Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №7

Тема: «Работа с двухсвязным списком»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

**I. Ответы на контрольные вопросы**

1. **Организация структуры двусвязного списка**

struct list {

int item;

list\* next;

list\* prev;

};

1. **В чем принципиальное отличие двусвязного от односвязного списка?**

В том, что в двухсвязном списке можно ходить как вперед через next, также и назад через previous.

1. **В чем отличие операции вставки в двусвязный список от вставки в односвязный список?**

В том, что в двухсвязном списке нужно указывать previous, а в односвязном только next.

1. **В чем отличие операции удаления из двусвязного списка от удаления из односвязного списка?**

В том, что в двухсвязном списке нужно указывать previous, а в односвязном только next.

1. **Как осуществляется операция перестановки элементов в двусвязном списке?**

Сначала мы присваиваем к нынешнему элементу адрес следующего элемента, затем к следующему элементу адрес предыдущего, попутно присваивая элементы, которые лежал до нынешнего, и после следующего элемента.

1. **Какие ошибки могут быть допущены при сцеплении двусвязных списков?**

1) Можно забыть про previous или неправильно его назначить;

2) Забыть обнулить второй список;

3) Не инициализировать один или оба списка;

1. **В чем разница при просмотре элементов слева направо и справа налево?**

Разница в том, что при просмотре слева направо цикл начинается с головы списка и заканчивается хвостом, а при справе налево, наоборот.

1. **Достоинства двусвязного списка**

1) лёгкость добавления и удаления элементов;

2) размер ограничен только объемом памяти компьютера и разрядностью указателей;

1. **Недостатки двусвязного списка**

1) сложность определения адреса элемента по его индексу в списке на поля;

2) указатели расходуется дополнительная память;

3) работа со списком медленнее, чем с массивами, так как к любому элементу списка можно обратиться, только пройдя все предшествующие ему элементы;

4) элементы списка могут быть расположены в памяти разреженно, что окажет негативный эффект на кэширование процессора;

5) над связными списками гораздо труднее производить параллельные векторные операции, такие как вычисление суммы;

**II. Выполнить упражнение**

**1. Постановка задачи**

Разработать алгоритм, блок схемы и программный код для работы с двусвязным списком:

* инициализация списка;
* добавление элемента в список;
* добавление элемента в список по значению после указанного элемента;
* добавление элементов в список по индексу после указанного элемента;
* добавление элемента в список перед указанным элементом по значению;
* добавление элемента в список перед указанным элементом по индексу;
* поиск элемента списка перед указанным элементом по индексу;
* поиск элемента списка перед указанным элементом по значению;
* поиск элемента списка после указанного элемента по индексу;
* поиск элемента списка после указанного элемента по значению;
* удаление элементов из списка по индексу после указанного элемента;
* удаление элементов из списка по значению после указанного элемента;
* удаление элемента по индексу из списка перед указанным элементом;
* удаление элемента по значению из списка перед указанным элементом;
* просмотр списка слева направо и справа налево;
* очистка списка.
* Разработать алгоритм, блок схему и программный код для организации перестановки 2 – х соседних элементов в двусвязном списке.
* Разработать алгоритм, блок схему и программный код для организации сортировки элементов двусвязного списка.

**2. Описание входных и выходных данных**

**Исходные данные:**

**void inition(list\*& he, list\*& ta);**- функция инициализации;

**void output list\* he, list\* ta);** - функция вывода;

**void output(list\* he, list\* ta);** - функция вывода наоборот;

**void clear(list\*& he, list\*& ta);** - функция очищения очереди;

**void search(list\*& he, list\*& ta);** - функция поиска;

**void add(list\*& he, list\*& ta);** - функция добавления;

**void deleLI(list\*& he, list\*& ta);** - функция удаления;

**void swapLI(list\*& he, list\*& ta); -** функция переустановки;

**void sort(list\*& he, list\*& ta); -** функция сортировки;

int main()



void inition(list\*& he, list\*& ta);



void add(list\*& he, list\*& ta)



void output(list\* he, list\* ta)



void output\_back(list\* he, list\* ta)



void deleLI(list\*& he, list\*& ta)



void search(list\* he, list\* ta)



void swapLI(list\*& he, list\*& ta)



void sort(list\*& he, list\*& ta)



void clear(list\*& he, list\*& ta)



void pushByIndex(list\*& he, list\*& ta)



void pushByIndex\_Before(list\*& he, list\*& ta)



void pushByIndex\_After(list\*& he, list\*& ta)



void pushByValue\_Before(list\*& he, list\*& ta)



void pushByValue\_After(list\*& he, list\*& ta)



void push(list\*& he, list\*& ta, int k)



void popByIndex(list\*& he, list\*& ta)



void popByIndex\_Before(list\*& he, list\*& ta)



void popByIndex\_After(list\*& he, list\*& ta)



void popByValue\_Before(list\*& he, list\*& ta)



void popByValue\_After(list\*& he, list\*& ta)



void popByValue(list\*& he, list\*& ta)



void pop(list\*& he, list\*& ta, int k)



void searchByIndex\_Value(list\* he, list\* ta)



void searchByIndex\_Before(list\* he, list\* ta)



void searchByIndex\_After(list\* he, list\* ta)



void searchByIndex(list\* he, list\* ta, int x)



void searchByValue\_Before(list\* he, list\* ta)



void searchByValue\_After(list\* he, list\* ta)  


int searchByValue(list\* he, list\* ta, int value)



void swapByIndex\_Before(list\*& he, list\*& ta)



void swapByIndex\_After(list\*& he, list\*& ta)



void swapByIndex(list\*& he, list\*& ta)



void swapByValue\_Before(list\*& he, list\*& ta)



void swapByValue\_After(list\*& he, list\*& ta)



void swapByValue(list\*& he, list\*& ta)



swapMain(list\*& he, list\*& ta, int x)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct list {

int item;

list\* next;

list\* prev;

