

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 ІІІ-12

Макович 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](https://www.youtube.com/watch?v=999IE-6b7_s))
- Тема №2 Класи (50 хв)  
([https://www.youtube.com/watch?v=ZbsukxxV5\\_Q&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=95](https://www.youtube.com/watch?v=ZbsukxxV5_Q&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=95))
- Тема №3 Список (70 хв)  
([https://www.youtube.com/watch?v=-25REjF\\_atI&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141](https://www.youtube.com/watch?v=-25REjF_atI&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141))
- Тема №4 Двобічний список (40 хв)  
([https://www.youtube.com/watch?v=QLzu2\\_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g](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 Епіс 6 Task 3 - Lab# programming: VNS Lab 10

```
1  ~ #include <iostream>
2  #include <cstring>
3  #include <fstream>
4  using namespace std;
5
6  ~ struct Node
7  {
8
9      char *data;
10     Node *next;
11     Node *prev;
12 };
13 ~ class List
14 {
15 private:
16     Node *head;
17     Node *tail;
18
19 public:
20     List()
21     {
22         head = nullptr;
23         tail = nullptr;
24     }
25     ~List()
26     {
27         Node *current = head;
28         while (current != nullptr)
29         {
30             Node *next = current->next;
31             delete current->data;
32             delete current;
33             current = next;
34         }
35     }
36     void printList()
37     {
38         if (head == nullptr)
39         {
40             cout << "List is empty" << endl;
41         }
42         else
43         {
44             Node *current = head;
45             while (current != nullptr)
46             {
47                 cout << current->data << " ";
48                 current = current->next;
49             }
50         }
51     }
52 }
```

```
47         cout << current->data << " ";
48         current = current->next;
49     }
50 }
51
52 void pushFront(const char *element)
53 {
54     char *value = strdup(element);
55     if (head == nullptr && tail == nullptr)
56     {
57         head = new Node{value, nullptr, nullptr};
58         tail = head;
59     }
60     else
61     {
62         head->prev = new Node{value, head, nullptr};
63         head = head->prev;
64     }
65 }
66
67 void removeElement(const char *value)
68 {
69     Node *current = head;
70     while (current != nullptr)
71     {
72         if (!strcmp(current->data, value))
73         {
74             if (current == head)
75             {
76                 if (head == tail)
77                 {
78                     head = nullptr;
79                     tail = nullptr;
80                     delete head;
81                 }
82                 else
83                 {
84                     Node *temp = head;
85                     head = head->next;
86                     delete temp;
87                 }
88             }
89             else if (current == tail)
90             {
91                 Node *temp = tail;
92                 tail = tail->prev;
93                 delete temp;
94             }
95             else
96             {
97                 Node *prev = current->prev;
98                 Node *next = current->next;
99                 prev->next = next;
100                next->prev = prev;
101                delete current;
102            }
103        }
104        current = current->next;
105    }
106 }
```

```

93         tail = tail->prev;
94         delete temp;
95     }
96     else
97     {
98
99         Node *temp = current;
100        current->prev->next = current->next;
101        current->next->prev = current->prev;
102        current = current->next;
103        delete temp;
104    }
105 }
106 current = current->next;
107 }
108 }
109 void addListToFile(const char *fileName)
110 {
111     ofstream fileIn;
112
113     fileIn.open(fileName);
114
115     Node *current = head;
116
117     while (current != nullptr)
118     {
119         fileIn << current->data << endl;
120         current = current->next;
121     }
122     fileIn.close();
123 }
124 void deleteList()
125 {
126     while (head != nullptr)
127     {
128         Node *temp = head;
129         head = head->next;
130         delete temp;
131     }
132     tail = nullptr;
133 }
134 void printListFromFile(const char *fileName)
135 {
136     ifstream fileOut;
137     fileOut.open(fileName);
138     char element[256];
139     while (fileOut >> element)

