 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Лабораторна робота №10**

з дисципліни «Алгоритмізація та програмування

на тему: «Двозв’язні списки»

Варіант №

**Виконав:**

**Перевірив:**

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2022

**Завдання**

1. Ознайомитися з теоретичними відомостями створення, обробки та видалення однозв’язних списків.

2. Відповідно до варіанту визначити структуру вузла двозв’язного списку, що в яксості даних зберігає ціле число, та розробити функції, що необхідні для роботи із заданим відповідно до варіанту контейнером: функцію додавання елемента (пам’ять під черговий елемент контейнера свід виділяти динамічно), функцію вилучення елемента, функцію друку всього змісту контейнера на екран, функцію одночасного видалення всього контейнера.

3. Розробити програмний застосунок, в якому реалізовано виконання індивідуального завдання, після чого всі створені контейнери видаляються (глобальні змінні не використовувати, функція main має бути призначена тільки для виклику функцій користувача):

**Завдання за варіантом:** Розробити функцію для розміщення у зворотному порядку елементів, які розташовані між найменшим і найбільшим елементами лінійного двоозв’язного списку(змінити треба коротший ланцюжок з двох можливих).

**Програмна реалізація:**

#include <iostream>

#include <iomanip>

using namespace std;

struct list

{

int data;

list\* next;

list\* prev;

};

void addElem2(list\*\* first, list\*\* cur)

{

list\* tmp = new list;

cin >> tmp->data;

if (\*cur == nullptr)

{

\*cur = \*first = tmp;

}

else

{

(\*cur)->next = tmp;

tmp->prev = \*cur;

tmp->next = \*first;

\*cur = tmp;

}

}

void outBack(list\* cur, list\* first)

{

list\* tmp = cur;

do

{

cout << tmp->data << " ";

tmp = tmp->prev;

} while (tmp != cur);

}

void outFwrd(list\* cur, list\* first)

{

list\* tmp = first;

do

{

cout << tmp->data << " ";

tmp = tmp->next;

} while (tmp != first);

}

void getMax(list\* first, list\*\* max)

{

int m = first->data;

\*max = first;

list\* tmp = first;

do

{

if (tmp->data > m)

{

\*max = tmp;

m = tmp->data;

}

tmp = tmp->next;

} while (tmp != first);

}

void getMin(list\* first, list\*\* min)

{

int m = first->data;

\*min = first;

list\* tmp = first;

do

{

tmp = tmp->next;

if (tmp->data < m)

{

\*min = tmp;

m = tmp->data;

}

} while (tmp != first);

}

void between2M(list\* first)

{

list\* max = nullptr;

list\* min = nullptr;

getMax(first, &max);

getMin(first, &min);

list\* tmp;

tmp = max;

int a = -1;

while (tmp != min)

{

a++;

tmp = tmp->next;

}

tmp = min;

int b = -1;

while (tmp != max)

{

b++;

tmp = tmp->next;

}

cout << endl;

if ((max->prev == min) || (max->next == min))

return;

if (a < b)

{

int\* arr = new int[a];

tmp = min->prev;

int i = 0;

while (tmp != max)

{

arr[i] = tmp->data;

i++;

tmp = tmp->prev;

}

i--;

tmp = min->prev;

while (tmp != max)

{

tmp->data = arr[i];

i--;

tmp = tmp->prev;

}

delete[] arr;

}

else

{

int\* arr = new int[b];

tmp = max->prev;

int i = 0;

while (tmp != min)

{

arr[i] = tmp->data;

i++;

tmp = tmp->prev;

}

tmp = max->prev;

i--;

while (tmp != min)

{

tmp->data = arr[i];

i--;

tmp = tmp->prev;

}

delete[] arr;

}

}

void deleteAll(list\*\* first, list\* cur)

{

list\* tmp;

cur->next = nullptr;

while ((\*first))

{

tmp = (\*first);

(\*first) = (\*first)->next;

delete tmp;

}

}

int main()

{

list\* first = nullptr;

list\* cur = nullptr;

int n;

cout << "How much elements you want ot enter?: ";

cin >> n;

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

{

cout << "Element Num." << i + 1 << "=";

addElem2(&first, &cur);

}

first->prev = cur;

cout << "All elements: ";

outFwrd(cur, first);

cout << endl;

between2M(first);

cout << "After changes: ";

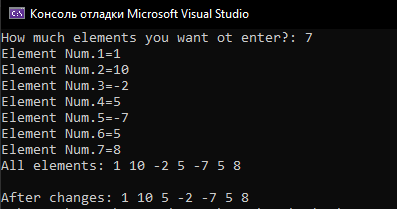
outFwrd(cur, first);

deleteAll(&first, cur);

return 0;

}

**Результат виконання програми:**

****

**Контрольні питання:**

1. **У чому полягають переваги і недоліки використання двозв'язних списків у порівнянні з однозв’язними?**

Переваги: можливість проходу в обидві сторони.

Недоліки: більша затратність пам’яті

**2. Чому при роботі з двозв'язним списком його обробку можна виконувати, починаючи з будь-якого елемента списку?**

Так як список двохзв’язний, то можна зробити так, щоб умовний перший елемент мав вказівник на умовний останній, отже список стане кільцевим і з будь-якого елемента можна дістатись до будь-якого іншого. Тому не має різниці з якого елемента починати перебір.

**3. Визначте набір допустимих операцій для роботи з лінійними двозв’язними списками.**

* Додавання нових елементів
* Видалення елментів
* Очищення повністю списку
* Перегляд списку
* Пошук у списку
* Сортування списку

**4. Визначте операції, що за швидкістю виконання кращі у двозв'язних списках ніж у однозв’язних.**

* Пошук у списку
* Сортування списку

**5. У чому полягають принципові відмінності для роботи з однозв'язними та двозв'язними списками.**

Принципова різниця полягає в тому, що двозв’язний список зберігає 2 вказівники – на попередній та наступний елементи**.** Це надає додаткові можливості, яких немає в однозв’язному.

**6. Чим відрізняється операція пошуку у однозв'язному і двозв'язному списках?**

У двозв’язному списку пошук може відбуватись у двох напрямках, коли в однозв’язному – в один. Це надає двозв’язному списку переваги у швидкості.

**7. Які методи сортування використовують для роботи із двозв’яними списками?**

* Сортування бульбашкою
* Швидке сортування
* Сортування злиттям
* Бінарне сортування

**8. Чи залежить час, витрачений на додавання елемента до двозв'язного списку, від кількості елементів у ньому?**

Ні, не залежить, оскільки елемент додається у умовний початок списку.

**9. У чому полягає різниця між лінійними та кільцевими зв'язними списками?**

У лінійного перший елемент замість посилання на останній має нульовий покажчик і останній елемент, замість покажчика на перший має нульовий покажчик.

**10. За рахунок чого робота зі зв'язними списками уповільнює виконання програми?**

Робота уповільнюється, тому що для того, щоб дістатись до потрібного елемента, треба пройти від умовно першого елемента до нього. Тобто перебір елементів, щоб дістатись до потрібного. Також відіграє роль перевірки списку, які необхідні, щоб унеможливити звернення до неіснуючого елементу.