



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(ла):

Студент(ка) групи ІІІ-12

Соснило Богдан Тарасович

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

Мета роботи: Ознайомитися з основними динамічними структурами даних (черга, стек, списки, дерево), вивчити їхні особливості та застосування в алгоритмах. Розробити і реалізувати алгоритми обробки цих структур для ефективного виконання різних задач.

Теоретичні відомості

1. Перенавантаження операторі виводу

<https://acode.com.ua/urok-141-perevantazhennya-operatoriv-vvodu-i-vyvodu/#toc-0>

2. Класи

<https://acode.com.ua/urok-121-klassy-ob-yekty-i-metody/#toc-1>
<https://acode.com.ua/urok-183-shablony-klassiv/>

3. Черга

<https://www.bestprog.net/uk/2019/09/26/c-queue-general-concepts-ways-to-implement-the-queue-implementing-a-queue-as-a-dynamic-array-ua/>

4. Стек

<https://www.bestprog.net/uk/2019/09/18/c-the-concept-of-stack-operations-on-the-stack-an-example-implementation-of-the-stack-as-a-dynamic-array-ua/>

5. Списки

<https://codelessons.dev/ru/spisok-list-v-s-polnyj-material/>
https://www.youtube.com/watch?v=-25REjF_atI

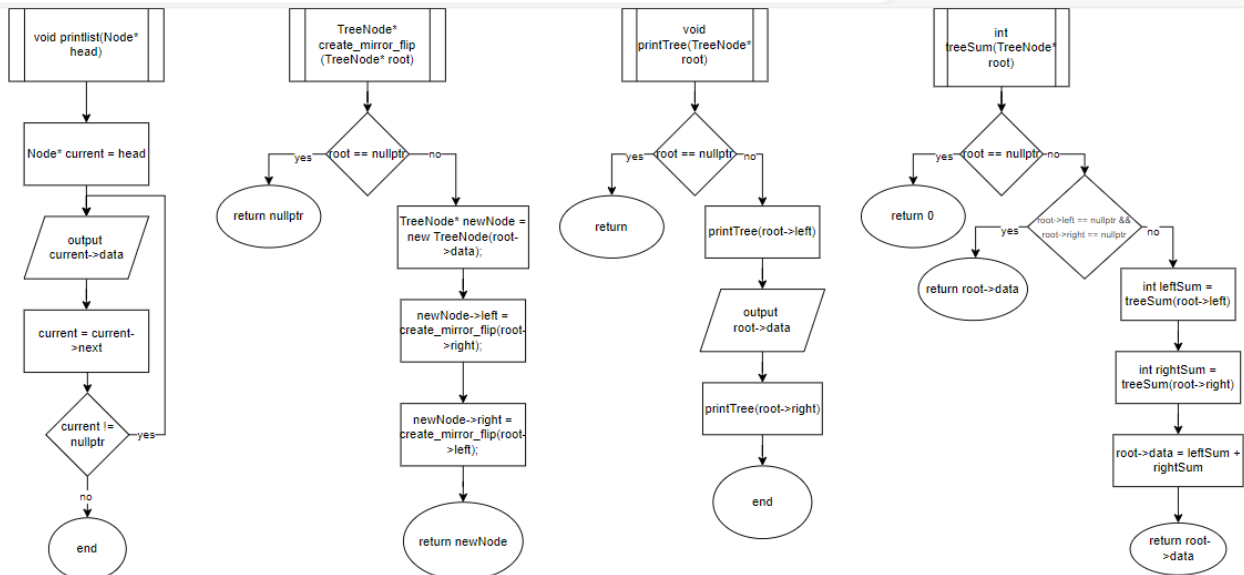
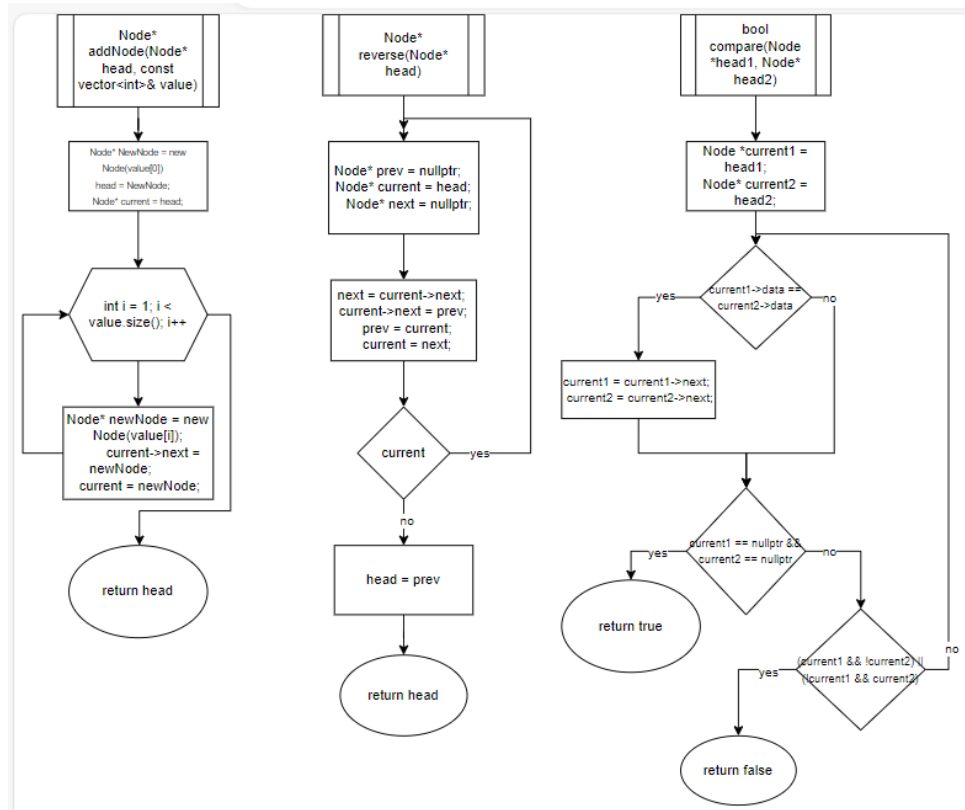
6. Дерева

