МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №7
на тему: “ЛІНІЙНІ СТРУКТУРИ ДАНИХ”
з дисципліни: "Алгоритми і структури даних”

Лектор:
доцент кафедри ПЗ
Коротєєва Т. О.

Виконав:
студент групи ПЗ-24
Губик А. С.

Прийняв:
асистент кафедри ПЗ
Вишневський О. К.

Львів – 2023

Тема роботи

ЛІНІЙНІ СТРУКТУРИ ДАНИХ

Мета роботи

Познайомитися з лінійними структурами даних (стек, черга, дек, список) та отримати навички програмування алгоритмів, що їх обробляють.

Індивідуальне завдання

Варіант 3: односторонній зв'язаний список символів.

Розробити програму, яка читає з клавіатури послідовність даних, жодне з яких не повторюється, зберігає їх до структури даних (згідно з варіантом), виводить на екран побудовану структуру та наступні характеристики:

1. кількість елементів;
2. мінімальний та максимальний елемент (для символів за кодом);
3. третій елемент з початку послідовності та другий з кінця послідовності;
4. елемент, що стоїть перед мінімальним елементом та елемент, що стоїть після максимального;
5. знайти позицію елемента, значення якого задається з клавіатури;
6. об'єднати дві структури в одну.

Теоретичні відомості

Стек, черга, дек, список відносяться до класу лінійних динамічних структур.

Зі стеку (stack) можна видалити тільки той елемент, який був у нього доданий останнім: стек працює за принципом «останнім прийшов – першим пішов» (last-in, first-out – LIFO).

З черги (queue), навпаки, можна видалити тільки той елемент, який знаходився в черзі довше за всіх: працює принцип «першим прийшов – першим пішов» (first-in, first-out – FIFO).

Дек - це впорядкована лінійна динамічно змінювана послідовність елементів, у якій виконуються такі умови: 1) новий елемент може приєднуватися з обох боків послідовності; 2) вибірка елементів можлива також з обох боків послідовності. Дек називають реверсивною чергою або чергою з двома боками.

У зв'язаному списку (або просто списку; linked list) елементи лінійно впорядковані, але порядок визначається не номерами, як у масиві, а вказівниками, що входять до складу елементів списку. Списки є зручним способом реалізації динамічних множин.

Елемент двобічно зв'язаного списку (doubly linked list) – це запис, що містить три поля: key (ключ) і два вказівники next (наступний) і prev (попередній). Крім цього, елементи списку можуть містити додаткові дані.

У кільцевому списку (circular list) поле prev голови списку вказує на хвіст списку, а поле next хвоста списку вказує на голову списку.

Хід роботи

```
#include <iostream>

struct Node {
    char data;
    Node* next;
};

class ForwardList {
private:
    Node* head;
    char max = 0, min = 127;
    size_t size = 0;

public:
    ForwardList() : head(nullptr) {}

    void push_front(char value) {
        Node* newNode = new Node;
        newNode->data = value;
        newNode->next = head;
        head = newNode;
        if(value > max) max = value;
        if(value < min) min = value;
        size++;
    }
    int find_pos(char elem){
        Node *tmp = head;
        int i = 0;
        for(; tmp->data != elem; tmp = tmp->next, i++)
        {
            if (tmp->next == nullptr ){
                i = -1;
                break;
            }
        }
        return i;

    }
    void append(const ForwardList& other) {
        if (other.head == nullptr) {
            return; // Nothing to append
        }

        if (head == nullptr) {
            head = new Node{other.head->data, nullptr};
        } else {
            Node* current = head;
            while (current->next != nullptr) {
                current = current->next;
            }
            current->next = new Node{other.head->data, nullptr};
```

```
    }

    Node* current = head;
    Node* otherCurrent = other.head->next;
    while (otherCurrent != nullptr) {
        current->next = new Node{otherCurrent->data, nullptr};
        current = current->next;
        otherCurrent = otherCurrent->next;
    }
}

void print() {
    Node* current = head;
    while (current != nullptr) {
        std::cout << current->data << " ";
        current = current->next;
    }
    std::cout << std::endl;
}
void weirdFunction() {
    Node* current = head;
    Node* tmp = head;
    for (int i = 0; i < 2; i++) {
        current = current->next;
    }
    for (int i = 0; i < size - 1; i++) {
        tmp = tmp->next;
    }
    std::cout << "Third element: " << current->data<< std::endl;
    std::cout << "Second from end element: " << tmp->data<< std::endl;
    std::cout << std::endl;
}
void strangeFunction() {
    Node* current = head;
    Node* tmp = head;
    int id1 = find_pos(min) - 1;
    int id2 = find_pos(max) + 1;

    if(id1 < 0){
        std::cout << "There's no element before min." << std::endl;
    }else{
        for (int i = 0; i < id1; i++)
        {
            current = current->next;
        }
        std::cout << "Element before min: " << current->data << std::endl;
    }

    if(id2 > size){
        std::cout << "There's no element after max." << std::endl;
    }else{
        for (int i = 0; i < id2; i++)
```

```
            {
                tmp = tmp->next;
            }
            std::cout << "Element after max: " << tmp->data << std::endl;
        }
    }
    void printMax() {
        std::cout <<"Max element is: "<< max;
        std::cout << std::endl;
    }
    void printMin() {
        std::cout <<"Min element is: "<< min;
        std::cout << std::endl;
    }
    void printSize() {
        std::cout <<"Length of array is: "<< size;
        std::cout << std::endl;
    }
};

int main() {
    ForwardList myForwardList;

    size_t elemCount;
    char elem;
    std::cout << "Enter number of elements: ";
    std::cin >> elemCount;
    std::cout << "Enter the elements: ";
    for (size_t i = 0; i < elemCount; i++)
    {
        std::cin >> elem;
        myForwardList.push_front(elem);

    }

    myForwardList.print();
    myForwardList.printSize();
    myForwardList.printMax();
    myForwardList.printMin();
    myForwardList.weirdFunction();
    myForwardList.strangeFunction();

    std::cout << "Enter the element to search: ";
    std::cin >> elem;
    int pos = myForwardList.find_pos(elem);
    if(pos == -1){
        std::cout << "There's no such element as: " << elem << '\n';
    }else{
        std::cout << "Position of element is: " << pos << '\n';
    }

    return 0;
```

```
}
```

```
artem@laptop:~/Progs++/ADSLabs/Lab7$ g++ -g main.cpp -o main
artem@laptop:~/Progs++/ADSLabs/Lab7$ ./main
Enter number of elements: 6
Enter the elements: qwerty
y t r e w q
Length of array is: 6
Max element is: y
Min element is: e
Third element: r
Second from end element: q

Element before min: r
Element after max: t
Enter the element to search: b
There's no such element as: b
artem@laptop:~/Progs++/ADSLabs/Lab7$
```

Рис. 1:

Висновок

Мабуть найменш використовувана структура даних, прохід по списку можливий тільки лінійний, елементи вставляються тільки спереду, проте маємо константний час вставки. Зазвичай односторонній список є реалізацією стеку, звичайно якщо використання динамічної пам’яті доцільне.