Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 6

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 10
Алготестер Лабораторної Роботи № 5
Алготестер Лабораторної Роботи № 7-8
Практичних Робіт до блоку № 6

Виконав:
Студент групи ШІ-12
Маковичч Маркіян

Львів 2024

Тема роботи: Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

Мета роботи: навчитись працювати з динамічними структурами, спробувати написати власні алгоритми для таких структур як: черга, стек, список та дерево.

Теоретичні відомості:

1) Структури
2) Класи
3) Список
4) Подвійний список
5) Бінарне дерево

Індивідуальний план опрацювання теорії:

- Тема №1 Структури (50 хв) (https://www.youtube.com/watch?v=999IE-6b7_s)
- Тема №2 Класи (50 хв) (https://www.youtube.com/watchv=ZbsukxxV5_Q&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=95)
- Тема №3 Список (70 хв) (https://www.youtube.com/watchv=-25REjF_atI&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141)
- Тема №4 Двозв’язний список (40 хв) (https://www.youtube.com/watch?v=QLzu2_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g)
- Тема №5 Бінарне дерево (50 хв) (https://www.youtube.com/watch?v=qBFzNW0ALxQ&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g)

Виконання роботи

Завдання №1 Epic 6 Task 3 - Lab# programming: VNS Lab 10

```cpp
#include <iostream>
#include <cstring>
#include <fstream>
using namespace std;

struct Node
{

    char *data;
    Node *next;
    Node *prev;
};
class List
{
private:
    Node *head;
    Node *tail;

public:
    List()
    {
        head = nullptr;
        tail = nullptr;
    }
    ~List()
    {
        Node *current = head;
        while (current != nullptr)
        {
            Node *next = current->next;
            delete current->data;
            delete current;
            current = next;
        }
    }
    void printList()
    {
        if (head == nullptr)
        {
            cout << "List is empty" << endl;
        }
        else
        {
            Node *current = head;
            while (current != nullptr)
            {
                cout << current->data << " ";
                current = current->next;
            }
        }
    }
```

```cpp
                cout << current->data << " ";
                current = current->next;
            }
        }
    }

    void pushFront(const char *element)
    {
        char *value = strdup(element);
        if (head == nullptr && tail == nullptr)
        {
            head = new Node{value, nullptr, nullptr};
            tail = head;
        }
        else
        {
            head->prev = new Node{value, head, nullptr};
            head = head->prev;
        }
    }

    void removeElement(const char *value)
    {
        Node *current = head;
        while (current != nullptr)
        {
            if (!strcmp(current->data, value))
            {
                if (current == head)
                {
                    if (head == tail)
                    {
                        head = nullptr;
                        tail = nullptr;
                        delete head;
                    }
                    else
                    {
                        Node *temp = head;
                        head = head->next;
                        delete temp;
                    }
                }
                else if (current == tail)
                {
                    Node *temp = tail;
                    tail = tail->prev;
```

```
                    tail = tail->prev;
                    delete temp;
                }
                else
                {

                    Node *temp = current;
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                    current = current->next;
                    delete temp;
                }
            }
            current = current->next;
        }
    }
    void addListToFile(const char *fileName)
    {
        ofstream fileIn;

        fileIn.open(fileName);

        Node *current = head;

        while (current != nullptr)
        {
            fileIn << current->data << endl;
            current = current->next;
        }
        fileIn.close();
    }
    void deleteList()
    {
        while (head != nullptr)
        {
            Node *temp = head;
            head = head->next;
            delete temp;
        }
        tail = nullptr;
    }
    void printListFromFile(const char *fileName)
    {
        ifstream fileOut;
        fileOut.open(fileName);
        char element[256];
        while (fileOut >> element)
```

```
    }
    void printListFromFile(const char *fileName)
    {
        ifstream fileOut;
        fileOut.open(fileName);
        char element[256];
        while (fileOut >> element)
        {

            pushFront(element);
        }
        printList();

        fileOut.close();
    }
};

int main()
{

    char fileName[100] = "test.txt";
    List list;

    list.pushFront("you");
    list.pushFront("are");
    list.pushFront("How");
    list.pushFront("Hello");
    list.printList();
    cout << endl;
    list.removeElement("are");
    list.removeElement("Hello");
    list.printList();
    list.addListToFile(fileName);
    cout << endl;
    list.deleteList();
    list.printList();
    list.printListFromFile(fileName);
    list.deleteList();
    cout << endl;
    list.printList();

    return 0;
}
```

```
PS C:\Users\Маркіян> cd "c:\Users\Марк?
Hello How are you
How you
List is empty
you How
List is empty
PS C:\Users\Маркіян\Desktop\epic_6>
```

