

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗВІТ
до лабораторної роботи №2
з навчальної дисципліни «Алгоритми і структури даних»
Тема: «Абстрактний тип даних “Список”»

Перевірив:
Стоянов Ю. М.
Підготував:
студент групи СП-12
Штокало Андрій Романович

Тернопіль 2023

Мета:

- набути навичок з реалізації АТД “Список”

Завдання:

1. Написати код реалізації логіки роботи основних операцій однозв'язного списку:
 - MAKENULL (створює порожній список)
 - END (повертає кінець списку)
 - FIRST (повертає початок списку)
 - INSERT (додає елемент до списку в задану позицію)
 - DELETE (видаляє елемент з списку)
 - LOCATE (знаходить позицію елементу в списку)
 - RETRIEVE (повертає значення елементу списку)
 - NEXT (повертає вказівник на наступний елемент списку)
2. Написати код реалізації логіки роботи основних операцій двозв'язного списку:
 - MAKENULL (створює порожній список)
 - END (повертає кінець списку)
 - FIRST (повертає початок списку)
 - INSERT (додає елемент до списку в задану позицію)
 - DELETE (видаляє елемент з списку)
 - LOCATE (знаходить позицію елементу в списку)
 - RETRIEVE (повертає значення елементу списку)
 - PREVIOUS (повертає вказівник на попередній елемент списку)
 - NEXT (повертає вказівник на наступний елемент списку)

IDE: Microsoft Visual Studio 2022

Завдання 1

Лістинг програми:

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
```

```

};

Node* head = NULL; // початок списку

// MAKENULL - створює порожній список
void MAKENULL() {
    head = NULL;
}

// END - повертає кінець списку
Node* END() {
    Node* p = head;
    while (p && p->next) {
        p = p->next;
    }
    return p;
}

// FIRST - повертає початок списку
Node* FIRST() {
    return head;
}

// INSERT - додає елемент до списку в задану позицію
void INSERT(int value, int pos) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = NULL;

    if (pos == 1) {
        newNode->next = head;
        head = newNode;
    }
    else {
        Node* prev = head;
        for (int i = 1; i < pos - 1 && prev; i++) {
            prev = prev->next;
        }
        if (prev) {
            newNode->next = prev->next;
            prev->next = newNode;
        }
    }
}

// DELETE - видаляє елемент з списку
void DELETE(int pos) {
    if (head) {
        if (pos == 1) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
        else {
            Node* prev = head;
            for (int i = 1; i < pos - 1 && prev->next; i++) {
                prev = prev->next;
            }
            if (prev->next) {

```

```

        Node* temp = prev->next;
        prev->next = prev->next->next;
        delete temp;
    }
}

// LOCATE - знаходити позицію елементу в списку
int LOCATE(int value) {
    Node* p = head;
    int pos = 1;
    while (p) {
        if (p->data == value) {
            return pos;
        }
        pos++;
        p = p->next;
    }
    return -1;
}

// RETRIEVE - повертає значення елементу списку
int RETRIEVE(int pos) {
    Node* p = head;
    int i = 1;
    while (p && i < pos) {
        p = p->next;
        i++;
    }
    if (p) {
        return p->data;
    }
    return -1;
}

// NEXT - повертає вказівник на наступний елемент списку
Node* NEXT(Node* p) {
    if (p) {
        return p->next;
    }
    return NULL;
}

void PRINTLIST() {
    Node* p = head;
    cout << "List: ";
    while (p) {
        cout << p->data << " ";
        p = p->next;
    }
    cout << endl;
}

int main() {
    int n;
    cout << "Enter number of elements in the list: ";
    cin >> n;
    // додавання елементів до списку

```

```

for (int i = 0; i < n; i++) {
    int value;
    cout << "Enter element " << i + 1 << ": ";
    cin >> value;
    INSERT(value, i + 1);
}

