**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

Лабораторная работа № 5

# Тема: “шаблоны классов в языке С++”

Проверил: Выполнил:

асс. каф. ПИ ст. гр. ПИ-18Б

Щедрин С.В. Моргунов А.Г.

\_\_\_\_.\_\_\_\_.2019г. \_\_\_\_.\_\_\_\_.2019г.

асс. каф. ПИ

Ногтев Е.А.

\_\_\_\_.\_\_\_\_.2019г.

Донецк – 2019

16 Добавить в шаблон класса CList метод, вставляющий перед первым вхождением элемента со значением X элемент со значением Y.

main.cpp

#include "stdafx.h"

#include <iostream>

#include <stdlib.h>

#include "CListDeleteMaxOddItems.h"

#define SIZE 256

using namespace std;

template <typename T>

class CList

{

private:

struct list

{

T data;

list \*next;

list \*prev;

};

list \*head, \*tale;

public:

CList() : head(nullptr), tale(nullptr) {};

~CList()

{

DeleteList();

}

void AddItem(T value)

{

if (head)

{

tale->next = new list;

tale->next->prev = tale;

tale = tale->next;

}

else

{

head = tale = new list;

tale->prev = nullptr;

}

tale->next = nullptr;

tale->data = value;

}

void DisplayList()

{

if (head)

{

for (list \*p = head; p != nullptr; p = p->next)

{

cout << p->data << "-->";

}

cout << "NULL" << endl;

}

else

{

cout << "Список пуст" << endl;

}

}

void DeleteList()

{

list \*p = head;

while (p)

{

head = head->next;

delete p;

p = head;

}

}

bool AddPrevItem(T value,T value2)

{

list \*o;

o = new list;

o->data = value2;

if ((head) && (head->data == value)) {

o->next = head;

head->prev = o;

head = o;

}

else// Если список существует и "голова" не равна значению

{

list \*p = head;

while (p->next)

{

if (p->next->data == value)

{

p->next->prev=o;

o->next = p->next;

p->next = o;

return true;

}

p = p->next;

}

}

cout<<"Error"<<endl;

return false;

}

};

template <>

bool CList<char\*>::AddPrevItem(char \*value,char\* value2)

{

list \*o;

o = new list;

o->data = value2;

if ((head) && (strcmp(head->data, value) == 0))

{

o->next = head;

head->prev = o;

head = o;

}

else

{

list \*p = head;

while (p->next != nullptr)

{

if (strcmp(p->next->data, value) == 0)

{

p->next->prev = o;

o->next = p->next;

p->next = o;

return true;

}

p = p->next;

}

}

return false;

}

template<>

void CList<char\*>::AddItem(char \*value)

{

if (head)

{

tale->next = new list;

tale->next->prev = tale;

tale = tale->next;

}

else

{

head = tale = new list;

tale->prev = nullptr;

}

tale->next = nullptr;

tale->data = new char[SIZE];

strcpy\_s(tale->data, SIZE, value);

}

template <>

void CList<char\*>::DeleteList()

{

list \*p = head;

while (p)

{

head = head->next;

delete p->data;

delete p;

p = head;

}

}

template<>

void CList<CListDeleteMaxOddItems>::DisplayList()

{

if (head)

{

for (list \*p = head; p != nullptr; p = p->next)

{

cout << p->data;

cout << " |\n" << " V\n";

}

cout << " NULL" << endl;

}

else

{

cout << "Список пуст" << endl;

}

}

template <>

void CList<CListDeleteMaxOddItems>::DeleteList()

{

list \*p = head;

while (p)

{

head = head->next;

p->data.deleteList();

delete p;

p = head;

}

head = tale = nullptr;

}

template <typename T>

void corrInput(T \*x)

{

while (!(cin >> \*x))

{

cout << "Повторите ввод" << endl;

cin.clear();

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

}

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

}

template<>

void corrInput(char \*x)

{

cin.getline(x, SIZE);

if (!(cin.good()))

{

cin.clear();

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

}

}

template<>

void corrInput(CListDeleteMaxOddItems \*x)

{

x->deleteList();

int value;

char answ;

do {

cout << "Введите число: ";

corrInput(&value);

x->addItem(value);

do {

cout << "Желаете продолжить ввод? (y/n)" << endl;

answ = getchar();

if ((answ != 'y') && (answ != 'n')) cout << "----Ошибка----" << endl;