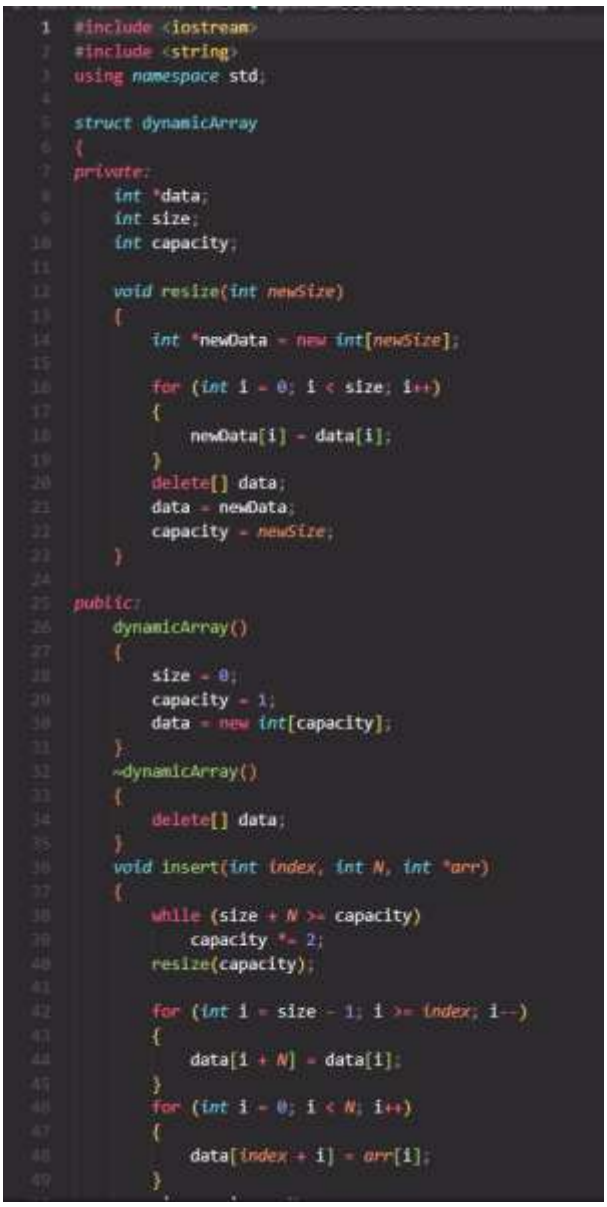
Завдання №2 Epic 6 Task 4 - Lab# programming: Algotester Lab 5

```cpp
#include <iostream>
#include <bitset>
#include <cmath>
#include <string>
#include <cstdint>

using namespace std;

void makeBoard(bitset<64> binBoard, int board[8][8])
{
    int count = 0;
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            board[i][j] = binBoard[count];
            count++;
        }
    }
}

int main()
{
    uint64_t a;
    cin >> a;
    bitset<64> binBoard(a);
    int board[8][8] = {};
    makeBoard(binBoard, board);
    int questions;
    cin >> questions;
    int coord[questions][2] = {};
    for (int i = 0; i < questions; i++)
    {
        cin >> coord[i][0] >> coord[i][1];
        coord[i][0]--;
        coord[i][1]--;
    }
    int row, column;
    for (int i = 0; i < questions; i++)
    {
```

```cpp
        cin >> coord[i][0] >> coord[i][1];
        coord[i][0]--;
        coord[i][1]--;
    }
    int row, column;
    for (int i = 0; i < questions; i++)
    {
        row = coord[i][0];
        column = coord[i][1];
        board[row][column] = (board[row][column] == 1) ? 0 : 1;
        for (int j = 0; j < 8; j++)
        {
            board[row][j] = (board[row][j] == 1) ? 0 : 1;
            board[j][column] = (board[j][column] == 1) ? 0 : 1;
        }
    };

    uint64_t result = 0;

    int count = 0;
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            if (board[i][j] == 1)
            {
                uint64_t c = pow(2, count);
                result += c;
            }
            count++;
        }
    }
    cout << result;

    return 0;
}
```

```
PS C:\Users\Маркіян> cd "c:\Users\Марк?ян\Desktop
lab_5_task_markiian_makovych }
0
4
1 1
8 8
1 8
8 1
9295429630892703873
PS C:\Users\Маркіян\Desktop\epic_6>
```