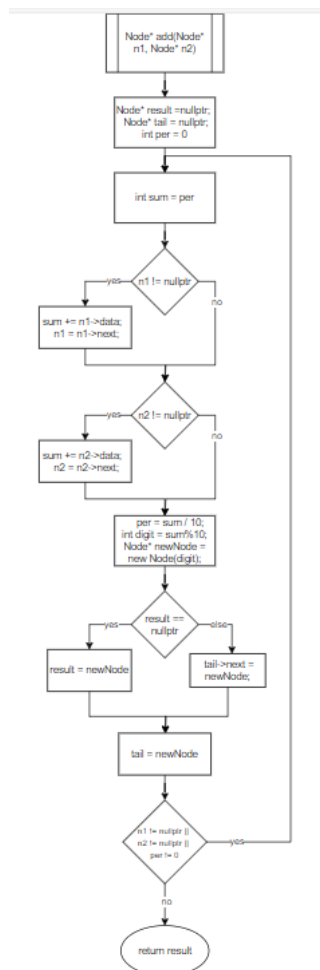
<https://www.youtube.com/watch?v=qBFzNW0ALxQ>
<https://www.geeksforgeeks.org/binary-tree-data-structure/>

Виконання роботи

Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7) (2 години)

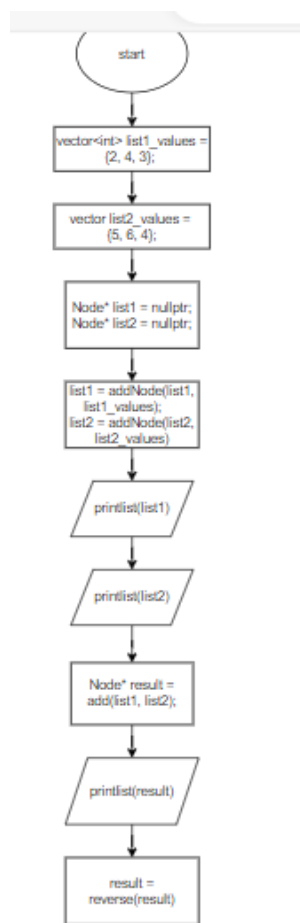
Функції:



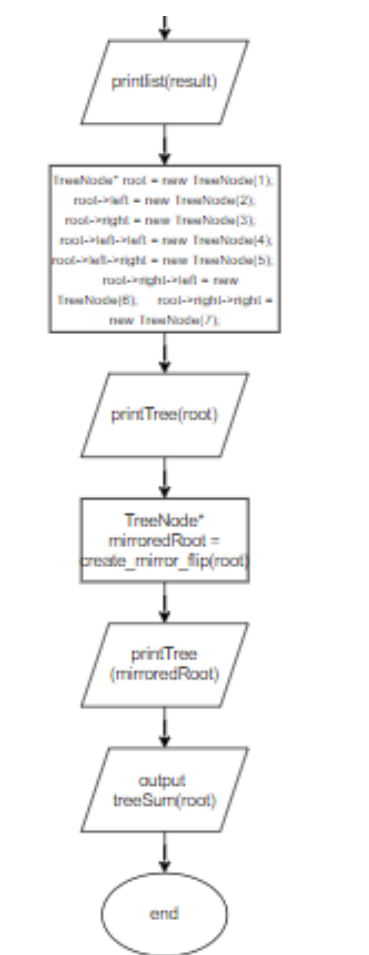


Основна програма:

1 частина



2 частина



Task 3 - Lab# programming: VNS Lab 10 (1 година)

4. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим номером, додати K елементів, починаючи із заданого номера;

```
1 #include <iostream>
2 #include <vector>
3 #include <fstream>
4 using namespace std;
5
6 struct Node{
7     int data;
8     Node* next;
9     Node(int value): data(value), next(nullptr){}
10 };
11
12 Node* addNode(Node* head, const vector<int>& value){
13     Node* NewNode = new Node(value[0]);
14     head = NewNode;
15     Node* current = head;
16     for (int i = 1; i < value.size(); i++) {
17         Node* newNode = new Node(value[i]);
18         current->next = newNode;
19         current = newNode;
20     }
21     return head;
22 }
23
24 void printlist(Node* head){
25     Node* current = head;
26     while(current != nullptr){
27         cout << current->data;
28         current = current->next;
29     }
30 }
31
32 void printFile(string file) {
33     ifstream inFile(file);
34     if (!inFile) {
35         cerr << "Error: Unable to open file " << file << endl;
36         return;
37     }
38
39     string content;
40     getline(inFile, content);
41     inFile.close();
42
43     if (content.empty()) {
44         cout << "Список порожній" << endl;
45     } else {
46         cout << "Вміст файлу: " << content << endl;
47     }
48 }
49
50 void addtoFile(string file, vector<int> node){
51     ofstream outFile(file);
52
53     if (!outFile) {
54         cerr << "Error: Unable to open file " << file << endl;
55         return;
56     }
57
58     for (int n : node) {
59         outFile << n << " ";
60     }
61     outFile.close();
62 }
63
64 void clearFile(string file) {
65     ofstream outFile(file, ofstream::trunc);
66 }
```

```
68     if (!outFile) {
69         cerr << "Error: Unable to open file " << file << endl;
70         return;
71     }
72     outFile.close();
73 }
74
75 Node* restoreFromFile(string file) {
76     ifstream inFile(file);
77     if (!inFile) {
78         cerr << "Error: Unable to open file " << file << endl;
79         return {};
80     }
81     Node* head = nullptr;
82     vector<int> node;
83     int value;
84     while (inFile >> value) {
85         node.push_back(value);
86     }
87
88     addNode(head, node);
89     inFile.close();
90
91     return head;
92 }
93
94 void clearList(Node*& head) {
95     while (head) {
96         Node* temp = head;
97         head = head->next;
98         delete temp;
99     }
100 }
101
102 void deleteAtPosition(Node*& head, int position) {
103     if (!head || position < 1) {
104         cout << "Некоректний номер елемента" << endl;
105         return;
106     }
107     if (position == 1) {
108         Node* temp = head;
109         head = head->next;
110         delete temp;
111         return;
112     }
113     Node* current = head;
114     for (int i = 1; current && i < position - 1; ++i) {
115         current = current->next;
116     }
117     if (!current || !current->next) {
118         cout << "Елемент із заданим номером не існує" << endl;
119         return;
120     }
121     Node* temp = current->next;
122     current->next = temp->next;
123     delete temp;
124 }
125
126 void addKElements(Node*& head, int position, int K) {
127     if (position < 1 || K < 1) {
128         cout << "Некоректний номер або кількість елементів" << endl;
129         return;
130     }
131     Node* current = head;
132
133     for (int i = 1; current && i < position; ++i) {
```

```

133     for (int i = 1; current && i < position; ++i) {
134         current = current->next;
135     }
136
137     if (!current) {
138         cout << "Позиція поза списком" << endl;
139         return;
140     }
141
142     for (int i = 0; i < k; ++i) {
143         int value;
144         cout << "Введіть значення для нового елемента: ";
145         cin >> value;
146
147         Node* newNode = new Node(value);
148         newNode->next = current->next;
149         current->next = newNode;
150         current = newNode;
151     }
152 }
153
154
155 int main() {
156
157     string fileName = "list.txt";
158     Node* head = nullptr;
159     int k;
160     vector<int> vec;
161     cout << "Введіть к-сть елементів у списку: ";
162     cin >> k;
163     for(int i = 0; i < k; ++i){
164         cin >> vec[i];
165     }
166     head = addNode(head, vec);
167     cout << "Початковий список: ";
168     printlist(head);
169     cout << endl;
170     deleteAtPosition(head, 2);
171     cout << "Після видалення елемента з позиції 2: ";
172     printlist(head);
173     cout << endl;
174     addKElements(head, 3, 3);
175     cout << "Після додавання 3 елементів із значенням від 50: ";
176     printlist(head);
177     cout << endl;
178     addToFile(fileName, vec);
179
180     clearList(head);
181     cout << "Після знищення списку: ";
182     printlist(head);
183     cout << endl;
184     head = restoreFromFile(fileName);
185     cout << "Після відновлення списку з файлу: ";
186     printFile(fileName);
187     cout << endl;
188     clearList(head);
189     cout << "Після остаточного знищення списку: ";
190     printlist(head);
191     cout << endl;
192     return 0;
193 }

```

```

Введіть к-сть елементів у списку: 5
1 2 3 4 5
Початковий список: 12345
Після видалення елемента з позиції 2: 1345
Введіть значення для нового елемента: 7
Введіть значення для нового елемента: 8
Введіть значення для нового елемента: 9
Після додавання 3 елементів із значенням від 50: 1347895
Після знищення списку:
Після відновлення списку з файлу: Вміст файлу: 1 2 3 4 5

Після остаточного знищення списку:

```

Task 4 - Lab# programming: Algotester Lab 5(30 хв)

У вас є карта гори розміром $N \times M$.

Також ви знаєте координати $\{x, y\}$, у яких знаходиться вершина гори.

Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

Клітинки які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

Вхідні дані

У першому рядку 2 числа N та M - розміри карти

у другому рядку 2 числа x та y - координати піку гори

Вихідні дані

N рядків по M елементів в рядку через пробіл - висоти карти.

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5
6  int main(){
7
8  int N, M;
9  cin >> N >> M;
10 int matrix[N][M];
11 int x,y;
12 cin >>x >> y;
13 x--;
14 y--;
15 for(int i = 0; i<N; i++){
16     for(int j = 0; j<M;j++){
17         int distance = abs(i-x)+abs(j-y);
18         matrix[i][j] = max(x,N-x-1) + max(y, M-y-1) - distance;
19     }
20 }
21 }
22
23 for (int i = 0; i<N; i++) {
24     for (int j = 0;j<M; j++) {
25         cout << matrix[i][j] << " ";
26     }
27     cout << endl;
28 }
29
30
31 return 0;
32 }
```

```
(4,4) (0,0)
3 4
3 3
0 1 2 1
1 2 3 2
2 3 4 3
```

Task 5 - Lab# programming: Algotester Lab 7-8(1) (1.5 години)

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  struct DynamicMas {
6  private:
7      int cap;
8      int size;
9      int* mas;
10
11     void resize(int newcap) {
12         int* mas1 = new int[newcap];
13         for (int i = 0; i < size; i++) {
14             mas1[i] = mas[i];
15         }
16         delete[] mas;
17         mas = mas1;
18         cap = newcap;
19     }
20
21 public:
22     DynamicMas() {
23         size = 0;
24         cap = 1;
25         mas = new int[cap];
26     }
27
28     ~DynamicMas() {
29         delete[] mas;
30     }
31
32     void getsize() {
33         cout << size << endl;
34     }
35
36     void getcapacity() {
37         cout << cap << endl;
38     }
39
40     void insert(int a, int N, int* numb) {
41         if (size + N >= cap) {
42             while (cap <= size + N) {
43                 cap *= 2;
44             }
45             resize(cap);
46         }
47         for (int i = size - 1; i >= a; i--) {
48             mas[i + N] = mas[i];
49         }
50
51         for (int i = 0; i < N; i++) {
52             mas[a + i] = numb[i];
53         }
54         size += N;
55
56     }
57
58     void erase(int a, int n) {
59         for (int i = a; i < size - n; i++) {
60             mas[i] = mas[i + n];
61         }
62         size -= n;
63     }
64
65     int& operator[](int a) {
66         return mas[a];
67     }
68
69     friend ostream& operator<<(ostream& out, const DynamicMas& arr) {
70         for (int i = 0; i < arr.size; i++) {
71             out << arr.mas[i] << " ";
72         }
73         return out;
74     }
75 };
76
77 int main() {
78     int Q;
79     cin >> Q;
80     DynamicMas vec;
81
82     for (int i = 0; i < Q; i++) {
83         string com;
84         cin >> com;
85         if (com == "size") {
86             vec.getsize();
87         } else if (com == "capacity") {
88             vec.getcapacity();
89         } else if (com == "insert") {
90             int a, b;
91             cin >> a >> b;
92             int* elem = new int[b];
93             for (int j = 0; j < b; j++) {
94                 cin >> elem[j];
95             }
96             vec.insert(a, b, elem);
97             delete[] elem;
98         } else if (com == "erase") {
99             int a, n;
100             cin >> a >> n;
101             vec.erase(a, n);
102         } else if (com == "get") {
103             int a;
104             cin >> a;
105             cout << vec[a] << endl;
106         } else if (com == "set") {
107             int index, value;
108             cin >> index >> value;
109             vec[index] = value;
110         } else if (com == "print") {
111             cout << vec << endl;
112         }
113     }
114
115     return 0;
116 }

```



```

12
size
0
capacity
1
insert 0 2
100 100
size
2
capacity
4
insert 0 2
102 102
size
4
capacity
8
insert 0 2
103 103
size
6
capacity
8
print
103 103 102 102 100 100

