Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Алгоритми та структури даних

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ №5

“Списки”

Варіант № 1

Оцінка «\_\_\_\_\_\_\_\_\_»

Дата «\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_

Виконав: студент 2 курсу

гр. ТВ-61

Артамонов Олексій Юрійович

Перевірила: Васильєва О.Б.

Київ – 2017

**Завдання:** У списку переставити у зворотному порядку всі елементи між першим та останнім елементами.

**Лістинг програми:**

#include "stdafx.h"

#include <iostream>

#include <conio.h>

using namespace std;

struct Node {

int val;

Node \*next, \*prev;

};

class List {

private:

Node \*head, \*tail;

int n, k;

public:

List() : head(NULL), tail(NULL) {

cout << "Enter count of elements: ";

cin >> n;

cout << "\n";

}

void Show();

void Add(int val);

void Enter(List & lst);

void Sort();

~List() {

while (head) {

tail = head->next;

if (tail)

tail->prev = NULL;

delete head;

head = tail;

}

}

};

void List::Enter(List & lst) {

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

cout << "Enter value: ";

cin >> k;

lst.Add(k);

}

cout << "\n";

}

void List::Add(int val) {

Node \*tmp = new Node;

tmp->next = NULL;

tmp->val = val;

if (head != NULL) {

tmp->prev = tail;

tail->next = tmp;

tail = tmp;

}

else {

tmp->prev = NULL;

head = tail = tmp;

}

}

void List::Show() {

Node \*tmp = tail;

tmp = head;

while (tmp != NULL) {

cout << tmp->val << "\t";

tmp = tmp->next;

}

cout << "\n";

}

void List::Sort() {

Node \*p = head->next;

Node \*q = tail->prev;

int tmp;

while (1) {

tmp = p->val;

p->val = q->val;

q->val = tmp;

p = p->next;

if (p == q)

break;

q = q->prev;

if (q == p)

break;

}

}

int main() {

List lst;

lst.Enter(lst);

lst.Show();

lst.Sort();

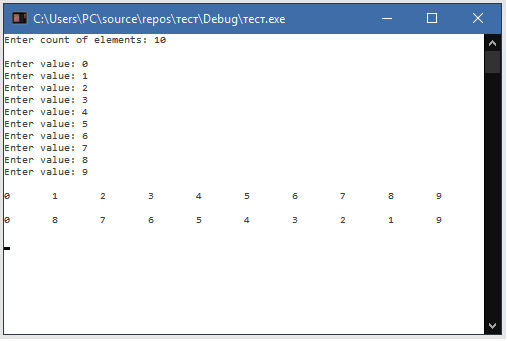
lst.Show();

\_getch();

return 0;

}

**Результати:**



**Контрольні питання:**

1. Якщо кожен елемент списку має в собі вказівник на наступний елемент (або на наступний і попередній у випадку двозв’язного списку).

2. Він міститься не в динамічний пам’яті.

3. У наступний елемент, на який він вказує – NULL.

4. Бо в однозв'язному списку, ми маємо вказівник тільки на наступний елемент, тому, щоб добратись до будь-якого, потрібне позиціонування на перший елемент.

5. Бо в двозв’язному ми маємо вказівник як на попередній, так і на наступний елемент, отже ми можемо йти по списку в обидва боки, тому ми можемо позиціонувати на будь-який елемент. 6. При вставці в початок списку, ми змінюємо значення самого вказівника початку списку (а не значення, на яке він посилається). При вставці в інші позиції, ми просто виділяємо місце в динамічній пам’яті, та пов’язуємо новий елемент з попереднім та наступним.

7. В двонаправленому ми враховуємо те, що маємо ще й вказівник на попередній елемент списку.

8. Для того, щоб уникнути помилок, наприклад, якщо ми будемо видаляти елемент пустого списку – компілятор видасть помилку.

9. Це потрібно, щоб очистити виділену пам’ять. У складних програмах, де пам’ять виділяється неодноразово, може виникнути переповнення пам’яті і програма просто «впаде». А при видалені пам’яті, ми одну й ту ж ділянку можемо використовувати неодноразово.

10. Недоліки: ми маємо доступ тільки до останнього елемента, щоб дістати нижній елемент – потрібно перед тим дістати всі, які вище нього. Переваги: для вирішення деяких задач, стек дуже корисний.

11. Недоліки: ми маємо доступ тільки до першого елемента, щоб дістати верхній елемент – потрібно перед тим дістати всі, які нижче нього. Переваги: для вирішення деяких задач, стек дуже корисний.

12. Стек використовується для обчислення постфіксних виразів, претворення інфіксного виразу в постфікс ний, задачі пошуку, зворотній польський запис. Черга в програмуванні використовується, як і в реальному житті, коли потрібно зробити якісь дії в порядку їх надходження, виконавши їх послідовно. Прикладом може служити організація подій в Windows. Коли користувач робить якась агресивна дія на додаток, то в додатку не викликається відповідна процедура (адже в цей момент додаток може здійснювати інші дії), а йому надсилається повідомлення, що містить інформацію про скоєний дії, це повідомлення ставиться в чергу, і тільки коли будуть оброблені повідомлення, що прийшли раніше, додаток виконає необхідну дію.

13. Останній елемент.

14. Перший і останній елемент.

15. Структура, що реалізує стек чи чергу не може містити в якості одного з полів змінну власного структурного типу.

16. Щоб не діставати елементи з пустого стеку/черги.

17. Ні, не можна. Для кожного елементу виділялась кількість пам’яті, яка необхідна для роботи тільки одного елемента. Якщо записати поверх цієї ділянки пам’яті інше значення, то виникне витічка пам’яті та (або) втратиться значення, яке розміщувалось по даній адресі раніше.

18. Аналогічно до питання №9.