Ocнова.сpp

-------------------------------------------------------------------------------------

#include"Menu.cpp"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "rus");

while (1)

{

system("cls");

cout << " ГЛАВНОЕ МЕНЮ" << endl;

cout << " 1 - Войти от имени администратора" << endl;

cout << " 2 - Войти от имени пользователя" << endl;

cout << " 0 - Выйти" << endl;

cout << "Выберите команду:";

switch (cin.get())

{

case '1':

{

MenuADM();

break;

}

case '2':

{

MenuUSER();

break;

}

case '0':

{

return 0;

}

}

}

}

Menu.h

#pragma once

//классы для работы с данными

#include<iostream>

#include"Shablon.cpp"

#include"locale.h"

#include"windows.h"

#include "Mouse.h"

#include"touchMouse.h"

#include "Devices.h"

#include"wiredMouse.h"

#include"Monitor.h"

#include"LCD.h"

#include"PDP.h"

#include<iomanip>

#include"TextFile.cpp"

#include"Iter.cpp"

#include"Algoritm.cpp"

#include<vector>

using namespace std;

inline void shapka(int nm);

template<typename T>

void ADM(Deque<T> \_c, T \_a , int \_k);

template<typename T>

void USER(Deque<T> \_c, T \_a, int \_k);

inline int MenuADM();

inline int MenuUSER();

Menu.cpp

-------------------------------------------------------------------------------------

#pragma once

#include "Menu.h"

void shapka(int nm)

{

if (nm == 1)

{

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

cout << setiosflags(ios::left)

<< setw(13) << "Год\_выпуска"

<< setw(11) << "Бренд"

<< setw(10) << "Цена"

<< setw(14) << "Чувст. мыши"

<< setw(12) << "Город"

<< setw(17) << "Частота курсора"

<< setw(15) << "Кол-во датчиков"

<< endl;

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

}

if (nm == 2)

{

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

cout << setiosflags(ios::left)

<< setw(13) << "Год\_выпуска"

<< setw(10) << "Бренд"

<< setw(10) << "Цена"

<< setw(14) << "Чувст. мыши"

<< setw(12) << "Материал"

<< setw(15) << "Улица"

<< setw(17) << " Длина провода"

<< setw(13) << "Время отклика"

<< endl;

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

}

if (nm == 3)

{

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

cout << setiosflags(ios::left)

<< setw(13) << "Год\_выпуска"

<< setw(12) << "Бренд"

<< setw(10) << "Цена"

<< setw(15) << "Разреш. экрана"

<< setw(15) << "Кол-во дюймов"

<< setw(14) << "Mатрица"

<< setw(11) << "Время вкл."

<< endl;

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

}

if (nm == 4)

{

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

cout << setiosflags(ios::left)

<< setw(13) << "Год\_выпуска"

<< setw(10) << "Бренд"

<< setw(10) << "Цена"

<< setw(16) << "Разреш. экрана"

<< setw(16) << "Кол-во дюймов"

<< setw(12) << "Плазма"

<< setw(10) << "Яркость"

<< endl;

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

}

}

template<typename T>

void USER(Deque<T> \_c, T \_a, int \_k)

{

Deque<T>\* data1 = new Deque<T>;

Algorithm<T> alg;

string k;

T\* tmp1;

int z = 10;

tmp1 = new T[z];

T ss;

int w;

while (1)

{

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

cout <<"1-Сортировка по контейнеру" << endl <<"2-поиск по контейнеру" << endl << "3-поиск по контейнеру с итератором" << endl

<< "4-Прочесть файл" << endl << "5-Выход" << endl;

w = enter();

switch (w)

{

case 1:

{

system("cls");

shapka(\_k);

data1->show(1);

alg.sort(\*data1);

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

shapka(\_k);

data1->show(1);

break;

}

case 2:

{

system("cls");

shapka(\_k);

data1->show(1);

cin >> ss;

alg.search(data1->Begin(), ss);

break;

}

case 3:

{

system("cls");

shapka(\_k);

data1->show(1);

cin >> ss;

alg.search2(data1->Begin(), data1->End(), ss);

break;

}

case 4:

{

system("cls");

textFile<T> b;

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

int count;

count = b.checkCount(q);

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

{

z++;

b.fromFile(tmp1[i], q, i);

data1->Push(tmp1[i]);

}

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

shapka(\_k);

data1->show(1);

break;

}

case 5:

{

return;

}

}

}

}

template<typename T>

void ADM(Deque<T> \_c, T \_a , int \_k)

{

system("cls");

Deque<T>\* data1 = new Deque<T>;

string k;

T\* tmp1;

textFile<T> b;

Algorithm<T> alg;

vector<T> vect;

T s;

T ss;

int w;

int z;

int count;

cout << "Выделить память " << endl;

cin >> z;

int cc;

tmp1 = new T[z];

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

{

cin >> s;

tmp1[i] = s;

data1->Push(s);

}

while (1)

{

rewind(stdin);

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

cout << "1-Добавить в конец deque" << endl << "2-Показать deque" << endl

<< "3-Удалить элемент" << endl << "4-Удалить любой элемент" << endl

<< "5-Очистить deque" << endl << "6-Узнать размер deque" << endl << "7-Сортировка" << endl

<< "8-Поиск по очереди" << endl << "9-Поиск по очереди(Iter)" << endl<< "10-Прочесть файл" << endl

<< "11-Записать в файл" << endl << "12-Очистка файла" << endl <<"13-Отмена действия " << endl

<< "14-Выход" << endl;

w = enter();

switch (w)

{

case 1:

{

system("cls");

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

vect.clear();

b.clear(q);

vect = data1->get();

for (int i = 0; i < data1->show\_size(); i++)