} while ((answ != 'y') && (answ != 'n'));

} while (answ == 'y');

}

int main()

{

system("color F0");

setlocale(0, "RUS");

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

CList<int> objInt;

bool bol=false;

int iVal,iVal2;

for (int i = 0; i < 5; i++)

{

cout << "Введите число: ";

corrInput(&iVal);

objInt.AddItem(iVal);

}

cout << endl;

objInt.DisplayList();

do{

cout << "Введите число, перед которым необходимо вставить элемент: ";

corrInput(&iVal);

cout << "Введите число, которое нужно вставить: ";

corrInput(&iVal2);

bol=objInt.AddPrevItem(iVal,iVal2);

}while(bol==false);

objInt.DisplayList();

cout << endl << endl;

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

CList<double> objDouble;

double dVal,dVal2;

for (int i = 0; i < 5; i++)

{

cout << "Введите число: ";

corrInput(&dVal);

objDouble.AddItem(dVal);

}

cout << endl;

objDouble.DisplayList();

do{

cout << "Введите число, перед которым необходимо вставить элемент: ";

corrInput(&dVal);

cout << "Введите число, которое нужно вставить: ";

corrInput(&dVal2);

bol=objDouble.AddPrevItem(dVal,dVal2);

}while(bol==false);

objDouble.DisplayList();

cout << endl << endl;

cout << "----------char\*------------" << endl;

CList<char\*> objChar;

char\* cVal = new char[SIZE];

char\* cVal2 = new char[SIZE];

for (int i = 0; i < 5; i++)

{

cout << "Введите строку: ";

corrInput(cVal);

objChar.AddItem(cVal);

}

cout << endl;

objChar.DisplayList();

do{

cout << "Введите строку, перед которой необходимо вставить элемент: ";

corrInput(cVal);

cout << "Введите строку, которую нужно вставить: ";

corrInput(cVal2);

bol=objChar.AddPrevItem(cVal,cVal2);

}while(bol==false);

objChar.DisplayList();

cout << endl << endl;

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

CList<CListDeleteMaxOddItems> objList;

CListDeleteMaxOddItems lVal, lVal2;

for (int i = 0; i < 5; i++)

{

cout << "Введите список: " << endl;

corrInput(&lVal);

objList.AddItem(lVal);

}

cout << endl;

objList.DisplayList();

do{

cout << "Введите список, перед которым необходимо вставить элемент: " << endl;

corrInput(&lVal);

cout << "Введите список, который нужно вставить: "<<endl;

corrInput(&lVal2);

bol=objList.AddPrevItem(lVal,lVal2);

}while(bol==false);

objList.DisplayList();

system("pause");

return 0;

}

/////////////////// CListDeleteMaxOddItems.h

#ifndef \_\_CListDeleteMaxOddItems\_H\_\_

#define \_\_CListDeleteMaxOddItems\_H\_

#include <iostream>

class CListDeleteMaxOddItems

{

struct list {

int value;

struct list \*next;

};

private:

list \*head\_, \*tale\_;

public:

CListDeleteMaxOddItems(); // конструктор

CListDeleteMaxOddItems(const CListDeleteMaxOddItems &); // конструктор копирования

~CListDeleteMaxOddItems(); // деструктор

int emptyList(); // метод проверки пуст ли список

void addItem(int val); // метод добавления элемента

void displayList(); // метод вывода списка

void deleteList(); // метод освобождения памяти, выделенной под список

void deleteMaxEvenItems(); // метод, удаляющий максимальный чётный элемент

bool operator == (const CListDeleteMaxOddItems &);

CListDeleteMaxOddItems & operator = (const CListDeleteMaxOddItems &);

friend std::ostream & operator << (std::ostream &, CListDeleteMaxOddItems &);

};

#endif \_\_CListDeleteMaxOddItems\_H\_\_

//////////////////// CListDeleteMaxOddItems.cpp

#include <iostream>

#include "CListDeleteMaxOddItems.h" // подключаем интерфейс класса

using namespace std;

CListDeleteMaxOddItems::CListDeleteMaxOddItems()

