Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

**про виконання лабораторних та практичних робіт блоку № 4**

На тему: «Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.»

***з дисципліни:*** «Мови та парадигми програмування»

до:

ВНС Лабораторної Роботи №4

ВНС Лабораторної Роботи №5

Алготестер Лабораторної Роботи №2

Алготестер Лабораторної Роботи №3

Практичних Робіт №4

***Виконав:***

студент групи ШІ-14

Сокаль Олег Ігорович

# **Тема роботи:**

Ознайомлення з одновимірними та багатовимірними масивами, ініціалізація одновимірних та багатовимірних масивів, принципи доступу до елементів масиву, вказівники та посилання, адресна арифметика, поняття про динамічні масиви, конструкцію std::vector

# **Мета роботи:**

Ознайомитись з різними видами масивів, отримати поняття про багатовимірні масиви. Ознайомитесь з вказівниками та посиланнями(що реалізовані на їх основі), зрозуміти принцип роботи та випадки застосування вказівників та посилань, отримати поняття про адресну арифметику і її застосування для індексації масивів, ознайомитись з динамічними масивами та методом їх ініціалізації, отримати поняття про модифіковані масиви std::array та std::vector

Завдання:

* Oleh Sokal - Epic 4 Task 1 - Theory Education Activities
* Oleh Sokal - Epic 4 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-8)
* Oleh Sokal - Epic 4 Task 3 - Lab# programming: VNS Lab 4
* Oleh Sokal - Epic 4 Task 4 - Lab# programming: VNS Lab 5
* Oleh Sokal - Epic 4 Task 5 - Lab# programming: Algotester Lab 3
* Oleh Sokal - Epic 4 Task 6 - Practice# programming: Class Practice Task
* Oleh Sokal - Epic 4 Task 7  - Practice# programming:  Self Practice Task
* Oleh Sokal - Epic 4 Task 8  - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* Oleh Sokal - Epic 4 Task 9 - Results Evaluation and Release

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Одновимірні та багатовимірні статичні масиви
* Тема №2: Вказівники
* Тема №3: Посилання
* Тема №4: Адресна арифметика, доступ до елементів масива через вказівники
* Тема №5: Динамічні одновимірні масиви, початкове уявлення про динамічні багатовимірні масиви
* Тема №6: Конструкції std::array та std::vector

1. Індивідуальний план опрацювання теорії:

* Тема №1: Одновимірні та багатовимірні статичні масиви
  + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Ініціалізація одновимірних масивів
    - Ініціалізація багатовимірних масивів
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
  + Звершення опрацювання теми: 26.11.23
* Тема №2: Вказівники
  + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Поняття про адреси змінних в пам’яті
    - Поняття про застосування поінтерів для збереження адрес
    - Використання поінтерів в масивах та функціях
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
  + Звершення опрацювання теми: 26.11.23

Тема №3: Посилання

* Джерела Інформації
  + - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Ознайомлення з посиланнями(референсами) та їх застосуванням як псевдонімів змінних
    - Поняття про принцип реалізації референсів через поінтери
    - Використання референсів в функціях
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23
* Тема №4: Адресна арифметика, доступ до елементів масива через вказівники
* Джерела Інформації

Практика в НУ “ЛП”

* + - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Зв’язок поняття поінтера з масивами
    - Доступ до елементів масива через вказівники
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23
* Тема №5: Динамічні одновимірні масиви, початкове уявлення про динамічні багатовимірні масиви
* Джерела Інформації
  + - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Ініціалізація динамічного одновимірного масиву
    - Методи ініціалізації динамічного двовимірного масиву
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23
* Тема №6: Конструкції std::array та std::vector

Джерела Інформації

* + - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Std::vector як модифікація динамічного масиву
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 29.11.23

# **Виконання роботи:**

**VNS LAB4**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 VNS Lab4

11.

1) Сформувати одновимірний масив цілих чисел, використовуючи генератор

випадкових чисел.

2) Роздрукувати отриманий масив.

3) Знищити 5 останніх елементів масиву.

4) Додати в початок масиву 3 елементи зі значенням M[I+1]+2.

5) Роздрукувати отриманий масив.