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

system("cls");

cin >> s;

\*tmp1 = s;

data1->Push(s);

system("cls");

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

system("cls");

break;

}

case 2:

{

system("cls");

shapka(\_k);

data1->show(1);

break;

}

case 3:

{

system("cls");

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

b.clear(q);

vect.clear();

vect = data1->get();

for (int i = 0; i < data1->show\_size(); i++)

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

system("cls");

data1->pop(1);

cout << "deque после удаления " << endl;

shapka(\_k);

data1->show(1);

break;

}

case 4:

{

system("cls");

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

b.clear(q);

vect.clear();

vect = data1->get();

for (int i = 0; i < data1->show\_size(); i++)

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

system("cls");

int i = 0;

cout << "Какой элемент удалить " << endl;

cin >> i;

data1->pop\_any(tmp1[i]);

cout << "deque после удаления " << endl;

shapka(\_k);

data1->show(1);

break;

}

case 5:

{

system("cls");

data1->delet();

break;

}

case 6:

{

system("cls");

cout << data1->show\_size() << endl;

break;

}

case 7:

{

system("cls");

shapka(\_k);

data1->show(1);

alg.sort(\*data1);

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

shapka(\_k);

data1->show(1);

break;

}

case 8:

{

system("cls");

shapka(\_k);

data1->show(1);

cin >> ss;

alg.search(data1->Begin(), ss);

break;

}

case 9:

{

system("cls");

shapka(\_k);

data1->show(1);

cin >> ss;

alg.search2(data1->Begin(), data1->End(), ss);

break;

}

case 10:

{

system("cls");

Deque<T> c;

T\* tmp2;

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

int count;

count = b.checkCount(q);

tmp2 = new T[count];

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

{

b.fromFile(tmp2[i], q, i);

c.Push(tmp2[i]);

}

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

shapka(\_k);

data1->show(0);

break;

}

case 11:

{

system("cls");

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

vector<T> ww;

ww = data1->get();

textFile<T> b;

for (int i = 0; i < data1->show\_size(); i++)

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

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

break;

}

case 12:

{

system("cls");

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

b.clear(q);

cout << "Файл очищен." << endl;

break;

}

case 13:

{

system("cls");

data1->delet();

string q;

cout << "Введите имя файла" << endl;

cin >> k;

q = k + ".txt";

int count;

count = b.checkCount(q);

tmp1 = new T[count];

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

{

b.fromFile(tmp1[i], q, i);

data1->Push(tmp1[i]);

}

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

shapka(\_k);

data1->show(0);

break;

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

break;

}

case 14:

{

return;

}

}

}

}

int MenuADM()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "Russian");

while (1)

{

system("cls");

cout << " МЕНЮ АДМИНА" << endl;

cout << " 1 - Редактировать информацию(touchMouse)" << endl;

cout << " 2 - Редактировать информацию(wiredMouse)" << endl;

cout << " 3 - Редактировать информацию(LCD)" << endl;

cout << " 4 - Редактировать информацию(PDP)" << endl;

cout << " 0 - Выйти" << endl;

cout << "Выберите команду:";

int k;

cin >> k;

switch (k)

{

case 1:

{

Deque<touchMouse> c;

touchMouse a;

int k = 1;

ADM(c, a , k);

break;

}

case 2:

{

Deque<wiredMouse> c;

wiredMouse a;

int k = 2;

ADM(c, a, k);

break;

}

case 3:

{

Deque<LCD> c;

LCD a;

int k = 3;

ADM(c, a, k);

break;

}

case 4:

{

Deque<PDP> c;

PDP a;

int k = 4;

ADM(c, a, k);

break;

}

case 0:

{

return 0;

}

}

cout << endl;

system("pause");

}

system("pause");

return 0;

}

int MenuUSER()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "Russian");

while (1)

{

system("cls");

cout << " МЕНЮ ПОЛЬЗОВАТЕЛЯ" << endl;

cout << " 1 - Показать информацию(touchMouse)" << endl;

cout << " 2 - Показать информацию(wiredMouse)" << endl;

cout << " 3 - Показать информацию(LCD)" << endl;

cout << " 4 - Показать информацию(PDP)" << endl;

cout << " 0 - Выйти" << endl;

cout << "Выберите команду:";

int k;

int z;

k = enter();

switch (k)

{

case 1:

{

Deque<touchMouse> c;

touchMouse a;

int k = 1;

USER(c, a, k);

break;

}

case 2:

{

Deque<wiredMouse> c;

wiredMouse a;

int k = 2;

USER(c, a, k);

break;

}

case 3:

{

Deque<LCD> c;

LCD a;

int k = 3;

USER(c, a, k);

break;

}

case 4:

{

Deque<PDP> c;

PDP a;

int k = 4;

USER(c, a, k);

break;

}

case 0:

{

return 0;

}

}

cout << endl;

system("pause");

}

system("pause");

return 0;

}

Device.h

#pragma once

#include<iostream>

#include "String.h"

#include<iomanip>

#include<fstream>

#include<sstream>

using namespace std;

class Devices

{

protected:

String yearOfIssue;

String brand;

int cost;

public:

Devices()

{

this->cost = 0;

}

Devices(char\* \_yearOfIssue, char\* \_brand, int cost)

{

this->yearOfIssue = \_yearOfIssue;

this->brand = \_brand;

this->cost = cost;

}

friend ostream& operator<<(ostream& on, Devices& ss);

