;

rab->pred = NULL;

rab->next = tmp;

tmp->pred = rab;

first = rab;

size++;

return;

}

if (poz == size + 1)

{

rab->data = note;

tmp = tail;

rab->pred = tmp;

rab->next = NULL;

tmp->next = rab;

tail = rab;

size++;

return;

}

while (poz > 1)

{

tmp = tmp->next;

poz--;

}

rab->data = note;

rab->next = tmp;

rab->pred = tmp->pred;

tmp->pred->next = rab;

tmp->pred = rab;

size++;

}

template<class TYPE>

void linklist<TYPE>::show\_size()

{

cout << "Размер: " << size << endl;

}

template<class TYPE>

void linklist<TYPE>::add\_data(TYPE data)

{

++size;

Node<TYPE>\* new\_element = new Node<TYPE>;

new\_element->data = data;

new\_element->next = nullptr;

new\_element->pred = nullptr;

if (!first || !tail)

{

first = tail = new\_element;

}

else

{

new\_element->next = first;

new\_element->pred = nullptr;

first->pred = new\_element;

first = new\_element;

}

}

template<class TYPE>

void linklist<TYPE>::sozd(TYPE note)

{

Node <TYPE>\* rab = new Node <TYPE>;

rab->next = NULL;

rab->data = note;

size++;

if (first != NULL)

{

rab->pred = tail;

tail->next = rab;

tail = rab;

}

else

{

rab->pred = NULL;

first = tail = rab;

}

}

template<class TYPE>

void linklist<TYPE>::pop\_data()

{

/\*cout << "Вывод листа" << endl;\*/

if (!first)

{

cout << "\nЛист пустой" << endl;

return;

}

Node<TYPE>\*n =first;

while (n)

{

cout << endl << n->data;

n = n->next;

}

cout << endl;

}

template<class TYPE>

void linklist<TYPE>::udalenie(TYPE value)

{

Node<TYPE>\*n = tail;

if (!first || !tail)

{

cout << "Лист пустой";

return;

}

if (n->pred == NULL && (n->data==value))

{

delete n;

size--;

tail = first = nullptr;

return;

}

while (n && !(n->data==value))

{

n = n->pred;

}

if (n->data==value)

{

if (n == tail)

{

tail = tail->pred;

tail->next = nullptr;

}

else if (n == first)

{

first = first->next;

first->pred = nullptr;

}

else

{

n->next->pred = n->pred;

n->pred->next = n->next;

}

delete n;

size--;

}

}

template<class TYPE>

TYPE linklist<TYPE>:: del(TYPE value)

{

Node<TYPE>\* n = tail;

TYPE per;

if (!first || !tail)

{

cout << "Лист пустой";

return value;

}

if (n->pred == NULL && (n->data == value))

{

per = n->data;

delete n;

size--;

tail = first = nullptr;

return per;

}

while (n && !(n->data == value))

{

n = n->pred;

}

if (n->data == value)

{

if (n == tail)

{

tail = tail->pred;

tail->next = nullptr;

}

else if (n == first)

{

first = first->next;

first->pred = nullptr;

}

else

{

n->next->pred = n->pred;

n->pred->next = n->next;

}

per = n->data;

delete n;

size--;

return per;

}

}

template<typename TYPE>

void linklist <TYPE>::printback()

{

Iterator<TYPE> it;

for (it = this->End(); it != this->Begin()->pred; --it)

{

cout << \*it;

cout << endl;

}

}

template<typename TYPE>

void linklist <TYPE>::printIterator()

{

Iterator<TYPE> it;

for (it = this->Begin(); it != this->End()->next; ++it)

{

cout << \*it;

cout << endl;

}

}

template<class TYPE>

vector<TYPE> linklist<TYPE>::get()

{

vector<TYPE> ww;

Iterator<TYPE> it;

for (it = this->Begin(); it != this->End()->next; ++it)

{

ww.push\_back(\*it);

}

return ww;

}

template<class TYPE>

int linklist<TYPE>::getrazmer()

{

return this->size;

}

template <class TYPE>

void linklist<TYPE>::ochistka()

{

while (first != nullptr)

{

Node<TYPE>\* n = first->next;

delete first;

first = n;

}

tail = nullptr;

size = 0;

}

template<typename TYPE>

class Iterator

{

private:

Node<TYPE>\* ptr;

public:

Iterator()

{

ptr = NULL;

}

Iterator(Node<TYPE>\* tmp)

{

ptr = tmp;

}

Iterator(const Iterator& tmp) : ptr(tmp.ptr) {}

~Iterator() { }

Iterator& operator++()