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

using namespace std;

int main() {

int random\_array[10] = {1, 5, 12, 97, 18, 36, 77, 19, 56, 23};

cout << "Початковий масив: ";

for (int i = 0; i < 10; ++i) {

cout << random\_array[i] << " ";

}

cout << endl;

int elements\_to\_remove = 6;

int new\_array\_size = 10 - elements\_to\_remove;

int new\_elements[3];

for (int i = 0; i < 3; ++i) {

new\_elements[i] = random\_array[i + 1] + 2;

}

for (int i = new\_array\_size; i >= 0; --i) {

random\_array[i + 3] = random\_array[i];

}

for (int i = 0; i < 3; ++i) {

random\_array[i] = new\_elements[i];

}

cout << "Оновлений масив: ";

for (int i = 0; i <= new\_array\_size + 3; ++i) {

cout << random\_array[i] << " ";

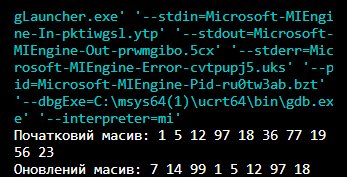
}

cout << endl;

return 0;

}

## **3. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 20 хв

**VNS LAB5**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №2 VNS Lab5

19. Задано рядок з N^2 цифр. Встановити чи можна, розбивши рядок на

підстрічки довжиною N, записати їх у рядки двовимірного масиву N x N по

одній цифрі в одному елементі так, щоб вони в першому стовпці

розташувалися в порядку зростання.

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <cmath>

using namespace std;

void ConvertToDoubleMassif(int n[], int k) {

int a = sqrt(k);

for (int i = 0; i < k - 1; ++i) {

for (int j = 0; j < k - i - 1; ++j) {

if (n[j] > n[j + 1]) {

int temp = n[j];

n[j] = n[j + 1];

n[j + 1] = temp;

}

}

}

int n1[a][a];

int s = 0;

for (int i = 0; i < a; i++) {

for (int j = 0; j < a; j++) {

n1[j][i] = n[s];

s++;

}

}

for (int i = 0; i < a; i++) {

for (int j = 0; j < a; j++) {

cout << n1[i][j] << " ";

}

cout << endl;

}

}

int main() {

int size, size1;

int arr[] = {6, 2, 5, 8};

int arr1[] = {4, 2, 3, 5, 9, 7, 1, 6, 0};

size = sizeof(arr)/ sizeof(arr[0]);

size1 = sizeof(arr1)/ sizeof(arr1[0]);

cout << "Matrix from arr:" << endl;

ConvertToDoubleMassif(arr, size);

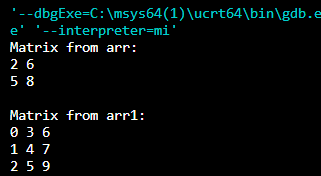
cout << "\nMatrix from arr1:" << endl;

ConvertToDoubleMassif(arr1, size1);

return 0;

}

## **3. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 30 хвилин

**Algotester Lab3V2**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання 3 Algotester Lab 3v2

**Вам дано 2 масиви розміром N та M. Значення у цих масивах унікальні.**

**Ваше завдання вивести у першому рядку кількість елементів, які наявні в обох масивах одночасно, у другому кількість унікальних елементів в обох масивах разом.**

## **2. Код програм з посиланням на зовнішні ресурси:**

## #include <iostream>

## using namespace std;

## int main() {

## int arr[100] = {0}, arr1[100] = {0};

## int n, n1;

## int unique = 0;

## int similar = 0;

## cin >> n;

## for (int i = 0; i < n; i++) {

## cin >> arr[i];

## }

## cin >> n1;

## for (int i = 0; i < n1; i++) {

## cin >> arr1[i];

## }

## for (int i = 0; i < n1; i++) {

## int found = 0;

## for (int j = 0; j < n; j++) {

## if (arr1[i] == arr[j]) {

## similar++;

## }

## }

## }

## unique= (n1+n) - similar;

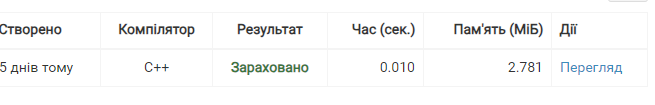
## cout << similar << endl;

## cout << unique << endl;

## return 0;

## }

## **3. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 10 хвилин

**Practice Task # 4**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №4 Class-Practice Task # 4

# **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:
   1. Реалізуйте рекурсивну функцію *isPalindrome*, яка перевіряє, чи заданий рядок є паліндромом.
2. Приклад визначення функції:
   1. *bool isPalindrome(const string& str, int start, int end);*
3. Перевантаження функцій:
   1. Перевантажте функцію *isPalindrome* для роботи з цілими значеннями.
   2. *bool isPalindrome*(ціле число);
4. Рекурсія:
   1. 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.