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

friend void operator <<= (ostream& stream, Devices& tmp);

friend void operator >>= (istream& is, Devices& tmp);

friend void operator <= (ostream& os, Devices& tmp);

friend void operator >= (istream& is, Devices& tmp);

inline String getName() { return yearOfIssue; }

inline void setName(const String os) { this->yearOfIssue = os; }

inline String getBrand() { return brand; }

inline void setBrand(const String os) { this->brand = os; }

inline int getCost() { return cost; }

inline void setCost(const int os) { this->cost = os; }

Devices& operator=(const Devices& other);

bool operator==(Devices& other);

bool operator==(char\* other);

bool operator > (const Devices& tmp);

~Devices()

{ }

};

Device.cpp

#include"Devices.h"

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

{

stream << tmp.yearOfIssue << "|" << tmp.brand << "|" << tmp.cost << "|" ;

}

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

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

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

{

int n = sizeof(tmp.yearOfIssue);

char\* t = new char[n];

t = tmp.yearOfIssue.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

n = sizeof(tmp.brand);

t = new char[n];

t = tmp.brand.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

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

}

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

{

int n;

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

char\* t = new char[n];

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

tmp.setName(t);

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

t = new char[n];

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

tmp.setBrand(t);

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

tmp.setCost(n);

}

ostream& operator<<(ostream& on, Devices& ss)

{

rewind(stdin);

return on << setiosflags(ios::left)

<< setw(13) << ss.yearOfIssue

<< setw(13) << ss.brand

<< setw(10) << ss.cost;

}

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

{

cout << "Введите год выпуска :";

in >> ss.yearOfIssue;

cout << "Введите бренд :";

in >> ss.brand;

cout << "Введите цену :";

int enter(istream & in);

int ch = enter(in);

ss.cost = ch;

return in;

}

Devices& Devices::operator=(const Devices& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

return \*this;

}

bool Devices::operator==(Devices& other)

{

if ((this->yearOfIssue == other.yearOfIssue) )

return true;

else

return false;

}

bool Devices::operator==(char\* other)

{

if ((this->yearOfIssue == other) )

return true;

else

return false;

}

bool Devices::operator > (const Devices& tmp)

{

if (this->yearOfIssue > tmp.yearOfIssue)

return true;

if (this->brand < tmp.brand)

return false;

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

return false;

}

Mouse.h

#pragma once

#include<iostream>

#include "Devices.h"

using namespace std;

class Mouse : public Devices

{

protected:

String material;

String mouseSensitivity;

public:

Mouse()

{

}

Mouse(char\* \_mouseSensitivity, char\* \_material, char\* \_yearOfIssue, char\* \_brand, int cost) :Devices(\_yearOfIssue, \_brand, cost)

{

this->mouseSensitivity = \_mouseSensitivity;

this->material = \_material;

}

inline String getMouseSensitivity() { return this->mouseSensitivity; }

inline void setMouseSensitivity(const String os) { this->mouseSensitivity = os; }

inline String getMaterial() { return this->material; }

inline void setMaterial(const String os) { this->material = os; }

friend ostream& operator<<(ostream& on, Mouse& ss);

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

friend void operator <<= (ostream& stream, Mouse& tmp);

friend void operator >>= (istream& is, Mouse& tmp);

friend void operator <= (ostream& os, Mouse& tmp);

friend void operator >= (istream& is, Mouse& tmp);

Mouse& operator=(const Mouse& other);

bool operator==(Mouse& other);

bool operator==(char\* other);

bool operator > (const Mouse& tmp);

~Mouse()

{ }

};

Mouse.cpp

#include"Mouse.h"

void operator <<= (ostream& stream, Mouse& tmp)

{

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

stream << tmp.mouseSensitivity << "|" << tmp.material << "|";

}

void operator >>= (istream& stream, Mouse& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

void operator <= (ostream& os, Mouse& tmp)

{

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

int n = sizeof(tmp.mouseSensitivity);

char\* t = new char[n];

t = tmp.mouseSensitivity.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

n = sizeof(tmp.material);

t = new char[n];

t = tmp.material.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

}

void operator >= (istream& is, Mouse& tmp)

{

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

int n;

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

char\* t = new char[n];

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

tmp.setMouseSensitivity(t);

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

t = new char[n];

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

tmp.setMaterial(t);

}

ostream& operator<<(ostream& on, Mouse& ss)

{

on << dynamic\_cast<Devices&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.mouseSensitivity

<< setw(15) << ss.material;

return on;

}

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

{

in >> dynamic\_cast<Devices&>(ss);

cout << "Введите уровень чувствительности мыши :";

in >> ss.mouseSensitivity;

cout << "Введите материал :";

in >> ss.material;

return in;

}

Mouse& Mouse::operator=(const Mouse& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->mouseSensitivity = other.mouseSensitivity;

this->material = other.material;

return \*this;

}

bool Mouse::operator==(Mouse& other)

{

if ((this->mouseSensitivity == other.mouseSensitivity))

return true;

else

return false;

}

bool Mouse::operator==(char\* other)

{

if ((this->mouseSensitivity == other))

return true;

else

return false;

}

bool Mouse::operator > (const Mouse& tmp)

{

if (this->mouseSensitivity > tmp.mouseSensitivity)

return true;

if (this->material < tmp.material)

return false;

}

Monitor.h

#pragma once

#include"Monitor.h"

#include<iostream>

#include "Devices.h"

#include "String.h"

using namespace std;

class Monitor : public Devices

{

protected:

String screenResolution;

int numberOfInches;

public:

Monitor() : Devices()

{

numberOfInches = 0;

}

Monitor(char\* \_screenResolution,int \_numberOfInches , char\* \_yearOfIssue, char\* \_brand , int cost) : Devices(\_yearOfIssue, \_brand, cost)

{

this->screenResolution = \_screenResolution;

this->numberOfInches = \_numberOfInches;