```

```

133     }
134 void printListFromFile(const char *fileName)
135 {
136     ifstream fileOut;
137     fileOut.open(fileName);
138     char element[256];
139     while (fileOut >> element)
140     {
141
142         pushFront(element);
143     }
144     printList();
145
146     fileOut.close();
147 }
148 };
149
150 int main()
151 {
152
153     char fileName[100] = "test.txt";
154     List list;
155
156     list.pushFront("you");
157     list.pushFront("are");
158     list.pushFront("How");
159     list.pushFront("Hello");
160     list.printList();
161     cout << endl;
162     list.removeElement("are");
163     list.removeElement("Hello");
164     list.printList();
165     list.addListToFile(fileName);
166     cout << endl;
167     list.deleteList();
168     list.printList();
169     list.printListFromFile(fileName);
170     list.deleteList();
171     cout << endl;
172     list.printList();
173
174     return 0;
175 }

```

```

PS C:\Users\Маркіян> cd "c:\Users\Маркіян\
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

```

1  #include <iostream>
2  #include <bitset>
3  #include <cmath>
4  #include <string>
5  #include <cstdlib>
6
7  using namespace std;
8
9  void makeBoard(bitset<64> binBoard, int board[8][8])
10 {
11     int count = 0;
12     for (int i = 0; i < 8; i++)
13     {
14         for (int j = 0; j < 8; j++)
15         {
16             board[i][j] = binBoard[count];
17             count++;
18         }
19     }
20 }
21
22 int main()
23 {
24     uint64_t a;
25     cin >> a;
26     bitset<64> binBoard(a);
27     int board[8][8] = {};
28     makeBoard(binBoard, board);
29     int questions;
30     cin >> questions;
31     int coord[questions][2] = {};
32     for (int i = 0; i < questions; i++)
33     {
34         cin >> coord[i][0] >> coord[i][1];
35         coord[i][0]--;
36         coord[i][1]--;
37     }
38     int row, column;
39     for (int i = 0; i < questions; i++)
40     {

```

```

34         cin >> coord[i][0] >> coord[i][1];
35         coord[i][0]--;
36         coord[i][1]--;
37     }
38     int row, column;
39     for (int i = 0; i < questions; i++)
40     {
41         row = coord[i][0];
42         column = coord[i][1];
43         board[row][column] = (board[row][column] == 1) ? 0 : 1;
44         for (int j = 0; j < 8; j++)
45         {
46             board[row][j] = (board[row][j] == 1) ? 0 : 1;
47             board[j][column] = (board[j][column] == 1) ? 0 : 1;
48         }
49     }
50
51     uint64_t result = 0;
52
53     int count = 0;
54     for (int i = 0; i < 8; i++)
55     {
56         for (int j = 0; j < 8; j++)
57         {
58             if (board[i][j] == 1)
59             {
60                 uint64_t c = pow(2, count);
61                 result += c;
62             }
63             count++;
64         }
65     }
66     cout << result;
67
68     return 0;
69 }

```

```

PS C:\Users\Маркіян> cd "c:\Users\Маркіян\Desktop\lab_5_task_markian_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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct dynamicArray
6 {
7     private:
8         int *data;
9         int size;
10        int capacity;
11
12        void resize(int newSize)
13        {
14            int *newData = new int[newSize];
15
16            for (int i = 0; i < size; i++)
17            {
18                newData[i] = data[i];
19            }
20            delete[] data;
21            data = newData;
22            capacity = newSize;
23        }
24
25    public:
26        dynamicArray()
27        {
28            size = 0;
29            capacity = 1;
30            data = new int[capacity];
31        };
32        ~dynamicArray()
33        {
34            delete[] data;
35        }
36        void insert(int index, int N, int *arr)
37        {
38            while (size + N >= capacity)
39                capacity *= 2;
40            resize(capacity);
41
42            for (int i = size - 1; i >= index; i--)
43            {
44                data[i + N] = data[i];
45            }
46            for (int i = 0; i < N; i++)
47            {
48                data[index + i] = arr[i];
49            }
50
51            size = size + N;
52            if (size == capacity)
53            {
54                capacity *= 2;
55                resize(capacity);
56            }
57        }
58        void erase(int index, int n)
59        {
60            for (int i = index; i < size - n; i++)
61            {
62                data[i] = data[i + n];
63            }
64            size = size - n;
65        }
66        int getSize()
67        {
68            return size;
69        }
70        int getCapacity()
71        {
72            return capacity;
73        }
74        int &operator[](int index)
75        {
76            return data[index];
77        }
78        friend ostream &operator<<(ostream &output, const dynamicArray &arr)
79        {
80            for (int i = 0; i < arr.size; i++)
81            {
82                output << arr.data[i] << ' ';
83            }
84            return output;
85        };
86
87    int main()
88    {
89        dynamicArray dynArr;
90        int Q;
91        cin >> Q;
92        string str;
93
94        for (int i = 0; i < Q; i++)
95        {
96            cin >> str;
```