};

int \_count = 0, p = 0;

void inition(list\*& he, list\*& ta);

void add(list\*& he, list\*& ta);

void output(list\* he, list\* ta);

void output\_back(list\* he, list\* ta);

void deleLI(list\*& he, list\*& ta);

void search(list\* he, list\* ta);

void swapLI(list\*& he, list\*& ta);

void sort(list\*& he, list\*& ta);

void clear(list\*& he, list\*& ta);

void pushByIndex(list\*& he, list\*& ta);

void pushByIndex\_Before(list\*& he, list\*& ta);

void pushByIndex\_After(list\*& he, list\*& ta);

void pushByValue\_Before(list\*& he, list\*& ta);

void pushByValue\_After(list\*& he, list\*& ta);

void push(list\*& he, list\*& ta, int k);

void popByIndex(list\*& he, list\*& ta);

void popByIndex\_Before(list\*& he, list\*& ta);

void popByIndex\_After(list\*& he, list\*& ta);

void popByValue\_Before(list\*& he, list\*& ta);

void popByValue\_After(list\*& he, list\*& ta);

void popByValue(list\*& he, list\*& ta);

void pop(list\*& he, list\*& ta, int k);

void searchByIndex\_Value(list\* he, list\* ta);

void searchByIndex\_Before(list\* he, list\* ta);

void searchByIndex\_After(list\* he, list\* ta);

void searchByIndex(list\* he, list\* ta, int x);

void searchByValue\_Before(list\* he, list\* ta);

void searchByValue\_After(list\* he, list\* ta);

int searchByValue(list\* he, list\* ta, int value);

void swapByValue\_Before(list\*& he, list\*& ta);

void swapByValue\_After(list\*& he, list\*& ta);

void swapByValue(list\*& he, list\*& ta);

void swapByIndex\_Before(list\*& he, list\*& ta);

void swapByIndex\_After(list\*& he, list\*& ta);

void swapByIndex(list\*& he, list\*& ta);

void swapMain(list\*& he, list\*& ta, int x);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

list\* \_head = new list;

list\* \_tail = new list;

bool f = true;

int choice;

\_head->item = 0;

\_head->next = NULL;

\_head->prev = NULL;

\_tail->item = 0;

\_tail->next = NULL;

\_tail->prev = NULL;

inition(\_head, \_tail);

while (f) {

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

cout << "| Главное Меню: |" << endl;

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

cout << "| 1. Добавление элемента |" << endl;

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

cout << "| 2. Вывод элементов |" << endl;

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

cout << "| 3. Вывод наоборот элементов |" << endl;

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

cout << "| 4. Удаление элемента |" << endl;

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

cout << "| 5. Поиск элемента |" << endl;

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

cout << "| 6. Поменять местами |" << endl;

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

cout << "| 7. Отсортировать список |" << endl;

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

cout << "| 8. Очистить список |" << endl;

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

cout << "| 9. Выход |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: add(\_head, \_tail); break;

case 2: output(\_head, \_tail); break;

case 3: output\_back(\_head, \_tail); break;

case 4: deleLI(\_head, \_tail); break;

case 5: search(\_head, \_tail); break;

case 6: swapLI(\_head, \_tail); break;

case 7: sort(\_head, \_tail); break;

case 8: clear(\_head, \_tail); break;

case 9: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

return 0;

}

void inition(list\*& he, list\*& ta) {

int x;

cout << "Инициализируйте первый элемент для списка:";

cin >> x;

system("cls");

list\* \_pv = new list;

\_pv->item = x;

he->next = \_pv;

\_pv->next = ta;

\_pv->prev = he;

ta->prev = \_pv;

cout << "Первый элемент со значением = " << x << " инициализирован\n" << endl;

\_count++;

}

void add(list\*& he, list\*& ta) {

if (\_count <= 0) {

inition(he, ta);

} else {

bool f = true;

int x;

while (f) {

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

cout << "| Меню добавления: |" << endl;

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

cout << "| 1. Добавление по индексу |" << endl;

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

cout << "| 2. Добавление до указанного индекса |" << endl;

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

cout << "| 3. Добавление после указанного индекса|" << endl;

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

cout << "| 4. Добавление до указанного элемента |" << endl;

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

cout << "|5. Добавление после указанного элемента|" << endl;

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

cout << "| 6. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; pushByIndex(he, ta); break;

case 2: f = false; pushByIndex\_Before(he, ta); break;

case 3: f = false; pushByIndex\_After(he, ta); break;

case 4: f = false; pushByValue\_Before(he, ta); break;

case 5: f = false; pushByValue\_After(he, ta); break;

case 6: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

}

void output(list\* he, list\* ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего выводить!!!" << endl;

} else {

int i = 1;

list\* \_pv = he->next;

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

cout << "|\tЭлементы\t: Индекс |\n";

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

while (\_pv != ta) {

cout << "|\t" << \_pv->item << "\t\t: " << i << "\t |" << endl;

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

\_pv = \_pv->next;

i++;

}

}

}

void output\_back(list\* he, list\* ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего выводить!!!" << endl;

} else {

int i = \_count;

list\* \_pv = ta->prev;

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

cout << "|\tЭлементы\t: Индекс |\n";

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

while (\_pv != he) {

cout << "|\t" << \_pv->item << "\t\t: " << i << "\t |" << endl;

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

\_pv = \_pv->prev;

i--;

}

}

}

void deleLI(list\*& he, list\*& ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего удалять!!!" << endl;

} else {

bool f = true;

int x;

while (f) {

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

cout << "| Меню удаления: |" << endl;

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

cout << "| 1. Удаление по индексу |" << endl;

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

cout << "| 2. Удаление до указанного индекса |" << endl;

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

cout << "| 3. Удаление после указанного индекса |" << endl;

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

cout << "| 4. Удаление до указанного элемента |" << endl;

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

cout << "| 5. Удаление после указанного элемента |" << endl;