Завдання №3 Epic 6 Task 5 - Lab# programming: Algotester Lab 7-8 variant 1



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

struct dynamicArray
{
private:
    int *data;
    int size;
    int capacity;

    void resize(int newSize)
    {
        int *newData = new int[newSize];

        for (int i = 0; i < size; i++)
        {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        capacity = newSize;
    }

public:
    dynamicArray()
    {
        size = 0;
        capacity = 1;
        data = new int[capacity];
    }
    ~dynamicArray()
    {
        delete[] data;
    }
    void insert(int index, int N, int *arr)
    {
        while (size + N >= capacity)
            capacity *= 2;
        resize(capacity);

        for (int i = size - 1; i >= index; i--)
        {
            data[i + N] = data[i];
        }
        for (int i = 0; i < N; i++)
        {
            data[index + i] = arr[i];
        }
```

```cpp
        size = size + N;
        if (size == capacity)
        {
            capacity *= 2;
            resize(capacity);
        }
    }
    void erase(int index, int n)
    {
        for (int i = index; i < size - n; i++)
        {
            data[i] = data[i + n];
        }
        size = size - n;
    }
    int getSize()
    {
        return size;
    }
    int getCapacity()
    {
        return capacity;
    }
    int &operator[](int index)
    {
        return data[index];
    }
    friend ostream &operator<<(ostream &output, const dynamicArray &arr)
    {
        for (int i = 0; i < arr.size; i++)
        {
            output << arr.data[i] << ' ';
        }
        return output;
    }
};

int main()
{
    dynamicArray dynArr;
    int Q;
    cin >> Q;
    string str;

    for (int i = 0; i < Q; i++)
    {
        cin >> str;
```

```
            int N;
            cin >> N;
            int arr[N];
            for (int i = 0; i < N; i++)
            {
                cin >> arr[i];
            }
            dynArr.insert(index, N, arr);
        }
        else if (str == "erase")
        {
            int index;
            cin >> index;
            int n;
            cin >> n;
            dynArr.erase(index, n);
        }
        else if (str == "size")
        {
            cout << dynArr.getSize() << endl;
        }
        else if (str == "capacity")
        {
            cout << dynArr.getCapacity() << endl;
        }
        else if (str == "get")
        {
            int index;
            cin >> index;
            cout << dynArr[index] << endl;
        }
        else if (str == "set")
        {
            int index;
            cin >> index;
            int newElement;
            cin >> newElement;
            dynArr[index] = newElement;
        }
        else if (str == "print")
        {
            cout << dynArr << endl;
        }
    }

    return 0;
```

```
12

size
0

insert 0 5
251 252 253 254 255

size
5
capacity
8
print
251 252 253 254 255

get 1
252
set 1 777
get 1
777

erase 1 3

get 1
255

size
2
print
251 255
PS C:\Users\Маркіян\Desktop\epic_6>
```

Завдання №3 Epic 6 Task 5 - Lab# programming: Algotester Lab 7-8 variant 2

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

template <typename T = int>
class dynamicArray
{
private:
    T *data;
    int size;
    int capacity;

    void resize(int newSize)
    {
        T *newData = new T[newSize];

        for (int i = 0; i < size; i++)
        {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        capacity = newSize;
    }

public:
    dynamicArray()
    {
        size = 0;
        capacity = 1;
        data = new T[capacity];
    }
    ~dynamicArray()
    {
        delete[] data;
    }
    void insert(int index, int N, T *arr)
    {

        while (size + N >= capacity)
            capacity *= 2;
        resize(capacity);

        for (int i = size - 1; i >= index; i--)
        {
            data[i + N] = data[i];
        }
        for (int i = 0; i < N; i++)
        {
            }
        size = size + N;
        if (size == capacity)
        {
            capacity *= 2;
            resize(capacity);
        }
    }
    void erase(int index, int n)
    {
        for (int i = index; i < size - n; i++)
        {
            data[i] = data[i + n];
        }
        size = size - n;
    }
    int getSize()
    {
        return size;
    }
    int getCapacity()
    {
        return capacity;
    }
    T &operator[](int index)
    {
        return data[index];
    }
    friend ostream &operator<<(ostream &output, const dynamicArray &arr)
    {
        for (int i = 0; i < arr.size; i++)
        {
            output << arr.data[i] << ' ';
        }
        return output;
    }
};

int main()
{
    dynamicArray<int> dynArr;
    int Q;
    cin >> Q;
    string str;

    for (int i = 0; i < Q; i++)
```

```
            int index;
            cin >> index;
            int N;
            cin >> N;
            int arr[N];
            for (int i = 0; i < N; i++)
            {
                cin >> arr[i];
            }
            dynArr.insert(index, N, arr);
        }
        else if (str == "erase")
        {
            int index;
            cin >> index;
            int n;
            cin >> n;
            dynArr.erase(index, n);
        }
        else if (str == "size")
        {
            cout << dynArr.getSize() << endl;
        }
        else if (str == "capacity")
        {
            cout << dynArr.getCapacity() << endl;
        }
        else if (str == "get")
        {
            int index;
            cin >> index;
            cout << dynArr[index] << endl;
        }
        else if (str == "set")
        {
            int index;
            cin >> index;
            int newElement;
            cin >> newElement;
            dynArr[index] = newElement;
        }
        else if (str == "print")
        {
            cout << dynArr << endl;
        }
    }
```

