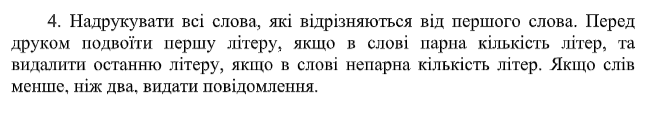
**Лабораторна робота № 3**

**ФБ-95 Прохоренко Ярослав**

Варіант 8

Завдання:



«includes.h»

#pragma once

#include <iostream>

#include <string>

using namespace std;

«Functions.h»

#pragma once

#include "list.h"

#include "array.h"

void checksBefore(List\* list);

void task(List\* list);

void list();

void arr();

«Functions.cpp»

#include "functions.h"

void checksBefore(List\* list)

{

system("cls");

cout << "-- Проверкa списка на работоспособность --\n\n";

cout << "Добавление в начало списка: " << (list->AddInTheBegin("first") ? "OK" : "ОШИБКА") << endl;

cout << "Добавление в конец списка: " << (list->AddInTheEnd("last") ? "OK" : "ОШИБКА") << endl;

cout << "Добавление в средину списка: " << (list->AddInTheMiddle("middle", 2) ? "OK" : "ОШИБКА") << endl;

list->ShowList();

cout << "Удаление со средины списка: " << (list->DeleteFromTheMiddle(2) ? "OK" : "ОШИБКА") << endl;

cout << "Удаление с конца списка: " << (list->DeleteFromTheEnd() ? "OK" : "ОШИБКА") << endl;

cout << "Удаление с начала списка: " << (list->DeleteFromTheBegin() ? "OK" : "ОШИБКА") << endl;

list->ShowList();

system("pause");

}

void task(List\* list)

{

system("cls");

cout << "Введите слова через пробел, для остановки введите точку(через пробел) и нажмите Enter: \n";

string tempString;

while (cin >> tempString && tempString != ".")

{

list->AddInTheEnd(tempString);

}

if (list->getCount() < 3)

{

cout << "Маленький список. Введите больше 2 слов" << endl;

return;

}

cout << "\n-- Исходное состяние: \n";

list->ShowList();

Node\* tempNode = new Node;

tempNode = list->getHead();

string headData = list->getHead()->data;

string tailData = list->getTail()->data;

while (headData == tailData)

{

list->DeleteFromTheEnd();

tailData = list->getTail()->data;

}

string mainWord = list->getHead()->data;

int n = list->getCount();

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

{

Node\* tempNodeHelper = tempNode->next;

string compareWord = tempNode->data;

if (mainWord == compareWord && tempNode != list->getHead())

{

tempNode->next->prev = tempNode->prev;

tempNode->prev->next = tempNode->next;

delete tempNode;

list->decCount();

}

else {

int sizeOfCompareWord = compareWord.size();

if (sizeOfCompareWord % 2 == 0)

{

compareWord.insert(0, 1, compareWord[0]);

tempNode->data = compareWord;

}

else if (sizeOfCompareWord % 2 == 1) {

tempNode->data = compareWord.substr(0, compareWord.size() - 1);

}

}

tempNode = tempNodeHelper;

}

cout << "\n-- Модифицированное состояние: \n";

list->ShowList();

system("pause");

}

void list()

{

List list;

checksBefore(&list);

task(&list);

}

void arr()

{

system("cls");

Array arr;

string tempString;

cout << "Введите слова через пробел, для остановки введите точку(через пробел) и нажмите Enter: \n";

while (cin >> tempString && tempString != ".")

{

arr.addInTheArray(tempString);

}

cout << "\n-- Исходное состяние: \n";

arr.showArray();

string mainWord = arr.getDataByIndex(0);

for (int i = 0; i < arr.getCount(); i++)

{

string compareWord = arr.getDataByIndex(i);

if (mainWord == compareWord && i != 0)

{

arr.deleteFromTheArray(i);

--i;

}

else {

int sizeOfCompareWord = compareWord.size();

if (sizeOfCompareWord % 2 == 0)

{

compareWord.insert(0, 1, compareWord[0]);

arr.changeDataByIndex(compareWord, i);

}

else if (sizeOfCompareWord % 2 == 1) {

arr.changeDataByIndex(compareWord.substr(0, compareWord.size() - 1), i);

}

}

}

cout << "\n-- Модифицированное состяние: \n";

arr.showArray();

system("pause");

}

«List.h»

#pragma once

#include "includes.h"

struct Node

{

string data;

Node\* next;

Node\* prev;

};

class List

{

private:

int count;

Node\* head;

Node\* tail;

public:

List();

int getCount();

void incCount();

void decCount();

bool AddInTheBegin(string data\_);

bool AddInTheEnd(string data\_);

bool AddInTheMiddle(string data\_, int position);

bool DeleteFromTheBegin();

bool DeleteFromTheEnd();

bool DeleteFromTheMiddle(int position);

void ShowList();

Node\* getHead();

Node\* getTail();

void setHead(Node\* head\_);

void setTail(Node\* tail\_);

};

«List.cpp»

#include "list.h"

void List::setHead(Node\* head\_)

{

head = head\_;

}

void List::setTail(Node\* tail\_)

{

tail = tail\_;

}

Node\* List::getHead()

{

return head;

}

Node\* List::getTail()

{

return tail;

}

bool List::AddInTheBegin(string data\_)

{

try

{

Node\* tempNode = new Node;

tempNode->data = data\_;

tempNode->prev = NULL;

if (head != NULL)

{

tempNode->next = head;

head->prev = tempNode;

head = tempNode;

}

else

{

tempNode->next = NULL;

head = tail = tempNode;

}

incCount();

return 1;

}

catch (...)