### Кроки реалізації

* Визначте та реалізуйте рекурсивну функцію isPalindrome для рядків.
* Визначте та реалізуйте перевантажену функцію isPalindrome для цілих чисел. Перетворити ціле число на рядок і використовуйте рядкову версію функції, щоб перевірити, чи це паліндром.

## **2. Код програм з посиланням на зовнішні ресурси:**

## **#include <iostream>**

## **#include <string>**

## **using namespace std;**

## **void Recursion(const string& str, int midleft, int midright) {**

## **if (str[midleft] == str[midright]) {**

## **if (midleft == 0) {**

## **cout << "YES!";**

## **return;**

## **}**

## **Recursion(str, midleft - 1, midright + 1);**

## **} else {**

## **cout << "NO!";**

## **}**

## **}**

## **void Recursion(const int& num, int midleft, int midright) {**

## **string str = to\_string(num);**

## **if (str[midleft] == str[midright]) {**

## **if (midleft == 0) {**

## **cout << "YES!";**

## **return;**

## **}**

## **Recursion(str, midleft - 1, midright + 1);**

## **} else {**

## **cout << "NO!";**

## **}**

## **}**

## **int main() {**

## **string str;**

## **cin >> str;**

## **int mid = str.length() / 2;**

## **int midleft = mid - 1;**

## **int midright = (str.length() % 2 == 0) ? mid : mid + 1;**

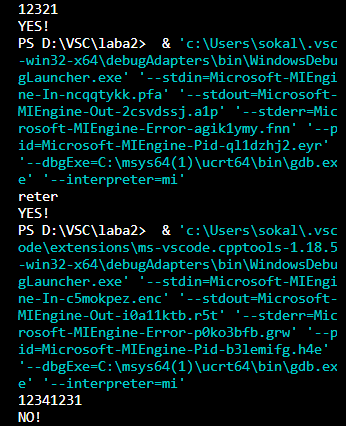
## **Recursion(str, midleft, midright);**

## **return 0;**

## **}**

## 

## **3 . Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 40 хвилин

**Self-Practice(1)(Дракон і Голод)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №5 Self-Practice(Дракон і Голод)

Дракон Віталій голодний. Перед тим як сідати їсти, він завжди генерує таку довільну послідовність з

n невід’ємних чисел, що її медіана дорівнює

m, а сума дорівнює-s

. Сьогодні Віталій дуже голодний, і думати не може. Тому таку послідовність доведеться згенерувати Вам.

Медіаною послідовності називається число, яке розташоване посередині, якщо послідовність посортувати за зростанням. Наприклад, медіаною послідовності [1, 4, 7, 8, 2] є число 4, а медіаною послідовності [1, 2, 3, 4] є число 3 (у випадку парної довжини береться число, яке є правіше).

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

using namespace std;

int main() {

int n, m, s, placeOfMid, rightPart;

cin >> n >> m >> s;

int arr[1000] = {0};

placeOfMid = n / 2 + 1;

rightPart = n - placeOfMid;

for (int i = rightPart; i < n; ++i) {

s += arr[i];

}

arr[placeOfMid] = m;

if ((s-m) < (rightPart \* arr[placeOfMid])) {

cout << "-1";

return 0;

}

for (int i = placeOfMid; i < n; i++) {

arr[i-1] = m;

s -= m;

}

arr[n - 1] = s;

for (int i = 0; i < n; ++i) {

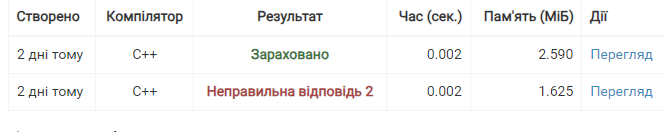
cout << arr[i] << " ";

}

return 0;

}

## **3. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 30 хвилин  
**Self-Practice(2)(Algotester Lab5V1)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №6 Self-Practice(Algotester Lab4 V3)

У світі Атод сестри Ліна і Рілай люблять грати у гру. У них є дошка із 8-ми рядків і 8-ми стовпців. На перетині

i-го рядкa і

j-го стовпця лежить магічна куля, яка може світитись магічним світлом (тобто у них є 64 кулі). На початку гри деякі кулі світяться, а деякі ні... Далі вони обирають

N куль і для кожної читають магічне заклиння, після чого всі кулі, які лежать на перетині стовпця і рядка обраної кулі міняють свій стан (ті що світяться - гаснуть, ті, що не світяться - загораються).