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

cout << "| 6. Удаление по значению |" << endl;

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

cout << "| 7. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; popByIndex(he, ta); break;

case 2: f = false; popByIndex\_Before(he, ta); break;

case 3: f = false; popByIndex\_After(he, ta); break;

case 4: f = false; popByValue\_Before(he, ta); break;

case 5: f = false; popByValue\_After(he, ta); break;

case 6: f = false; popByValue(he, ta); break;

case 7: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

}

void search(list\* he, list\* ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего искать!!!" << endl;

} else {

bool f = true;

int x;

while (f) {

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

cout << "| Меню поиска: |" << endl;

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

cout << "| 1. Поиск по индексу |" << endl;

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

cout << "| 2. Поиск до указанного индекса |" << endl;

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

cout << "| 3. Поиск после указанного индекса |" << endl;

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

cout << "| 4. Поиск до указанного элемента |" << endl;

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

cout << "| 5. Поиск после указанного элемента |" << endl;

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

cout << "| 6. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; searchByIndex\_Value(he, ta); break;

case 2: f = false; searchByIndex\_Before(he, ta); break;

case 3: f = false; searchByIndex\_After(he, ta); break;

case 4: f = false; searchByValue\_Before(he, ta); break;

case 5: f = false; searchByValue\_After(he, ta); break;

case 6: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

}

void swapLI(list\*& he, list\*& ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего менять местами!!!" << endl;

} else {

bool f = true;

int x;

while (f) {

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

cout << "| Меню swap'а: |" << endl;

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

cout << "| 1. Поменять местами по индексу |" << endl;

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

cout << "| 2. Поменять местами по значению |" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; swapByIndex(he, ta); break;

case 2: f = false; swapByValue(he, ta); break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

}

void sort(list\*& he, list\*& ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего сортировать!!!" << endl;

} else {

list\* \_pv = new list;

list\* \_PV = new list;

for (int i = 1; i <= \_count; i++) {

\_pv = he->next;

while (\_pv->next != ta) {

if (\_pv->item > \_pv->next->item) {

\_PV = \_pv->next;

\_pv->prev->next = \_PV;

\_PV->next->prev = \_pv;

\_pv->next = \_PV->next;

\_PV->prev = \_pv->prev;

\_pv->prev = \_PV;

\_PV->next = \_pv;

} else {

\_pv = \_pv->next;

}

}

}

}

}

void clear(list\*& he, list\*& ta) {

if (\_count <= 0) {

cout << "Лист пуст, нечего очищать!!!" << endl;

} else {

list\* \_pv = new list;

\_pv = he->next;

while (\_pv != ta) {

list\* \_PV = new list;

\_PV = \_pv;

\_pv = \_pv->next;

delete \_PV;

}

\_count = 0;

he->next = 0;

cout << "Лист успешно очищен!!!" << endl;

}

}

/\*PUSH по Индексу

ДО // ПОСЛЕ

\*/

void pushByIndex(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс куда добавить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

push(he, ta, x);

cout << " в индекс = " << x << endl;

}

}

}

void pushByIndex\_Before(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс куда добавить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == 1){

f = false;

push(he, ta, x);

cout << " в индекс = " << x << endl;

} else {

f = false;

push(he, ta, x - 1);

cout << " в индекс = " << x << endl;

}

}

}

void pushByIndex\_After(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс куда добавить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

push(he, ta, x + 1);

cout << " в индекс = " << x << endl;

}

}

}

/\*PUSH по Значению

ДО // ПОСЛЕ

\*/

void pushByValue\_Before(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите значение элемента до которого добавить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

push(he, ta, check);

cout << " до элемента " << x << endl;

}

}

}

void pushByValue\_After(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите значение элемента до которого добавить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

push(he, ta, check + 1);

cout << " после элемента " << x << endl;

}

}

}

//PUSH

void push(list\*& he, list\*& ta, int k) {

list\* \_pv = new list;

int x;

cout << "Введите новый элемент для добавления:";

cin >> x;

system("cls");

\_pv->item = x;

if (k == 1) {

\_pv->next = he->next;

he->next->prev = \_pv;

\_pv->prev = he;

he->next = \_pv;

} else if (k == (\_count + 1)) {

list\* \_PV = new list;

\_PV = he->next;

while (\_PV->next != ta) {

\_PV = \_PV->next;

}

\_pv->next = \_PV->next;

\_PV->next->prev = \_pv;

\_PV->next = \_pv;

\_pv->prev = \_PV;

} else {

list\* \_PV = new list;

int j = 1;

\_PV = he->next;

while (\_PV != ta) {

if (j == (k - 1)) {

\_pv->next = \_PV->next;

\_PV->next->prev = \_pv;

\_pv->prev = \_PV;

\_PV->next = \_pv;

break;

}

j++;

\_PV = \_PV->next;

}

}

\_count++;

cout << "Элемент " << x << " успешно добавлен";

}

/\*POP по Индексу

ДО // ПОСЛЕ

\*/

void popByIndex(list\*& he, list\*& ta){

bool f = true;

int x;

while (f) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

pop(he, ta, x);

}

}

}

void popByIndex\_Before(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == 1){

cout << "До 1 индекса ничего нет" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

pop(he, ta, x - 1);

}

}

}

void popByIndex\_After(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == \_count) {

cout << "После " << \_count << " ничего нет " << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

pop(he, ta, x + 1);

}

}

}

/\*POP по Значению

ДО // ПОСЛЕ

\*/

void popByValue\_Before(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите значение элемента до которого удалить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else if (check == 1) {

cout << "Нельзя удалить до " << x << " \nПотому что там ничего нет!!!" << endl;

} else {

f = false;

pop(he, ta, (check - 1));

}

}

}

void popByValue\_After(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите значение элемента до которого удалить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else if (check == \_count) {

cout << "Нельзя удалить после " << x << " \nПотому что там ничего нет!!!" << endl;