{

return 0;

}

}

bool List::AddInTheEnd(string data\_)

{

try

{

Node\* tempNode = new Node;

tempNode->data = data\_;

tempNode->next = NULL;

if (head != NULL)

{

tempNode->prev = tail;

tail->next = tempNode;

tail = tempNode;

}

else {

tempNode->prev = NULL;

head = tail = tempNode;

}

incCount();

}

catch (...)

{

return 0;

}

}

bool List::AddInTheMiddle(string data\_, int position)

{

Node\* tempNode = new Node;

Node\* tempNodeHelper = new Node;

if (position <= 1 || position > count) return 0;

if (count < 2 || head == NULL) return 0;

try

{

tempNodeHelper = head;

for (int i = 1; i < position; i++)

tempNodeHelper = tempNodeHelper->next;

tempNode->prev = tempNodeHelper->prev;

tempNodeHelper->prev->next = tempNode;

tempNodeHelper->prev = tempNode;

tempNode->next = tempNodeHelper;

tempNode->data = data\_;

incCount();

return 1;

}

catch (...)

{

return 0;

}

}

bool List::DeleteFromTheBegin()

{

try

{

if (count == 0) return 0;

if (count < 2)

{

if (head != NULL)

{

delete head;

head = NULL;

decCount();

}

}

else

{

Node\* tempNode = new Node;

tempNode = head->next;

head->next->prev = NULL;

delete head;

head = tempNode;

decCount();

}

return 1;

}

catch (...)

{

return 0;

}

}

bool List::DeleteFromTheEnd()

{

try

{

if (count == 0) return 0;

if (count < 2)

{

if (head != NULL)

{

delete head;

head = NULL;

decCount();

}

}

else

{

Node\* tempNode = new Node;

tempNode = tail->prev;

tail->prev->next = NULL;

delete tail;

tail = tempNode;

decCount();

}

return 1;

}

catch (...)

{

return 0;

}

}

bool List::DeleteFromTheMiddle(int position)

{

try

{

if (position <= 1 || position >= count || count < 3) return 0;

Node\* tempNode = new Node;

tempNode = head;

for (int i = 1; i < position; i++)

tempNode = tempNode->next;

tempNode->next->prev = tempNode->prev;

tempNode->prev->next = tempNode->next;

delete tempNode;

decCount();

return 1;

}

catch (...)

{

return 0;

}

}

void List::ShowList()

{

if (count == 0)

{

cout << "--------------\n";

cout << "Пусто\n";

cout << "--------------\n";

}

else

{

Node\* tempNode = new Node;

tempNode = head;

cout << "--------------\n";

while (tempNode != NULL)

{

cout << tempNode->data << " ";

tempNode = tempNode->next;

}

cout << "\n--------------\n";

}

}

List::List()

{

count = 0;

head = NULL;

tail = NULL;

}

int List::getCount()

{

return count;

}

void List::incCount()

{

count++;

}

void List::decCount()

{

count--;

}

«Array.h»

#pragma once

#include "includes.h"

class Array

{

private:

int count;

string\* arr;

public:

Array();

bool addInTheArray(string data);

bool deleteFromTheArray(int index);

bool changeDataByIndex(string data, int index);

string getDataByIndex(int index);

int getCount();

void showArray();

};

«Array.cpp»

#include "array.h"

int Array::getCount()

{

return count;

}

string Array::getDataByIndex(int index)

{

return arr[index];

}

bool Array::changeDataByIndex(string data, int index)

{

arr[index] = data;

return 1;

}

bool Array::deleteFromTheArray(int index)

{

try

{

string\* tempArray = new string[count];

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

{

tempArray[i] = arr[i];

}

delete[] arr;

arr = new string[count - 1];

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

if (i != index)

{

arr[j] = tempArray[i];

}

else

--j;

--count;

delete[] tempArray;

return 1;

}

catch (...)

{

return 0;

}

}

bool Array::addInTheArray(string data)

{

try

{

string\* tempArray = new string[count + 1];

memcpy(tempArray, arr, count \* sizeof(string));

tempArray[count] = data;

arr = tempArray;

count++;

return 1;

}

catch (...)

{

return 0;

}

}

void Array::showArray()

{

if (count == 0)

{

cout << "--------------\n";

cout << "Пусто\n";

cout << "--------------\n";

}

else

{

cout << "--------------\n";

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

{

cout << arr[i] << " ";

}

cout << "\n--------------\n";

}

}

Array::Array()

{

count = 0;

arr = new string[count];

}

«main.cpp»

#include "functions.h"

int main()

{

setlocale(0, "ru");

cout << "1. Динамический массив\n2. Список\n\n> ";

char ans;

cin >> ans;

switch (ans)

{

case '1':

arr();

break;

case '2':

list();

break;

default:

cout << "Ошибка ввода!";

break;

}

return 0;

}

**Контакти**

Почта: [yarpro-ipt23@lll.kpi.ua](mailto:yarpro-ipt23@lll.kpi.ua) /

[prohorenko.yaroslav01@gmail.com](mailto:prohorenko.yaroslav01@gmail.com)

Телеграм: @AugFitzR