

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



## Звіт

**про виконання лабораторних та практичних робіт блоку № 5**

На тему: «Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами.

Створення й використання бібліотек.»

**з дисципліни:** «Основи програмування»

до:

ВНС Лабораторної Роботи № 6

ВНС Лабораторної Роботи № 8

ВНС Лабораторної Роботи № 9

Алготестер Лабораторної Роботи №4

Алготестер Лабораторної Роботи №6

Практичних Робіт до блоку №5

**Виконав:**

Студент групи ШІ-13  
Бойко Роман Андрійович

Львів 2024



**Тема роботи:** Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.

**Мета роботи:** Навчитися записувати і зчитувати інформацію з файлу стилями мов С та С++. Базово розібратися що таке бібліотека і де \\ використовують.

### Теоретичні відомості:

- файли
- рядкові змінні та символи
- бібліотеки

Джерела:

-  С++ • Теорія • Урок 166 • Робота з файлами (стиль мови С)
-  С++ • Теорія • Урок 171 • Робота з файлами (стиль мови С++)
- базові знання про вектори і списки

## Виконання роботи

Особистий варіант - VNS - 1, Algotester Lab 4 - 1,2, Algotester Lab 6 - 2

### Завдання 1: Practice task 1

**Реалізувати функцію створення файла і запису в нього даних:**

```
enum FileOpResult { Success, Failure, ... };
```

```
FileOpResult write_to_file(char *name, char *content);
```

*Умови задачі:*

- створити файл із заданим ім'ям; якщо файл існує – перезаписати його вміст
- написати код стійкий до різних варіантів вхідних параметрів
- name – ім'я, може не включати шлях
- записати у файл вміст стрічки content, прочитати content із стандартного вводу
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, запису даних, чи закриття файла.

**Код:**

```

1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4  enum FileOpResult
5  {
6      Success,
7      Failure
8  };
9  FileOpResult write_to_file(char *name, char *content);
10 int main()
11 {
12     char name[100];
13     char st[250];
14     cout << "Enter name of the file: ";
15     cin.getline(name, 100);
16     cout << "Enter some text: ";
17     cin.getline(st, 250);
18
19     FileOpResult f = write_to_file(name, st);
20     if (f == Success)
21     {
22         cout << "Success" << endl;
23     }
24     else if (f == Failure)
25     {
26         cout << "Failure" << endl;
27     }
28 }

```

```

29 FileOpResult write_to_file(char *name, char *content)
30 {
31
32     FILE *fileStream = fopen(name, "w");
33     if (fileStream != nullptr)
34     {
35         size_t write = fwrite(content, sizeof(char), strlen(content), fileStream);
36         fclose(fileStream);
37         if (write != strlen(content))
38         {
39             fclose(fileStream);
40             return Failure;
41         }
42         fclose(fileStream);
43     }
44     else
45     {
46         return Failure;
47     }
48
49     return Success;
50 }

```

## Вивід в терміналі:

```

Enter name of the file: texter
Enter some text: lafhioahf sopdfhosdf iwehroiwherh weprjopew
Success

```

## Вивід у файлі:

```
1  lafhioahf sopdfhosdf iwehroiwherh weprjopew
```

Час виконання ~ 25 хв

## Завдання 2: Practice task 2

**Реалізувати функцію створення файла і запису в нього даних:**

```
enum FileOpResult { Success, Failure, ... };
```

```
FileOpResult copy_file(char *file_from, char *file_to);
```

Умови задачі:

- копіювати вміст файла з ім'ям file\_from у файл з ім'ям file\_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів
- file\_from, file\_to – можуть бути повним або відносним шляхом
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, читання чи запису даних, закриття файла.

## Код

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  using namespace std;
5  enum FileOpResult
6  {
7      Success,
8      Failure
9  };
10 FileOpResult copy_file(char *file1, char *file2);
11 int main()
12 {
13     const int SIZE = 5;
14     int value = 5;
15     string name = "Hello, my name is Roman!";
16     double arr[SIZE] = {4.45, 12.43534, -4.3453, 1.01, 90.329};
17     char file1Name[] = "File1";
18     char file2Name[] = "CopyFile1";
19     ofstream fileStream(file1Name);
20     if (!fileStream.is_open())
21     {
22         return Failure;
23     }
24     fileStream << value << "\n"
25     << name << "\n";
```

```

26     for (int i = 0; i < SIZE; i++)
27     {
28         fileStream << arr[i] << " ";
29     }
30     fileStream.close();
31     FileOpResult res = copy_file(file1Name, file2Name);
32     if (res == Success)
33     {
34         cout << "Success!" << endl;
35     }
36     else
37     {
38         cout << "Failure!" << endl;
39     }
40 }
41
42 FileOpResult copy_file(char *file1, char *file2)
43 {
44     string line;
45     ifstream firstFile(file1);
46     ofstream destFile(file2);
47     if (!firstFile.is_open() || !destFile.is_open())
48     {
49         return Failure;
50     }
51     destFile << firstFile.rdbuf();
52     if (!firstFile.good() || !destFile.good())
53     {
54         return Failure;
55     }
56     firstFile.close();
57     destFile.close();
58     return Success;
59 }

```

**Вивід в терміналі:**

```
Success!
```

**Файл 1:**

```

1  5
2  Hello, my name is Roman!
3  4.45 12.4353 -4.3453 1.01 90.329

```

**Файл 2:**