}

String getScreenResolution() { return screenResolution; }

void setScreenResolution(const String os) { this->screenResolution = os; }

int getNumberOfInches() { return numberOfInches; }

void setNumberOfInches(const int os) { this->numberOfInches = os; }

friend ostream& operator<<(ostream& on, Monitor& ss);

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

friend void operator <<= (ostream& stream, Monitor& tmp);

friend void operator >>= (istream& is, Monitor& tmp);

friend void operator <= (ostream& os, Monitor& tmp);

friend void operator >= (istream& is, Monitor& tmp);

Monitor& operator=(const Monitor& other);

bool operator==(Monitor& other);

bool operator==( char\* other);

bool operator > (const Monitor& tmp);

~Monitor()

{ }

};

Monitor.cpp

#include"Monitor.h"

void operator <<= (ostream& stream, Monitor& tmp)

{

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

stream << tmp.screenResolution << "|" << tmp.numberOfInches << "|";

}

void operator >>= (istream& stream, Monitor& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

void operator <= (ostream& os, Monitor& tmp)

{

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

int n = sizeof(tmp.screenResolution);

char\* t = new char[n];

t = tmp.screenResolution.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

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

}

void operator >= (istream& is, Monitor& tmp)

{

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

int n;

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

char\* t = new char[n];

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

tmp.setScreenResolution(t);

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

tmp.setNumberOfInches(n);

}

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

{

in >> dynamic\_cast<Devices&>(ss);

cout << "Введите разрешение экрана :";

in >> ss.screenResolution;

cout << "Введите кол-во дюймов :";

int enter(istream & in);

int ch = enter(in);

ss.numberOfInches = ch;

return in;

}

ostream& operator<<(ostream& on, Monitor& ss)

{

on << dynamic\_cast<Devices&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.screenResolution

<< setw(15) << ss.numberOfInches;

return on;

}

Monitor& Monitor::operator=(const Monitor& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->screenResolution = other.screenResolution;

this->numberOfInches = other.numberOfInches;

return \*this;

}

bool Monitor::operator==(Monitor& other)

{

if ((this->screenResolution == other.screenResolution))

return true;

else

return false;

}

bool Monitor::operator==( char\* other)

{

if ((this->screenResolution == other))

return true;

else

return false;

}

bool Monitor::operator > (const Monitor& tmp)

{

if (this->screenResolution > tmp.screenResolution)

return true;

if (this->numberOfInches < tmp.numberOfInches)

return false;

}

touchMouse.h

#pragma once

#include<iostream>

#include"Mouse.h"

using namespace std;

class touchMouse : public Mouse

{

protected:

int numberOfSensors;

String cursorFrequency;

public:

touchMouse() : Mouse()

{

this->numberOfSensors = 0;

}

touchMouse(char\* \_cursorFrequency, int \_numberOfSensors , char\* \_mouseSensitivity, char\* \_material, char\* \_yearOfIssue, char\* \_brand, int cost)

: Mouse(\_mouseSensitivity, \_material, \_yearOfIssue, \_brand, cost)

{

this->cursorFrequency = \_cursorFrequency;

this->numberOfSensors = \_numberOfSensors;

}

String getCursorFrequency() { return this->cursorFrequency; }

void setCursorFrequency(const String os) { this->cursorFrequency = os; }

int getNumberOfSensors() { return this->numberOfSensors; }

void setNumberOfSensors(const int os) { this->numberOfSensors = os; }

friend ostream& operator<<(ostream& on, touchMouse& ss);

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

friend void operator <<= (ostream& stream, touchMouse& tmp);

friend void operator >>= (istream& is, touchMouse& tmp);

friend void operator <= (ostream& os, touchMouse& tmp);

friend void operator >= (istream& is, touchMouse& tmp);

touchMouse& operator=(const touchMouse& other);

bool operator==(touchMouse& other);

bool operator==(char\* other);

bool operator > (const touchMouse& tmp);

~touchMouse()

{ }

};

touchMouse.cpp

#include"touchMouse.h"

void operator <<= (ostream& stream, touchMouse& tmp)

{

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

stream << tmp.cursorFrequency << "|" << tmp.numberOfSensors << "|";

stream << endl;

}

void operator >>= (istream& stream, touchMouse& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

void operator <= (ostream& os, touchMouse& tmp)

{

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

int n = sizeof(tmp.cursorFrequency);

char\* t = new char[n];

t = tmp.cursorFrequency.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

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

}

void operator >= (istream& is, touchMouse& tmp)

{

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

int n;

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

char\* t = new char[n];

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

tmp.setCursorFrequency(t);

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

tmp.setNumberOfSensors(n);

}

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

{

in >> dynamic\_cast<Mouse&>(ss);

cout << "Введите уровень частоты курсора :";

in >> ss.cursorFrequency;

cout << "Введите кол-во датчиков :";

int enter(istream & in);

int ch = enter(in);

ss. numberOfSensors = ch;

return in;

}

ostream& operator<<(ostream& on, touchMouse& ss)

{

on << dynamic\_cast<Mouse&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.cursorFrequency

<< setw(15) << ss.numberOfSensors

<< endl;

return on;

}

touchMouse& touchMouse::operator=(const touchMouse& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->mouseSensitivity = other.mouseSensitivity;

this->material = other.material;

this->cursorFrequency = other.cursorFrequency;

this->numberOfSensors = other.numberOfSensors;

return \*this;

}

bool touchMouse::operator==(touchMouse& other)

{

if (this->cursorFrequency == other.cursorFrequency)

{

return true;

}

else

{

return false;

}

}

bool touchMouse::operator==(char\* other)

{

if ((this->cursorFrequency == other))

return true;

else

return false;

}

bool touchMouse::operator > (const touchMouse& tmp)