```

101     int N;
102     cin >> N;
103     int arr[N];
104     for (int i = 0; i < N; i++)
105     {
106         cin >> arr[i];
107     }
108     dynArr.insert(index, N, arr);
109 }
110 else if (str == "erase")
111 {
112     int index;
113     cin >> index;
114     int n;
115     cin >> n;
116     dynArr.erase(index, n);
117 }
118 else if (str == "size")
119 {
120     cout << dynArr.getSize() << endl;
121 }
122 else if (str == "capacity")
123 {
124     cout << dynArr.getCapacity() << endl;
125 }
126 else if (str == "get")
127 {
128     int index;
129     cin >> index;
130     cout << dynArr[index] << endl;
131 }
132 else if (str == "set")
133 {
134     int index;
135     cin >> index;
136     int newElement;
137     cin >> newElement;
138     dynArr[index] = newElement;
139 }
140 else if (str == "print")
141 {
142     cout << dynArr << endl;
143 }
144 }
145
146 return 0;

```

```

PS C:\Users\Маркіян\Desktop> cd C:\Users\Маркіян\Desktop
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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 template <typename T = int>
6 class dynamicArray
7 {
8 private:
9     T *data;
10    int size;
11    int capacity;
12
13    void resize(int newSize)
14    {
15        T *newData = new T[newSize];
16
17        for (int i = 0; i < size; i++)
18        {
19            newData[i] = data[i];
20        }
21        delete[] data;
22        data = newData;
23        capacity = newSize;
24    }
25
26 public:
27    dynamicArray()
28    {
29        size = 0;
30        capacity = 1;
31        data = new T[capacity];
32    }
33    ~dynamicArray()
34    {
35        delete[] data;
36    }
37    void Insert(int index, int N, T *arr)
38    {
39
40        while (size + N >= capacity)
41            capacity *= 2;
42        resize(capacity);
43
44        for (int i = size - 1; i >= index; i--)
45        {
46            data[i + N] = data[i];
47        }
48        for (int i = 0; i < N; i++)
49        {
```

```
51         }
52         size = size + N;
53         if (size == capacity)
54         {
55             capacity *= 2;
56             resize(capacity);
57         }
58     }
59    void erase(int index, int n)
60    {
61        for (int i = index; i < size - n; i++)
62        {
63            data[i] = data[i + n];
64        }
65        size = size - n;
66    }
67    int getSize()
68    {
69        return size;
70    }
71    int getCapacity()
72    {
73        return capacity;
74    }
75    T &operator[](int index)
76    {
77        return data[index];
78    }
79    friend ostream &operator<<(ostream &output, const dynamicArray &arr)
80    {
81        for (int i = 0; i < arr.size; i++)
82        {
83            output << arr.data[i] << ' ';
84        }
85        return output;
86    }
87 };
88
89 int main()
90 {
91     dynamicArray<int> dynArr;
92     int Q;
93     cin >> Q;
94     string str;
95
96     for (int i = 0; i < Q; i++)
```



