

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



Звіт

про виконання лабораторних та практичних робіт блоку № 4

На тему: «Цикли. Вкладені Цикли. Завершення виконання циклів.
Функції. Простір імен. Перевантаження функцій. Функції з змінною
кількістю параметрів. Рекурсія. Вбудовані функції.»

з дисципліни: «Основи програмування»

до:

Практичних Робіт до блоку № 4

Виконав:

Студент групи ШІ-11

Яровой Павло Олегович

Тема роботи: Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами

Мета роботи: Навчитися працювати з одновимірними та двовимірними масивами, використовувати вказівники та посилання для оптимізації роботи з динамічними масивами, розібратися зі структурами даних і вкладеними структурами для зберігання складних даних, а також освоїти алгоритми обробки та маніпуляції масивами та структурами.

Теоретичні відомості:

1)Перелік тем:

1. Класи пам'яті у C++
2. Вступ до Масивів і Вказівників
3. Одновимірні Масиви
4. Вказівники та Посилання
5. Двовимірні Масиви
6. Динамічні Масиви
7. Структури Даних
8. Вкладені Структури
9. Використання структур

10. Алгоритми обробки та робота з масивами та структурами

2)Індивідуальний план опрацювання теорії:

Тема №1

Джерела:

[C++ Storage Classes](#)

[Memory Classes in C++](#)

Тема №2

Джерела:

[Introduction to Arrays in C++](#)

[Pointers in C++](#)

Тема №3

Джерела:

[C++ One-Dimensional Arrays](#)

[Array Basics in C++](#)

Тема №4

Джерела:

Pointers and References in C++
References in C++

Тема №5

Джерела:

Two-Dimensional Arrays in C++
2D Arrays in C++

Тема №6

Джерела:

Dynamic Arrays in C++
C++ Dynamic Memory

Тема №7

Джерела:

Structures in C++
Introduction to Data Structures

Тема №8

Джерела:

Nested Structures in C++
C++ Nested Structures

Тема №9

Джерела:

C++ Structs and their Use
Uses of Structures in C++

Тема №10

Джерела:

Algorithms for Arrays in C++

Working with Arrays and Structures in C++

Виконання роботи:

1)Перелік завдань:

- John Black - Epic 4 Task 1 - Theory Education Activities
- John Black - Epic 4 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-8)
- John Black - Epic 4 Task 3 - Lab# programming: VNS Lab 4(варіант 20)
- John Black - Epic 4 Task 4 - Lab# programming: VNS Lab 5(варіант 20)
- John Black - Epic 4 Task 5 - Lab# programming: Algotester Lab 2(варіант 1)
- John Black - Epic 4 Task 6 - Lab# programming: Algotester Lab 3(варіант 3)

- John Black - Epic 4 Task 7 - Practice# programming: Class Practice Task
- John Black - Epic 4 Task 8 - Practice# programming: Self Practice Task
- John Black - Epic 4 Task 9 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
- John Black - Epic 4 Task 10 - Results Evaluation and Release

2) Умови завдань:

Task 3:

20.

- 1) Реалізувати з використанням масиву двонаправлене кільце (перегляд можливий в обидва боки, від останнього елемента можна перейти до першого).
- 2) Роздрукувати отриманий масив, починаючи з K-ого елемента і до K-1 (по кільцю вліво).
- 3) Додати в кільце після елементів з індексами кратними 5 елементи, які дорівнюють 0.
- 4) Роздрукувати отриманий масив, починаючи з K-ого елемента (і до K+1 по кільцю вправо).

Task 4:

20. Знайти мінімальний з неповторюваних елементів двовимірного масиву.

Task 5.

Lab 2v1

Limits: 1 sec., 256 MiB

У вас є дорога, яка виглядає як N чисел.

Після того як ви по ній пройдете - вашу втому можна визначити як різницю максимального та мінімального елементу.

Ви хочете мінімізувати втому, але все що ви можете зробити - викинути одне число з дороги, тобто забрати його з масиву.

В результаті цієї дії, яку мінімальну втому ви можете отримати в кінці дороги?

Input

У першому рядку ціле число N - кількість чисел

У другому рядку масив r , який складається з N цілих чисел

Output

Єдине ціле число m - мінімальна втома, яку можна отримати

Task 6

Lab 3v3

Limits: 1 sec., 256 MiB

Вам дана стрічка s .

Ваше завдання зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

Input

У першому рядку стрічка S

Output

Стрічка $S_{compressed}$

Task 7:

Перевірка чи слово або число є паліндромом

Задача

Реалізувати програму, яка перевіряє, чи дане слово чи число є паліндромом за допомогою рекурсії.

Паліндром — це послідовність символів, яка однаково читається вперед і назад (наприклад, «radar», «level», «12321»).