Завдання №4 Epic 6 Task 6 - Practice# programming: Class Practice Task

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

struct Node
{
    int data;
    Node *next;
};
Node *pushBack(Node *head, int element)
{
    Node *newEl = new Node{element, nullptr};

    if (head == nullptr)
    {
        head = newEl;
    }
    else
    {
        Node *current = head;
        while (current->next != nullptr)
        {
            current = current->next;
        }
        current->next = newEl;
    }
    return head;
};
void printList(Node *head)
{
    if (head == nullptr)
    {
        cout << "List is empty" << endl;
    }
    Node *current = head;

    while (current != nullptr)
    {
        cout << current->data << " ";
        current = current->next;
    }
}
void printNum(Node *head)
{
    if (head == nullptr)
    {
        cout << "List is empty" << endl;
    }
    Node *current = head;

    while (current != nullptr)
    {
        cout << current->data;
        current = current->next;
    }
}
Node *reverse(Node *head)
{
    if (head == nullptr)
    {
        cout << "List is empty" << endl;
        return 0;
    }
    else
    {
        Node *current = head;
        Node *next = nullptr;
        Node *temp = nullptr;
        while (current != nullptr)
        {
            next = current->next;
            current->next = temp;
            temp = current;
            current = next;
        }
        return temp;
    }
};
bool compare(Node *headF, Node *headS)
{

    Node *current1 = headF;
    Node *current2 = headS;

    while (current1 != nullptr && current2 != nullptr)
    {
        if (current1->data != current2->data)
        {
            return false;
        };
        current1 = current1->next;
        current2 = current2->next;
    }
    if (current1 != nullptr || current2 != nullptr)
    {
        return false;
```

```
    }
    if (current1 != nullptr || current2 != nullptr)
    {
        return false;
    }
    else
    {
        return true;
    }
};

Node *addNum(Node *num1, Node *num2)
{
    Node *current1 = num1;
    Node *current2 = num2;
    Node *sum = nullptr;
    int tempSum = 0;
    int over = 0;

    while (current2 != nullptr)
    {
        tempSum = current1->data + current2->data + over;

        if (tempSum <= 9)
        {
            sum = pushBack(sum, tempSum);
            over = 0;
        }
        if (tempSum > 9)
        {
            sum = pushBack(sum, tempSum % 10);
            over = tempSum / 10;
        }
        current1 = current1->next;
        current2 = current2->next;
    }
    if (current1 != nullptr)
    {
        while (current1 != nullptr)
        {
            tempSum = current1->data + over;
            if (tempSum <= 9)
            {
                sum = pushBack(sum, tempSum);
                over = 0;
            }
            if (tempSum > 9)
            {
                sum = pushBack(sum, tempSum % 10);
                over = tempSum / 10;
            }
            current1 = current1->next;
        }
    }
    else if (over != 0)
    {
        sum = pushBack(sum, over);
    }
    return sum;
};

struct treeNode
{
    int data;
    treeNode *left;
    treeNode *right;

    treeNode(int value) : data(value), left(nullptr), right(nullptr) {};
};

treeNode *insert(treeNode *node, int value)
{
    if (node == nullptr)
    {
        return new treeNode(value);
    }
    else
    {
        if (value > node->data)
        {
            node->right = insert(node->right, value);
        }
        else
        {
            node->left = insert(node->left, value);
        }
        return node;
    }
};
void printTree(treeNode *node)
{
```