```

101     int index;
102     cin >> index;
103     int N;
104     cin >> N;
105     int arr[N];
106     for (int i = 0; i < N; i++)
107     {
108         cin >> arr[i];
109     }
110     dynArr.insert(index, N, arr);
111 }
112 else if (str == "erase")
113 {
114     int index;
115     cin >> index;
116     int n;
117     cin >> n;
118     dynArr.erase(index, n);
119 }
120 else if (str == "size")
121 {
122     cout << dynArr.getSize() << endl;
123 }
124 else if (str == "capacity")
125 {
126     cout << dynArr.getCapacity() << endl;
127 }
128 else if (str == "get")
129 {
130     int index;
131     cin >> index;
132     cout << dynArr[index] << endl;
133 }
134 else if (str == "set")
135 {
136     int index;
137     cin >> index;
138     int newElement;
139     cin >> newElement;
140     dynArr[index] = newElement;
141 }
142 else if (str == "print")
143 {
144     cout << dynArr << endl;
145 }
146 }

```

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

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct Node
6  {
7      int data;
8      Node *next;
9  };
10 Node *pushBack(Node *head, int element)
11 {
12     Node *newEl = new Node{element, nullptr};
13
14     if (head == nullptr)
15     {
16         head = newEl;
17     }
18     else
19     {
20         Node *current = head;
21         while (current->next != nullptr)
22         {
23             current = current->next;
24         }
25         current->next = newEl;
26     }
27     return head;
28 };
29 void printList(Node *head)
30 {
31     if (head == nullptr)
32     {
33         cout << "List is empty" << endl;
34     }
35     Node *current = head;
36
37     while (current != nullptr)
38     {
39         cout << current->data << " ";
40         current = current->next;
41     }
42 }
43 void printNum(Node *head)
44 {
45     if (head == nullptr)
46     {
47         cout << "List is empty" << endl;
48     }
49     Node *current = head;
```

```
49     Node *current = head;
50
51     while (current != nullptr)
52     {
53         cout << current->data;
54         current = current->next;
55     }
56 }
57 Node *reverse(Node *head)
58 {
59     if (head == nullptr)
60     {
61         cout << "List is empty" << endl;
62         return 0;
63     }
64     else
65     {
66         Node *current = head;
67         Node *next = nullptr;
68         Node *temp = nullptr;
69         while (current != nullptr)
70         {
71             next = current->next;
72             current->next = temp;
73             temp = current;
74             current = next;
75         }
76         return temp;
77     }
78 };
79 bool compare(Node *headF, Node *headS)
80 {
81
82     Node *current1 = headF;
83     Node *current2 = headS;
84
85     while (current1 != nullptr && current2 != nullptr)
86     {
87         if (current1->data != current2->data)
88         {
89             return false;
90         };
91         current1 = current1->next;
92         current2 = current2->next;
93     }
94     if (current1 != nullptr || current2 != nullptr)
95     {
96         return false;
```

```

93     }
94     if (current1 != nullptr || current2 != nullptr)
95     {
96         return false;
97     }
98     else
99     {
100         return true;
101     }
102 };
103
104 Node *addNum(Node *num1, Node *num2)
105 {
106     Node *current1 = num1;
107     Node *current2 = num2;
108     Node *sum = nullptr;
109     int tempSum = 0;
110     int over = 0;
111
112     while (current2 != nullptr)
113     {
114         tempSum = current1->data + current2->data + over;
115
116         if (tempSum <= 9)
117         {
118             sum = pushBack(sum, tempSum);
119             over = 0;
120         }
121         if (tempSum > 9)
122         {
123             sum = pushBack(sum, tempSum % 10);
124             over = tempSum / 10;
125         }
126         current1 = current1->next;
127         current2 = current2->next;
128     }
129     if (current1 != nullptr)
130     {
131         while (current1 != nullptr)
132         {
133             tempSum = current1->data + over;
134             if (tempSum <= 9)
135             {
136                 sum = pushBack(sum, tempSum);
137                 over = 0;
138             }
139             if (tempSum > 9)