Мета Задачі

Навчитися користуватися механізмами перевантаження функції та використовувати рекурсію для вирішення задач обчислення.

Вимоги:

1. Визначення функції:
 - a. Реалізуйте рекурсивну функцію *isPalindrome*, яка перевіряє, чи заданий рядок є паліндромом.
2. Приклад визначення функції:
 - a. *bool isPalindrome(const string& str, int start, int end);*
3. Перевантаження функцій:
 - a. Перевантажте функцію *isPalindrome* для роботи з цілими значеннями.
 - b. *bool isPalindrome(ціле число);*
4. Рекурсія:
 - a. Рекурсивна функція для рядків перевірить символи в поточній початковій і кінцевій позиціях. Якщо вони збігаються, він буде рекурсивно перевіряти наступні позиції, поки початок не перевищить кінець, після чого рядок буде визначено як паліндром.

Task 8.1 Self practice

Lab 2v2

Limits: 1 sec., 256 MiB

У вас є масив r розміром N . Також вам дано 3 цілих числа.

Спочатку ви маєте видалити з масиву ці 3 числа, які вам дані. Після цього перетворити цей масив у масив сум, розміром $N_{new} - 1$ (розмір нового масиву після видалення елементів), який буде відображати суми с елементів нового масиву.

Далі необхідно вивести масив сум на екран.

Input

У першому рядку ціле число N - кількість чисел

У другому рядку масив r , який складається з N цілих чисел

У третьому рядку 3 цілих числа, a, b, c , які треба видалити з масиву

Output

У першому рядку ціле число M - кількість чисел у масиві, який буде виведено

У наступному рядку M чисел - новий масив

Task 8.2 Self practice

Коля, Вася і темна кімната

Обмеження: 2 сек., 256 МБ

Блукаючи коридорами «Екстралогіки» в пошуках дверей, які треба помалювати, Коля і Вася, здається, знайшли темну кімнату!

Досліджуючи кімнату, Коля і Вася поміж старих тенісних м'ячків і ракеток, столів, вазонів, шаф із одягом знайшли n дверей. Знайшовши таке багатство, будівельники не встояли перед спокусою і вирішили всі ці двері помалювати.

Коля і Вася мають із собою k пензлів із фарбою. Насиченість фарби на i -му пензлі дорівнює b_i . Кожні двері будівельники планують помалювати рівно одним пензлем. Після того, як двері з площею S були помальовані пензлем із насиченістю фарби B , краса дверей стане рівною SB , а насиченість фарби на цьому пензлі зменшиться на одиницю, тобто стане рівною $B - 1$. Якщо насиченість фарби на деякому пензлі стане рівною нулю, то він висихає, і дверей ним більше не можна малювати.

Будівельники, як істинні митці та майстри своєї справи, хочуть зробити сумарну красу всіх дверей якомога більшою. Допоможіть їм — скажіть, якої максимальної сумарної краси дверей можна досягнути.

Вхідні дані

У першому рядку задано два натуральні числа n і k — кількість дверей і пензлів відповідно.

Другий рядок містить n натуральних чисел s_i — площі дверей.

У третьому рядку задано k натуральних чисел b_i — початкові насиченості фарб на пензлях.

Гарантується, що фарби на пензлях вистачить для того, щоб помалювати всі двері.

Вихідні дані

У єдиному рядку виведіть натуральне число — максимальну сумарну красу дверей.

Активация Windows

Перейдіть до розділу "Настройки", щоб активувати Windows.

Task 3 - Lab# programming: VNS Lab 5 Task 1 Варіант 20


```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void printRing(const vector<int>& ring, int k, bool left) {
6      int n = ring.size();
7      cout << "Ring printed " << (left ? "left" : "right") << " from index " << k << ": ";
8      for (int i = 0; i < n; ++i) {
9          cout << ring[(left ? k - i + n : k + i) % n] << " ";
10     }
11     cout << endl;
12 }
13
14 void insertZeros(vector<int>& ring) {
15     for (int i = 0; i < ring.size(); ++i) {
16         if (i % 5 == 0) {
17             ring.insert(ring.begin() + i + 1, 0);
18             ++i;
19         }
20     }
21 }
22
23 int main() {
24     int size, k;
25     cout << "Enter the size of the ring: ";
26     cin >> size;
27     cout << "Enter the starting index (0-based): ";
28     cin >> k;
29
30     vector<int> ring(size);
31     cout << "Enter the elements of the ring: ";
32     for (int i = 0; i < size; ++i) {
33         cin >> ring[i];
34     }
35
36     printRing(ring, k, true);
37     insertZeros(ring);
38     printRing(ring, k, false);
39
40     return 0;
41 }