// тестування операцій
while (true) {
    cout << "\nSelect an operation to perform:" << endl;
    cout << "1. MAKENULL" << endl;
    cout << "2. END" << endl;
    cout << "3. FIRST" << endl;
    cout << "4. INSERT" << endl;
    cout << "5. DELETE" << endl;
    cout << "6. LOCATE" << endl;
    cout << "7. RETRIEVE" << endl;
    cout << "8. NEXT" << endl;
    cout << "8. PRINTLIST" << endl;
    cout << "0. Exit" << endl;
    int choice;
    cin >> choice;
    switch (choice) {
        case 1:
            MAKENULL();
            cout << "List is now empty" << endl;
            break;
        case 2: {
            Node* last = END();
            if (last) {
                cout << "Last element in the list is: " << last->data <<
endl;
            }
            else {
                cout << "List is empty" << endl;
            }
            break;
        }
        case 3: {
            Node* first = FIRST();
            if (first) {
                cout << "First element in the list is: " << first->data
<< endl;
            }
            else {
                cout << "List is empty" << endl;
            }
            break;
        }
        case 4: {
            int value, pos;
            cout << "Enter value to insert: ";
            cin >> value;
            cout << "Enter position to insert: ";
            cin >> pos;
            INSERT(value, pos);
            cout << "Element inserted successfully" << endl;
            break;
        }
    }
}

```

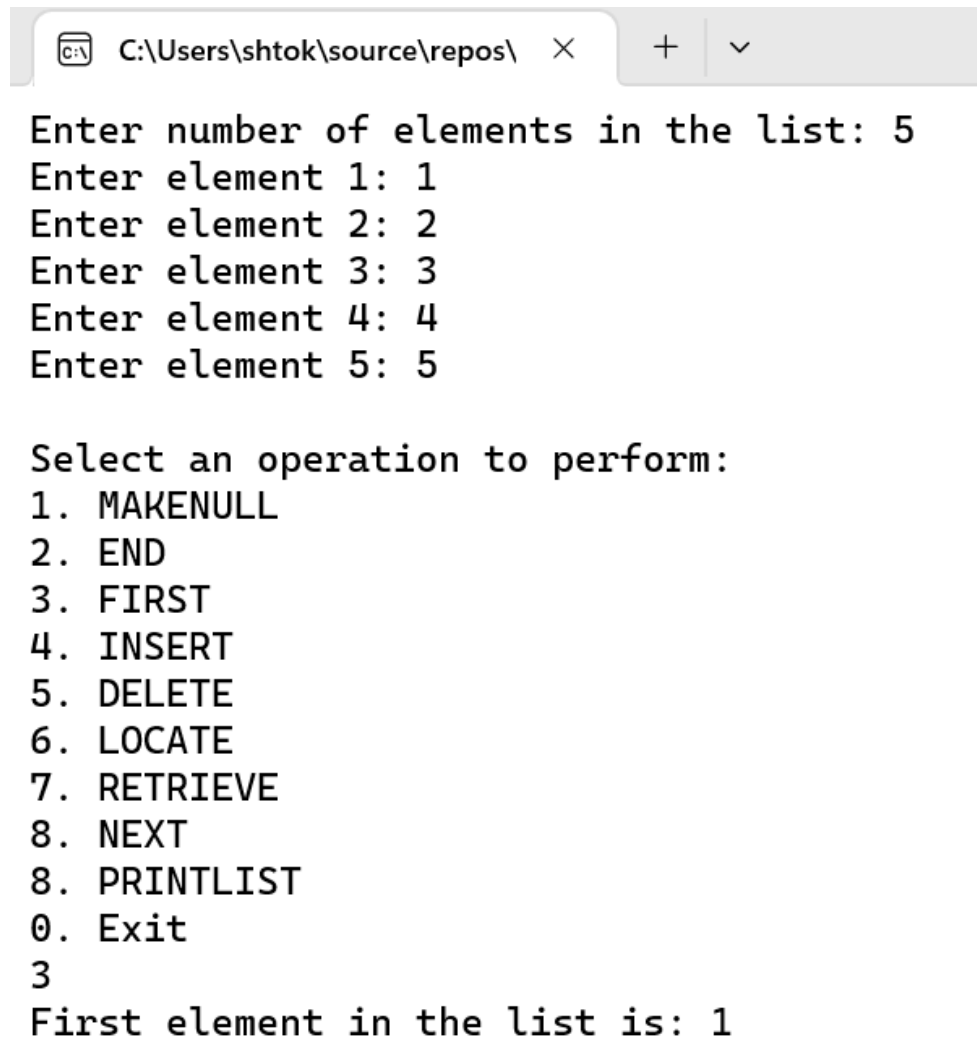
```

case 5: {
    int pos;
    cout << "Enter position of element to delete: ";
    cin >> pos;
    DELETE(pos);
    cout << "Element deleted successfully" << endl;
    break;
}
case 6: {
    int value;
    cout << "Enter value to locate: ";
    cin >> value;
    int pos = LOCATE(value);
    if (pos == -1) {
        cout << "Element not found in the list" << endl;
    }
    else {
        cout << "Element found at position " << pos << endl;
    }
    break;
}
case 7: {
    int pos;
    cout << "Enter position of element to retrieve: ";
    cin >> pos;
    int value = RETRIEVE(pos);
    if (value == -1) {
        cout << "Invalid position or list is empty" << endl;
    }
    else {
        cout << "Element retrieved: " << value << endl;
    }
    break;
}
case 8: {
    int pos;
    cout << "Enter position of element to get next element: ";
    cin >> pos;
    Node* p = head;
    for (int i = 1; i < pos && p; i++) {
        p = p->next;
    }
    if (p && p->next) {
        cout << "Next element is: " << p->next->data << endl;
    }
    else {
        cout << "Invalid position or no next element" << endl;
    }
    break;
}
case 9: {
    PRINTLIST();
    break;
}
case 0:
    cout << "Invalid choice, please enter a valid option!\n";
    break;
}
}

```

