 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Лабораторна робота №9**

з дисципліни «Алгоритмізація та програмування

на тему: «Однозв’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 однозв’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 однозв’язного списку.

**Програмна реалізація:**

#include <iostream>

#include <iomanip>

using namespace std;

struct list

{

int data;

list\* next;

};

void between2Ms(list\* first)

{

list\* tmp = first;

list\* max = tmp;

list\* min = tmp;

while (tmp != nullptr)

{

if (tmp->data < min->data)

min = tmp;

else if (tmp->data > max->data)

max = tmp;

tmp = tmp->next;

}

tmp = first;

while (1)

{

if (tmp == min)

{

tmp = tmp->next;

while (tmp != max)

{

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

tmp = tmp->next;

}

return;

}

else if (tmp == max)

{

tmp = tmp->next;

while (tmp != min)

{

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

tmp = tmp->next;

}

return;

}

tmp = tmp->next;

}

}

void addElem(list\*\* first)

{

list\* tmp = new list;

cin >> tmp->data;

tmp->next = (\*first);

(\*first) = tmp;

}

void printElems(list\* cur)

{

list\* tmp = cur;

while (tmp != nullptr)

{

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

tmp = tmp->next;

}

}

void deleteAll(list\*\* first)

{

list\* tmp;

while ((\*first))

{

tmp = (\*first);

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

delete tmp;

}

}

int main()

{

list\* first = 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 << "=";

addElem(&first);

}

cout << "All elements: ";

printElems(first);

cout << "\nElements between max and min: ";

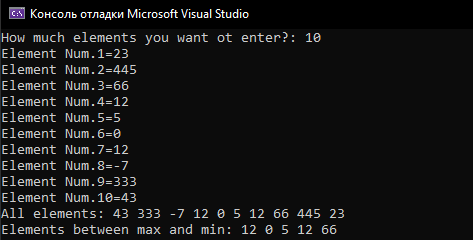
between2Ms(first);

deleteAll(&first);

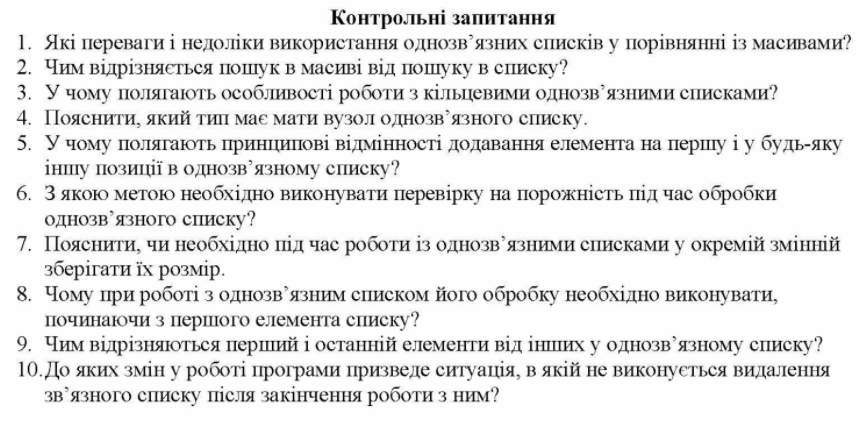
return 0;

}

**Результат виконання програми:**

****

**Контрольні питання:**



1. Переваги списків над масивами:

Можливість додавати нові вузли у кінець. Масив має статичний розмір, і, якщо, вільного місця там немає, доведеться створювати масив більшого розміру, копіювати у нього елементи «старого» масиву і тільки після цього додавати новий елемент;

Можливість видаляти вузол і звільнювати пам'ять, яку він займав. У масиві можна лише зсунути елементи і розглядати його, як масив меншого розміру. Пам'ять при цьому не звільняється;

Можливість вставляти вузол у середину списку. При умові, що масив не заповнений до кінця, можна «розсунути» елементи і вставити між ними необхідний. Якщо ж масив повний — доведеться створювати новий масив більшого розміру, копіювати елементи і вставляти новий.

Недоліки списків перед масивами:

Відсутність індексованого доступу до елементів списку;

Зайвий час на прохід по списку для пошуку/видалення/додавання елементу у кінець;

Використання більшого об'єму пам'яті за рахунок покажчиків на наступний вузол.

1. У масиві пошук відбувається за допомогою перебору індексів масиву, а пошук в списку відбувається за переходом від вузла до вузла за допомогою покажчиків на наступний вузол.
2. Кільцеві (кругові, циклічні) списки є різновидом зв'язкових списків. Вони можуть бути однозв'язними або двозв'язковими. Їхньою відмінністю є те, що умовний останній елемент зберігає посилання на перший елемент, а перший (у разі двозв'язкового списку) - на останній, тому список виходить замкнутим або кільцевим.
3. Списки можуть бути типизованими або нетипізованими. Якщо список типизований, тип його елементів заданий, і всі його елементи повинні мати типи, сумісні із заданим типом елементів списку. Найчастіше списки типізовані.
4. При додаванні елементу на першу позицію, ми не маємо змінювати показники інших елементів, на відміну від додавання на будь-яку іншу позицію. Наприклад, для додавання на 2 позицію, ми маємо змінити покажчик на наступний елемент у першого елемента.
5. Якщо порожньо, то це означає, що це останній елемент, адже він не вказує на наступний елемент.
6. Ні, це не є обов’язковим. Для перебору списку можна використовувати цикл, де критерієм зупинки буде те, що вказівник на наступний елемент буде “nullptr”.
7. Адже кожен наступний елемент однобічно зв’язаний з попереднім. Тобто, щоб добратися до н-ного елемента ланцюжка, потрібно почати з першого, який містить посилання на наступний і т.д.
8. На перший елемент не вказує жоден інший. Останній елемент замість показника на наступний елемент має “nullptr”.
9. Об’єкти видаляються з метою звільнення пам’яті, з якою працює програма. Якщо ця пам’ять не звільняється, в подальшому надмірне заповнення (переповнення) оперативної пам’яті комп’ютера можна сповільнити хід виконання програми або й зовсім припинити її роботу.