} else {

f = false;

pop(he, ta, (check + 1));

}

}

}

void popByValue(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите значение элемента до которого удалить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

pop(he, ta, (check));

}

}

}

//POP

void pop(list\*& he, list\*& ta, int k) {

list\* \_pv = new list;

int x;

if (k == 1) {

\_pv = he->next;

he->next = \_pv->next;

\_pv->next->prev = he;

x = \_pv->item;

cout << "Элемент " << \_pv->item << " успешно удален" << endl;

delete \_pv;

} else if (k == \_count) {

list\* \_PV = new list;

list\* \_\_PV = new list;

\_PV = he;

while (\_PV->next != ta) {

\_\_PV = \_PV;

\_PV = \_PV->next;

}

x = \_PV->item;

cout << "Элемент " << \_PV->item << " успешно удален" << endl;

delete \_PV;

\_\_PV->next = ta;

ta->prev = \_\_PV;

} else {

int j = 1;

list\* \_PV = new list;

list\* \_\_PV = new list;

\_PV = he->next;

while (\_PV != ta) {

if (j == k) {

\_\_PV->next = \_PV->next;

\_PV->next->prev = \_\_PV;

x = \_PV->item;

cout << "Элемент " << \_PV->item << " успешно удален" << endl;

delete \_PV;

break;

}

j++;

\_\_PV = \_PV;

\_PV = \_PV->next;

}

}

\_count--;

if (\_count == 0) {

he->next = 0;

}

}

/\*SEARCH по Значению

ДО // ПОСЛЕ

\*/

void searchByIndex\_Value(list\* he, list\* ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

searchByIndex(he, ta, x);

}

}

}

void searchByIndex\_Before(list\* he, list\* ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == 1){

cout << "До 1 индекса ничего нет" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

searchByIndex(he, ta, (x - 1));

}

}

}

void searchByIndex\_After(list\* he, list\* ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x >= \_count) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == \_count){

cout << "После " << \_count << " ничего нет " << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

searchByIndex(he, ta, (x + 1));

}

}

}

//SEARCH по Индексу

void searchByIndex(list\* he, list\* ta, int x) {

int i = 1;

list\* \_pv = new list;

\_pv = he->next;

while (\_pv != ta) {

if (i == x) {

cout << "Искомый элемент:" << endl;

cout << x << "| " << \_pv->item << endl;

}

i++;

\_pv = \_pv->next;

}

}

/\*SEARCH по Значению

ДО // ПОСЛЕ

\*/

void searchByValue\_Before(list\* he, list\* ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else if (check == 1) {

cout << "Ничего нет до значения " << x << " \nПовторите заново ввод!!!" << endl;

} else {

f = false;

searchByIndex(he, ta, (check - 1));

}

}

}

void searchByValue\_After(list\* he, list\* ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

} else if (check == \_count) {

cout << "Ничего нет после значения " << x << " \nПовторите заново ввод!!!" << endl;

} else {

f = false;

searchByIndex(he, ta, (check + 1));

}

}

}

//SEARCH по Значению

int searchByValue(list\* he, list\* ta, int value) {

int i = 1, j, kol = 0, k = 0, x;

bool f = true;

list\* \_pv = new list;

\_pv = he->next;

while (\_pv != ta) {

if (\_pv->item == value) {

kol++;

j = i;

}

i++;

\_pv = \_pv->next;

}

if (kol == 0) {

return 0;

} else if (kol == 1) {

return j;

} else {

while (f) {

for (i = 0; i < kol; i++) {

cout << i + 1 << "\t|" << value << endl;

}

cout << "Мы нашли " << kol << " элемента с введённым значение, выберите 1 из них:";

cin >> x;

system("cls");

if (x <= 0 || x > kol) {

cout << "Ошибка при вводе!!!" << endl;

} else {

f = false;

i = 1;

\_pv = he->next;

while (\_pv != ta) {

if (\_pv->item == value) {

j = i;

k++;

}

if (k == x) {

break;

}

i++;

\_pv = \_pv->next;

}

return j;

}

}

}

return 0;

}

//SWAP по индексу

void swapByIndex\_Before(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс:";

cin >> x;

system("cls");

if (x <= 0 || x >= (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == 1) {

cout << "До первого индекса ничего нет!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

swapMain(he, ta, x - 1);

}

}

}

void swapByIndex\_After(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

cout << "Введите индекс:";

cin >> x;

system("cls");

if (x <= 0 || x >= (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (x == \_count) {

cout << "После индекса " << \_count << " ничего нет!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

swapMain(he, ta, x);

f = false;

}

}

}

void swapByIndex(list\*& he, list\*& ta) {

bool f = true;

int x;

while (f) {

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

cout << "| Меню swap'а по индексу: |" << endl;

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

cout << "| 1. Поменять местами до задаваемого индекса |" << endl;

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

cout << "|2. Поменять местами после задаваемого индекса|" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; swapByIndex\_Before(he, ta); break;

case 2: f = false; swapByIndex\_After(he, ta); break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

//SWAP по Значению

void swapByValue\_Before(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Такого элемента нету!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (check == 1) {

cout << "Элемент " << x << " находится на первом индексе и не может быть перестановлен!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

swapMain(he, ta, check - 1);

}

}

}

void swapByValue\_After(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x);

if (check == 0) {

cout << "Такого элемента нету!!!" << endl << "Повторите заново ввод!!!" << endl;

} else if (check == \_count) {

cout << "Элемент " << x << " находится на последнем индексе и не может быть перестановлен!!!" << endl << "Повторите заново ввод!!!" << endl;

} else {

f = false;

swapMain(he, ta, check);

}

}

}

void swapByValue(list\*& he, list\*& ta) {

bool f = true;

int x, check;

while (f) {

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

cout << "| Меню swap'а по значению: |" << endl;

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

cout << "| 1. Поменять местами до задаваемого значения |" << endl;