```

Task 4 - Lab# programming: VNS Lab 5(вариант 20)

```

1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5
6  int findMinUnique(const vector<vector<int>>& matrix) {
7      string elements = "";
8
9      for (const auto& row : matrix) {
10         for (int elem : row) {
11             elements += to_string(elem) + " ";
12         }
13     }
14
15     int minUnique = INT_MAX;
16     for (const auto& row : matrix) {
17         for (int elem : row) {
18             string target = to_string(elem) + " ";
19             size_t first = elements.find(target);
20             size_t last = elements.rfind(target);
21             if (first == last) {
22                 if (elem < minUnique) {
23                     minUnique = elem;
24                 }
25             }
26         }
27     }
28
29     return (minUnique == INT_MAX) ? -1 : minUnique;
30 }
31
32 int main() {
33     int rows, cols;
34
35     cout << "Enter the number of rows: ";
36     cin >> rows;
37     cout << "Enter the number of columns: ";
38     cin >> cols;
39
40     vector<vector<int>> matrix(rows, vector<int>(cols));
41
42     cout << "Enter the elements of the matrix row by row:" << endl;
43     for (int i = 0; i < rows; ++i) {
44         for (int j = 0; j < cols; ++j) {
45             cin >> matrix[i][j];

```

```

46     }
47 }
48
49 int minUnique = findMinUnique(matrix);
50
51 if (minUnique != -1) {
52     cout << "The minimum unique element is: " << minUnique << endl;
53 } else {
54     cout << "No unique elements found." << endl;
55 }
56
57 return 0;
58 }
59

```

Task 5 - Lab# programming: Algotester Lab 2(вариант 1)

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  int main() {
8      int N;
9      cin >> N;
10
11     vector<int> r(N);
12     for(int i = 0; i < N; i++) {
13         cin >> r[i];
14     }
15
16     auto minimal = min_element(r.begin(), r.end());
17     auto maximal = max_element(r.begin(), r.end());
18
19     vector<int> copy_1 = r;
20     copy_1.erase(find(copy_1.begin(), copy_1.end(), *maximal));
21
22     int diff_1 = *max_element(copy_1.begin(), copy_1.end()) - *min_element(copy_1.begin(), copy_1.end());
23
24     vector<int> copy_2 = r;
25     copy_2.erase(find(copy_2.begin(), copy_2.end(), *minimal));
26
27     int diff_2 = *max_element(copy_2.begin(), copy_2.end()) - *min_element(copy_2.begin(), copy_2.end());
28
29     cout << min(diff_1, diff_2);
30     return 0;
31 }
32
33

```

Task 6 - Lab# programming: Algotester Lab 3(вариант 3)

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main() {
7      string s;
8      cin >> s;
9
10     string result;
11     int count = 1;
12
13     for (int i = 1; i < s.length(); i++) {
14         if (s[i] == s[i - 1]) {
15             count++;
16         } else {
17             result += s[i - 1];
18             if (count > 1) {
19                 result += to_string(count);
20             }
21             count = 1;
22         }
23     }
24
25     result += s[s.length() - 1];
26     if (count > 1) {
27         result += to_string(count);
28     }
29
30     cout << result << endl;
31
32     return 0;
33 }

```

Task 7 - Practice# programming: Class Practice Task

```
1  #include <iostream>
2  #include <string>
3  #include <cctype>
4  using namespace std;
5
6  bool isPalindrome(const string& str, int start, int end)
7  |   if (start >= end) {
8  |       return true;
9  |   }
10 |   if (tolower(str[start]) != tolower(str[end])) {
11 |       return false;
12 |   }
13 |   return isPalindrome(str, start + 1, end - 1);
14 | }
15
16 bool isPalindrome(int num) {
17 |   if (num < 0) return false;
18 |   int original = num;
19 |   int reversed = 0;
20 |   while (num > 0) {
21 |       reversed = reversed * 10 + num % 10;
22 |       num /= 10;
23 |   }
24 |   return original == reversed;
25 | }
26
27 int main() {
28 |   string s;
29 |   cout << "Enter a word: ";
30 |   cin >> s;
31 |   if (isPalindrome(s, 0, s.length() - 1)) {
32 |       cout << s << " is a palindrome" << endl;
33 |   } else {
34 |       cout << s << " isnt a palindrome" << endl;
35 |   }
36
37 |   int num;
```

```
37     int num;
38     cout << "Enter a number: ";
39     cin >> num;
40     if (isPalindrome(num)) {
41         cout << num << " is a palindrome" << endl;
42     } else {
43         cout << num << " isnt a palindrome" << endl;
44     }
45
46     return 0;
47 }
48
```