Також вони вирішили трохи Вам допомогти і придумали спосіб як записати стан дошки одним числом

a із 8-ми байт, а саме (див. Примітки):

* Молодший байт задає перший рядок матриці;
* Молодший біт задає перший стовпець рядку;
* Значення біту каже світиться куля чи ні (0 - ні, 1 - так);

Тепер їх цікавить яким буде стан дошки після виконання

N заклинань і вони дуже просять Вас їм допомогти.

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

using namespace std;

int main() {

unsigned long long n;

char arr[64] = {0};

unsigned long long a;

unsigned long long b = 0;

int untouchablePlace;

int correctPlace;

int rows, columns;

cin >> a;

cin >> n;

for (int i = 0; i < 64; i++) {

if ((a >> i) & 1) {

arr[i] = 1;

}

}

for (int i = 0; i < n; i++) {

cin >> rows >> columns;

untouchablePlace = (rows - 1) \* 8 + columns - 1;

arr[untouchablePlace] = (arr[untouchablePlace] == 1) ? 0 : 1;

for (int j = 0; j < 8; j++) {

correctPlace = j \* 8 + columns - 1;

if (correctPlace != untouchablePlace) {

arr[correctPlace] = (arr[correctPlace] == 1) ? 0 : 1;

}

}

for (int j = 0; j < 8; j++) {

correctPlace = (rows - 1) \* 8 + j;

if (correctPlace != untouchablePlace) {

arr[correctPlace] = (arr[correctPlace] == 1) ? 0 : 1;

}

}

}

for (int i = 63; i >= 0; i--) {

if (arr[i] == 1) {

b |= (1ULL << i);

}

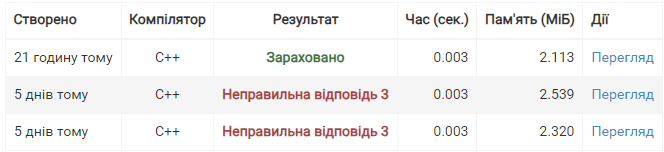
}

cout << b << endl;

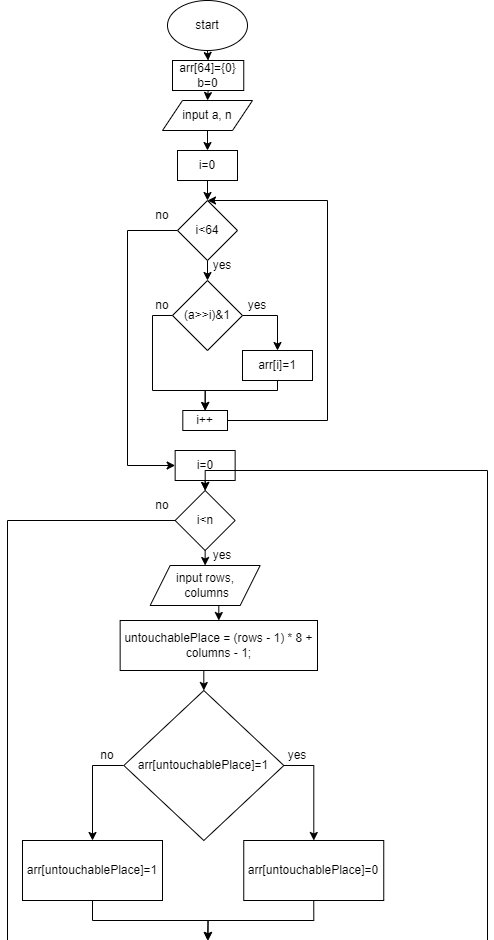
return 0;