```

З використанням шаблону класу:

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  template <typename shablon>
6  class DynamicMas {
7  private:
8      int cap;
9      int size;
10     shablon* mas;
11
12     void resize(int newcap) {
13         shablon* newArr = new shablon[newcap];
14         for (int i = 0; i < size; i++) {
15             newArr[i] = mas[i];
16         }
17         delete[] mas;
18         mas = newArr;
19         cap = newcap;
20     }
21
22 public:
23     DynamicMas() {
24         size = 0;
25         cap = 1;
26         mas = new shablon[cap];
27     }
28
29     ~DynamicMas() {
30         delete[] mas;
31     }
32
33     void getSize() const {
34         cout << size << endl;
35     }
36
37     void getCapacity() const {
38         cout << cap << endl;
39     }
40
41     void insert(int pos, int count, shablon* elements) {
42         if (size + count >= cap) {
43             while (cap <= size + count) {
44                 cap *= 2;
45             }
46             resize(cap);
47         }
48         for (int i = size - 1; i >= pos; i--) {
49             mas[i + count] = mas[i];
50         }
51
52         for (int i = 0; i < count; i++) {
53             mas[pos + i] = elements[i];
54         }
55         size += count;
56
57     }
58
59     void erase(int pos, int count) {
60         for (int i = pos; i < size - count; i++) {
61             mas[i] = mas[i + count];
62         }
63         size -= count;
64     }
65
66     shablon& operator[](int index) {
67         return mas[index];
68     }
69
70     friend ostream& operator<<(ostream& out, const DynamicMas<shablon>& dynamicMas) {
71         for (int i = 0; i < dynamicMas.size; i++) {
72             out << dynamicMas.mas[i] << " ";
73         }
74         return out;
75     };
76
77 int main() {
78     int Q;
79     cin >> Q;
80     DynamicMas<int> vec;
81
82     for (int i = 0; i < Q; i++) {
83         string command;
84         cin >> command;
85         if (command == "size") {
86             vec.getSize();
87         } else if (command == "capacity") {
88             vec.getCapacity();
89         } else if (command == "insert") {
90             int pos, count;
91             cin >> pos >> count;
92             int* elements = new int[count];
93             for (int j = 0; j < count; j++) {
94                 cin >> elements[j];
95             }
96             vec.insert(pos, count, elements);
97             delete[] elements;
98         } else if (command == "erase") {
99             int pos, count;
100             cin >> pos >> count;
101             vec.erase(pos, count);
102         } else if (command == "get") {
103             int index;
104             cin >> index;
105             cout << vec[index] << endl;
106         } else if (command == "set") {
107             int index, value;
108             cin >> index >> value;
109             vec[index] = value;
110         } else if (command == "print") {
111             cout << vec << endl;
112         }
113     }
114
115     return 0;
116 }

```

