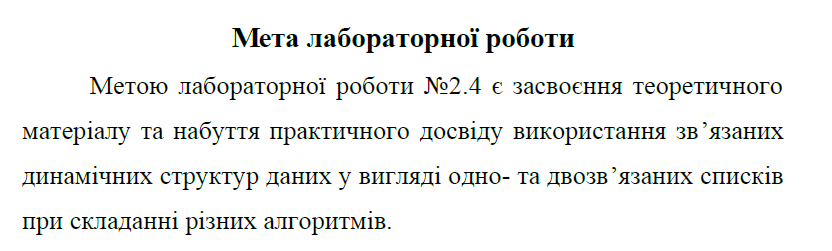
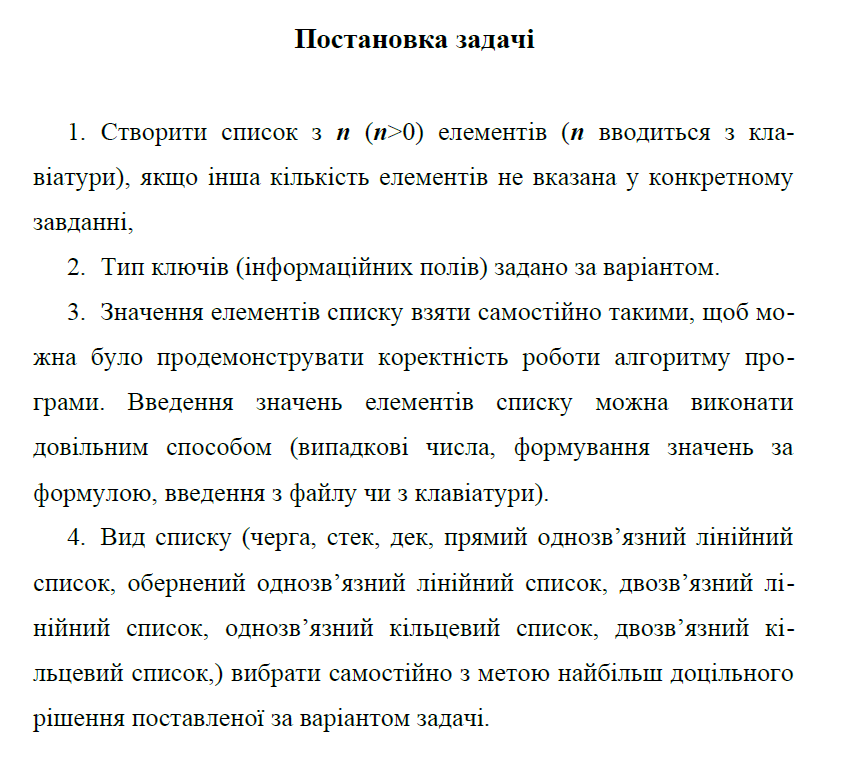
**Лабораторна робота №2.4**

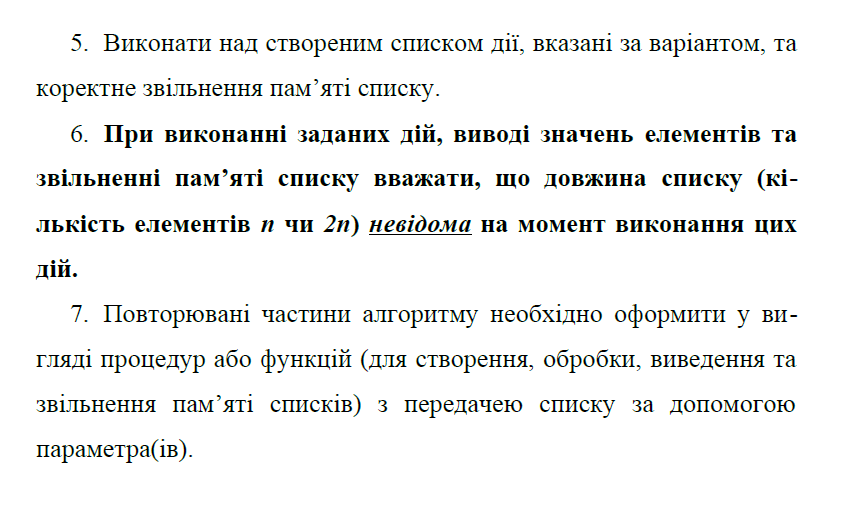
З дисципліни «Структури даних та алгоритми»

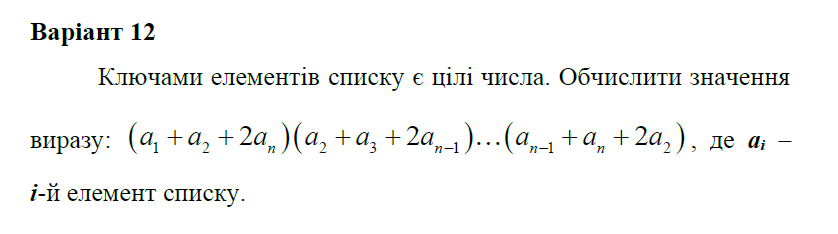
Зв’язні динамічні структури

Виконав: Землянський Едуард

Група: КВ-22

****





**Текст програми**

// [1, 2, 3, 4, 5]

// (1 + 2 + 2\*5) (2 + 3 + 2\*4) (3 + 4 + 2\*3) (4 + 5 + 2\*2)

// [1, 2, 3, 4, 5, 6, 7, 8]

// (1 + 2 + 2\*8) (2 + 3 + 2\*7) (3 + 4 + 2\*6) (4 + 5 + 2\*5) (5 + 6 + 2\*4) (6 + 7 + 2\*3) (7 + 8 + 2\*2)

#include <iostream>

#include <cstdlib>

using namespace std;

class Node{

    public:

        int data;

        Node \* prev, \* next;

    public:

        Node(int data){

            this -> data = data;

            this -> prev = this -> next = NULL;

        }

};

class LinkedList{

    public:

        Node \* head, \* tail;

    public:

        LinkedList(){

            head = tail = NULL;

        }

        ~LinkedList(){

            while(head != NULL)

                pop\_front();

        }

        Node \* push\_front(int data){

            Node \* ptr = new Node(data);

            ptr -> next = head;

            if (head != NULL)

                head -> prev = ptr;

            if (tail == NULL)

                tail = ptr;

            head = ptr;

            return ptr;

        }

        Node \* push\_back(int data){

            Node \* ptr = new Node(data);

            ptr -> prev = tail;

            if (tail != NULL)

                tail -> next = ptr;

            if (head == NULL)

                head = ptr;

            tail = ptr;

            return ptr;

        }

        void pop\_front(){

            if (head == NULL) return;

            Node \* ptr = head -> next;

            if (ptr != NULL)

                ptr -> prev = NULL;

            else

                tail = NULL;

            delete head;

            head = ptr;

        }

        void pop\_back(){

            if (tail == NULL) return;

            Node \* ptr = tail -> prev;

            if (ptr != NULL)

                ptr -> next = NULL;

            else

                head = NULL;

            delete tail;

            tail = ptr;

        }

        Node \* GetAt(int index){

            Node \* ptr = head;

            int n = 0;

            while (n != index)

            {

                if (ptr == NULL)

                    return ptr;

                ptr = ptr -> next;

                n++;

            }

            return ptr;

        }

        Node \* operator [] (int index) {

            return GetAt(index);

        }

        Node \* insert (int index, int data){

            Node \* right = GetAt(index);

            if (right == NULL)

                return push\_back(data);

            Node \* left = right -> prev;

            if (left == NULL)

                return push\_front(data);

            Node \* ptr = new Node(data);

            ptr -> prev = left;

            ptr -> next = right;

            left -> next = ptr;

            right -> prev = ptr;

        }

        void erase(int index){

            Node \* ptr = GetAt(index);

            if (ptr == NULL)

                return;

            if (ptr -> prev == NULL){

                pop\_front();

                return;

            }

            if (ptr -> next == NULL){

                pop\_back();

                return;

            }

            Node \* left = ptr -> prev;

            Node \* right = ptr -> next;

            left -> next = right;

            right -> prev = left;

            delete ptr;

        }

};

int main() {

    srand(time(0));

    LinkedList lst;

    int length = 15;

    for (int i = 1; i < length + 1; i++){

        lst.push\_back(rand() % 10 + 1); // [1; 10] rand int

    }

    cout << "Integer list: ";

    for(Node \* ptr = lst.head; ptr != NULL; ptr = ptr -> next){

        cout << ptr -> data << " ";

    }

    cout << endl;

    Node \* first = lst.head;

    Node \* last = lst.tail;

    int sum = 0;

    long long int mult = 1;

    while (first -> next != NULL || last -> prev != NULL){

        cout << "(" << first -> data << " + " << first -> next -> data << " + " << last -> data << ")";

        sum = first -> data + first -> next -> data + last -> data;

        mult \*= sum;

        first = first -> next;

        last = last -> prev;

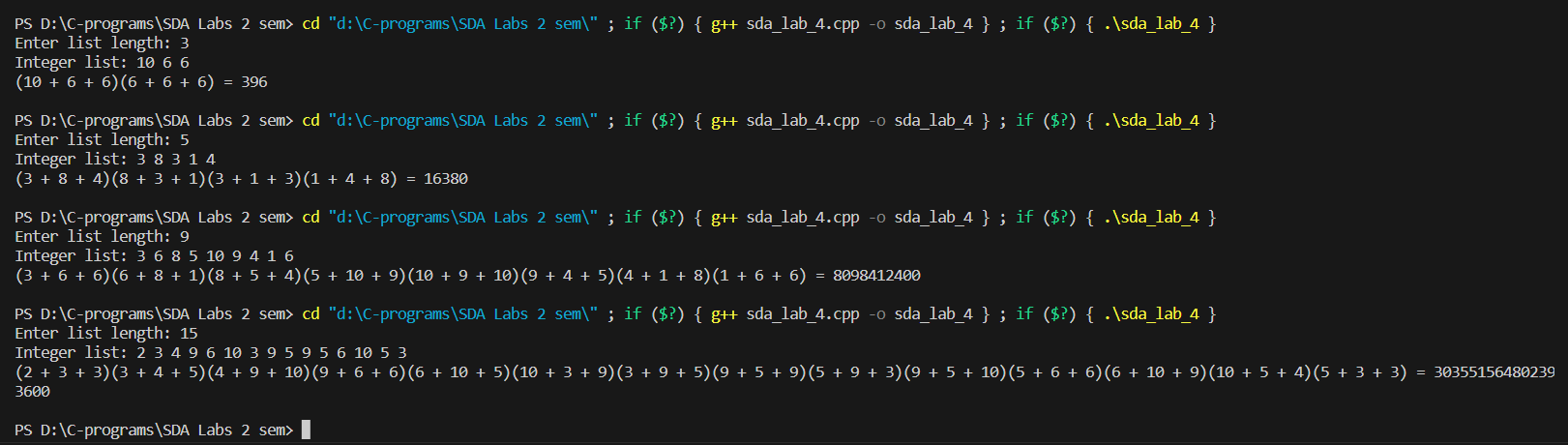
    }

    cout << " = " << mult << endl;

    return 0;

}

**Тестування, результати для різних вхідних даних:**

****