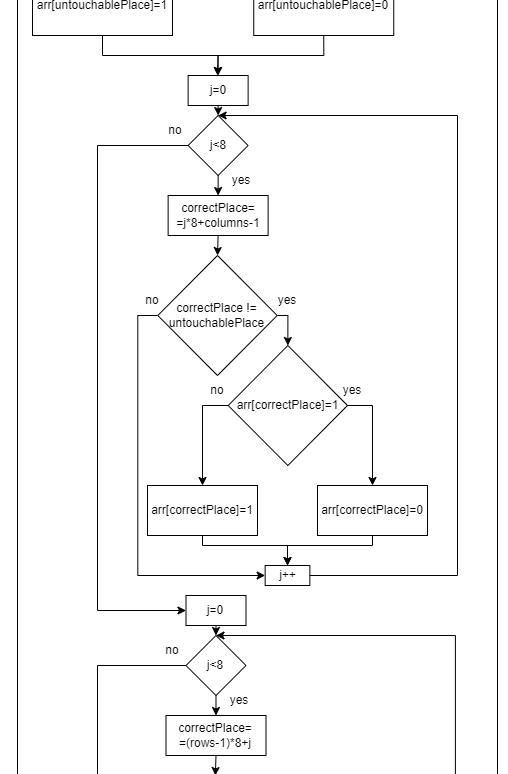
}

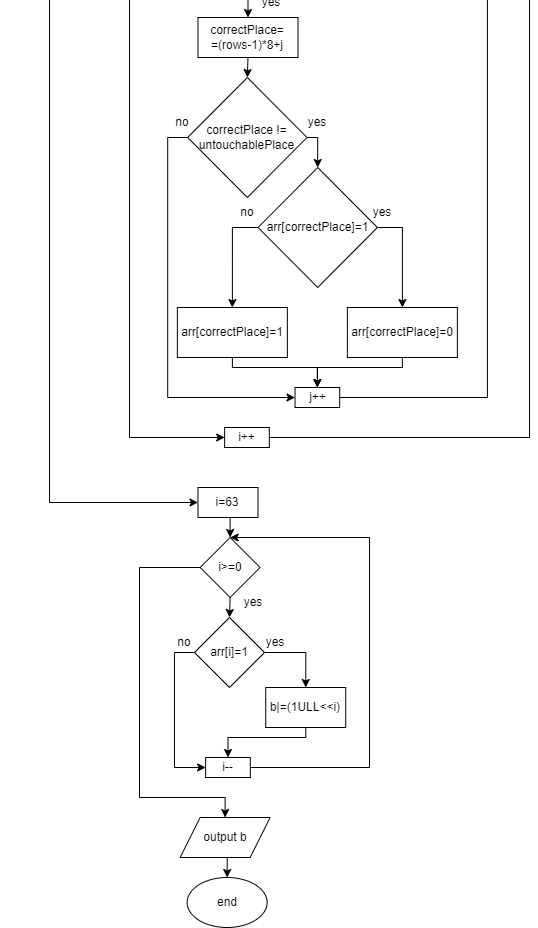
## **3. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 4 години;



****

****

**Algotester Lab2V2**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання 3 Algotester Lab 2v2

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 сусідніх елементів нового масиву.

Далі необхідно вивести масив сум на екран.

## **2. Код програм з посиланням на зовнішні ресурси:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main() {**

**int N, m;**

**cin >> N;**

**vector<int> r(N);**

**int a, b, c;**

**for (int i = 0; i < N; i++) {**

**cin >> r[i];**

**}**

**cin >> a >> b >> c;**

**for (int i = 0; i < N; i++) {**

**if (r[i] == a || r[i] == b || r[i] == c) {**

**r.erase(r.begin() + i);**

**i--;**

**N--;**

**}**

**}**

**m=N-1;**

**if(N<2){**

**m=0;**

**}**

**cout << m << endl;**

**for (int i = 0; i < m; i++) {**

**cout << r[i] + r[i + 1] << endl;**

**}**

**return 0;**

**}**

# **Висновки:**

Отже, в ході роботи над епіком 4 я отримав знання про те, що таке масиви і які вони бувають. Були вивчені одновимірні та двовимірні масиви, принципи їх ініціалізації та доступу до членів. Окремо було вивчено поняття поінтерів та референсів, зокрема як їх ефективно використовувати в функціях та масивах. Було розглянуто поняття адресної арифметики та доступу до елементів масиву через вказівники. Я вивчив поняття динамічних масивів,як їх ініціалізовувати та яку роль в цьому грають вказівники. Крім того, в якості додаткової інформації я вивчив std::vector як модифіковані різновид динамічного масиву.

Pull-Request: