Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



# Звіт

про виконання

**Лабораторних та практичних робіт №4*:*** «Мови та парадигми програмування»

***з розділу***: «Лінійні та розгалужені алгоритми. Умовні оператори. Константи,

змінні»

***Виконав:***

студент групи ШІ-11 Сабадило Марко

Львів 2023

### Одновимірні масиви:

#### **Одновимірні масиви:**

**Одновимірні масиви у мові C++ дозволяють зберігати однакові типи даних у послідовності. Елементи масиву індексуються числами, починаючи з 0.**

**Джерела:**

* [**cppreference.com - Arrays**](https://en.cppreference.com/w/cpp/language/array)

### Двовимірні масиви:

#### **Двовимірні масиви:**

**Двовимірні масиви представляють собою таблиці, де дані розташовані у вигляді рядків і стовпців. Елементи двовимірного масиву індексуються двома числами.**

**Джерела:**

* [**cppreference.com - Arrays**](https://en.cppreference.com/w/cpp/language/array)

### Вказівники та посилання:

#### **Вказівники:**

**Вказівники вказують на адресу пам'яті об'єкта. Вони дозволяють працювати з динамічною пам'яттю та передавати адреси об'єктів у функції.**

**Джерела:**

* [**cppreference.com - Pointers**](https://en.cppreference.com/w/cpp/language/pointer)

#### **Посилання:**

**Посилання - це альтернативний спосіб роботи з об'єктами, який дозволяє звертатися до об'єкта за його оригінальною назвою.**

**Джерела:**

* [**cppreference.com - References**](https://en.cppreference.com/w/cpp/language/reference)

### Динамічні масиви:

#### **Динамічні масиви:**

**Динамічні масиви створюються в режимі виконання за допомогою оператора** new**. Це дозволяє задавати розмір масиву під час виконання програми.**

**Джерела:**

* [**cppreference.com - Dynamic memory management**](https://en.cppreference.com/w/cpp/memory/new)

### Структури даних:

#### **Структури даних:**

**Структури даних в C++ дозволяють комбінувати різні типи даних під однією назвою. Структури дозволяють організовувати дані у складний об'єкт.**

**Джерела:**

* [**cppreference.com - Structures**](https://en.cppreference.com/w/cpp/language/struct)

### Вкладені структури:

#### **Вкладені структури:**

**Вкладені структури в C++ дозволяють вкладати одну структуру всередину іншої. Це корисно для організації складних структур даних.**

**Джерела:**

* [**cppreference.com - Structures**](https://en.cppreference.com/w/cpp/language/struct)

### Алгоритми обробки та робота з масивами та структурами:

#### **Алгоритми обробки масивів та структур:**

**Алгоритми обробки масивів та структур даних включають в себе різні методи сортування, пошуку та інші операції.**

**Джерела:**

* [**cppreference.com - Algorithms**](https://en.cppreference.com/w/cpp/algorithm)