```

```

139         if (tempSum > 9)
140         {
141             sum = pushBack(sum, tempSum % 10);
142             over = tempSum / 10;
143         }
144         current1 = current1->next;
145     }
146 }
147 else if (over != 0)
148 {
149     sum = pushBack(sum, over);
150 }
151 return sum;
152 };
153
154 struct treeNode
155 {
156     int data;
157     treeNode *left;
158     treeNode *right;
159
160     treeNode(int value) : data(value), left(nullptr), right(nullptr) {};
161 };
162
163 treeNode *insert(treeNode *node, int value)
164 {
165     if (node == nullptr)
166     {
167         return new treeNode(value);
168     }
169     else
170     {
171         if (value > node->data)
172         {
173             node->right = insert(node->right, value);
174         }
175         else
176         {
177             node->left = insert(node->left, value);
178         }
179         return node;
180     }
181 };
182 void printTree(treeNode *node)
183 {

```

```

187     }
188     cout << node->data << " ";
189     printTree(node->left);
190     printTree(node->right);
191 }
192
193 treeNode *mirror(treeNode *node)
194 {
195     if (!node)
196         return 0;
197
198     swap(node->left, node->right);
199     mirror(node->left);
200     mirror(node->right);
201     return node;
202 };
203
204 treeNode *treeSum(treeNode *node)
205 {
206     if ((node == nullptr) || ((node->left == nullptr) && (node->right == nullptr)))
207         return nullptr;
208
209     treeSum(node->left);
210     treeSum(node->right);
211
212     node->data = 0;
213     if (node->right != nullptr)
214     {
215         node->data += node->right->data;
216     }
217     if (node->left != nullptr)
218     {
219         node->data += node->left->data;
220     }
221     return node;
222 }
223
224
225 int main()
226 {
227     cout << "TASK 1" << endl;
228     cout << endl;
229     Node *head = nullptr;
230
231     for (int i = 1; i < 16; i++)
232     {
233         head = pushBack(head, i);
234     }
235     cout << "First list:" << endl;
236     printList(head);
237     Node *head1 = reverse(head);
238     cout << endl;
239     cout << "Reversed List:" << endl;
240     printList(head1);
241
242     cout << endl;
243     cout << endl;
244     cout << "TASK 2" << endl;
245     cout << endl;
246
247     Node *headF = nullptr;
248     Node *headS = nullptr;
249
250     headF = pushBack(headF, 1);
251     headF = pushBack(headF, 5);
252     headF = pushBack(headF, 3);
253     headF = pushBack(headF, 1);
254     headF = pushBack(headF, 5);
255     headF = pushBack(headF, 7);
256
257     headS = pushBack(headS, 1);
258     headS = pushBack(headS, 5);
259     headS = pushBack(headS, 3);
260     headS = pushBack(headS, 100);
261     headS = pushBack(headS, 5);
262     headS = pushBack(headS, 7);
263
264     if (compare(headF, headS))
265     {
266         cout << "List are identical" << endl;
267     }
268     else
269     {
270         cout << "List are different" << endl;
271     }
272
273     cout << endl;
274     cout << "TASK 3" << endl;

```

```

220 {
227     cout << "TASK 1" << endl;
228     cout << endl;
229     Node *head = nullptr;
230
231     for (int i = 1; i < 16; i++)
232     {
233         head = pushBack(head, i);
234     }
235     cout << "First list:" << endl;
236     printList(head);
237     Node *head1 = reverse(head);
238     cout << endl;
239     cout << "Reversed List:" << endl;
240     printList(head1);
241
242     cout << endl;
243     cout << endl;
244     cout << "TASK 2" << endl;
245     cout << endl;
246
247     Node *headF = nullptr;
248     Node *headS = nullptr;
249
250     headF = pushBack(headF, 1);
251     headF = pushBack(headF, 5);
252     headF = pushBack(headF, 3);
253     headF = pushBack(headF, 1);
254     headF = pushBack(headF, 5);
255     headF = pushBack(headF, 7);
256
257     headS = pushBack(headS, 1);
258     headS = pushBack(headS, 5);
259     headS = pushBack(headS, 3);
260     headS = pushBack(headS, 100);
261     headS = pushBack(headS, 5);
262     headS = pushBack(headS, 7);
263
264     if (compare(headF, headS))
265     {
266         cout << "List are identical" << endl;
267     }
268     else
269     {
270         cout << "List are different" << endl;
271     }
272
273     cout << endl;
274     cout << "TASK 3" << 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(newTree);

```

#### 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

1325789051324789045178941257895417893478943587943789345789

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 MiB

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 хвилини тому Зеник побіг до магазину вибрати 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 надуті 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 надуває кожні  $z_i$  кульок, перепочиває  $y_i$  секунд.