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

cout << "|2. Поменять местами после задаваемого значения|" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; swapByValue\_Before(he, ta); break;

case 2: f = false; swapByValue\_After(he, ta); break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

//SWAP

void swapMain(list\*& he, list\*& ta, int x) {

list\* \_pv = new list;

list\* \_PV = new list;

int i = 1;

\_pv = he->next;

while (\_pv != ta) {

if (i == x) {

break;

}

i++;

\_pv = \_pv->next;

}

\_PV = \_pv->next;

\_pv->prev->next = \_PV;

\_PV->next->prev = \_pv;

\_pv->next = \_PV->next;

\_PV->prev = \_pv->prev;

\_pv->prev = \_PV;

\_PV->next = \_pv;

cout << "Были swap'нуты элементы " << \_pv->item << " и " << \_PV->item << endl;

}

**1.Постановка задачи**

* Разработать алгоритм, блок схему и программный код для организации слияния двусвязных списков.

**2. Описание входных и выходных данных**

**void Concon(list\* he1, list\* ta1, list\* he2, list\* ta2, int& count1, int& count2); -** Функция для меню конкатенации;

**void con(list\* he1, list\* ta1, list\* he2, list\* ta2); -** функция конкантенации;

void Concon(list\* he1, list\* ta1, list\* he2, list\* ta2, int& count1, int& count2)



void con(list\* he1, list\* ta1, list\* he2, list\* ta2)



**3.Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

using namespace std;

struct list {

int item;

list\* next;

list\* prev;

};

void inition(list\* he, list\* ta);

void inition\_count0(list\*& he, list\*& ta, int& \_count);

void listnum1(list\* he, list\* ta, int& \_count);

void listnum2(list\* he, list\* ta, int& \_count);

void Concon(list\* he1, list\* ta1, list\* he2, list\* ta2, int& count1, int& count2);

void con(list\* he1, list\* ta1, list\* he2, list\* ta2);

void add(list\*& he, list\*& ta, int& \_count);

void output(list\* he, list\* ta, int& \_count);

void output\_back(list\* he, list\* ta, int& \_count);

void deleLI(list\*& he, list\*& ta, int& \_count);

void search(list\*& he, list\*& ta, int& \_count);

void swapLI(list\*& he, list\*& ta, int& \_count);

void sort(list\*& he, list\*& ta, int& \_count);

void clear(list\*& he, list\*& ta, int& \_count);

void push(list\*& he, list\*& ta, int k, int& \_count);

void pop(list\*& he, list\*& ta, int k, int& \_count);

void searchByIndex(list\* he, list\* ta, int x, int& \_count);

int searchByValue(list\* he, list\* ta, int value, int& \_count);

void swapMain(list\*& he, list\*& ta, int x, int& \_count);

void swapByIndex(list\*& he, list\*& ta, int& \_count);

void swapByValue(list\*& he, list\*& ta, int& \_count);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int n = 2;

list\* \_head = new list[n];

list\* \_tail = new list[n];

bool f = true;

int choice, \_count1 = 1, \_count2 = 1;

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

\_head[i].item = 0;

\_head[i].next = NULL;

\_head[i].prev = NULL;

\_tail[i].item = 0;

\_tail[i].next = NULL;

\_tail[i].prev = NULL;

inition(&\_head[i], &\_tail[i]);

}

while (f) {

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

cout << "| Главное Меню: |" << endl;

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

cout << "| 1. Список № 1 |" << endl;

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

cout << "| 2. Список № 2 |" << endl;

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

cout << "| 3. Конкантинация |" << endl;

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

cout << "| 4. Выход |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: listnum1(&\_head[0], &\_tail[0], \_count1); break;

case 2: listnum2(&\_head[1], &\_tail[1], \_count2); break;

case 3: Concon(&\_head[0], &\_tail[0], &\_head[1], &\_tail[1], \_count1, \_count2); break;

case 4: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

return 0;

}

void listnum1(list\* he, list\* ta, int& \_count) {

bool f = true;

int choice;

while (f) {

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

cout << "| Главное листа № 1: |" << endl;

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

cout << "| 1. Добавление элемента |" << endl;

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

cout << "| 2. Вывод элементов |" << endl;

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

cout << "| 3. Вывод наоборот элементов |" << endl;

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

cout << "| 4. Удаление элемента |" << endl;

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

cout << "| 5. Поиск элемента |" << endl;

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

cout << "| 6. Поменять местами |" << endl;

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

cout << "| 7. Отсортировать список |" << endl;

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

cout << "| 8. Очистить список |" << endl;

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

cout << "| 9. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: add(he, ta, \_count); break;

case 2: output(he, ta, \_count); break;

case 3: output\_back(he, ta, \_count); break;

case 4: deleLI(he, ta, \_count); break;

case 5: search(he, ta, \_count); break;

case 6: swapLI(he, ta, \_count); break;

case 7: sort(he, ta, \_count); break;

case 8: clear(he, ta, \_count); break;

case 9: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

void listnum2(list\* he, list\* ta, int& \_count) {

bool f = true;

int choice;

while (f) {

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

cout << "| Главное листа № 2: |" << endl;

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

cout << "| 1. Добавление элемента |" << endl;

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

cout << "| 2. Вывод элементов |" << endl;

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

cout << "| 3. Вывод наоборот элементов |" << endl;

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

cout << "| 4. Удаление элемента |" << endl;

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

cout << "| 5. Поиск элемента |" << endl;

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

cout << "| 6. Поменять местами |" << endl;

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

cout << "| 7. Отсортировать список |" << endl;

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

cout << "| 8. Очистить список |" << endl;