```
    }
    cout << node->data << " ";
    printTree(node->left);
    printTree(node->right);
}

treeNode *mirror(treeNode *node)
{
    if (!node)
        return 0;

    swap(node->left, node->right);
    mirror(node->left);
    mirror(node->right);
    return node;
};

treeNode *treeSum(treeNode *node)
{
    if ((node == nullptr) || ((node->left == nullptr) && (node->right == nullptr)))
        return nullptr;

    treeSum(node->left);
    treeSum(node->right);

    node->data = 0;
    if (node->right != nullptr)
    {
        node->data += node->right->data;
    }
    if (node->left != nullptr)
    {
        node->data += node->left->data;
    }

    return node;
}

int main()
{
    cout << "TASK 1" << endl;
    cout << endl;
    Node *head = nullptr;

    for (int i = 1; i < 16; i++)
    {
        head = pushBack(head, i);
```

```
{
    cout << "TASK 1" << endl;
    cout << endl;
    Node *head = nullptr;

    for (int i = 1; i < 16; i++)
    {
        head = pushBack(head, i);
    }
    cout << "First list:" << endl;
    printList(head);
    Node *head1 = reverse(head);
    cout << endl;
    cout << "Reversed List:" << endl;
    printList(head1);

    cout << endl;
    cout << endl;
    cout << "TASK 2" << endl;
    cout << endl;

    Node *headF = nullptr;
    Node *headS = nullptr;

    headF = pushBack(headF, 1);
    headF = pushBack(headF, 5);
    headF = pushBack(headF, 3);
    headF = pushBack(headF, 1);
    headF = pushBack(headF, 5);
    headF = pushBack(headF, 7);

    headS = pushBack(headS, 1);
    headS = pushBack(headS, 5);
    headS = pushBack(headS, 3);
    headS = pushBack(headS, 100);
    headS = pushBack(headS, 5);
    headS = pushBack(headS, 7);

    if (compare(headF, headS))
    {
        cout << "List are identical" << endl;
    }
    else
    {
        cout << "List are different" << endl;
    }

    cout << endl;
```

```
string num1, num2, temp;
cout << "First number" << endl;
cin >> num1;
cout << "Second number" << endl;
cin >> num2;

if (num2.length() > num1.length())
{
    temp = num1;
    num1 = num2;
    num2 = temp;
}
Node *n1 = nullptr;
Node *n2 = nullptr;

for (int i = num1.length() - 1; i >= 0; i--)
{
    n1 = pushBack(n1, (int)num1[i] - 48);
}
for (int i = num2.length() - 1; i >= 0; i--)
{
    n2 = pushBack(n2, (int)num2[i] - 48);
}
Node *sum = addNum(n1, n2);

cout << "Sum is equal to: ";
printNum(reverse(sum));
cout << endl;
cout << endl;
cout << "TASK 4" << endl;
cout << endl;

treeNode *root = nullptr;

root = insert(root, 10);
root = insert(root, 7);
root = insert(root, 1);
root = insert(root, 6);
root = insert(root, 20);
root = insert(root, 23);
root = insert(root, 12);
root = insert(root, 3);
cout << "Binary tree:" << endl;
printTree(root);
treeNode *newTree = mirror(root);
cout << endl;
cout << "Mirrored tree:" << endl;
printTree(root);
```

```
TASK 1

First list:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Reversed List:
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

TASK 2

List are different

TASK 3

First number
12348957138998302149813298132723189741324789
Second number
13257890513247890451789412578954178934789435879437893457 89
Sum is equal to: 1325789051324801394136080256197567706777076311133530670578