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

#### Вхідні дані

У першому рядку дано два цілих числа  $m$  і  $n$  — кількість кульок, які потрібно надуті, 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.

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  int main()
6  {
7      int m, n;
8      cin >> m >> n;
9
10     int t, z, y;
11     int left = 0, right = 2000 * 2000;
12     int result = right;
13
14     int boys[2000][3];
15
16     for (int i = 0; i < n; i++)
17     {
18         cin >> boys[i][0] >> boys[i][1] >> boys[i][2];
19     }
20
21     while (left <= right)
22     {
23         int mid = (left + right) / 2;
24         int totalBalloons = 0;
25
26         for (int i = 0; i < n; i++)
27         {
28             t = boys[i][0];
29             z = boys[i][1];
30             y = boys[i][2];
31
32             int cycles = mid / (z * t + y);
33
34             int balloons = cycles * (z * t + y);
35             totalBalloons += balloons;
36         }
37
38         if (totalBalloons < m)
39             left = mid + 1;
40         else
41             right = mid;
42     }
43
44     cout << right << endl;
45     return 0;
46 }

```

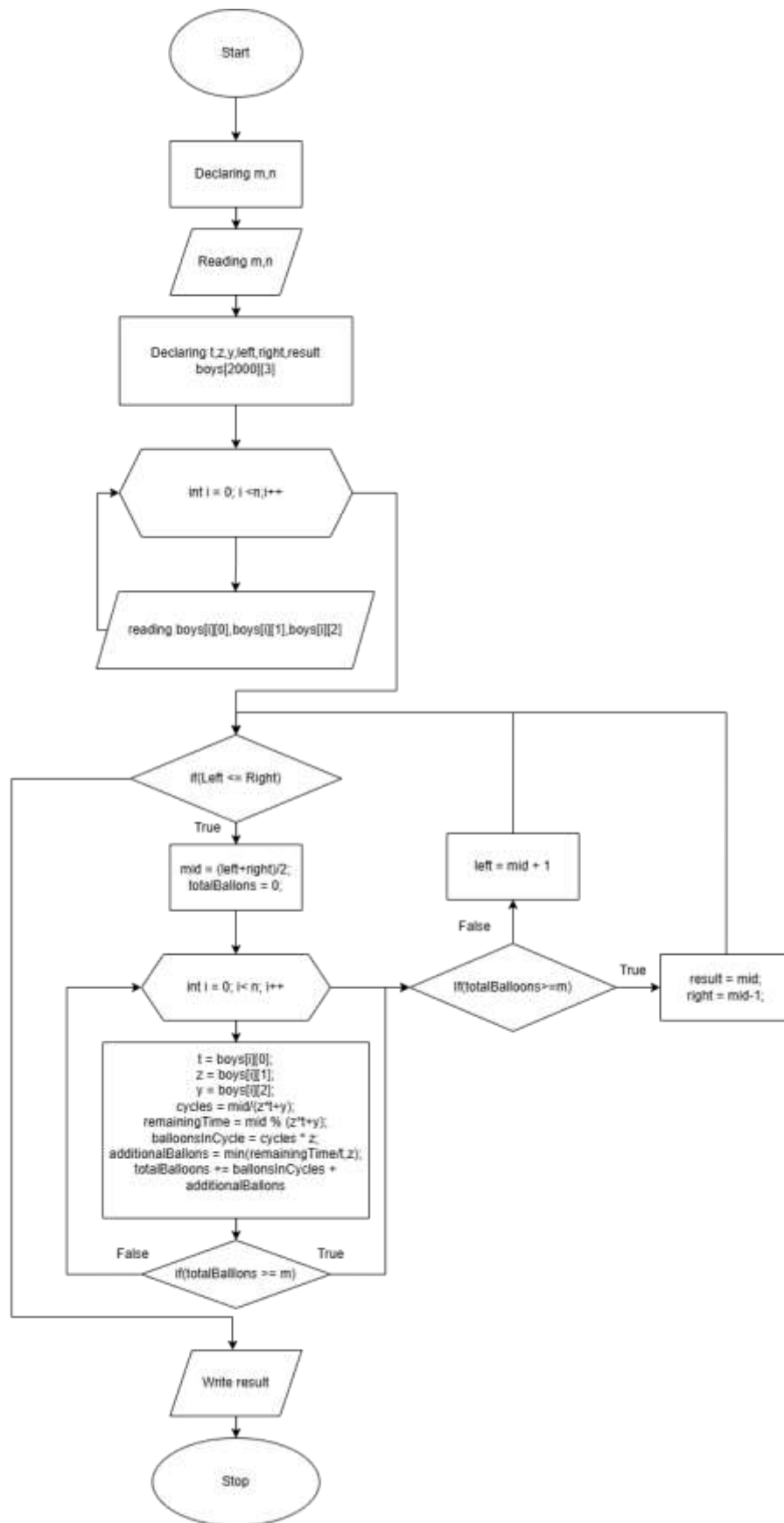
```