```
12
size
0
capacity
1
insert 0 2
100 100
size
2
capacity
4
insert 0 2
102 102
size
4
capacity
8
insert 0 2
103 103
size
6
capacity
8
print
103 103 102 102 100 100
```

7 годин тому	C++ 23	Зараховано	0.006	1.383	Перегляд
7 годин тому	C++ 23	Зараховано	0.006	1.285	Перегляд

Task 6 – Practice# programming: Class Practice Task(2 години)

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  struct Node {
7      int data;
8      Node* next;
9      Node(int value): data(value), next(nullptr) {}
10 };
11
12 Node* addNode(Node* head, const vector<int>& value) {
13     if (value.empty()) return nullptr;
14     Node* newNode = new Node(value[0]);
15     head = newNode;
16     Node* current = head;
17     for (int i = 1; i < value.size(); i++) {
18         Node* temp = new Node(value[i]);
19         current->next = temp;
20         current = temp;
21     }
22     return head;
23 }
24
25 Node* reverse(Node* head) {
26     Node* prev = nullptr;
27     Node* current = head;
28     Node* next = nullptr;
29     while (current) {
30         next = current->next;
31         current->next = prev;
32         prev = current;
33         current = next;
34     }
35     head = prev;
36     return head;
37 }
38
39 bool compare(Node* head1, Node* head2) {
40     Node* current1 = head1;
41     Node* current2 = head2;
42     while (current1 && current2) {
43         if (current1->data != current2->data) {
44             return false;
45         }
46         current1 = current1->next;
47         current2 = current2->next;
48     }
49     return current1 == nullptr && current2 == nullptr;
50 }
51
52 Node* add(Node* n1, Node* n2) {
53     Node* result = nullptr;
54     Node* tail = nullptr;
55     int per = 0;
56
57     while (n1 != nullptr || n2 != nullptr || per != 0) {
58         int sum = per;
59
60         if (n1 != nullptr) {
61             sum += n1->data;
62             n1 = n1->next;
63         }
64
65         if (n2 != nullptr) {
66             sum += n2->data;
67             n2 = n2->next;

```

```

68     }
69
70     per = sum / 10;
71     int digit = sum % 10;
72
73     Node* newNode = new Node(digit);
74
75     if (result == nullptr) {
76         result = newNode;
77     } else {
78         tail->next = newNode;
79     }
80     tail = newNode; // Необхідно оновити tail
81 }
82
83 return result;
84 }
85
86 void printlist(Node* head) {
87     Node* current = head;
88     while (current != nullptr) {
89         cout << current->data << " ";
90         current = current->next;
91     }
92     cout << endl;
93 }
94
95 struct TreeNode {
96     int data;
97     TreeNode* left;
98     TreeNode* right;
99     TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}
100 };
101
102 TreeNode* create_mirror_flip(TreeNode* root) {
103     if (root == nullptr) {
104         return nullptr;
105     }
106
107     TreeNode* newNode = new TreeNode(root->data);
108     newNode->left = create_mirror_flip(root->right);
109     newNode->right = create_mirror_flip(root->left);
110
111     return newNode;
112 }
113
114 void printTree(TreeNode* root) {
115     if (root == nullptr) {
116         return;
117     }
118     printTree(root->left);
119     cout << root->data << " ";
120     printTree(root->right);
121 }
122
123 int treeSum(TreeNode* root) {
124     if (root == nullptr) {
125         return 0;
126     }
127
128     int leftSum = treeSum(root->left);
129     int rightSum = treeSum(root->right);
130
131

```

```

130     int rightSum = treeSum(root->right);
131
132     return root->data + leftSum + rightSum;
133 }
134
135 int main() {
136     vector<int> list1_values = {2, 4, 3};
137     vector<int> list2_values = {5, 6, 4};
138
139     Node* list1 = nullptr;
140     Node* list2 = nullptr;
141
142     list1 = addNode(list1, list1_values);
143     list2 = addNode(list2, list2_values);
144
145     cout << "List 1: ";
146     printlist(list1);
147
148     cout << "List 2: ";
149     printlist(list2);
150
151     Node* result = add(list1, list2);
152
153     cout << "Sum of List 1 and List 2: ";
154     printlist(result);
155
156     result = reverse(result);
157
158     cout << "Reversed sum: ";
159     printlist(result);
160
161     TreeNode* root = new TreeNode(1);
162     root->left = new TreeNode(2);
163     root->right = new TreeNode(3);
164     root->left->left = new TreeNode(4);
165     root->left->right = new TreeNode(5);
166     root->right->left = new TreeNode(6);
167     root->right->right = new TreeNode(7);
168
169     cout << "Original Tree: ";
170     printTree(root);
171     cout << endl;
172
173     TreeNode* mirroredRoot = create_mirror_flip(root);
174
175     cout << "Mirrored Tree: ";
176     printTree(mirroredRoot);
177     cout << endl;
178
179     cout << "Sum of Tree Nodes: " << treeSum(root) << endl;
180
181     return 0;
182 }