{

if (ptr->next == NULL)

{

ptr = NULL;

return \*this;

}

ptr = ptr->next;

return \*this;

}

Iterator& operator--()

{

if (ptr->pred == NULL)

{

ptr = NULL;

return \*this;

}

ptr = ptr->pred;

return \*this;

}

TYPE& operator\*()

{

return ptr->data;

}

Node<TYPE>\* operator&()

{

return ptr;

}

bool operator == (const Node<TYPE>\* tmp)

{

if (this->ptr == tmp)

{

return true;

}

return false;

}

bool operator != (const Node<TYPE>\* tmp)

{

if (this->ptr != tmp)

{

return true;

}

return false;

}

Iterator& operator=(const Node <TYPE>& tmp)

{

if (this->ptr != tmp)

{

ptr = tmp;

}

return \*this;

}

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*StackOtmena.h

#pragma once

#include <iostream>

#include <process.h>

#include <string>

#include <conio.h>

#include <iomanip>

#include <algorithm>

#include <Windows.h>

#include <vector>

using namespace std;

template<class TYPE>

struct Value

{

TYPE data;//данный

Value\* next;//указатель на след элемент

};

template<class TYPE>

class Stack

{

private:

Value<TYPE>\* first;

int razmer;

public:

Stack()

{

first = nullptr;

razmer = 0;

}

Stack(const Stack<TYPE>& obj)

{

first = obj.first;

razmer = obj.razmer;

}

TYPE del();

void add\_data(TYPE data);

void pop\_data();

void ochistka();

void show\_razmer();

~Stack()

{

ochistka();

}

};

template<class TYPE>

void Stack<TYPE>::show\_razmer()

{

cout << "Размер: " << razmer << endl;

}

template<class TYPE>

void Stack<TYPE>::add\_data(TYPE data)

{

++razmer;

Value<TYPE>\* new\_element = new Value<TYPE>;

new\_element->data = data;

new\_element->next = nullptr;

if (!first)

{

first= new\_element;

}

else

{

new\_element->next = first;

first = new\_element;

}

}

template<class TYPE>

void Stack<TYPE>::pop\_data()

{

if (!first)

{

cout << "\nСтек пустой" << endl;

return;

}

Value<TYPE>\* n = first;

while (n)

{

cout << endl << n->data;

n = n->next;

}

cout << endl;

}

template<class TYPE>

TYPE Stack<TYPE>::del()

{

TYPE str;

Value<TYPE>\* n = first;

first = first->next;

str = n->data;

delete n;

return str;

}

template <class TYPE>

void Stack<TYPE>::ochistka()

{

while (first != nullptr)

{

Value<TYPE>\* n = first->next;

delete first;

first = n;

}

razmer = 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*String.h

#pragma once

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <process.h>

#include <windows.h>

#include <string>

#include <sstream>

using namespace std;

class String

{

char\* mas;

int razmer;

public:

String() : razmer(0)

{

mas = nullptr;

}

String(const String& ss)

{

this->razmer = strlen(ss.mas);

this->mas = new char[razmer + 1];

strcpy\_s(this->mas, strlen(ss.mas) + 1, ss.mas);

}

String(int length)

{

razmer = length;

mas = new char[length + 1];

}

friend ostream& operator<<(ostream& on, String const& ss);

friend istream& operator>>(istream& in, String& ss);

String& operator=(const String& obj);

String& operator+=(const String& obj);

String operator+(const String& obj);

bool operator>(const String& obj);

bool operator<(const String& obj);

bool operator==(String& obj);

friend bool operator!=(const String& obj, const String& obj1);

char& operator[](int s);

char\* operator = (const char\* str);

void operator = (char str[]); // для константных строк

bool operator == (const String& tmp) const;

bool operator != (const char\* tmp) const;

bool operator ==(string str);

/\*bool operator == (const char\* tmp) const;\*/

void operator += (char\* str)

{

int len = this->razmer + strlen(str);

char\* tmp\_str = new char[this->razmer + 1];

strcpy\_s(tmp\_str,strlen(this->mas)+1, this->mas);

if (this->razmer != 0)

delete this->mas;

mas = new char[len + 1];

strcpy\_s(mas,strlen(tmp\_str)+1, tmp\_str);

strcat\_s(mas,strlen(mas)+strlen(str)+1, str);

this->razmer = len;

}

~String();

};

/\*String.cpp

#include "String.h"

istream& operator>>(istream& in, String& ss)