{

if (this->cursorFrequency > tmp.cursorFrequency)

return true;

if (this->numberOfSensors < tmp.numberOfSensors)

return false;

}

wiredMouse.h

#pragma once

#include<iostream>

#include"Mouse.h"

#include "String.h"

using namespace std;

class wiredMouse : public Mouse

{

protected:

int responseTime; //время отклика

int wireLength; //длина провода

String scrollingWay; //сп-б прокрутки

public:

wiredMouse() : Mouse()

{

this->responseTime = 0;

this->wireLength = 0;

}

wiredMouse(char\* \_scrollingWay, int \_wireLength, int \_responseTime, char\* \_mouseSensitivity, char\* \_material, char\* \_yearOfIssue, char\* \_brand, int cost)

: Mouse(\_mouseSensitivity, \_material, \_yearOfIssue, \_brand, cost)

{

this->scrollingWay = \_scrollingWay;

this->wireLength = \_wireLength;

this->responseTime = \_responseTime;

}

String getScrollingWay() { return this->scrollingWay; }

void setScrollingWay(const String os) { this->scrollingWay = os; }

int getResponseTime() { return this->responseTime; }

void setResponseTime(const int os) { this->responseTime = os; }

int getWireLength() { return this->wireLength; }

void setWireLength(const int os) { this->wireLength = os; }

friend ostream& operator<<(ostream& on, wiredMouse& ss);

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

friend void operator <<= (ostream& stream, wiredMouse& tmp);

friend void operator >>= (istream& is, wiredMouse& tmp);

friend void operator <= (ostream& os, wiredMouse& tmp);

friend void operator >= (istream& is, wiredMouse& tmp);

wiredMouse& operator=(const wiredMouse& other);

bool operator==(wiredMouse& other);

bool operator==( char\* other);

bool operator > (const wiredMouse& tmp);

~wiredMouse()

{ }

};

wiredMouse.cpp

#include"wiredMouse.h"

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

{

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

stream << tmp.scrollingWay << "|" << tmp.wireLength << "|" << tmp.responseTime << "|";

stream << endl;

}

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

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

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

}

}

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

{

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

int n = sizeof(tmp.scrollingWay);

char\* t = new char[n];

t = tmp.scrollingWay.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

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

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

}

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

{

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

int n;

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

char\* t = new char[n];

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

tmp.setScrollingWay(t);

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

tmp.setWireLength(n);

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

tmp.setResponseTime(n);

}

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

{

in >> dynamic\_cast<Mouse&>(ss);

cout << "Введите способ прокрутки :";

in >> ss.scrollingWay;

cout << "Введите длину провода :";

int enter(istream & in);

int ch = enter(in);

ss. wireLength = ch;

cout << "Введите время отклика :";

int enter(istream & in);

int ch = enter(in);

ss. responseTime = ch;

return in;

}

ostream& operator<<(ostream& on, wiredMouse& ss)

{

on << dynamic\_cast<Mouse&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.scrollingWay

<< setw(15) << ss.wireLength

<< setw(15) << ss.responseTime

<< endl;

return on;

}

wiredMouse& wiredMouse::operator=(const wiredMouse& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->mouseSensitivity = other.mouseSensitivity;

this->material = other.material;

this->scrollingWay = other.scrollingWay;

this->wireLength = other.wireLength;

this->responseTime = other.responseTime;

return \*this;

}

bool wiredMouse::operator==(wiredMouse& other)

{

if (this->scrollingWay == other.scrollingWay)

{

return true;

}

else

{

return false;

}

}

bool wiredMouse::operator==( char\* other)

{

if ((this->scrollingWay == other))

return true;

else

return false;

}

bool wiredMouse::operator > (const wiredMouse& tmp)

{

if (this->scrollingWay > tmp.scrollingWay)

return true;

if (this->wireLength < tmp.wireLength)

return false;

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

return false;

}

LCD.h

#pragma once

#include<iostream>

#include"Monitor.h"

class LCD : public Monitor

{

protected:

String matrix;

int turnOnSpeed;

public:

LCD() : Monitor()

{

this->turnOnSpeed = 0;

}

LCD(char\* \_matrix, int \_turnOnSpeed , char\* \_screenResolution,int \_numberOfInches, char\* \_yearOfIssue, char\* \_brand,int cost)

: Monitor(\_screenResolution, \_numberOfInches,\_yearOfIssue, \_brand, cost )

{

this->matrix = \_matrix;

this->turnOnSpeed = \_turnOnSpeed;

}

String getMatrix() { return matrix; }

void setMatrix(const String os) { this->matrix = os; }

int getTurnOnSpeed() { return turnOnSpeed; }

void setTurnOnSpeed(const int os) { this->turnOnSpeed = os; }

friend ostream& operator<<(ostream& on, LCD& ss);

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

friend void operator <<= (ostream& stream, LCD& tmp);

friend void operator >>= (istream& is, LCD& tmp);

friend void operator <= (ostream& os, LCD& tmp);

friend void operator >= (istream& is, LCD& tmp);

LCD& operator=(const LCD& other);

bool operator==(LCD& other);

bool operator==( char\* other);

bool operator > (const LCD& tmp);

~LCD()

{ }

};

LCD.cpp

#include"LCD.h"

void operator <<= (ostream& stream, LCD& tmp)

{

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

stream << tmp.matrix << "|" << tmp.turnOnSpeed << "|";

stream << endl;

}

void operator >>= (istream& stream, LCD& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

void operator <= (ostream& os, LCD& tmp)

{

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

int n = sizeof(tmp.matrix);

char\* t = new char[n];

t = tmp.matrix.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

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

}

void operator >= (istream& is, LCD& tmp)

{

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

int n;

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

char\* t = new char[n];