1. Зображення, що містить чорно-білий, мистецтво, дизайн, візерунок

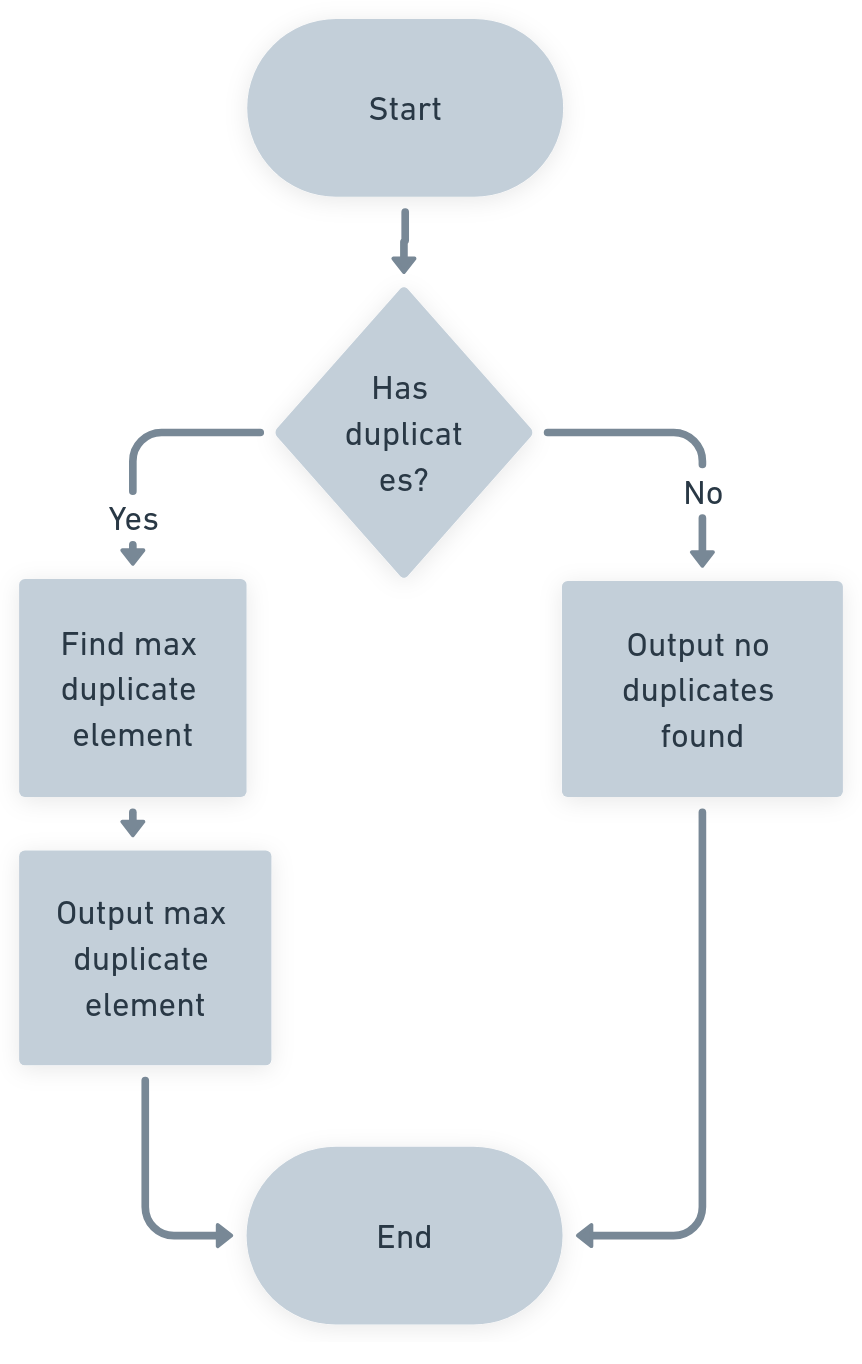
   Автоматично згенерований опис

Зображення, що містить знімок екрана, чорно-білий, дизайн

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, схема, дизайн

Автоматично згенерований опис



Зображення, що містить текст, знімок екрана, чорно-білий, дизайн

Автоматично згенерований опис

**Код програм з посиланням на зовнішні ресурси:**

Завдання № 1

|  |
| --- |
| #include <iostream> #include <unordered\_set> #include <vector>  using namespace std;  int main() {  // Введення розмірів масивів  int N, M;  cin >> N;   // Введення першого масиву  vector<int> arr1(N);  for (int i = 0; i < N; ++i) {  cin >> arr1[i];  }   cin >> M;   // Введення другого масиву  vector<int> arr2(M);  for (int i = 0; i < M; ++i) {  cin >> arr2[i];  }   // Знайдемо кількість елементів, які наявні в обох масивах одночасно  unordered\_set<int> set1(arr1.begin(), arr1.end());  unordered\_set<int> set2(arr2.begin(), arr2.end());   int common\_elements = 0;  for (int element : set1) {  if (set2.find(element) != set2.end()) {  common\_elements++;  }  }   // Знайдемо кількість унікальних елементів в обох масивах  unordered\_set<int> union\_set(arr1.begin(), arr1.end());  union\_set.insert(arr2.begin(), arr2.end());   int unique\_elements = union\_set.size();   // Виведемо результат  cout << common\_elements << endl;  cout << unique\_elements << endl;   return 0; } |