```

1  5
2  Hello, my name is Roman!
3  4.45 12.4353 -4.3453 1.01 90.329

```

**Час виконання завдання ~ 40 хвилин**

### **Завдання 3: Algotester Lab 4 variant 1**

**Вам дано 2 цілих чисел масиви, розміром N та M.**

**Ваше завдання вивести:**

- 1. Різницю N-M**
- 2. Різницю M-N**
- 3. Їх перетин**
- 4. Їх об'єднання**
- 5. Їх симетричну різницю**

#### **Вхідні дані**

**У першому рядку ціле число N - розмір масиву 1**

**У другому рядку N цілих чисел - елементи масиву 1**

**У третьому рядку ціле число M - розмір масиву 2**

**У четвертому рядку M цілих чисел - елементи масиву 2**

#### **Вихідні дані**

**Вивести результат виконання 5 вищезазначених операцій у форматі:**

**У першому рядку ціле число N - розмір множини**

**У наступному рядку N цілих чисел - посортована у порядку зростання множина**

**Пам'ятайте, ви маєте написати 2 варіанти розв'язку, один з використанням засобів STL (`std::set_intersection`, `std::set_symmetric_difference`, `std::set_difference`, `std::set_union`), інший зі своєю реалізацією. Своє сортування можна не писати.**

**Варіант розв'язку 1:**

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  void differenceOf2Arrays(int arr1[], int arr2[], int size1, int size2);
5  void intersectionOf2Arrays(int arr1[], int arr2[], int size1, int size2);
6  bool isPresent(int arr[], int size, int elem);
7  void unionOf2Arrays(int arr1[], int arr2[], int size1, int size2);
8  void symmetricDifferenceOf2Arrays(int arr1[], int arr2[], int size1, int size2);
9  void removeDublicates(int arr[], int &size);
10 int main()
11 {
12     int N, M;
13     cin >> N;
14     int arr1[N];
15     for (int i = 0; i < N; i++)
16     {
17         cin >> arr1[i];
18     }
19     cin >> M;
20     int arr2[M];
21     for (int i = 0; i < M; i++)
22     {
23         cin >> arr2[i];
24     }
25
26     removeDublicates(arr1, N);
27     removeDublicates(arr2, M);
28     differenceOf2Arrays(arr1, arr2, N, M);
29     differenceOf2Arrays(arr2, arr1, M, N);
30     intersectionOf2Arrays(arr1, arr2, N, M);
31     unionOf2Arrays(arr1, arr2, N, M);
32     symmetricDifferenceOf2Arrays(arr1, arr2, N, M);
33 }

```

```

34 void differenceOf2Arrays(int arr1[], int arr2[], int size1, int size2)
35 {
36     int dif[size1];
37     int k = 0;
38     for (int i = 0; i < size1; i++)
39     {
40         bool find = false;
41         for (int j = 0; j < size2; j++)
42         {
43             if (arr1[i] == arr2[j])
44             {
45                 find = true;
46                 break;
47             }
48         }
49         if (!find)
50         {
51             dif[k] = arr1[i];
52             k++;
53         }
54     }
55     sort(dif, dif + k);
56     cout << k << endl;
57     for (int i = 0; i < k; i++)
58     {
59         cout << dif[i] << " ";
60     }
61     cout << "\n\n";
62 }

```

```

63 void intersectionOf2Arrays(int arr1[], int arr2[], int size1, int size2)
64 {
65     int inters[size1];
66     int k = 0;
67     for (int i = 0; i < size1; i++)
68     {
69         bool find = false;
70         for (int j = 0; j < size2; j++)
71         {
72             if (arr1[i] == arr2[j])
73             {
74                 find = true;
75                 break;
76             }
77         }
78         if (find)
79         {
80             inters[k] = arr1[i];
81             k++;
82         }
83     }
84     sort(inters, inters + k);
85     cout << k << endl;
86     for (int i = 0; i < k; i++)
87     {
88         cout << inters[i] << " ";
89     }
90     cout << "\n\n";
91 }

```

```

92 bool isPresent(int arr[], int size, int elem)
93 {
94     for (int i = 0; i < size; i++)
95     {
96         if (elem == arr[i])
97         {
98             return true;
99         }
100     }
101     return false;
102 }
103 void unionOf2Arrays(int arr1[], int arr2[], int size1, int size2)
104 {
105     int uni[size1 + size2];
106     int k = 0;
107     for (int i = 0; i < size1; i++)
108     {
109         if (!isPresent(uni, k, arr1[i]))
110         {
111             uni[k] = arr1[i];
112             k++;
113         }
114     }
115     for (int i = 0; i < size2; i++)
116     {
117         if (!isPresent(uni, k, arr2[i]))
118         {
119             uni[k] = arr2[i];
120             k++;
121         }
122     }

```



```

123     sort(uni, uni + k);
124     cout << k << endl;
125     for (int i = 0; i < k; i++)
126     {
127         cout << uni[i] << " ";
128     }
129     cout << "\n\n";
130 }
131 void symmetricDifferenceOf2Arrays(int arr1[], int arr2[], int size1, int size2)
132 {
133     int dif[size1 + size2];
134     int k = 0;
135     for (int i = 0; i < size1; i++)
136     {
137         bool find = false;
138         for (int j = 0; j < size2; j++)
139         {
140             if (arr1[i] == arr2[j])
141             {
142                 find = true;
143                 break;
144             }
145         }
146         if (!find)
147         {
148             dif[k] = arr1[i];
149             k++;
150         }
151     }

```

```

152     for (int i = 0; i < size2; i++)
153     {
154         bool find = false;
155         for (int j = 0; j < size1; j++)
156         {
157             if (arr2[i] == arr1[j])
158             {
159                 find = true;
160                 break;
161             }
162         }
163         if (!find)
164         {
165             dif[k] = arr2[i];
166             k++;
167         }
168     }
169     sort(dif, dif + k);
170     cout << k << endl;
171     for (int i = 0; i < k; i++)
172     {
173         cout << dif[i] << " ";
174     }
175     cout << "\n\n";
176 }
177 void removeDuplicates(int arr[], int &size)
178 {
179     int k = 0;
180     for (int i = 0; i < size; i++)
181     {
182         if (i == 0 || arr[i] != arr[i - 1])
183         {
184             arr[k] = arr[i];
185             k++;
186         }
187     }
188     size = k;
189 }

```

## Вивід в терміналі:

```
5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3

3
6 7 8

2
4 5

8
1 2 3 4 5 6 7 8

6
1 2 3 6 7 8
```

Час виконання завдання ~ 25 хвилин

## Варіант розв'язку - 2

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  int main()
7  {
8      int N, M;
9      cin >> N;
10     int num;
11     vector<int> set1;
12     for (int i = 0; i < N; i++)
13     {
14         cin >> num;
15         set1.push_back(num);
16     }
17     cin >> M;
18     vector<int> set2;
19     for (int i = 0; i < M; i++)
20     {
21         cin >> num;
22         set2.push_back(num);
23     }
24
25     cout << endl;
26     sort(set1.begin(), set1.end());
27     sort(set2.begin(), set2.end());
28     vector<int> dif1;
29     set_difference(set1.begin(), set1.end(), set2.begin(), set2.end(), inserter(dif1, dif1.end()));
30     cout << dif1.size() << endl;
31     for (int e : dif1)
```