TASK 4

Binary tree:
10 7 1 6 3 20 12 23
Mirrored tree:
10 20 23 12 7 1 6 3

TASK 5

Tree Sum:
38 35 23 12 3 3 3 3
```

Завдання №5 Epic 6 Task 7 - Practice# programming: Self Practice Task

Дивні кульки

Обмеження: 2 сек., 256 МіБ

Дітвора хоче Нового року. Усі зібралися святкувати вдома в Зеника, поки його батьки поїхали на засніжені Альпи. Дітвора хоче мандаринок, реклами коли на телебаченні, кульок. Реклама вже давно є, декілька хвилин тому Зеник побіг до магазину вибирати мандаринки, а дівчата залишилися наводити лад у квартирі. Коли все було прибрано, вони згадали, що не виконали однієї своєї мрії — сотні надутих кульок навколо. Тішило те, що Марічка десь у спальні батьків Зеника знайшла кульки. «Хм, що вони там роблять?» — здивувалася Марічка. І зрозуміла вона це аж через декілька років. Дівчата спробували надути кульки, але зрозуміли, що вони занадто... дівчата(?!), щоб їх надувати. Ух ця вже дівоча логіка. Змусили хлопців надувати кульки. Але хлопці теж не дурні — поставили свої умови.

Хлопців є n, і кожен з них надуває одну кульку за t_i секунд, але після того, як надував кожні z_i кульок, перепочиває y_i секунд.

От дівчатам і стало цікаво, скільки потрібно найменше секунд, щоб надути m кульок.

Вхідні дані

У першому рядку дано два цілих числа m і n — кількість кульок, які потрібно надути, і кількість хлопців у розпорядженні дівчат.

У наступних n рядках дано по три цілих числа — t_i, z_i, y_i.

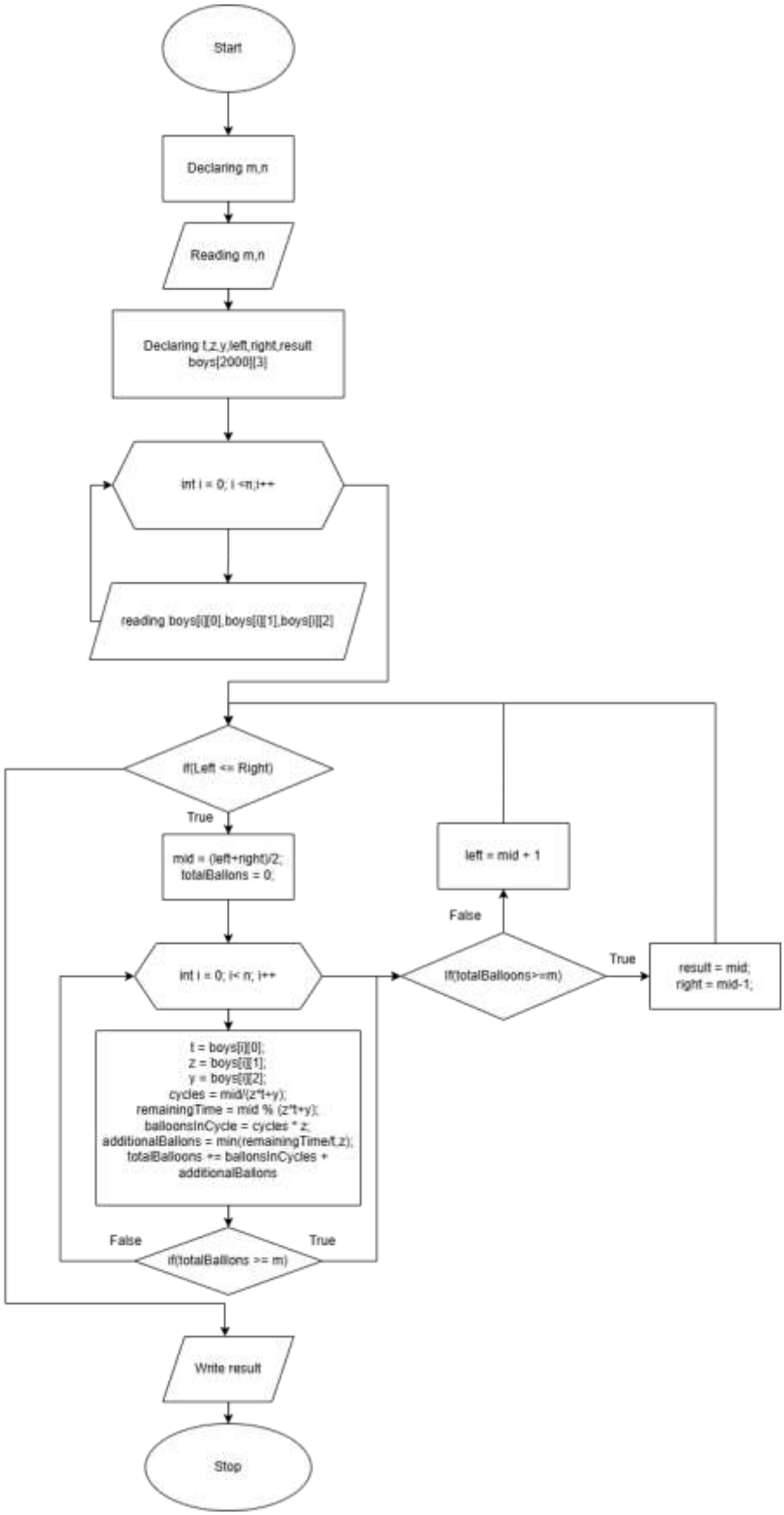
Вихідні дані

В одному рядку виведіть ціле число — найменший час, через який будуть надуті всі кульки.

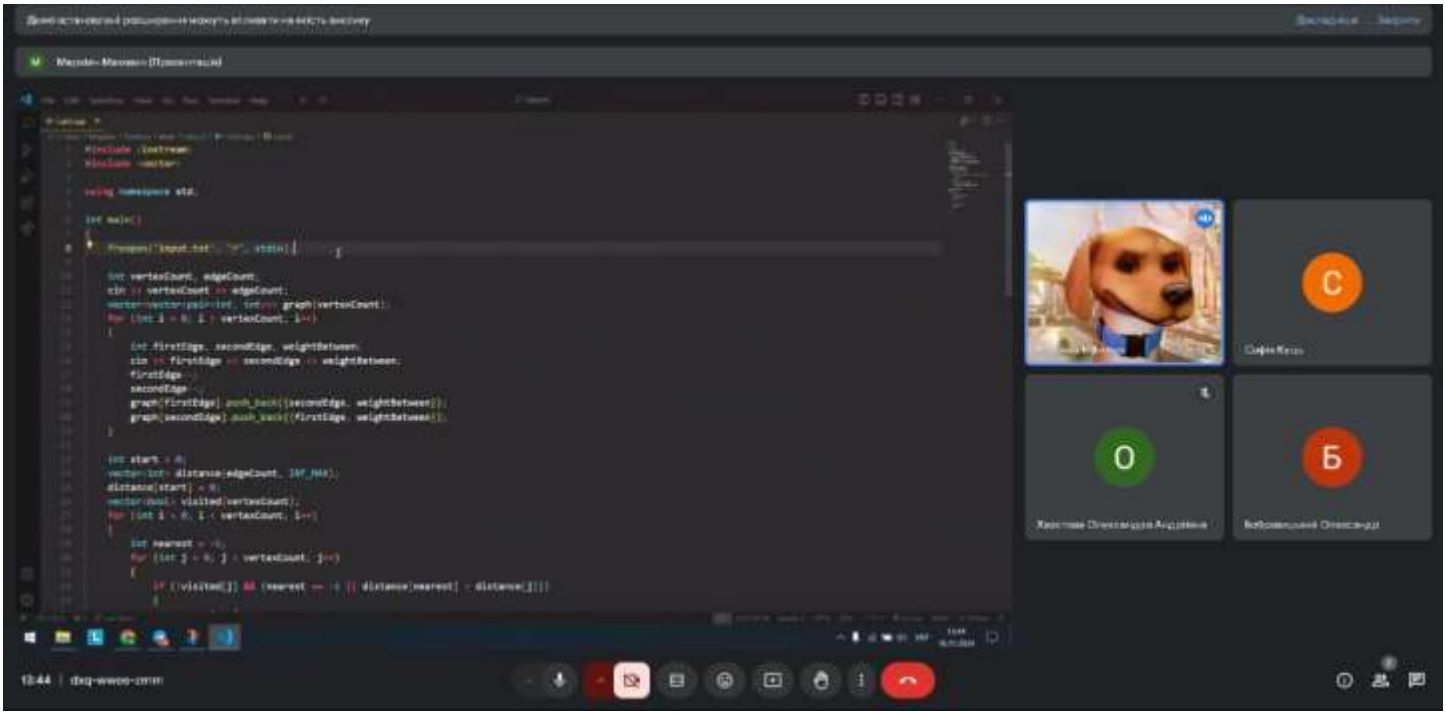
```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int m, n;
    cin >> m >> n;