{

if(ss.mas!=NULL)

delete[] ss.mas;

delete[] ss.mas;

char buf[80];

rewind(stdin);

in.getline(buf, sizeof buf);

ss.razmer = strlen(buf);

ss.mas = new char[ss.razmer + 1];

strcpy\_s(ss.mas, strlen(buf) + 1, buf);

return in;

}

ostream& operator<<(ostream& on, const String& ss)

{

on << ss.mas;

return on;

}

String& String :: operator =(const String& obj)

{

if (this != &obj)

{

delete[] mas;

razmer = obj.razmer;

mas = new char[razmer + 1];

strcpy\_s(mas, strlen(obj.mas) + 1, obj.mas);

}

return \*this;

}

String& String::operator+=(const String& obj)

{

\*this = \*this + obj;

return \*this;

}

//перегруженный оператор присваивания String - char \*

char\* String::operator = (const char\* str)

{

delete[] this->mas;

this->razmer = strlen(str);

this->mas = new char[this->razmer + 1];

strcpy\_s(this->mas,strlen(str)+1, str);

return mas;

}

//перегруженный оператор присваивания String - char str[]

void String::operator = (char str[])

{

delete[] this->mas;

this->razmer = strlen(str);

this->mas = new char[this->razmer + 1];

strcpy\_s(this->mas,strlen(str)+1, str);

}

//перегруженный оператор сравнени на равенство String - String

bool String::operator == (const String& tmp) const

{

if ((strcmp(this->mas, tmp.mas))==0)

return true;

else

return false;

}

bool String::operator == (String& tmp)

{

if ((strcmp(this->mas, tmp.mas)) == 0)

return true;

else

return false;

}

bool String::operator ==(string str)

{

if ((strcmp(this->mas, str.c\_str())) == 0)

return true;

else

return false;

}

bool String::operator != (const char\* tmp) const

{

if ((strcmp(mas, tmp)) != 0)

return true;

else

return false;

}

bool operator!=(const String& obj, const String& obj1)

{

if ((strcmp(obj.mas, obj1.mas)) != 0)

return true;

else

return false;

}

char& String::operator[](int s)

{

if (s<0 || s>razmer)

{

cout << "Выход за пределы массива" << endl;

exit(1);

}

return mas[s];

}

String String::operator+(const String& obj)

{

String temp;

int dlina = strlen(mas);

int dlina1 = strlen(obj.mas);

temp.razmer = dlina + dlina1;

temp.mas = new char[dlina + dlina1 + 1];

int i;

strcpy\_s(temp.mas, strlen(mas) + 1, mas);

strcat\_s(temp.mas, strlen(mas) + strlen(obj.mas) + 1, obj.mas);

return temp;

}

bool String::operator>(const String& obj)

{

if ((strcmp(this->mas, obj.mas)) > 0)

return true;

}

bool String::operator<(const String& obj)

{

if ((strcmp(this->mas, obj.mas)) < 0)

return false;

}

String::~String()

{

delete[] this->mas;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*TXTFILE.h

#pragma once

#include <iostream>

#include <fstream>

#include "Shablon.h"

using namespace std;

class TXTFILE

{

public:

TXTFILE() {};

TXTFILE(string \_title);

~TXTFILE() {};

template<class TYPE>

void toFile(TYPE& obj, string \_filename);

template<class TYPE>

void fromFile(TYPE& obj, string \_filename, int i);

int checkCount(string \_filename);

void clear(string \_filename);

};

template<class TYPE>

void TXTFILE::toFile(TYPE& obj, string \_filename)

{

ofstream ofs(\_filename, ofstream::app);

if (!ofs)

{

cout << "Не удалось открыть файл: " << \_filename;

system("pause");

return;

}

ofs <<= obj;

ofs << endl;

ofs.close();

}

template<class TYPE>

void TXTFILE::fromFile(TYPE& obj, string \_filename, int i)

{

ifstream ifs(\_filename, ifstream::in);

if (!ifs)

{

cout << "Не удалось открыть файл: " << \_filename;

system("pause");

return;

}

string s;

while (i > 0)

{

getline(ifs, s);

i--;

}

ifs >>= obj;

ifs.close();

}

int TXTFILE::checkCount(string \_filename)

{

ifstream ifs(\_filename, ifstream::in);

if (!ifs)

{

cout << "Не удалось открыть файл: " << \_filename;

system("pause");

return -1;

}

int count = 0;

string ss;

while (getline(ifs, ss))

{

count++;

}

ifs.close();

return count;

}

void TXTFILE::clear(string \_filename)

{

ofstream ofs(\_filename, ios::out,ios::trunc);

ofs.close();

}