[Посилання на Pull reqest](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1105/files#diff-71b432da4c7f19c42b5df74848c27e374827f1469e20011282bc258b80cab1ed)

Завдання № 2

|  |
| --- |
| ##include <iostream> #include <vector> #include <algorithm>  using namespace std;  int main() {  // Введення розмірів та елементів першого масиву  int N;  cin >> N;  vector<int> arrayA(N);  for (int i = 0; i < N; ++i) {  cin >> arrayA[i];  }   // Введення розмірів та елементів другого масиву  int M;  cin >> M;  vector<int> arrayB(M);  for (int i = 0; i < M; ++i) {  cin >> arrayB[i];  }   // Завдання 1: кількість спільних елементів  int commonElementsCount = 0;  for (int i : arrayA) {  if (std::find(arrayB.begin(), arrayB.end(), i) != arrayB.end()) {  commonElementsCount++;  }  }   // Завдання 2: кількість унікальних елементів в обох масивах  vector<int> combinedArray;  combinedArray.reserve(arrayA.size() + arrayB.size());  combinedArray.insert(combinedArray.end(), arrayA.begin(), arrayA.end());  combinedArray.insert(combinedArray.end(), arrayB.begin(), arrayB.end());   // Сортування та видалення дублікатів  sort(combinedArray.begin(), combinedArray.end());  combinedArray.erase(unique(combinedArray.begin(), combinedArray.end()), combinedArray.end());   // Виведення результатів  cout << commonElementsCount << endl;  cout << combinedArray.size() << endl;   return 0; } |

[Посилання на Pull reqest](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1105/files#diff-cea101c53d9f83607a5f364c35b319d2ac81d57a7f4836984747d38b32619884)

Завдання № 3

|  |
| --- |
| ##include <iostream>  using namespace std;  const int MAX\_SIZE = 10;  class CircularArray { private:  int array[MAX\_SIZE];  int size;  int front;  int rear;  public:  CircularArray(int s) : size(s), front(-1), rear(-1) {  for (int i = 0; i < MAX\_SIZE; ++i) {  array[i] = -1;  }  }   void addElement(int value) {  if (front == -1 && rear == -1) {  front = rear = 0;  } else {  rear = (rear + 1) % size;  if (rear == front) {  front = (front + 1) % size;  }  }  array[rear] = value;  }   void printArray(int start, int end, int direction) {  int i = start;  while (true) {  cout << array[i] << " ";   if (i == end) {  break;  }   i = (i + direction + size) % size;  }  cout << endl;  }   void addSpecialElements() {  int firstElement = array[front];  int thirdLastElement = array[(rear - 2 + size) % size];   addElement(firstElement);  addElement(thirdLastElement);  addElement(thirdLastElement);  addElement(thirdLastElement);  }   // Гетери для отримання значень приватних членів  int getFront() const {  return front;  }   int getRear() const {  return rear;  } };  int main() {  int K;   cout << "Enter the size of the circular array: ";  cin >> K;   CircularArray circularArray(K);   for (int i = 1; i <= K; ++i) {  circularArray.addElement(i);  }   cout << "Original circular array (left direction): ";  circularArray.printArray(circularArray.getFront(), circularArray.getRear(), 1);   circularArray.addSpecialElements();   cout << "Modified circular array (right direction): ";  circularArray.printArray(circularArray.getFront(), circularArray.getRear(), -1);   return 0; } |
|  |

[Посилання на pullreqet](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1105/commits/72b995a7d793a506bf32b40cd86101bb35e992d0" \l "diff-d7dfdf4b7fa190bb240e2c61c8238ceb46f529733fa37d60589e384ccea53853)

Завдання № 3.2

|  |
| --- |
| ##include <iostream> #include <unordered\_map> #include <vector> #include <cstdlib> #include <ctime> // Для використання функцій time та srand  using namespace std;  // Функція для пошуку максимального повторюваного елемента int findMaxDuplicate(const vector<vector<int>>& matrix) {  unordered\_map<int, int> countMap;  int maxDuplicate = INT\_MIN;   // Підрахунок кількості повторюваних елементів  for (const auto& row : matrix) {  for (int element : row) {  countMap[element]++;  if (countMap[element] > 1 && element > maxDuplicate) {  maxDuplicate = element;  }  }  }   return maxDuplicate; }  // Функція для рандомного заповнення масиву void fillMatrixRandomly(vector<vector<int>>& matrix) {  srand(time(NULL)); // Ініціалізація генератора випадкових чисел  for (auto& row : matrix) {  for (int& element : row) {  element = rand() % 10 + 1; // Рандомні значення від 1 до 10  }  } }  // Функція для виведення масиву void printMatrix(const vector<vector<int>>& matrix) {  for (const auto& row : matrix) {  for (int element : row) {  cout << element << " ";  }  cout << endl;  } }  int main() {  // Розмір масиву  const int rows = 4;  const int cols = 4;   // Створення та заповнення масиву  vector<vector<int>> matrix(rows, vector<int>(cols));  fillMatrixRandomly(matrix);   // Виведення заповненого масиву  cout << "Randomly filled matrix:" << endl;  printMatrix(matrix);   // Знаходження та виведення максимального повторюваного елемента  int result = findMaxDuplicate(matrix);  if (result != INT\_MIN) {  cout << "\nMax Duplicate Element: " << result << endl;  } else {  cout << "\nNo duplicates found." << endl;  }   return 0; } |
|  |

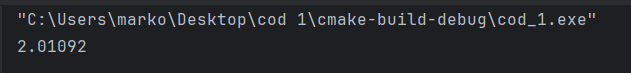
[Посидання на Pullreqest](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1105/files#diff-f6e6ea5cf135f77fdf5ec14d0285611971d58518f2ea3bb30b9ed0288072f75a)

Завдання № 4

|  |
| --- |
| #include <iostream> #include <string>  using namespace std;  bool isPalindrome(const string& str, int start, int end) {  if (start >= end) {  return true;  }   // рекурсивний виклик для перевірки наступних символів  return (str[start] == str[end]) && isPalindrome(str, start + 1, end - 1); }  bool isPalindrome(int number) {  // конвертація цілого числа в рядок для використання рядкової версії функції  string str = to\_string(number);    return isPalindrome(str, 0, str.length() - 1); }  int main() {  string word;  cout << "Enter a word: ";  cin >> word;   if (isPalindrome(word, 0, word.length() - 1)) {  cout << " word is palindrome." << endl;  } else {  cout << "word is not palindrome." << endl;  }   int number;  cout << "Enter an integer: ";  cin >> number;   if (isPalindrome(number)) {  cout << "integer is palindrome." << endl;  } else {  cout << " integer is not palindrome." << endl;  }   return 0; } |

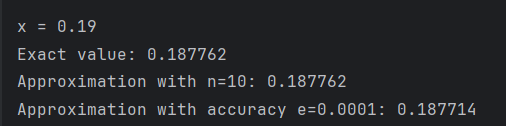
[Посилання на Pull reqest](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1105/files#diff-9717b2137c51facd2ea927d2aeb871747154966ca5295b257de437ab37b5b3fc)

Завданя № 1



Час, витрачений на виконання завдання - 20 хвилин

Завдання № 2



Час, витрачений на виконання завдання - 20 хвилин

Завдання № 3

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

Час, витрачений на виконання завдання - 20 хвилин

Завдання № 3.2

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Час, витрачений на виконання завдання - 20 хвилин

Завдання № 4

Зображення, що містить текст, знімок екрана, Шрифт, алгебра

Автоматично згенерований опис