    int t, z, y;
    int left = 0, right = 2000 * 2000;
    int result = right;

    int boys[2000][3];

    for (int i = 0; i < n; i++)
    {
        cin >> boys[i][0] >> boys[i][1] >> boys[i][2];
    }

    while (left <= right)
    {
        int mid = (left + right) / 2;
        int totalBalloons = 0;

        for (int i = 0; i < n; i++)
        {
            t = boys[i][0];
            z = boys[i][1];
            y = boys[i][2];

            int cycles = mid / (z * t + y);
```

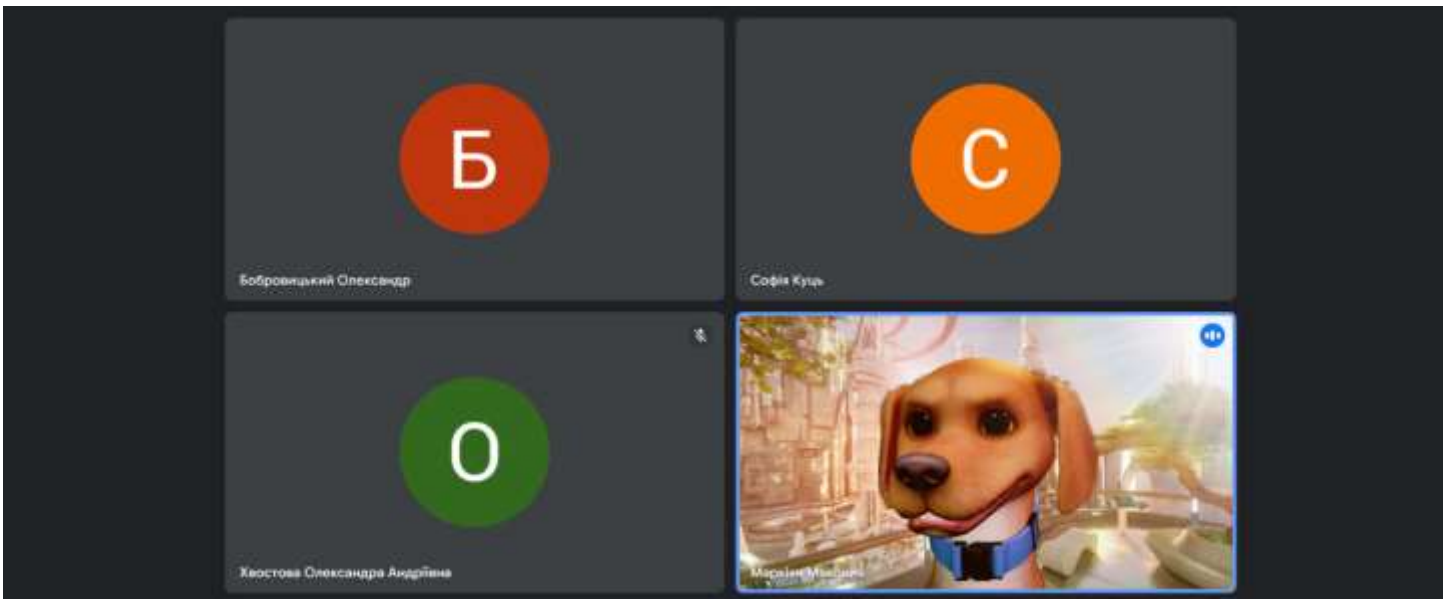
```
            int cycles = mid / (z * t + y);

            int remainingTime = mid % (z * t + y);

            int balloonsInCycles = cycles * z;
            int additionalBalloons = min(remainingTime / t, z);

            totalBalloons += balloonsInCycles + additionalBalloons;

            if (totalBalloons >= m)
            {
                break;
            }
        }

        if (totalBalloons >= m)
        {
            result = mid;
            right = mid - 1;
        }
        else
        {
            left = mid + 1;
        }
    }

    cout << result << endl;

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
21 хвилину тому	C++ 23	Зараховано	0.004	1.172	Перегляд


Start
Declaring m,n
Reading m,n
Declaring t,z,y,left,right,result
boys[2000][3]
int i = 0; i <n;i++
reading boys[i][0],boys[i][1],boys[i][2]
if(Left <= Right)
True
mid = (left+right)/2;
totalBallons = 0;
int i = 0; i< n; i++
t = boys[i][0];
z = boys[i][1];
y = boys[i][2];
cycles = mid/(z*t+y);
remainingTime = mid % (z*t+y);
balloonsInCycle = cycles * z;
additionalBallons = min(remainingTime/t,z);
totalBalloons += ballonsInCycles +
additionalBallons
False
True
if(totalBallons >= m)
if(totalBalloons>=m)
True
result = mid;
right = mid-1;
False
left = mid + 1
Write result
Stop

Робота в команді





Висновок: під час виконання лабораторної роботи я навчився краще працювати і розуміти динамічні структури. Навчився писати власні алгоритми для роботи з такими структурами.