```

32     {
33         cout << e << " ";
34     }
35     cout << endl;
36     vector<int> dif2;
37     set_difference(set2.begin(), set2.end(), set1.begin(), set1.end(), inserter(dif2, dif2.end()));
38     cout << dif2.size() << endl;
39     for (int e : dif2)
40     {
41         cout << e << " ";
42     }
43     cout << endl;
44     vector<int> inter;
45     set_intersection(set1.begin(), set1.end(), set2.begin(), set2.end(), inserter(inter, inter.end()));
46     cout << inter.size() << endl;
47     for (int e : inter)
48     {
49         cout << e << " ";
50     }
51     cout << endl;
52     vector<int> un;
53     set_union(set1.begin(), set1.end(), set2.begin(), set2.end(), inserter(un, un.end()));
54     cout << un.size() << endl;
55     for (int e : un)
56     {
57         cout << e << " ";
58     }
59
60     vector<int> sym;
61     set_symmetric_difference(set1.begin(), set1.end(), set2.begin(), set2.end(), inserter(sym, sym.end()));
62     cout << sym.size() << endl;
63     for (int e : sym)
64     {
65         cout << e << " ";
66     }
67     cout << endl;
68 }
69

```

## Вивід в терміналі:

```

5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3

3
6 7 8

2
4 5

8
1 2 3 4 5 6 7 8

6
1 2 3 6 7 8

```

**Час виконання завдання ~ 25 хвилин**

## Завдання 4: Algotester Lab 4 variant 2

Вам дано масив  $a$  з  $N$  цілих чисел.

Спочатку видаліть масиву  $a$  усі елементи що повторюються, наприклад масив  $[1, 3, 3, 4]$  має перетворитися у  $[1, 3, 4]$ .

Після цього оберніть посортовану версію масиву  $a$  на  $K$ , тобто при  $K=3$  масив  $[1, 2, 3, 4, 5, 6, 7]$  перетвориться на  $[4, 5, 6, 7, 1, 2, 3]$ .

Виведіть результат.

**Пам'ятайте, ви маєте написати 2 варіанти розв'язку, один з використанням засобів STL (`std::unique`, `std::sort`, `std::rotate`), інший зі своєю реалізацією.**

### Розв'язок - 1:

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  void deleteDublicates(int arr[], int &size);
5  void reversedArray(int arr[], int size, int index);
6  int main()
7  {
8      int N, K;
9      cin >> N >> K;
10     int a[N];
11     for (int i = 0; i < N; i++)
12     {
13         cin >> a[i];
14     }
15     sort(a, a + N);
16     deleteDublicates(a, N);
17     cout << N << endl;
18     reversedArray(a, N, K);
19 }
20 void deleteDublicates(int arr[], int &size)
21 {
22     int k = 0;
23     for (int i = 0; i < size; i++)
24     {
25         if (i == 0 || arr[i] != arr[i - 1])
26         {
27             arr[k] = arr[i];
28             k++;
29         }
30     }
31     size = k;
32 }
```

```

33 void reversedArray(int arr[], int size, int index)
34 {
35     int k = 0;
36     int newArr[size];
37     for (int i = index; i < size; i++)
38     {
39         newArr[k] = arr[i];
40         k++;
41     }
42     for (int i = 0; i < index; i++)
43     {
44         newArr[k] = arr[i];
45         k++;
46     }
47     for (int i = 0; i < k; i++)
48     {
49         cout << newArr[i] << " ";
50     }
51 }

```

**Результат виконання програми:**

```

10 3
1 2 2 3 3 3 4 5 6 7
7
4 5 6 7 1 2 3

```

**Час виконання завдання ~ 25 хвилин**

**Розв'язок - 2**

```

1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <algorithm>
5  using namespace std;
6
7  int main()
8  {
9      int N, K;
10     cin >> N >> K;
11     int num;
12     vector<int> unique_El;
13     for (int i = 0; i < N; i++)
14     {
15         cin >> num;
16         unique_El.push_back(num);
17     }
18     sort(unique_El.begin(), unique_El.end());
19     unique_El.erase(unique(unique_El.begin(), unique_El.end()), unique_El.end());
20     int size = unique_El.size();
21     if (size > 0)
22     {
23         K = K % size;
24         rotate(unique_El.begin(), unique_El.begin() + K, unique_El.end());
25     }
26     cout << unique_El.size() << endl;
27     for (int e : unique_El)
28     {
29         cout << e << " ";
30     }
31 }

```

**Вивід в терміналі:**

```
10 3
1 2 2 3 3 3 4 5 6 7
7
4 5 6 7 1 2 3
```

**Час виконання завдання ~ 20 хвилин**

### **Завдання 5: Algotester Lab 6 variant 2**

У вас є шахова дошка розміром 8×8 та дуже багато фігур.

Кожна клітинка може мати таке значення:

- Пуста клітинка O
- Пішак P
- Тура R
- Кінь N
- Слон B
- Король K
- Королева Q

Вам дають позиції фігур на дошці (всі фігури одного кольору, кількість королів може бути > 1).

Далі йдуть Q запитів з координатами клітинки {x,y}{x,y}. На кожен запит ви маєте вивести стрічку сі - посортовані за алфавітом букви фігур, які атакують цю клітинку (пішаки атакують вниз).

У випадку, якщо на клітинці стоїть якась фігура - виведіть символ X.

У випадку, якщо клітинку не атакують - виведіть O.

Наявність фігури у певній клітинці не блокує атаку для іншої фігури. Тобто якщо між турою та клітинкою стоїть інша фігура - вважається що тура атакує цю клітинку.

**Код:**

```

1  #include <iostream>
2  #include <vector>
3  #include <set>
4  using namespace std;
5
6  bool isInBoard(int x, int y)
7  {
8      return (x >= 0 && x < 8 && y >= 0 && y < 8);
9  }
10 set<char> PawnAttack(vector<vector<char>> &board, int x, int y)
11 {
12     set<char> at;
13     if (isInBoard(x - 1, y - 1) && board[x - 1][y - 1] == 'P')
14     {
15         at.insert('P');
16     }
17     if (isInBoard(x - 1, y + 1) && board[x - 1][y + 1] == 'P')
18     {
19         at.insert('P');
20     }
21     return at;
22 }

```

```

23 set<char> KingAttack(vector<vector<char>> &board, int x, int y)
24 {
25     set<char> at;
26     for (int i = -1; i <= 1; i++)
27     {
28         for (int j = -1; j <= 1; j++)
29         {
30             if (i != 0 || j != 0)
31             {
32                 int newX = x + i;
33                 int newY = y + j;
34                 if (isInBoard(newX, newY) && board[newX][newY] == 'K')
35                 {
36                     at.insert('K');
37                 }
38             }
39         }
40     }
41     return at;
42 }

```

```

set<char> RookAttack(vector<vector<char>> &board, int x, int y)
{
    set<char> at;
    for (int i = 0; i < 8; i++)
    {
        if (i != x && board[i][y] == 'R')
            at.insert('R');
        if (i != y && board[x][i] == 'R')
            at.insert('R');
    }
    return at;
}