Task 8.1 - Practice# programming: Self Practice Task#1

```
1  #include <iostream>
2  #include <unordered_set>
3  using namespace std;
4
5  int main() {
6      int N;
7      cin >> N;
8
9      int r[N];
10     for (int i = 0; i < N; ++i) {
11         cin >> r[i];
12     }
13
14     int a, b, c;
15     cin >> a >> b >> c;
16
17     unordered_set<int> to_remove = {a, b, c};
18     int filtered[N];
19     int filtered_size = 0;
20
21     for (int i = 0; i < N; ++i) {
22         if (to_remove.find(r[i]) == to_remove.end())
23             filtered[filtered_size++] = r[i];
24     }
25
26
27     int M = filtered_size - 1;
28     if (M <= 0) {
29         cout << 0 << endl;
30         return 0;
31     }
32
33     int sums[M];
34     for (int i = 0; i < M; ++i) {
35         sums[i] = filtered[i] + filtered[i + 1];
36     }
37
38     cout << M << endl;
39     for (int i = 0; i < M; ++i) {
40         cout << sums[i] << " ";
41     }
42     cout << endl;
43
44     return 0;
45 }
```

Task 8.2 - Practice# programming: Self Practice Task#2

```
1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <algorithm>
5
6  using namespace std;
7
8  int main() {
9      int n, k;
10     cin >> n >> k;
11
12     vector<int> s(n), b(k);
13     for (int i = 0; i < n; i++) {
14         cin >> s[i];
15     }
16     for (int i = 0; i < k; i++) {
17         cin >> b[i];
18     }
19
20     sort(s.rbegin(), s.rend());
21     sort(b.rbegin(), b.rend());
22
23     long long totalBeauty = 0;
24     multiset<int> brushes(b.begin(), b.end());
25
26     for (int i = 0; i < n; i++) {
27         if (brushes.empty()) {
28             break;
29         }
30
31         auto it = brushes.end();
32         --it;
33         int maxBrush = *it;
34         totalBeauty += static_cast<long long>(s[i]) * maxBrush;
35
36         brushes.erase(it);
37         if (maxBrush > 1) {
38             brushes.insert(maxBrush - 1);
39         }
40     }
41
42     cout << totalBeauty << endl;
43
44     return 0;
45 }
```

5) Результати виконання завдань та фактично затрачений час

Task 3 - Lab# programming: VNS Lab 4(варіант 20)

```
Enter the size of the ring: 6
Enter the starting index (0-based): 2
Enter the elements of the ring: 1 2 5 10 11 15
Ring printed left from index 2: 5 2 1 15 11 10
Ring printed right from index 2: 2 5 10 11 0 15 1 0
```

Фактично затрачений час: 15 хв

Task 4 - Lab# programming: VNS Lab 5(варіант 20)

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter the elements of the matrix row by row:
1 2 4
4 3 1
5 2 1
```

Фактичний час затрачений на виконання: 20хв

Task 5 - Lab# programming: Algotester Lab 2(варіант 1)

```
5
1 2 2 4 4
2
```


✓ Lab 2v1

Compiler	Result	Time (sec.)	Memory (MiB)	Actions
C++ 23	Accepted	0.003	1.246	View

Фактичний час затрачений на виконання: 20хв

Task 6 - Lab# programming: Algotester Lab 3(варіант 3)

```
AAAAAAAAAAAAAAAAQQQQQQQQRRRRRRRFSHHHHOO
A15Q9R8FSH502
```

<div>  Lab 3v3 </div>				
Compiler	Result	Time (sec.)	Memory (MiB)	Actions
C++ 23	Accepted	0.003	1.230	View

Фактичний час затрачений на виконання: 30хв

Task 7 - Practice# programming: Class Practice Task


```
Enter a word: Radar
Radar is a palindrome
Enter a number: 1234567
1234567 isnt a palindrome
```

```
Enter a word: wall
wall isnt a palindrome
Enter a number: 12344321
12344321 is a palindrome
```

Фактичний час затрачений на виконання: 40хв

Task 8.1 - Practice# programming: Self Practice Task#1

```
5
5 4 3 2 1
4 5 6
2
5 3
```

<div>  Lab 2v2 </div>				
Compiler	Result	Time (sec.)	Memory (MiB)	Actions
C++ 23	Accepted	0.003	1.234	View

Фактичний час затрачений на виконання: 30хв

Task 8.2 - Practice# programming: Self Practice Task#2

```
3 4
4 3 7
1 10 9 2
133
```

Задача	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)
0033 - Коля, Вася і таємна кімната	C++ 23	Зараховано	0.102	5.832

Фактичний час затрачений на виконання: 50хв

6) Робота з кодадою

Відео-зустріч:



Висновок: У межах практичних та лабораторних робіт блоку №4 я вивчив низку нових понять, таких як: масиви різних типів, вказівники, посилання, ознайомився з динамічними структурами даних та алгоритмами їх обробки. Отримані знання та навички дозволяють ефективно працювати з великими обсягами даних, забезпечуючи їхню структурованість, оптимізацію та логічну цілісність, що є важливим для розробки гнучких і продуктивних програм.