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

cout << "| 9. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: add(he, ta, \_count); break;

case 2: output(he, ta, \_count); break;

case 3: output\_back(he, ta, \_count); break;

case 4: deleLI(he, ta, \_count); break;

case 5: search(he, ta, \_count); break;

case 6: swapLI(he, ta, \_count); break;

case 7: sort(he, ta, \_count); break;

case 8: clear(he, ta, \_count); break;

case 9: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

void Concon(list\* he1, list\* ta1, list\* he2, list\* ta2, int& count1, int& count2) {

bool f = true;

int choice;

if (count1 == 0 && count2 == 0) {

cout << "Оба списка не инициализованы!!!" << endl;

}

else if (count1 == 0) {

cout << "Список № 1 не инициализован!!!" << endl;

}

else if (count2 == 0) {

cout << "Список № 2 не инициализован!!!" << endl;

}

else {

while (f) {

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

cout << "| Меню конкетенации из двух списков: |" << endl;

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

cout << "| 1. 1 + 2 |" << endl;

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

cout << "| 2. 2 + 1 |" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> choice;

system("cls");

switch (choice) {

case 1: con(he1, ta1, he2, ta2); cout << "№ 2 список присоединен к № 1, и № 2 список пуст!!!" << endl; count1 += count2; count2 = 0; f = false; break;

case 2: con(he2, ta2, he1, ta1); cout << "№ 1 список присоединен к № 2, и № 1 список пуст!!!" << endl; count2 += count1; count1 = 0; f = false; break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

}

void con(list\* he1, list\* ta1, list\* he2, list\* ta2) {

ta1->prev->next = he2->next;

he2->next->prev = ta1->prev;

he2->next = 0;

ta2->prev->next = ta1;

ta1->prev = ta2->prev;

ta2->prev = 0;

}

void add(list\*& he, list\*& ta, int& \_count) {

if (\_count == 0) {

inition\_count0(he, ta, \_count);

}

else {

bool f = true, f1 = true;

int x, check;

while (f) {

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

cout << "| Меню добавления: |" << endl;

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

cout << "| 1. Добавление по индексу |" << endl;

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

cout << "| 2. Добавление до указанного индекса |" << endl;

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

cout << "| 3. Добавление после указанного индекса|" << endl;

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

cout << "| 4. Добавление до указанного элемента |" << endl;

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

cout << "|5. Добавление после указанного элемента|" << endl;

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

cout << "| 6. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

if (x == 1) {

while (f1) {

cout << "Введите индекс куда добавить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

push(he, ta, x, \_count);

cout << " в индекс = " << x << endl;

}

}

}

else if (x == 2) {

while (f1) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (x == 1) {

f1 = false;

f = false;

push(he, ta, x, \_count);

cout << " до индекса " << x << endl;

}

else {

f1 = false;

f = false;

push(he, ta, x - 1, \_count);

cout << " до индекса " << x << endl;

}

}

}

else if (x == 3) {

while (f1) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x >= (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

push(he, ta, x + 1, \_count);

cout << " после индекса " << x << endl;

}

}

}

else if (x == 4) {

if (\_count <= 0) {

cout << "Лист пуст, нельзя добавить до или после!!!" << endl;

}

else {

while (f1) {

cout << "Введите значение элемента до которого добавить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

push(he, ta, check, \_count);

cout << " до элемента " << x << endl;

}

}

}

}

else if (x == 5) {

if (\_count <= 0) {

cout << "Лист пуст, нельзя добавить после или после!!!" << endl;

}

else {

while (f1) {

cout << "Введите значение элемента после которого добавить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

push(he, ta, (check + 1), \_count);

cout << " после элемента " << x << endl;

}

}

}

}

else if (x == 6) {

f = false;

}

else {

cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl;

}

}

}

}

void output(list\* he, list\* ta, int& \_count) {

if (\_count <= 0) {

cout << "Лист пуст, нечего выводить!!!" << endl;

}

else {

int i = 1;

list\* \_pv = he->next;

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

cout << "|\tЭлементы\t: Индекс |\n";

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

while (\_pv != ta) {

cout << "|\t" << \_pv->item << "\t\t: " << i << "\t |" << endl;

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

\_pv = \_pv->next;

i++;

}

}

}

void output\_back(list\* he, list\* ta, int& \_count) {

if (\_count <= 0) {

cout << "Лист пуст, нечего выводить!!!" << endl;

}

else {

int i = \_count;

list\* \_pv = ta->prev;

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

cout << "|\tЭлементы\t: Индекс |\n";

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

while (\_pv != he) {

cout << "|\t" << \_pv->item << "\t\t: " << i << "\t |" << endl;

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

\_pv = \_pv->prev;

i--;

}

}

}

void deleLI(list\*& he, list\*& ta, int& \_count) {

if (\_count <= 0) {

cout << "Лист пуст, нечего удалять!!!" << endl;

}

else {

bool f = true, f1 = true;

int x, check;

while (f) {

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

cout << "| Меню удаления: |" << endl;

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

cout << "| 1. Удаление по индексу |" << endl;

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

cout << "| 2. Удаление до указанного индекса |" << endl;

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

cout << "| 3. Удаление после указанного индекса |" << endl;

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

cout << "| 4. Удаление до указанного элемента |" << endl;

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

cout << "| 5. Удаление после указанного элемента |" << endl;

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

cout << "| 6. Удаление по значению |" << endl;

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

cout << "| 7. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

if (x == 1) {

while (f1) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

pop(he, ta, x, \_count);

}

}

}

else if (x == 2) {

while (f1) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 1 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

pop(he, ta, x - 1, \_count);

}

}

}

else if (x == 3) {

while (f1) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 || x >= \_count) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

pop(he, ta, x + 1, \_count);

}

}

}

else if (x == 4) {

while (f1) {

cout << "Введите значение элемента до которого удалить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

if (check == 1) {

cout << "Нельзя удалить до " << x << " \nПотому что там ничего нет!!!" << endl;

}

else {

pop(he, ta, (check - 1), \_count);

}

}

}

}

else if (x == 5) {

while (f1) {

cout << "Введите значение элемента после которого удалить:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

if (check == \_count) {

cout << "Нельзя удалить после " << x << " \nПотому что там ничего нет!!!" << endl;