```

```

55 set<char> BishopAttack(vector<vector<char>> &board, int x, int y)
56 {
57     set<char> at;
58     for (int i = 1; x - i >= 0 && y - i >= 0; i++)
59     {
60         if (isInBoard(x - i, y - i) && board[x - i][y - i] == 'B')
61         {
62             at.insert('B');
63             break;
64         }
65     }
66     for (int i = 1; x - i >= 0 && y + i < 8; i++)
67     {
68         if (isInBoard(x - i, y + i) && board[x - i][y + i] == 'B')
69         {
70             at.insert('B');
71             break;
72         }
73     }
74     for (int i = 1; x + i < 8 && y - i >= 0; i++)
75     {
76         if (isInBoard(x + i, y - i) && board[x + i][y - i] == 'B')
77         {
78             at.insert('B');
79             break;
80         }
81     }

```

```

82     for (int i = 1; x + i < 8 && y + i < 8; i++)
83     {
84         if (isInBoard(x + i, y + i) && board[x + i][y + i] == 'B')
85         {
86             at.insert('B');
87             break;
88         }
89     }
90     return at;
91 }

```

```

92 set<char> QueenAttack(vector<vector<char>> &board, int x, int y)
93 {
94     set<char> at;
95     for (int i = 0; i < 8; i++)
96     {
97         if (i != x && board[i][y] == 'Q')
98             at.insert('Q');
99         if (i != y && board[x][i] == 'Q')
100             at.insert('Q');
101     }
102     for (int i = 1; x - i >= 0 && y - i >= 0; i++)
103     {
104         if (isInBoard(x - i, y - i) && board[x - i][y - i] == 'Q')
105         {
106             at.insert('Q');
107             break;
108         }
109     }
110     for (int i = 1; x - i >= 0 && y + i < 8; i++)
111     {
112         if (isInBoard(x - i, y + i) && board[x - i][y + i] == 'Q')
113         {
114             at.insert('Q');
115             break;
116         }
117     }

```



```

118     for (int i = 1; x + i < 8 && y - i >= 0; i++)
119     {
120         if (isInBoard(x + i, y - i) && board[x + i][y - i] == 'Q')
121         {
122             at.insert('Q');
123             break;
124         }
125     }
126     for (int i = 1; x + i < 8 && y + i < 8; i++)
127     {
128         if (isInBoard(x + i, y + i) && board[x + i][y + i] == 'Q')
129         {
130             at.insert('Q');
131             break;
132         }
133     }
134     return at;
135 }

```

```

136 set<char> KnightAttack(vector<vector<char>> &board, int x, int y)
137 {
138     set<char> at;
139     int knightMoves[8][2] = {
140         {-2, -1}, {-1, -2}, {1, -2}, {2, -1}, {2, 1}, {1, 2}, {-1, 2}, {-2, 1}};
141     for (int i = 0; i < 8; ++i)
142     {
143         int newX = x + knightMoves[i][0];
144         int newY = y + knightMoves[i][1];
145         if (isInBoard(newX, newY) && board[newX][newY] == 'N')
146         {
147             at.insert('N');
148         }
149     }
150     return at;
151 }

```

```

152 int main()
153 {
154     vector<vector<char>> board(8, vector<char>(8));
155     for (int i = 0; i < 8; i++)
156     {
157         for (int j = 0; j < 8; j++)
158         {
159             cin >> board[i][j];
160         }
161     }
162     int Q;
163     cin >> Q;
164     for (int i = 0; i < Q; i++)
165     {
166         int x, y;
167         cin >> x >> y;
168         x--;
169         y--;
170         set<char> res;
171         if (board[x][y] != '0')
172         {
173             cout << "X" << endl;
174             continue;
175         }

```

```

176     auto pawn = PawnAttack(board, x, y);
177     res.insert(pawn.begin(), pawn.end());
178     auto king = KingAttack(board, x, y);
179     res.insert(king.begin(), king.end());
180     auto rook = RookAttack(board, x, y);
181     res.insert(rook.begin(), rook.end());
182     auto bishop = BishopAttack(board, x, y);
183     res.insert(bishop.begin(), bishop.end());
184     auto knight = KnightAttack(board, x, y);
185     res.insert(knight.begin(), knight.end());
186     auto queen = QueenAttack(board, x, y);
187     res.insert(queen.begin(), queen.end());
188     if (res.empty())
189     {
190         cout << "0" << endl;
191     }
192     else
193     {
194         for (char ch : res)
195         {
196             cout << ch;
197         }
198         cout << endl;
199     }
200 }
201 }

```

**Вивід в терміналі:**

```

00000000
0R000000
00N00000
0000P000
00000000
00000000
00000000
K0Q00000
0000000R

7
8 1
KR
1 2
NR
5 4
NP
5 1
Q
6 2
KQR
8 4
QR
6 7
0

```

**Час виконання завдання ~ 4 години**

## Завдання 6: VNS Lab 6

Задано рядок, що складається із символів. Символи поєднуються в слова.

Слова одне від одного відокремлюються одним або декількома пробілами.

Наприкінці тексту ставиться крапка. Текст містить не більше 255 символів.

Виконати ввід рядка, використовуючи функцію `gets(s)` і здійснити обробку

рядка у відповідності зі своїм варіантом.

Надрукувати найдовше й найкоротше слово в цьому рядку.

Код:

```
1  #include <iostream>
2  #include <string>
3  #include <stdio.h>
4  #include <string.h>
5  using namespace std;
6  void words(char s[]);
7  int main()
8  {
9      const int SIZE = 256;
10     char s[SIZE];
11     gets(s);
12     words(s);
13 }
14 void words(char s[])
15 {
16     char *token;
17     token = strtok(s, " ");
18     char *shortest = token;
19     char *longest = token;
20     while (token != nullptr)
21     {
22         if (strlen(shortest) > strlen(token))
23         {
24             shortest = token;
25         }
26         if (strlen(longest) < strlen(token))
27         {
28             longest = token;
29         }
30         token = strtok(nullptr, " ");
31     }
32     cout << "The longest word is: " << longest << endl;
33     cout << "The shortest word is: " << shortest << endl;
34 }
```

## Вивід в терміналі:

```
efwohgowehg wopej dfohwehf0 wpoefjew-jf
The longest word is: efwohgowehg
The shortest word is: wopej
```