```

```

List 1: 2 4 3
List 2: 5 6 4
Sum of List 1 and List 2: 7 0 8
Reversed sum: 8 0 7
Original Tree: 4 2 5 1 6 3 7
Mirrored Tree: 7 3 6 1 5 2 4
Sum of Tree Nodes: 28

```

Task 7 - Practice# programming: Self Practice Task

```
1  #include <iostream>
2  #include <string>
3  #include <algorithm>
4
5  using namespace std;
6
7  int main() {
8      string s;
9      cin >> s;
10
11     int n = s.length();
12
13     int changes1 = 0, changes2 = 0;
14
15     for (int i = 0; i < n; ++i) {
16         if (i % 2 == 0) {
17             if (s[i] != 'B') {
18                 changes1++;
19             }
20         } else {
21             if (s[i] != 'Y') {
22                 changes1++;
23             }
24         }
25     }
26
27     for (int i = 0; i < n; ++i) {
28         if (i % 2 == 0) {
29             if (s[i] != 'Y') {
30                 changes2++;
31             }
32         } else {
33             if (s[i] != 'B') {
34                 changes2++;
35             }
36         }
37     }
38
39     cout << min(changes1, changes2) << endl;
40
41     return 0;
42 }
```

```
YYBBB
2
```

Зустрічі з командою

З командою зустрічалися двічі, на зустрічах обговорювали питання та прогрес по епіку.

zoom Workplace Meeting Khrystyna Ivaniv's screen

docs.google.com/spreadsheets/d/1CZyTYh-dTUigwJme78puTBwc2if_-2/edit?gid=1552872159#gid=1552872159

Epic 6 - Team Individual Tasks

Файл Змінити Вигляд Вставити Формат Дані Інструменти Довідка

100% Лише коментарі

	C	D	E	F	G	H
1						
2		Програмний Код №1	Програмний Код №2	Програмний Код №3	Програмний Код №3	Програмний Код №4
3	Студент	Файл 1	Файл 2	Файл 3.1	Файл 3.2	Файл 4
4		VNS Lab 10 - Task 1-N	Algotester Lab 5	Algotester Lab 7-8	Algotester Lab 7-8	Class Practice Work
5		Варіант №	Варіант №	Варіант №	Варіант №	Код з практичних по темі на заняттях
6						
7	екзесті	epic_6_pactice_and_labs_john_black				
8	екзесті та в гілці на ПК	vns_lab_10_task_john_black.c	algotester_lab_5_task_john_black.c	algotester_lab_7_8_variant_1_j	algotester_lab_7_8_variant_2_j	practice_work_team_tasks_john_black.c
9		pp	ack.cpp	ohn_black.cpp	ohn_black.cpp	n_black.cpp
10		YES	YES	YES	YES/NO	YES
11	екзесті та в гілці на ПК	epic_6_pactice_and_labs_report_john_black.docx				
12	якою у звіті	YES	YES	YES	YES/NO	YES
13	тематичні задачі	YES	YES	YES	YES/NO	YES
14	у звіті	YES	YES	YES	YES/NO	YES
15	и у звіті	YES	YES	YES	YES/NO	YES
16						
17	твіна	4	2	3	3	YES
18	ювін	24	3	1	1	YES
19	яч	23	2	1	1	YES
20	ровін	22	3	3	3	YES
21	уліварівна	21	2	2	2	YES
22		20	3	1	1	YES

ШІ-11 ШІ-12 ШІ-13

Khrystyna Ivaniv
Сирватка Олександр
Богдан
Женя

Висновок: в ході цього етапу я здобув навички роботи з різними структурами даних, такими як черга, список, стек та дерево, а також оволодів алгоритмами для обробки динамічних структур.