32     int cycles = mid / (z * t + y);
33
34     int remainingTime = mid % (z * t + y);
35
36     int balloonsInCycles = cycles * z;
37     int additionalBalloons = min(remainingTime / t, z);
38
39     totalBalloons += balloonsInCycles + additionalBalloons;
40
41     if (totalBalloons >= m)
42     {
43         break;
44     }
45 }
46
47 if (totalBalloons >= m)
48 {
49     result = mid;
50     right = mid - 1;
51 }
52 else
53 {
54     left = mid + 1;
55 }
56 }
57
58 cout << result << endl;
59
60 return 0;
61 }

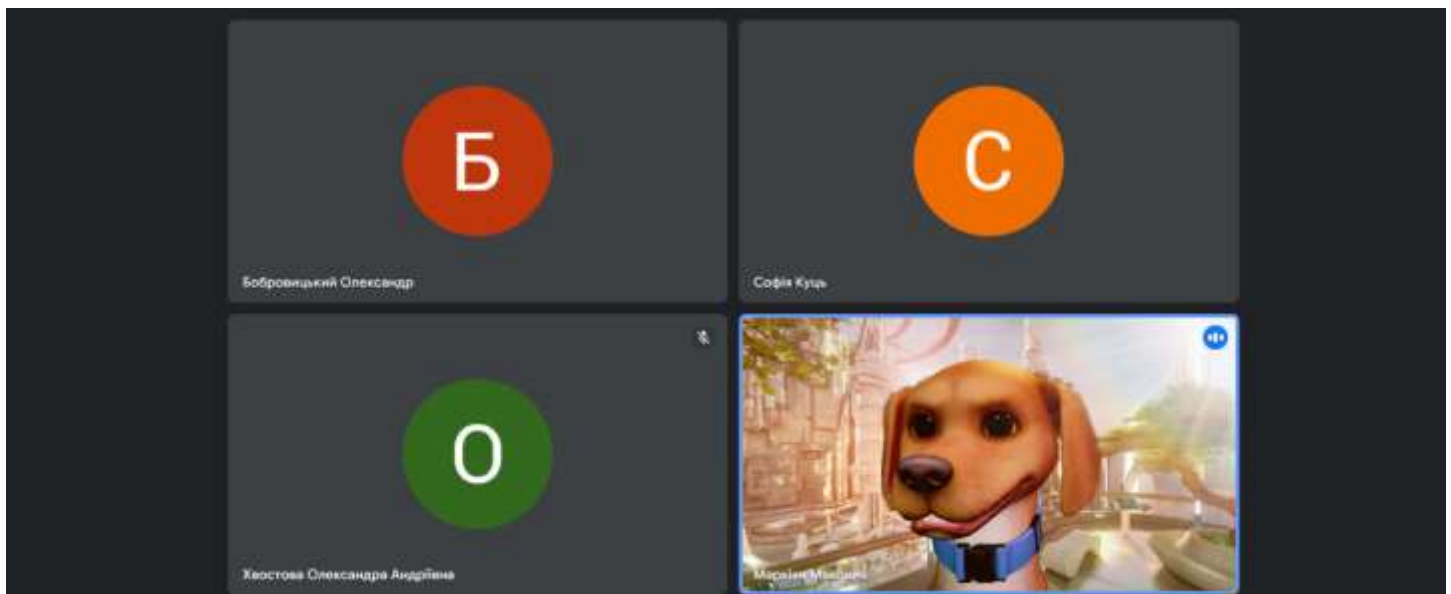
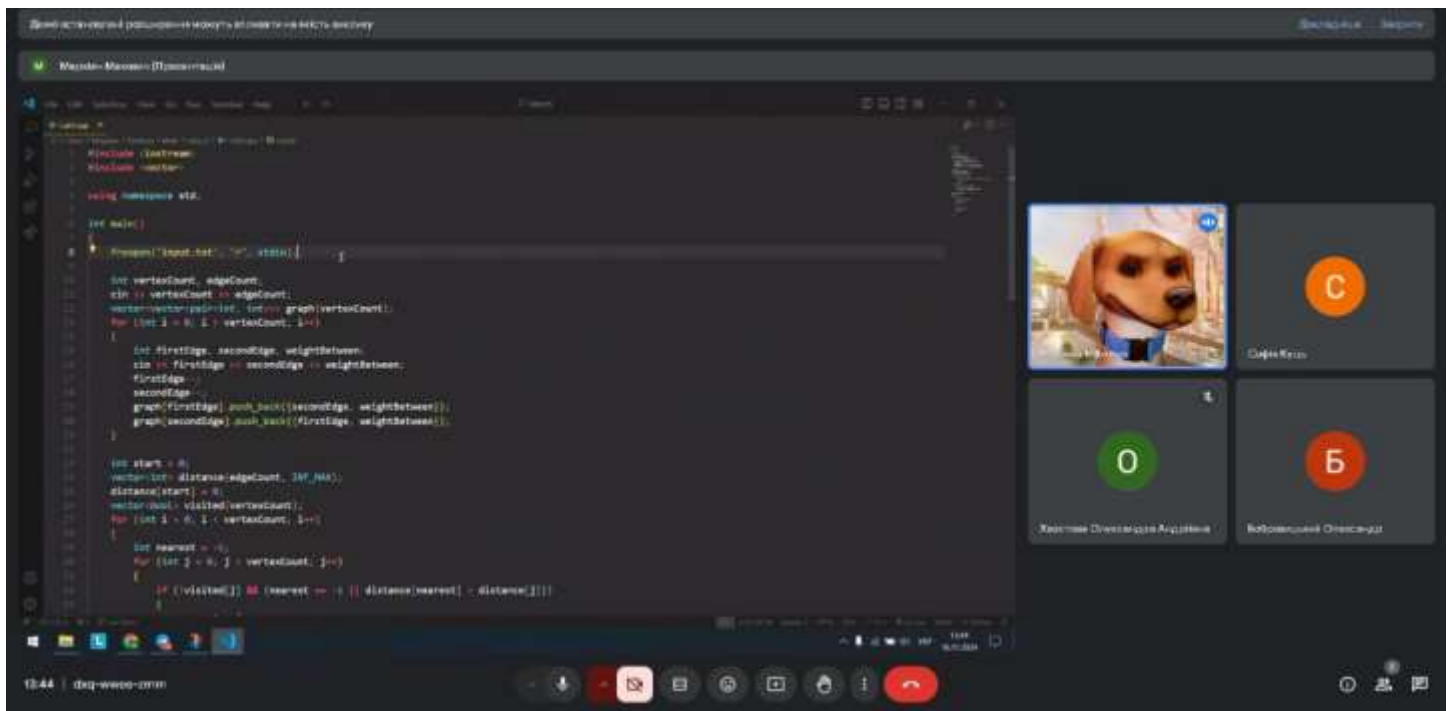
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
21 хвилину тому	C++ 23	Зараховано	0.004	1.172	<a href="#">Перегляд</a>





## 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.