**Час виконання завдання: 30 хвилин**

## Завдання 7: VNS Lab 8

**Сформувати двійковий файл із елементів, заданої у варіанті структури,**

**роздрукувати його вміст, виконати знищення й додавання елементів у**

**відповідності зі своїм варіантом, використовуючи для пошуку елементів що**

**знищуються чи додаються, функцію. Формування, друк, додавання й знищення**

**елементів оформити у вигляді функцій. Передбачити повідомлення про**

**помилки при відкритті файлу й виконанні операцій вводу/виводу.**

**Структура "Абітурієнт":**

**- прізвище, ім'я, по батькові;**

**- рік народження;**

**- оцінки вступних іспитів (3);**

**- середній бал атестата.**

**Знищити елемент із зазначеним номером, додати елемент після елемента із**

**зазначеним прізвищем.**

**Код:**

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  const int SIZE = 50;
5  const int ArrS = 3;
6  struct Abiturient
7  {
8      char firstName[SIZE];
9      char secondName[SIZE];
10     int age;
11     int marks[ArrS];
12     int average;
13 };
14 enum FileOpResult
15 {
16     Success,
17     Failure
18 };
19
20 void addAbit(Abiturient &a);
21 void outputAbit(Abiturient &a);
22 FileOpResult writeDataToFile(Abiturient *a, int size, char *filename);
23 FileOpResult readDataFromFile(char *filename);
24 FileOpResult DeleteElemFromFile(char *filename, int index);
25 FileOpResult AddElemBySurname(char *filename, char surn[]);
26
```

```

27  int main()
28  {
29      int size, index;
30      char surn[SIZE];
31      char filename[] = "Labs";
32      cout << "Enter amount of abiturients you want to add: ";
33      cin >> size;
34      Abiturient *a = new Abiturient[size];
35      for (int i = 0; i < size; i++)
36      {
37
38          cin.ignore();
39          cout << "Abiturient: " << i + 1 << endl;
40          addAbit(a[i]);
41      }
42      FileOpResult r = writeDataToFile(a, size, filename);
43      if (r == Success)
44      {
45          cout << "Information added succesfully!" << endl;
46      }
47      else
48      {
49          cerr << "Error in adding information to file!" << endl;
50      }
51      readDataFromFile(filename);
52      cout << "\nEnter index of abiturient you wanna to delete: ";

```

```

53      cin >> index;
54      FileOpResult r2 = DeleteElemFromFile(filename, index);
55      if (r2 == Success)
56      {
57          cout << "Information deleted by index succesfully!" << endl;
58      }
59      else
60      {
61          cerr << "Error in deleting information from the file!" << endl;
62      }
63      cout << "\nEnter surname of abiturient: ";
64      cin.ignore();
65      cin.getline(surn, SIZE);
66      FileOpResult r3 = AddElemBySurname(filename, surn);
67      if (r3 == Success)
68      {
69          cout << "Information added by surname succesfully!" << endl;
70      }
71      else
72      {
73          cerr << "Error in adding information to the file!" << endl;
74      }
75      delete[] a;
76  }
77

```

```

78 void addAbit(Abiturient &a)
79 {
80     cout << "Enter name: ";
81     cin.getline(a.firstName, SIZE);
82     cout << "Enter surname: ";
83     cin.getline(a.secondName, SIZE);
84     cout << "Enter age: ";
85     cin >> a.age;
86     int sum = 0;
87     for (int j = 0; j < ArrS; j++)
88     {
89         cout << "Enter mark " << j + 1 << ": ";
90         cin >> a.marks[j];
91         sum += a.marks[j];
92     }
93     int averageMark = sum / 3;
94     a.average = averageMark;
95 }
96

```

```

109 FileOpResult writeDataToFile(Abiturient *a, int size, char *filename)
110 {
111     FILE *fileStream = fopen(filename, "wb");
112     if (fileStream != nullptr)
113     {
114         size_t write = fwrite(a, sizeof(Abiturient), size, fileStream);
115
116         if (write != size)
117         {
118             return Failure;
119         }
120         fclose(fileStream);
121         return Success;
122     }
123     else
124     {
125         return Failure;
126     }
127 }
128

```

```

129 FileOpResult readDataFromFile(char *filename)
130 {
131     FILE *fileStream = fopen(filename, "rb");
132     Abiturient a;
133     if (!fileStream)
134     {
135         return Failure;
136     }
137     while (fread(&a, sizeof(Abiturient), 1, fileStream))
138     {
139         outputAbit(a);
140     }
141     fclose(fileStream);
142     return Success;
143 }
144

```

```

145 FileOpResult DeleteElemFromFile(char *filename, int index)
146 {
147     FILE *fileStream = fopen(filename, "rb");
148     Abiturient a;
149     if (!fileStream)
150     {
151         return Failure;
152     }
153     vector<Abiturient> abiture;
154     while (fread(&a, sizeof(Abiturient), 1, fileStream))
155     {
156         abiture.push_back(a);
157     }
158     fclose(fileStream);
159     if (index < 0 || index >= abiture.size())
160     {
161         return Failure;
162     }

```

```

163     abiture.erase(abiture.begin() + index);
164     FILE *filestream = fopen(filename, "wb");
165     if (!fileStream)
166     {
167         return Failure;
168     }
169     if (fileStream != nullptr)
170     {
171         size_t write = fwrite(abiture.data(), sizeof(abiture), abiture.size(), fileStream);
172         if (write != abiture.size())
173         {
174             return Failure;
175         }
176         fclose(fileStream);
177         return Success;
178     }
179     else
180     {
181         return Failure;
182     }
183 }

```

```

184 FileOpResult AddElemBySurname(char *filename, char surn[])
185 {
186     FILE *fileStream = fopen(filename, "rb");
187     Abiturient a;
188     if (!fileStream)
189     {
190         return Failure;
191     }
192     vector<Abiturient> abiture;
193     while (fread(&a, sizeof(Abiturient), 1, fileStream))
194     {
195         abiture.push_back(a);
196     }
197     fclose(fileStream);
198     Abiturient abit;
199     cout << "Adding new abiturient: " << endl;
200     addAbit(abit);
201     bool found = true;

```

```

201     bool found = true;
202     for (int i = 0; i < abiture.size(); i++)
203     {
204         bool surnFound = true;
205         for (int j = 0; j < SIZE; j++)
206         {
207             if (abiture[i].secondName[j] != surn[j])
208             {
209                 surnFound = false;
210                 break;
211             }
212             if (abiture[i].secondName[j] == '\0' && surn[j] == '\0')
213             {
214                 break;
215             }
216         }
217         if (surnFound)
218         {
219             abiture.insert(abiture.begin() + i + 1, abit);
220             found = true;
221             break;
222         }
223     }
224 }

```