}

else {

pop(he, ta, (check + 1), \_count);

}

}

}

}

else if (x == 6) {

while (f1) {

cout << "Введите значение элемента который будем удалять:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

pop(he, ta, check, \_count);

}

}

}

else if (x == 7) {

f = false;

}

else {

cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl;

}

}

}

}

void search(list\*& he, list\*& ta, int& \_count) {

bool f = true, f1 = true;

int x, check;

while (f) {

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

cout << "| Меню поиска: |" << endl;

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

cout << "| 1. Поиск по индексу |" << endl;

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

cout << "| 2. Поиск до указанного индекса |" << endl;

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

cout << "| 3. Поиск после указанного индекса |" << endl;

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

cout << "| 4. Поиск до указанного элемента |" << endl;

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

cout << "| 5. Поиск после указанного элемента |" << endl;

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

cout << "| 6. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

if (x == 1) {

while (f1) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

searchByIndex(he, ta, x, \_count);

}

}

}

else if (x == 2) {

while (f1) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (x == 1) {

cout << "До 1 индекса ничего нет" << endl << "Повторите заново ввод!!!" << endl;

} else {

f1 = false;

f = false;

searchByIndex(he, ta, (x - 1), \_count);

}

}

}

else if (x == 3) {

while (f1) {

cout << "Введите индекс где искать:";

cin >> x;

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (x == \_count) {

cout << "После индекса " << \_count << " ничего нет" << endl << "Повторите заново ввод!!!" << endl;

} else {

f1 = false;

f = false;

searchByIndex(he, ta, (x + 1), \_count);

}

}

}

else if (x == 4) {

while (f1) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

if (check == 1) {

cout << "Нельзя искать до " << x << " индекса\nПотому что там ничего нет!!!" << endl;

}

else {

searchByIndex(he, ta, (check - 1), \_count);

}

}

}

}

else if (x == 5) {

while (f1) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Введенного элемента не существует." << endl << "Повторите заново ввод!!!" << endl;

}

else {

f1 = false;

f = false;

if (check == 1) {

cout << "Нельзя искать после " << x << " индекса\nПотому что там ничего нет!!!" << endl;

}

else {

searchByIndex(he, ta, (check + 1), \_count);

}

}

}

}

else if (x == 6) {

f = false;

}

else {

cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl;

}

}

}

void swapLI(list\*& he, list\*& ta, int& \_count) {

bool f = true;

int x;

while (f) {

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

cout << "| Меню swap'а: |" << endl;

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

cout << "| 1. Поменять местами по индексу |" << endl;

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

cout << "| 2. Поменять местами по значению |" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1: f = false; swapByIndex(he, ta, \_count); break;

case 2: f = false; swapByValue(he, ta, \_count); break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

void sort(list\*& he, list\*& ta, int& \_count) {

if (\_count == 0) {

cout << "Лист пуст, нечего сортировать!!!" << endl;

}

else {

list\* \_pv = new list;

list\* \_PV = new list;

for (int i = 1; i <= \_count; i++) {

\_pv = he->next;

while (\_pv->next != ta) {

if (\_pv->item > \_pv->next->item) {

\_PV = \_pv->next;

\_pv->prev->next = \_PV;

\_PV->next->prev = \_pv;

\_pv->next = \_PV->next;

\_PV->prev = \_pv->prev;

\_pv->prev = \_PV;

\_PV->next = \_pv;

}

else {

\_pv = \_pv->next;

}

}

}

}

}

void clear(list\*& he, list\*& ta, int& \_count) {

if (\_count <= 0) {

cout << "Лист пуст, нечего очищать!!!" << endl;

}

else {

list\* \_pv = new list;

\_pv = he->next;

while (\_pv != ta) {

list\* \_PV = new list;

\_PV = \_pv;

\_pv = \_pv->next;

delete \_PV;

}

\_count = 0;

he->next = 0;

cout << "Лист успешно очищен!!!" << endl;

}

}

void inition(list\* he, list\* ta) {

int x;

cout << "Инициализируйте первый элемент для списка:";

cin >> x;

system("cls");

list\* \_pv = new list;

\_pv->item = x;

he->next = \_pv;

\_pv->next = ta;

\_pv->prev = he;

ta->prev = \_pv;

cout << "Первый элемент со значением = " << x << " инициализирован\n" << endl;

}

void inition\_count0(list\*& he, list\*& ta, int& \_count) {

int x;

cout << "Инициализируйте первый элемент для списка:";

cin >> x;

system("cls");

list\* \_pv = new list;

\_pv->item = x;

he->next = \_pv;

\_pv->next = ta;

\_pv->prev = he;

ta->prev = \_pv;

\_count++;

cout << "Первый элемент со значением = " << x << " инициализирован\n" << endl;

}

void searchByIndex(list\* he, list\* ta, int x, int& \_count) {

if (\_count <= 0) {

cout << "Лист пуст, нечего искать!!!" << endl;

}

else {

int i = 1;

bool f = true;

list\* \_pv = new list;

while (f) {

system("cls");

if (x <= 0 || x > (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f = false;

\_pv = he->next;

while (\_pv != ta) {

if (i == x) {

cout << "Искомый элемент:" << endl;

cout << x << "| " << \_pv->item << endl;

}

i++;

\_pv = \_pv->next;

}

}

}

}

}

int searchByValue(list\* he, list\* ta, int value, int& \_count) {

int i = 1, j, kol = 0, k = 0, x;

bool f = true;

list\* \_pv = new list;

\_pv = he->next;

while (\_pv != ta) {

if (\_pv->item == value) {

kol++;

j = i;

}

i++;

\_pv = \_pv->next;

}

if (kol == 0) {

return 0;

}

else if (kol == 1) {

return j;

}

else {

while (f) {

for (i = 0; i < kol; i++) {

cout << i + 1 << "\t|" << value << endl;