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

tmp.setMatrix(t);

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

tmp.setTurnOnSpeed(n);

}

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

{

in >> dynamic\_cast<Monitor&>(ss);

cout << "Введите какая матрица установлена :";

in >> ss.matrix;

cout << "Введите время включения :";

int enter(istream & in);

int ch = enter(in);

ss.turnOnSpeed = ch;

return in;

}

ostream& operator<<(ostream& on, LCD& ss)

{

on << dynamic\_cast<Monitor&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.matrix

<< setw(15) << ss.turnOnSpeed

<< endl;

return on;

}

LCD& LCD::operator=(const LCD& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->screenResolution = other.screenResolution;

this->numberOfInches = other.numberOfInches;

this->matrix = other.matrix;

this->turnOnSpeed = other.turnOnSpeed;

return \*this;

}

bool LCD::operator==(LCD& other)

{

if ((this->matrix == other.matrix))

return true;

else

return false;

}

bool LCD::operator==( char\* other)

{

if ((this->matrix == other))

return true;

else

return false;

}

bool LCD::operator > (const LCD& tmp)

{

if (this->matrix > tmp.matrix)

return true;

if (this->turnOnSpeed < tmp.turnOnSpeed)

return false;

}

PDP.h

#pragma once

#include<iostream>

#include"Monitor.h"

using namespace std;

class PDP : public Monitor

{

protected:

String plasma;

String brightnessLevel;

public:

PDP() : Monitor()

{

}

PDP(char\* \_plasma, char\* \_brightnessLevel, char\* \_screenResolution,int \_numberOfInches, char\* \_yearOfIssue, char\* \_brand,int cost)

: Monitor(\_screenResolution, \_numberOfInches, \_yearOfIssue, \_brand, cost)

{

this->plasma = plasma;

this->brightnessLevel = \_brightnessLevel;

}

String getPlasma() { return plasma; }

void setPlasma(const String os) { this->plasma = os; }

String getBrightnessLevel() { return brightnessLevel; }

void setBrightnessLevel(const String os) { this->brightnessLevel = os; }

friend ostream& operator<<(ostream& on, PDP& ss);

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

friend void operator <<= (ostream& stream, PDP& tmp);

friend void operator >>= (istream& is, PDP& tmp);

friend void operator <= (ostream& os, PDP& tmp);

friend void operator >= (istream& is, PDP& tmp);

PDP& operator=(const PDP& other);

bool operator==(PDP& other);

bool operator==( char\* other);

bool operator > (const PDP& tmp);

~PDP()

{ }

};

PDP.cpp

#include"PDP.h"

void operator <<= (ostream& stream, PDP& tmp)

{

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

stream << tmp.plasma << "|" << tmp.brightnessLevel << "|";

stream << endl;

}

void operator >>= (istream& stream, PDP& tmp)

{

string s, s1;

if (getline(stream, s))

{

stringstream ss;

ss << s;

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

getline(ss, s1, '|');

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

}

}

void operator <= (ostream& os, PDP& tmp)

{

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

int n = sizeof(tmp.plasma);

char\* t = new char[n];

t = tmp.plasma.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

n = sizeof(tmp.brightnessLevel);

t = new char[n];

t = tmp.brightnessLevel.c\_str();

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

os.write(reinterpret\_cast<char\*>(&t), sizeof(t));

}

void operator >= (istream& is, PDP& tmp)

{

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

int n;

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

char\* t = new char[n];

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

tmp.setPlasma(t);

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

t = new char[n];

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

tmp.setBrightnessLevel(t);

}

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

{

in >> dynamic\_cast<Monitor&>(ss);

cout << "Введите какая плазма установлена :";

in >> ss.plasma;

cout << "Введите уровень яркости :";

in >> ss.brightnessLevel;

return in;

}

ostream& operator<<(ostream& on, PDP& ss)

{

on << dynamic\_cast<Monitor&>(ss);

on << setiosflags(ios::left)

<< setw(15) << ss.plasma

<< setw(15) << ss.brightnessLevel

<< endl;

return on;

}

PDP& PDP::operator=(const PDP& other)

{

this->yearOfIssue = other.yearOfIssue;

this->brand = other.brand;

this->cost = other.cost;

this->screenResolution = other.screenResolution;

this->numberOfInches = other.numberOfInches;

this->plasma = other.plasma;

this->brightnessLevel = other.brightnessLevel;

return \*this;

}

bool PDP::operator==(PDP& other)

{

if ((this->plasma == other.plasma))

return true;

else

return false;

}

bool PDP::operator==( char\* other)

{

if ((this->plasma == other))

return true;

else

return false;

}

bool PDP::operator > (const PDP& tmp)

{

if (this->plasma > tmp.plasma)

return true;

if (this->brightnessLevel < tmp.brightnessLevel)

return false;

}

Exp\_vvod.h

#pragma once

#include"Exception.h"

inline char\* exp(istream& in)

{

int fe;

char\* str = new char[80];

do

{

rewind(stdin);

try

{

fe = 0;

in.getline(str, 80);

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

{

if (str[i] == ' ')

i++;

if(str[i] <= '2023' || str[i] >= '2000')

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;

}

inline int enter()

{

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;

}

inline int enter(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;

}

Exception.h

#pragma once

#include<iostream>

using namespace std;

class Exp\_vvod

{

private:

int nm;

char ch[80];

public:

Exp\_vvod(const Exp\_vvod& num)

{

nm = num.nm;

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

}

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

{

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

nm = number;

}

void show()

{

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

{

cout << ch[i];

}

cout << endl;

}

~Exp\_vvod()

{ }

};

Iter.h

#pragma once

#include<iostream>

#include"Shablon.h"

using namespace std;

template<typename T>

class Iter

{

private:

Node<T>\* ptr;

public:

Iter()

{

ptr = NULL;

}

Iter(Node<T>\* tmp)

{

ptr = tmp;

}

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

~Iter() { }

Iter& operator++();

Iter& operator--();

T& operator\*();

Node<T>\* operator & ();

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

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

Iter& operator=(const Node<T>& tmp);

};

Iter.cpp

#include"Iter.h"

template<typename T>

Iter<T>& Iter<T>::operator++()

{

if (ptr->next == NULL)

{

ptr = NULL;

return \*this;

}

ptr = ptr->next;

return \*this;

}

template<typename T>

Iter<T>& Iter<T>::operator--()

{

if (ptr->pred == NULL)

{

ptr = NULL;

return \*this;

}

ptr = ptr->pred;

return \*this;

}

template<typename T>

T& Iter<T>::operator\*()

{

return ptr->node;

}

template<typename T>

Node<T>\* Iter<T>::operator&()

{

return ptr;

}

template<typename T>

bool Iter<T>::operator==(const Node<T>\* tmp)

{

if (this->ptr == tmp)

{

return true;

}

return false;

}

template<typename T>

bool Iter<T>::operator != (const Node<T>\* tmp)

{

if (this->ptr != tmp)

{

return true;

}

return false;

}

template<typename T>

Iter<T>& Iter<T>::operator=(const Node <T>& tmp)

{

if (this->ptr != tmp)

{

ptr = tmp;

}

return \*this;

}

String.h

#pragma once

#pragma warning(disable:4996)

#include <iostream>

#include"Exp\_vvod.h"

using namespace std;

class String

{

private:

char\* str;

long int length;

public:

String(const char\* p)

{

if (p == NULL)

{

this->length = 0;

this->str = new char[this->length + 1];

strcpy(this->str, "");

}

else

{

length = strlen(p);

this->str = new char[this->length + 1];

strcpy(this->str, p);

}

}

String(int \_length = 100)

{

this->length = \_length;

this->str = new char[this->length + 1];

strcpy(this->str, "");

}

String(char str[])

{

this->length = strlen(str);

this->str = new char[this->length + 1];

strcpy(this->str, str);

}

String(const String& str)

{

this->length = str.length;

this->str = new char[this->length + 1];

strcpy(this->str, str.str);

}

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

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

String& operator=(const String& other);

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

String operator()(int index1, int length);

String operator+(const String& other);

bool operator==(const String& tmp);

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

bool operator!=(const String& other);

bool operator<(const String& other);

bool operator>(const String& other);

char& operator[](int index);

void operator = (char\* str);

void operator = (const char str[]);

char\* c\_str()

{

return this->str;

}

int slength()

{

return this->length;

}

~String()

{

delete[] this->str;

}

};

String.cpp

#include"String.h"

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

{

rewind(stdin);

extern char\* exp(istream & in);

const char\* ch = exp(in);

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

return in;

}

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

{

return on << ss.str;

}

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

{

length = strlen(other.str);

this->str = new char[length + 1];

strcpy\_s(str, strlen(other.str) + 1, other.str);

return \*this;

}

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

{

delete[] this->str;

this->length = strlen(str);

this->str = new char[this->length + 1];

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

}

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

{

delete[] this->str;

this->length = strlen(str);

this->str = new char[this->length + 1];

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

}

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

{

return (this->operator=(this->operator+(other)));

}

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

{

String newStr;

int thisLength = strlen(this->str);

int otherLength = strlen(other.str);

newStr.length = thisLength + otherLength;

newStr.str = new char[thisLength + otherLength + 1];

int i = 0;

for (; i < thisLength; i++)

{

newStr.str[i] = this->str[i];

}

for (int j = 0; j < otherLength; j++, i++)

{

newStr.str[i] = other.str[j];

}

newStr.str[thisLength + otherLength] = '\0';

return newStr;

}

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

{

return this->str[index];

}

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

{

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

return true;

}

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

{

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

return false;

}

String String::operator()(int index1, int length1)

{

String obrstr;

obrstr.str = new char[length + 1];

for (int i = 0; i < length + 1; i++)

{

obrstr.str[i] = this->str[i];

}

for (int count = index1; count < length1 + 1; ++count)

{

cout << obrstr.str[count];

}

return obrstr;

}

bool String::operator!=(const String& other)

{

return !(this->operator==(other));

}

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

{

if (!strcmp(this->str, tmp.str))

return true;

else

return false;

}

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

{

if (strcmp(this->str, tmp) == 0)

return true;

else

return false;

}

Shablon.h

#pragma once

#include <iostream>

#include<vector>

using namespace std;

using std::bad\_alloc;

template <typename T>

struct Node

{

T node;//само значение

Node\* next;//указатель на след элемент

Node\* pred;//указатель на пред элемент

};

template <typename T>

class Deque

{

private:

Node<T>\* head;

Node<T>\* tail;

int size;

template <typename T>

friend class Iter;

public:

Deque()

{

head = nullptr;

tail = nullptr;

size = 0;

}

~Deque()

{

delet();

}

Node<T>\* Begin() { return head; }

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

void Push(T tmp);

int show\_size();

void show(int index);

void pop(int index);

void pop\_any(T value);

void delet();

vector<T> get();

};

Shablon.cpp

#include"Shablon.h"

template<typename T>

vector<T> Deque<T>::get()

{

vector<T> v;

Iter<T> it;

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

{

v.push\_back(\*it);

}

return v;

}

template <typename T>

void Deque<T>::Push(T tmp)

{

try

{

++size;