```

225     if (!found)
226     {
227         cout << "Surname is not found!" << endl;
228         return Failure;
229     }
230     FILE *filestream = fopen(filename, "wb");
231     if (!fileStream)
232     {
233         return Failure;
234     }
235     if (fileStream != nullptr)
236     {
237         size_t write = fwrite(abiture.data(), sizeof(abiture), abiture.size(), fileStream);
238         if (write != abiture.size())
239         {
240             return Failure;
241         }
242         fclose(fileStream);
243         return Success;
244     }
245     else
246     {
247         return Failure;
248     }
249 }

```

**Вивід в терміналі:**



```
Enter amount of abituriants you want to add: 2
Abiturient: 1
Enter name: Roman
Enter surname: Boyko
Enter age: 18
Enter mark 1: 99
Enter mark 2: 100
Enter mark 3: 99
Abiturient: 2
Enter name: Andriy
Enter surname: Kovalenko
Enter age: 17
Enter mark 1: 90
Enter mark 2: 80
Enter mark 3: 70
Information added succesfully!
Name: Roman
Second name: Boyko
Age: 18
Mark 1: 99
Mark 2: 100
Mark 3: 99
Average mark: 99
Name: Andriy
Second name: Kovalenko
Age: 17
Mark 1: 90
Mark 2: 80
Mark 3: 70
Average mark: 80
```

```
Enter index of abiturient you wanna to delete: 0
Information deleted by index succesfully!

Enter surname of abiturient: Ivanenko
Adding new abiturient:
Enter name: Kin
Enter surname: Hop
Enter age: 10
Enter mark 1: 11
Enter mark 2: 12
Enter mark 3: 13
Information added by surname succesfully!
```

**Час виконання завдання ~ 1 година**

## **Завдання 8: VNS Lab 9**

**Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію**

- 1) Скопіювати у файл F2 тільки парні рядки з F1.**
- 2) Підрахувати розмір файлів F1 й F2 (у байтах).**

**Код:**

```

1  #include <iostream>
2  using namespace std;
3  long fileSize(FILE *filename);
4  int main()
5  {
6      int k = 0;
7      FILE *fileStream = fopen("lab9_1", "r");
8      FILE *fileStream2 = fopen("lab9_2", "w");
9      if (!fileStream || !fileStream2)
10     {
11         cout << "There is an error in opening the file!" << endl;
12         return 1;
13     }
14     const size_t SIZE = 100;
15     char string[SIZE];
16     while (fgets(string, SIZE, fileStream))
17     {
18         if (k % 2 == 0 || k == 0)
19         {
20             fputs(string, fileStream2);
21         }
22         k++;
23     }
24     long size1 = fileSize(fileStream);
25     long size2 = fileSize(fileStream2);
26     cout << "Operation with files is succesfully completed!" << endl;
27     cout << "Number of rows in file 1: " << k << endl;
28     cout << "Size of file 1: " << size1 << " bytes" << endl;
29     cout << "Size of file 2: " << size2 << " bytes" << endl;
30     fclose(fileStream);
31     fclose(fileStream2);
32 }

```

```

33 long fileSize(FILE *filename)
34 {
35     fseek(filename, 0, SEEK_END);
36     long size = ftell(filename);
37     fseek(filename, 0, SEEK_SET);
38     return size;
39 }

```

**Вивід в терміналі:**

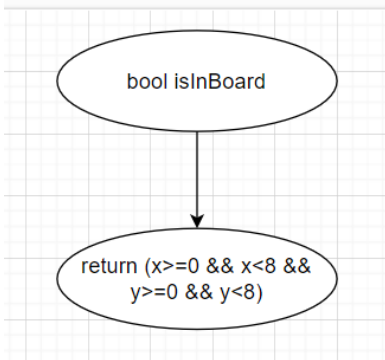
```

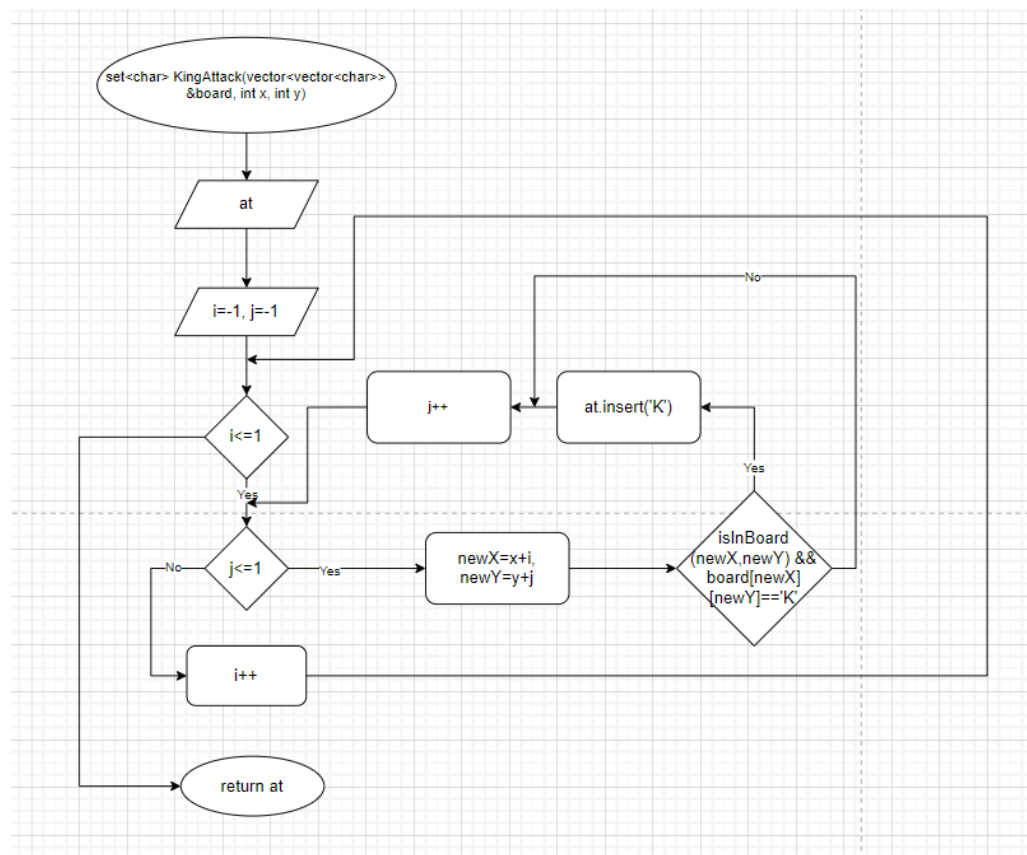
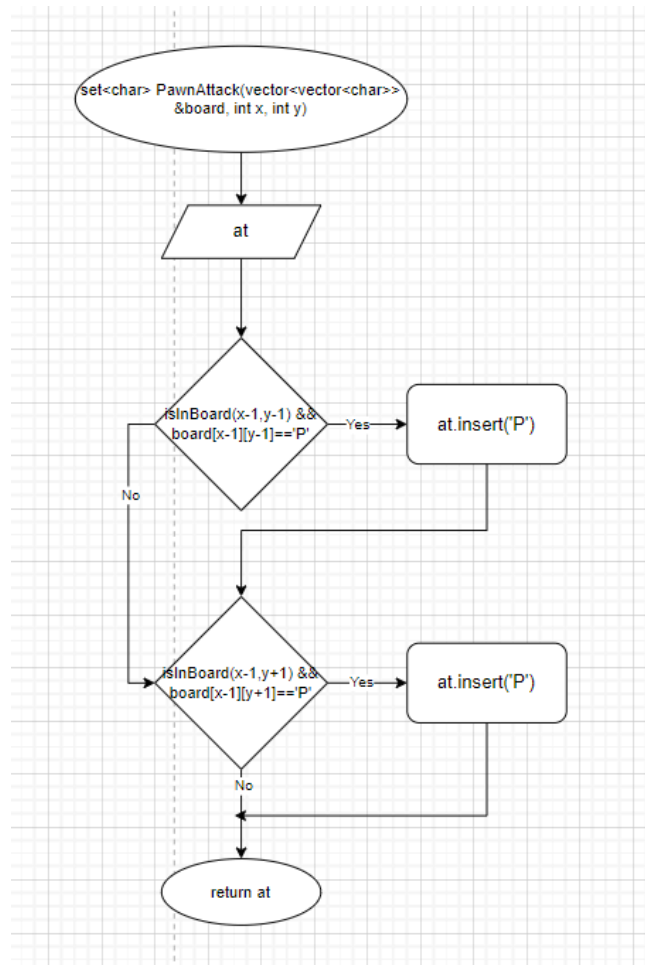
Operation with files is succesfully completed!
Number of rows in file 1: 17
Size of file 1: 1248 bytes
Size of file 2: 711 bytes