{

head\_ = NULL;

tale\_ = NULL;

}

CListDeleteMaxOddItems::CListDeleteMaxOddItems(const CListDeleteMaxOddItems & copy)

{

if (copy.head\_)

{

list \*insertP, \*copyP;

head\_ = tale\_ = new list;

head\_->value = copy.head\_->value;

head\_->next = NULL;

insertP = head\_;

copyP = copy.head\_->next;

while (copyP)

{

insertP->next = new list;

insertP = insertP->next;

insertP->value = copyP->value;

copyP = copyP->next;

}

insertP->next = NULL;

tale\_ = insertP;

}

else

{

head\_ = tale\_ = NULL;

}

}

CListDeleteMaxOddItems::~CListDeleteMaxOddItems()

{

deleteList();

}

int CListDeleteMaxOddItems::emptyList()

{

if (head\_ == NULL)

{

//cout << "The list is empty" << endl;

return 1;

}

return 0;

}

void CListDeleteMaxOddItems::addItem(int value)

{

list \*item;

item = new list;

if (emptyList() != 0) // проверка пустой ли список

{

item->next = NULL;

item->value = value;

head\_ = tale\_ = item;

}

else

{

tale\_->next = item;

tale\_ = item;

item->next = NULL;

item->value = value;

}

cout << "Item has been added" << endl;

}

void CListDeleteMaxOddItems::displayList()

{

if (emptyList() != 0)

{

return;

}

list \*p = head\_;

cout << "List:" << endl;

while (p != NULL)

{

cout << p->value << "-->";

p = p->next;

}

cout << "NULL" << endl;

}

void CListDeleteMaxOddItems::deleteList()

{

if (emptyList() != 0)

{

return;

}

list \*p;

while (head\_ != NULL)

{

p = head\_;

head\_ = head\_->next;

delete p;

}

tale\_ = NULL;

//cout << "List deleted" << endl;

}

void CListDeleteMaxOddItems::deleteMaxEvenItems()

{

if (emptyList() != 0)

{

return;

}

int maxVal = 0;

list \*help = NULL;

for (list \*p = head\_; p->next != NULL; p = p->next)

{

if ((p->next->value % 2 == 0) && (p->next->value != 0))

{

if ((maxVal < p->next->value) || (maxVal == 0))

{

maxVal = p->next->value;

help = p;

}

}

}

if ((head\_->value % 2 == 0) && (maxVal < head\_->value))

{

help = head\_;

head\_ = head\_->next;

delete help;

if (head\_ == NULL)

{

tale\_ = NULL;

}

else

if (head\_->next == NULL)

{

tale\_ = head\_;

}

}

else

if (maxVal != 0)

{

list \*del = help->next;

help->next = del->next;

delete del;

if (help->next == NULL)

{

tale\_ = help;

}

}

}

bool CListDeleteMaxOddItems::operator == (const CListDeleteMaxOddItems & other)

{

if ((head\_ == NULL) || (other.head\_ == NULL))

{

if (head\_ == other.head\_)

{

return true;

}

else

return false;

}

else

{

list \*first = head\_, \*second = other.head\_;

while (first && second)

{

if (first->value != second->value)

{

return false;

}

first = first->next;

second = second->next;

}

if (first == second)

{

return true;

}

else return false;

}

}

CListDeleteMaxOddItems & CListDeleteMaxOddItems::operator = (const CListDeleteMaxOddItems & copy)

{

if (this != &copy)

{

if (copy.head\_)

{

list \*insertP, \*copyP;

head\_ = tale\_ = new list;

head\_->value = copy.head\_->value;

head\_->next = NULL;

insertP = head\_;

copyP = copy.head\_->next;

while (copyP)

{

insertP->next = new list;

insertP = insertP->next;

insertP->value = copyP->value;

copyP = copyP->next;

}

insertP->next = NULL;

tale\_ = insertP;

}

else

{

head\_ = tale\_ = NULL;

}

}

return \*this;

}

ostream & operator << (ostream &out, CListDeleteMaxOddItems &obj)

{

if (obj.head\_)

{

CListDeleteMaxOddItems::list \*p = obj.head\_;

while (p)

{

out << p->value << "-->";

p = p->next;

}

out << "NULL" << endl;

}

return out;

}