}

cout << "Мы нашли " << kol << " элемента с введённым значение, выберите 1 из них:";

cin >> x;

system("cls");

if (x <= 0 || x > kol) {

cout << "Ошибка при вводе!!!" << endl;

}

else {

f = false;

i = 1;

\_pv = he->next;

while (\_pv != ta) {

if (\_pv->item == value) {

j = i;

k++;

}

if (k == x) {

break;

}

i++;

\_pv = \_pv->next;

}

return j;

}

}

}

return 0;

}

void push(list\*& he, list\*& ta, int k, int& \_count) {

list\* \_pv = new list;

int x;

cout << "Введите новый элемент для добавления:";

cin >> x;

system("cls");

\_pv->item = x;

if (k == 1) {

\_pv->next = he->next;

he->next->prev = \_pv;

\_pv->prev = he;

he->next = \_pv;

}

else if (k == (\_count + 1)) {

list\* \_PV = new list;

\_PV = he->next;

while (\_PV->next != ta) {

\_PV = \_PV->next;

}

\_pv->next = \_PV->next;

\_PV->next->prev = \_pv;

\_PV->next = \_pv;

\_pv->prev = \_PV;

}

else {

list\* \_PV = new list;

int j = 1;

\_PV = he->next;

while (\_PV != ta) {

if (j == (k - 1)) {

\_pv->next = \_PV->next;

\_PV->next->prev = \_pv;

\_pv->prev = \_PV;

\_PV->next = \_pv;

break;

}

j++;

\_PV = \_PV->next;

}

}

\_count++;

cout << "Элемент " << x << " успешно добавлен";

}

void pop(list\*& he, list\*& ta, int k, int& \_count) {

list\* \_pv = new list;

int x;

if (k == 1) {

\_pv = he->next;

he->next = \_pv->next;

\_pv->next->prev = he;

x = \_pv->item;

cout << "Элемент " << \_pv->item << " успешно удален" << endl;

delete \_pv;

}

else if (k == \_count) {

list\* \_PV = new list;

list\* \_\_PV = new list;

\_PV = he;

while (\_PV->next != ta) {

\_\_PV = \_PV;

\_PV = \_PV->next;

}

x = \_PV->item;

cout << "Элемент " << \_PV->item << " успешно удален" << endl;

delete \_PV;

\_\_PV->next = ta;

ta->prev = \_\_PV;

}

else {

int j = 1;

list\* \_PV = new list;

list\* \_\_PV = new list;

\_PV = he->next;

while (\_PV != ta) {

if (j == k) {

\_\_PV->next = \_PV->next;

\_PV->next->prev = \_\_PV;

x = \_PV->item;

cout << "Элемент " << \_PV->item << " успешно удален" << endl;

delete \_PV;

break;

}

j++;

\_\_PV = \_PV;

\_PV = \_PV->next;

}

}

\_count--;

if (\_count == 0) {

he->next = 0;

}

}

void swapMain(list\*& he, list\*& ta, int x, int& \_count) {

if (\_count == 0) {

cout << "Лист пуст, нечего менять местами!!!" << endl;

}

else {

list\* \_pv = new list;

list\* \_PV = new list;

int i = 1;

\_pv = he->next;

while (\_pv != ta) {

if (i == x) {

break;

}

i++;

\_pv = \_pv->next;

}

\_PV = \_pv->next;

\_pv->prev->next = \_PV;

\_PV->next->prev = \_pv;

\_pv->next = \_PV->next;

\_PV->prev = \_pv->prev;

\_pv->prev = \_PV;

\_PV->next = \_pv;

cout << "Были swap'нуты элементы " << \_pv->item << " и " << \_PV->item << endl;

}

}

void swapByIndex(list\*& he, list\*& ta, int& \_count) {

bool f = true, f2 = true;

int x;

while (f) {

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

cout << "| Меню swap'а по индексу: |" << endl;

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

cout << "| 1. Поменять местами до задаваемого индекса |" << endl;

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

cout << "|2. Поменять местами после задаваемого индекса|" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1:

f = false;

while (f2) {

cout << "Введите индекс:";

cin >> x;

system("cls");

if (x <= 0 || x >= (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (x == 1) {

cout << "До первого индекса ничего нет!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

swapMain(he, ta, x - 1, \_count);

f2 = false;

}

}; break;

case 2:

f = false;

while (f2) {

cout << "Введите индекс:";

cin >> x;

system("cls");

if (x <= 0 || x >= (\_count + 1)) {

cout << "Вы вышли за границы списка!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (x == \_count) {

cout << "После индекса " << \_count << " ничего нет!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

swapMain(he, ta, x, \_count);

f2 = false;

}

}; break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

void swapByValue(list\*& he, list\*& ta, int& \_count) {

bool f = true, f2 = true;

int x, check;

while (f) {

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

cout << "| Меню swap'а по значению: |" << endl;

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

cout << "| 1. Поменять местами до задаваемого значения |" << endl;

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

cout << "|2. Поменять местами после задаваемого значения|" << endl;

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

cout << "| 3. Назад |" << endl;

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

cout << endl;

cout << "Введите номер операции:";

cin >> x;

system("cls");

switch (x) {

case 1:

f = false;

while (f2) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Такого элемента нету!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (check == 1) {

cout << "Элемент " << x << " находится на первом индексе и не может быть перестановлен!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f2 = false;

swapMain(he, ta, check - 1, \_count);

}

}; break;

case 2:

f = false;

while (f2) {

cout << "Введите искоемый элемент:";

cin >> x;

system("cls");

check = searchByValue(he, ta, x, \_count);

if (check == 0) {

cout << "Такого элемента нету!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else if (check == \_count) {

cout << "Элемент " << x << " находится на последнем индексе и не может быть перестановлен!!!" << endl << "Повторите заново ввод!!!" << endl;

}

else {

f2 = false;

swapMain(he, ta, check, \_count);

}

}; break;

case 3: f = false; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}