Node<T>\* tmp1 = new Node<T>; //выделяем память под элемент

tmp1->node = tmp;

tmp1->next = nullptr;

tmp1->pred = nullptr;

if (!head || !tail) // очередь не создана

{

tail = tmp1; //указатель на хвост очереди

head = tmp1; //указатель на начало очереди

}

else

{

tmp1->next = head;//указатель на хвост очереди

head->pred = tmp1;//хвосту даем новый элемент

head = tmp1;//передвиг указатель хвоста на новый элемент

}

}

catch (bad\_alloc e)

{

cout << "\n Исключение bad\_alloc: невозможно разместить данные в памяти" << e.what() << endl;

return;

}

}

//функция вывода очереди если index == 0 то с хвоста или с головы == 1

template<typename T>

void Deque<T>::show(int index)

{

try

{

Node<T>\* n = tail;

Node<T>\* m = head;

if (index != 0 && index != 1)

throw 1;

if (!tail)

{

cout << "Очередь пуста " << endl;

return;

}

if (index == 0) {

while (n)

{

cout << n->node;

n = n->next;

}

}

if (!head)

{

cout << "Очередь пуста " << endl;

return;

}

if (index == 1) {

while (n)

{

cout << n->node;

n = n->pred;

}

}

return;

}

catch (int i)

{

if (i == 1)

{

cout << "\nОшибка инициализации: "

"выберете либо 0(хвост) , либо 1(голова)";

return;

}

}

}

template<typename T>

void Deque<T>::pop(int index)

{

try

{

if (index != 0 && index != 1)

throw 1;

if (!head || !tail)

{

cout << "Очередь пустая";

return;

}

if (index == 0)//если хвост

{

Node<T>\* n = tail;

tail = tail->next;

tail->pred = NULL;

delete n;

size--;

}

else if (index == 1)//если голова

{

Node<T>\* n = head;

head = head->pred;

head->next = NULL;

delete n;

size--;

}

}

catch (int i)

{

if (i == 1)

{

cout << "\nОшибка инициализации: "

"выберете либо 0(хвост) , либо 1(голова)";

return;

}

}

}

template<typename T>

void Deque<T>::pop\_any(T value)

{

Node<T>\* n = tail;//указат на хвост очереди

if (!head || !tail)

{

cout << "Очередь пустая";

return;

}

if (n->next == NULL && (n->node == value))

{

delete n;

size--;

tail = head = nullptr;

return;

}

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

{

n = n->next;

}//идем до хвоста или до элемент

if ((n->node == value))//найден элемент

{

if (n == tail)//если хвост

{

tail = tail->next;

tail->pred = nullptr;

}

else if (n == head)//если голова

{

head = head->pred;

head->next = nullptr;

}

else//если что-то другое

{

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

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

}

delete n;

size--;

}

}

template<typename T>

int Deque<T>::show\_size()

{

//cout << "Размер: " << size << endl;

return size;

}

template<typename T>

void Deque<T>::delet()

{

while (head != nullptr)

{

Node<T>\* n = head->next;

delete head;

head = n;

}

tail = nullptr;

size = 0;

}

TextFile.h

#pragma once

#include<iostream>

#include<fstream>

#include<sstream>

using namespace std;

template<class T>

class textFile

{

public:

textFile() {};

textFile(string \_title);

~textFile() {};

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

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

int checkCount(string \_filename);

void clear(string \_filename);

};

TextFile.cpp

#include"TextFile.h"

template<class T>

void textFile<T>::toFile(T& obj, string \_filename)

{

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

if (!ofs)

{

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

system("pause");

return;

}

ofs <<= obj;

ofs.close();

}

template<class T>

void textFile<T>::fromFile(T& 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();

}

template<class T>

int textFile<T>::checkCount(string \_filename)

{

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

if (!ifs)

{

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

system("pause");

return -1;

}

int count = 0;

string s;

while (getline(ifs, s))

{

count++;

}

ifs.close();

return count;

}

template<class T>

void textFile<T>::clear(string \_filename)

{

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

ofs.close();

}

Algoritm.h

#pragma once

#include "Shablon.h"

#include"Iter.h"

template <class T>

class Algorithm

{

public:

Algorithm();

~Algorithm();

Deque<T>& search(Node<T>\* beg, T \_obj);

Deque<T>& search2(Node<T>\* beg, Node<T>\* end, T \_obj);

void sort(Deque<T>& \_a);

};

Algoritm.cpp

#include"Algoritm.h"

template <class T>

Algorithm<T>::Algorithm()

{

}

template <class T>

Algorithm<T>::~Algorithm()

{

}

template<class T>

Deque<T>& Algorithm<T>::search(Node<T>\* beg, T \_obj)

{

Node<T>\* rab = beg;

Deque<T> temp;

while (rab != NULL)

{

if (rab->node == \_obj)

{

cout << rab->node;

temp.Push(rab->node);

}

rab = rab->next;

}

return temp;

}

template<class T>

Deque<T>& Algorithm<T>::search2(Node<T>\* beg, Node<T>\* end, T \_obj)

{

Node<T>\* rab = beg;

Deque<T> temp;

Iter<T> it;

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

{

if (\*it == \_obj)

{

cout << \*it;

temp.Push(\*it);

}

}

return temp;

}

template<class T>

void Algorithm<T>::sort(Deque<T>& a)

{

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

Iter<T> it\_beg = a.Begin();

Iter<T> it\_next = a.Begin()->next;

while (it\_beg != NULL)

{

it\_next = it\_beg;

++it\_next;

while (it\_next != NULL)

{

if ((\*it\_beg) > (\*it\_next))

{

tmp->node = \*it\_beg;

\*it\_beg = \*it\_next;

\*it\_next = tmp->node;

}

++it\_next;

}

++it\_beg;

}

}