```
    return 0;  
}
```

Результат:



```
C:\Users\shtok\source\repos\  
Enter number of elements in the list: 5  
Enter element 1: 1  
Enter element 2: 2  
Enter element 3: 3  
Enter element 4: 4  
Enter element 5: 5  
  
Select an operation to perform:  
1. MAKENULL  
2. END  
3. FIRST  
4. INSERT  
5. DELETE  
6. LOCATE  
7. RETRIEVE  
8. NEXT  
8. PRINTLIST  
0. Exit  
3  
First element in the list is: 1
```

Рисунок 1 - Результат тестування завдання 1

Завдання 2

Лістинг програми:

```
#include <iostream>  
using namespace std;  
  
struct node {  
    int data;  
    node* next;  
    node* prev;  
};  
  
node* head = NULL;  
node* tail = NULL;  
  
void MAKENULL() {  
    head = NULL;
```

```

        tail = NULL;
    }

node* END() {
    return tail;
}

node* FIRST() {
    return head;
}

void INSERT(int x, int pos) {
    node* temp = new node;
    temp->data = x;
    temp->prev = NULL;
    temp->next = NULL;
    if (head == NULL && tail == NULL) {
        head = temp;
        tail = temp;
        return;
    }
    if (pos == 1) {
        temp->next = head;
        head->prev = temp;
        head = temp;
        return;
    }
    node* cur = head;
    for (int i = 1; i < pos - 1; i++) {
        cur = cur->next;
    }
    temp->next = cur->next;
    temp->prev = cur;
    if (cur->next == NULL) {
        tail = temp;
    }
    else {
        cur->next->prev = temp;
    }
    cur->next = temp;
}

void DELETE(int pos) {
    if (head == NULL) {
        cout << "List is empty" << endl;
        return;
    }
    if (pos == 1) {
        node* temp = head;
        head = head->next;
        if (head == NULL) {
            tail = NULL;
        }
        else {
            head->prev = NULL;
        }
        delete temp;
        return;
    }
}

```



```

        node* cur = head;
        for (int i = 1; i < pos; i++) {
            cur = cur->next;
        }
        if (cur == tail) {
            tail = tail->prev;
            tail->next = NULL;
            delete cur;
            return;
        }
        cur->prev->next = cur->next;
        cur->next->prev = cur->prev;
        delete cur;
    }

int LOCATE(int x) {
    node* cur = head;
    int pos = 1;
    while (cur != NULL) {
        if (cur->data == x) {
            return pos;
        }
        cur = cur->next;
        pos++;
    }
    return -1; // element not found
}

int RETRIEVE(int pos) {
    node* cur = head;
    for (int i = 1; i < pos; i++) {
        cur = cur->next;
    }
    return cur->data;
}

node* PREVIOUS(node* p) {
    return p->prev;
}

node* NEXT(node* p) {
    return p->next;
}

void PRINTLIST() {
    node* cur = head;
    while (cur != NULL) {
        cout << cur->data << " ";
        cur = cur->next;
    }
    cout << endl;
}

int main() {
    int n, x, pos;
    cout << "Enter number of items: ";
    cin >> n;
    cout << "Enter the items: " << endl;
    for (int i = 1; i <= n; i++) {

```

```

        cin >> x;
        INSERT(x, i);
    }
    int choice;
    do {
        cout << "Choose an operation: " << endl;
        cout << "1. MAKENULL" << endl;
        cout << "2. END" << endl;
        cout << "3. FIRST" << endl;
        cout << "4. INSERT" << endl;
        cout << "5. DELETE" << endl;
        cout << "6. LOCATE" << endl;
        cout << "7. RETRIEVE" << endl;
        cout << "8. PREVIOUS" << endl;
        cout << "9. NEXT" << endl;
        cout << "10. PRINTLIST" << endl;
        cout << "0. Exit" << endl;
        cin >> choice;
        switch (choice) {
            case 1:
                MAKENULL();
                cout << "List is now empty" << endl;
                break;
            case 2:
                if (tail == NULL) {
                    cout << "List is empty" << endl;
                }
                else {
                    cout << "End of the list: " << tail->data << endl;
                }
                break;
            case 3:
                if (head == NULL) {
                    cout << "List is empty" << endl;
                }
                else {
                    cout << "Beginning of the list: " << head->data << endl;
                }
                break;
            case 4:
                cout << "Enter value and position to insert: ";
                cin >> x >> pos;
                INSERT(x, pos);
                cout << "Element inserted" << endl;
                break;
            case 5:
                cout << "Enter position to delete: ";
                cin >> pos;
                DELETE(pos);
                cout << "Element deleted" << endl;
                break;
            case 6:
                cout << "Enter element to locate: ";
                cin >> x;
                pos = LOCATE(x);
                if (pos == -1) {
                    cout << "Element not found" << endl;
                }
                else {

```

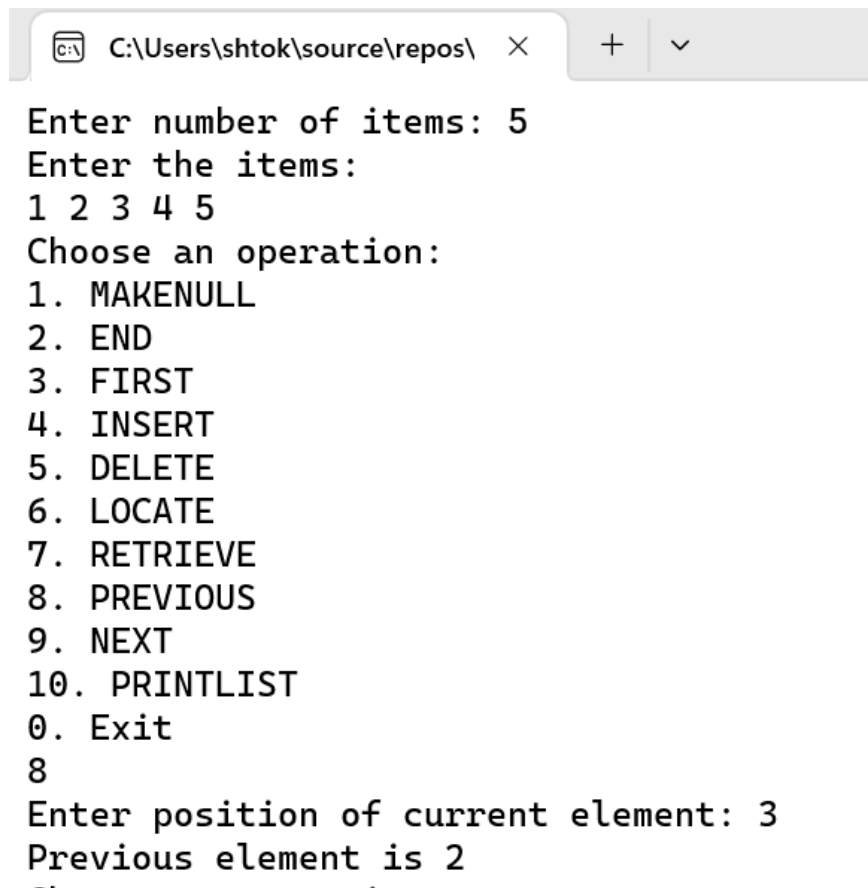
```

        cout << "Element found at position " << pos << endl;
    }
    break;
case 7:
    cout << "Enter position to retrieve element from: ";
    cin >> pos;
    x = RETRIEVE(pos);
    cout << "Element at position " << pos << " is " << x << endl;
    break;
case 8:
    cout << "Enter position of current element: ";
    cin >> pos;
    if (pos == 1) {
        cout << "No previous element" << endl;
    }
    else {
        node* cur = head;
        for (int i = 1; i < pos; i++) {
            cur = cur->next;
        }
        node* prev = PREVIOUS(cur);
        cout << "Previous element is " << prev->data << endl;
    }
    break;
case 9:
    cout << "Enter position of current element: ";
    cin >> pos;
    if (pos == n) {
        cout << "No next element" << endl;
    }
    else {
        node* cur = head;
        for (int i = 1; i < pos; i++) {
            cur = cur->next;
        }
        node* next = NEXT(cur);
        cout << "Next element is " << next->data << endl;
    }
    break;
case 10:
    cout << "List contents: ";
    PRINTLIST();
    break;
case 0:
    cout << "Exiting program" << endl;
    break;
default:
    cout << "Invalid choice" << endl;
}
} while (choice != 0);

return 0;
}

```

Результат:



```
C:\Users\shtok\source\repos\  X  +  v

Enter number of items: 5
Enter the items:
1 2 3 4 5
Choose an operation:
1. MAKENULL
2. END
3. FIRST
4. INSERT
5. DELETE
6. LOCATE
7. RETRIEVE
8. PREVIOUS
9. NEXT
10. PRINTLIST
0. Exit
8
Enter position of current element: 3
Previous element is 2
..
```

Рисунок 2 - Результат тестування завдання 2

Висновки:

- здобуто навички з реалізації АТД “Список”