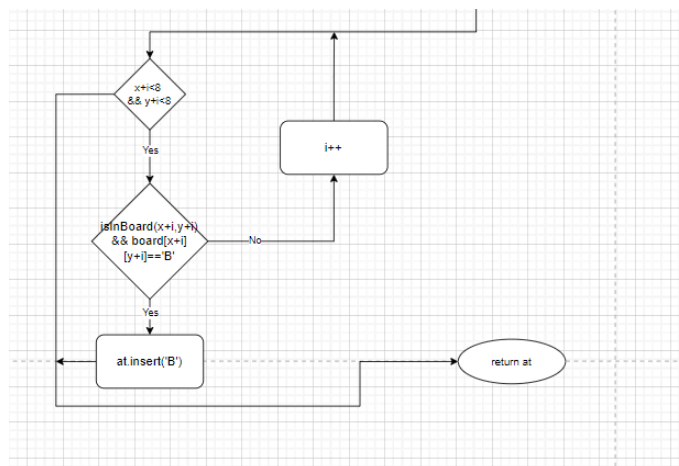
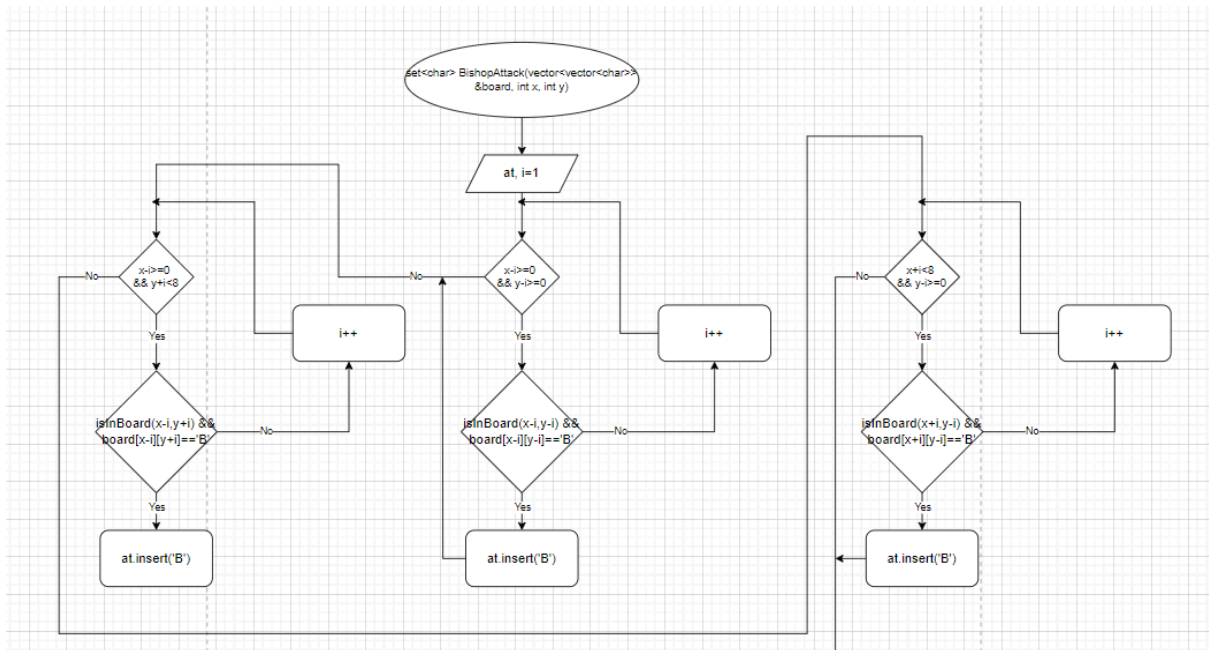
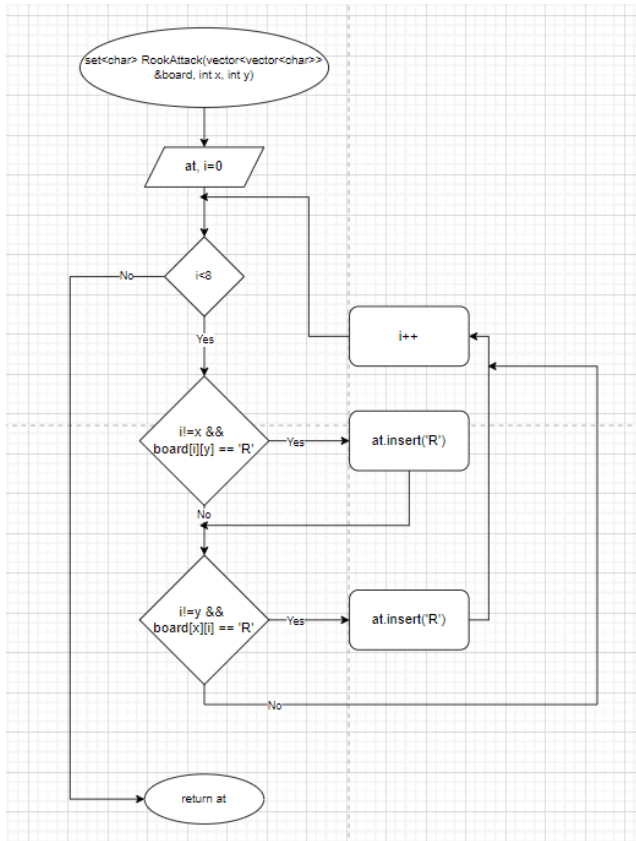
```

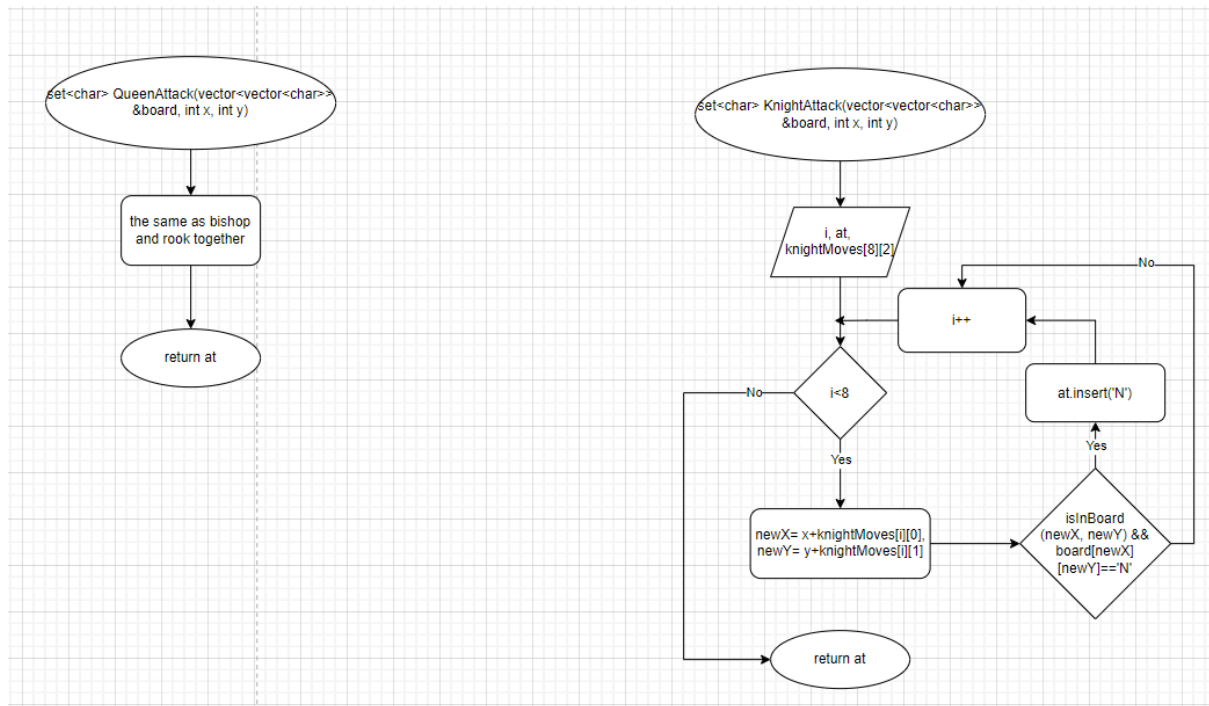
**Час виконання завдання ~ 30 хвилин**

**Блок-схема до задачі Algotester Lab 6:**









## Завдання 9: Algotester Lab 4 Variant 3

Вам дано масив, який складається з  $N$  додатніх цілих чисел.

Ваше завдання - розділити його на три частини, по остачі від ділення на 3, по зростанню остачі (тобто спочатку йдуть числа, у яких остача 0, далі числа з остачею 1 і тоді нарешті числа з остачею 2).

Далі необхідно ті елементи, остача від ділення на 3 яких парна посортувати по зростанню, а ті, у яких остача 1 - по спаданню.

Після цього видаліть усі дублікати з масиву.

Виведіть результуючий масив.

Код:

```

1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <algorithm>
5  using namespace std;
6
7  int main()
8  {
9      int N, num;
10     cin >> N;
11     vector<int> vec;
12     vector<int> vec0;
13     vector<int> vec1;
14     vector<int> vec2;
15     for (int i = 0; i < N; i++)
16     {
17         cin >> num;
18         vec.push_back(num);
19     }
20     for (int i = 0; i < vec.size(); i++)
21     {
22         if (vec[i] % 3 == 0)
23         {
24             vec0.push_back(vec[i]);
25         }
26     }
  
```

```

26         else if (vec[i] % 3 == 1)
27         {
28             vec1.push_back(vec[i]);
29         }
30         else if (vec[i] % 3 == 2)
31         {
32             vec2.push_back(vec[i]);
33         }
34     }
35     sort(vec0.begin(), vec0.end());
36     sort(vec1.begin(), vec1.end());
37     sort(vec2.begin(), vec2.end());
38     reverse(vec1.begin(), vec1.end());
39     vector<int> res;
40     res.insert(res.end(), vec0.begin(), vec0.end());
41     res.insert(res.end(), vec1.begin(), vec1.end());
42     res.insert(res.end(), vec2.begin(), vec2.end());
43     auto l = unique(res.begin(), res.end());
44     res.erase(l, res.end());
45     cout << res.size() << endl;
46     for (int num : res)
47     {
48         cout << num << " ";
49     }
50 }

```

**Вивід в терміналі:**

```

10
1 33 4 8 6 5 2 7 5 0
9
0 6 33 7 4 1 2 5 8

```

**Час виконання завдання ~ 20 хвилин**

**Висновок:**

**У цьому епіку я навчився працювати з файлами, розібрався як працюють вектори і списки а також ознайомився із бібліотеками.**