ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ГАБРОВО

Катедра „Компютърни системи и технологии“

**КУРСОВА РАБОТА**

Тема: Множество е съвкупност от не повтарящи се елементи. Дадени са два едномерни масива с естествени числа от интервала [0…299]. Да се състави програма, която проверява дали всички числа от първия масив се съдържат във втория масив. Да се създаде трети масив, който да съдържа само повтарящите се числа и да се намери най-голямата стойност в него. За всяка обработка да се използва отделна функция.

Разработил: Ваня Тодорова Ванева

ф. номер: 22472128

курс: 1 / Първи

специалност: СКИ

Проверил:............................

Подпис: .............

Дата: ...............

1. **Теоретична част**
   1. **Дефиниране и използване на структури масиви в програмен език C++**

Какво е масив

Масив или Array е съвкупност от еднотипни данни, организирани последователно.

Всеки елемент е достъпен чрез индекс (index), който показва мястото на елемента с масива. Последователността на индексите е нарастваща, като започват от 0 до n - 1, където n е размерът на масива.



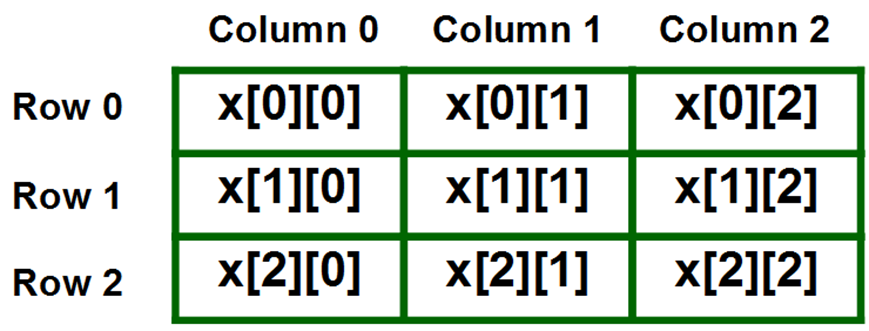
Видове

Размерът на масива не се променя след като е зададен, т.е. той е фиксиран(на статичните масиви).

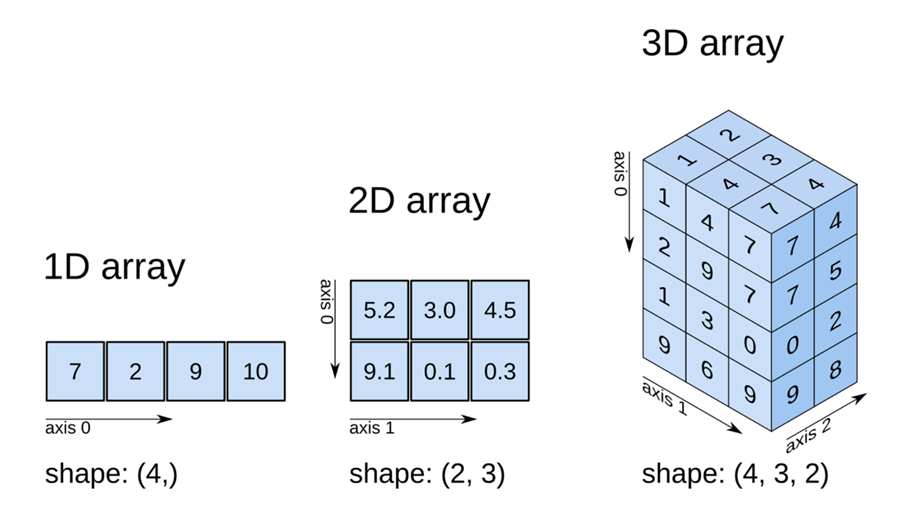
Динамичните масиви имат променлив размер, който се сменя по време на изпълнение на програмата.

Съществуват и многомерни масиви. Примери за многомерни масиви са двумерни, тримерни и т.н.

Двумерните (матрици) са с два индекса, един за ред и един за стълб. Тези масиви са за съхраняване на данни в табличен вид. Представляват масиви, съдържащи масиви. Всеки ред може да се представи като отделен масив.



Тримерните масиви са прекалено сложни за представяне, затова не се ползва често. Имат три index-а.



Указатели

Указателите са променливи, които съхраняват/сочат към адрес на друга променлива в паметта.

**int a;**

**int \*p = &a;**

**\*** е оператор за указател, **&a** връща адреса на **a** в паметта.

Масивите и указателите са свързани в C++. Като декларираме масив, името на масива действа като указател към първия елемент на масива.

**int arr[5] = {1, 2, 3, 4, 5};**

**int \*p = arr;**

В този случай **\*p** е еквивалентно на **arr[0].**

Въпреки, че името на масива е като указател на първия елемент, то не е променлива – не може да се модифицира.

Дефиниране на масив

Едномерен

Статичен едномерен масив се дефинира по този начин с предварително зададен размер.

**тип име[размер];**

Или компилаторът може сам да избере размера, когато елементите са инициализирани.

**тип име[] = {елемент1, елемент2};**

Динамичните едномерни масиви се дефинират с **new.**

**тип\* име = new тип[променлива, задавана от клавиатурата];**

„\*“ след типа е указател, който сочи към началото на динамично разпределения масив.

И се изтриват с помощта на **delete.**

**delete[] arr; //**Освобождаване на паметта.

Може да се ползва **std::vector** (готов динамичен контейнер). Автоматично управлява паметта.

**#include <vector>**

**vector<тип> име;**

**име.push\_back(1); //**Добавяне на елементи

**име.pop\_back(); //**Премахване на последния елемент

Двумерен

Статичен двумерен масив се дефинира по този начин.

**тип име[брой\_редове][брой\_колони];**

Или може да се инициализира с предварително зададени стойности без размер.

**тип име[][] = {**

**{елемент1, елемент2, елемент3},**

**{елемент4, елемент5, елемент6}**

**};**

Динамичния двумерен масив се дефинира с два указателя за двата индекса.

**тип\*\* име = new тип\*[брой\_редове];**

Освобождаването на паметта става по същия начин като едномерните масиви.

Използване на **std::vector** за двумерен масив

**#include <vector>**

**vector<vector<тип>> име(брой\_редове, vector<тип>(брой\_колони));**

Добавяне и премахване на елементи от двумерен масив става чрез вложени цикли (външен и вътрешен).

Използване на структури масив

Достъпът до елементите от масив се осъществява чрез посочване на индекса на елемента.

**string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};**

**cout << cars[0]; //**“Volvo“

Промяна на елемент от масив става чрез индекса отново.

**string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};**

**cars[0] = "Opel";**

**cout << cars[0]; //**“Opel“

Обхождане на масив става с цикли, най-често с **for**.

**for (int i = 0; i < 4; i++) {**

**cout << cars[i] << endl; //**Извежда всеки елемент на нов ред

**}**

Обхождане на масив чрез указатели, като работим директно с адресите на елементите.

**string \*p = cars;**

**for (int i = 0; i < 4; i++) {**

**cout << \*(p + i) << endl; //**Извежда всеки елемент чрез указателя

**}**

* 1. **Сортиране на масиви**

Сортирането на масиви в C++ е процес, при който елементите на масива се нареждат във възходящ или низходящ ред. Това се постига чрез

Ръчно имплементирани алгоритми за сортиране

Bubble Sort - Сравнява съседни елементи в масива. Ако те не са в правилния ред, ги разменя. След всяко пълно преминаване през масива, най-големият/най-малкият елемент отива до края на масива. Така, докато не са подредени всички елементи.

Selection Sort - Избира най-малкия/най-големия елемент и го поставя на правилното място (началото на несортираната част), като го разменя с текущия елемент. Процесът се повтаря за останалите елементи, докато целият масив е подреден.

Insertion Sort - Започва с втория елемент (първият се счита за подреден). Взема текущия елемент и го сравнява с елементите от вече подредената част на масива. Придвижва по-големите елементи от подредената част надясно, за да направи място за текущия елемент.

Използване на готови функции

**std::sort** от библиотека **<algorithm>** - Подава се масива на функцията и тя автоматично го сортира във възходящ или низходящ ред.

**#include <algorithm>**

**int arr[5] = {3, 1, 4, 1, 5};**

**sort(arr, arr + 5); //arr** е указател към първия елемент, **arr+5** е края на масива

**sort(arr, arr + 5, greater<int>()); //**Сортиране в низходящ ред

Сортиране на масив от низове се осъществява чрез ASCII стойностите на символите в низа. Може да бъде чувствително към главни и малки букви. Например на „А“ ASCII стойността е 65, а на „а“ - 97.

* 1. **Използвана литература**

<https://cplusplus.com/doc/tutorial/pointers/>

<https://cplusplus.com/reference/vector/vector/>

<https://www.digitalocean.com/community/tutorials/sort-in-c-plus-plus>

<https://www.geeksforgeeks.org/bubble-sort-algorithm/>

<https://www.geeksforgeeks.org/insertion-sort-algorithm/>

<https://www.geeksforgeeks.org/selection-sort-algorithm-2/>

<https://www.geeksforgeeks.org/sort-c-stl/>

<https://www.guru99.com/bg/cpp-dynamic-array.html>

<https://introprogramming.info/intro-csharp-book/read-online/glava7-masivi/>

<https://kst.tugab.bg/dl/week5/PIK-Lec-5.pdf>

<https://www.programist.bg/c++/masiv3.html>

<https://www.scaler.com/topics/how-to-sort-a-string-in-cpp/>

<https://softuni.bg/blog/cplusplus-arrays-specifics>

<https://softwarerecs.stackexchange.com/questions/62286/visualizing-n-dimensional-arrays>

<https://www.w3schools.com/cpp/cpp_arrays.asp>

<https://www.w3schools.com/cpp/cpp_pointers.asp>

1. **Практическо изпълнение**
   1. **Програмен код**

**#include <iostream>**

**#include <cstdlib>**

**#include <ctime>**

**#define MAX\_VALUE 299**

**// Function to fill an array with random numbers**

**void fillArray(int\* array, int length) {**

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

**array[i] = rand() % (MAX\_VALUE + 1); // Random numbers from 0 to MAX\_VALUE**

**printf("%d ", array[i]);**

**}**

**}**

**// Function to find common elements between two arrays**

**int findCommonElements(int\* firstArray, int\* secondArray, int length, int\* thirdArray) {**

**int thirdArrayIndex = 0;**

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

**for (int j = 0; j < length; j++) {**

**if (firstArray[i] == secondArray[j]) {**

**thirdArray[thirdArrayIndex] = firstArray[i];**

**thirdArrayIndex++;**

**break; // To avoid duplicates**

**}**

**}**

**}**

**return thirdArrayIndex;**

**}**

**// Function to print an array**

**void printArray(const char\* title, int\* array, int length) {**

**printf("\n\n%s:\n", title);**

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

**printf("%d ", array[i]);**

**}**

**}**

**int main() {**

**int length;**

**// Input array length**

**printf("Length of arrays: ");**

**scanf\_s("%d", &length);**

**if (length <= 0 || length > MAX\_VALUE) {**

**printf("Invalid length. Please enter a value between 1 and %d.\n", MAX\_VALUE);**

**return 1;**

**}**

**// Declaring arrays**

**int firstArray[MAX\_VALUE], secondArray[MAX\_VALUE], thirdArray[MAX\_VALUE];**

**// Seeding random number generator**

**srand((unsigned int)time(NULL));**

**// Fill first array**

**printf("\nFirst Array Items: \n");**

**fillArray(firstArray, length);**

**// Fill second array**

**printf("\n\nSecond Array Items: \n");**

**fillArray(secondArray, length);**

**// Find common elements**

**int thirdArrayIndex = findCommonElements(firstArray, secondArray, length, thirdArray);**

**// Print third array if it has elements**

**if (thirdArrayIndex > 0) {**

**printArray("Third Array", thirdArray, thirdArrayIndex);**

**// Find largest element in third array**

**int largest = thirdArray[0];**

**int largestIndex = 0;**

**for (int i = 1; i < thirdArrayIndex; i++) {**

**if (thirdArray[i] > largest) {**

**largest = thirdArray[i];**

**largestIndex = i;**

**}**

**}**

**printf("\n\nLargest element in Third Array: %d at index [%d]", largest, largestIndex);**

**}**

**else {**

**printf("\n\nNo common elements found between Array 1 and 2.");**

**}**

**return 0;**

**}**

* 1. **Описание на решението**

В началото се дефинира константа **MAX\_VALUE**, която задава максималния размер на масивите и най-голямата стойност, която може да бъде генерирана в тях.

Потребителя въвежда дължината на масивите чрез променливата **length**. Ако тя не е в интервала [1, **MAX\_VALUE**], програмата дава грешка и не продължава.

Декларират се три масива с дължина **MAX\_VALUE**. Първия и втория с произволни числа и третия, който ще е за еднаквите елементи от двата масива.

Функцията **srand((unsigned int)time(NULL))** инициализира генератора на случайни числа със стойност, базирана на текущото време, за да се осигурят различни резултати при всяко стартиране.

**rand() % (MAX\_VALUE + 1)** генерира числа в диапазона [0, **MAX\_VALUE**].

Първият и вторият масив се запълват с тези случайни числа, които се извеждат на екрана.

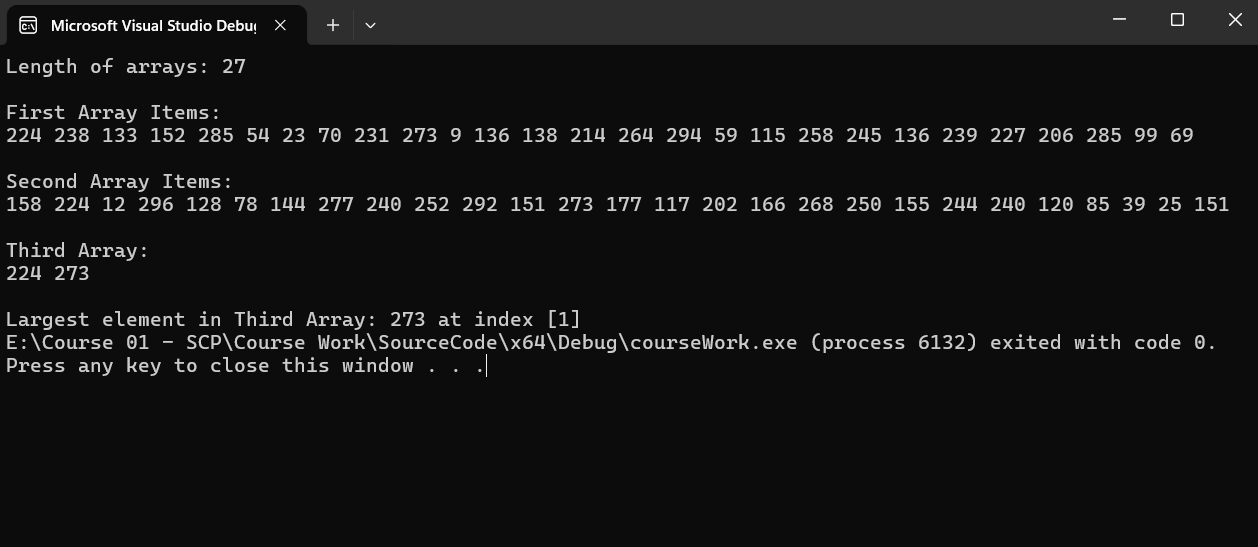
Програмата сравнява всеки елемент от първия елемент с всеки елемент от втория. Ако се намери съвпадение, елемента се добавя в третия масив.

Програмата използва **break**, за да се избегнат повторения на елементи, след като се открие съвпадение.

Ако третия масив съдържа елементи, те се принтират на екрана. Иначе се дава съобщение, че няма съвпадения между първия и втория масив.

Програмата търси най-голямата стойност в третия масив, ако не е празен. Използва се цикъл, който сравнява текущата най-голяма стойност с всеки елемент от масива. Променливата **largest** запазва най-голямата стойност, а **largestIndex** index-а на елемента с най-голяма стойност. Принтират се и двете променливи.

* 1. **Екрани**



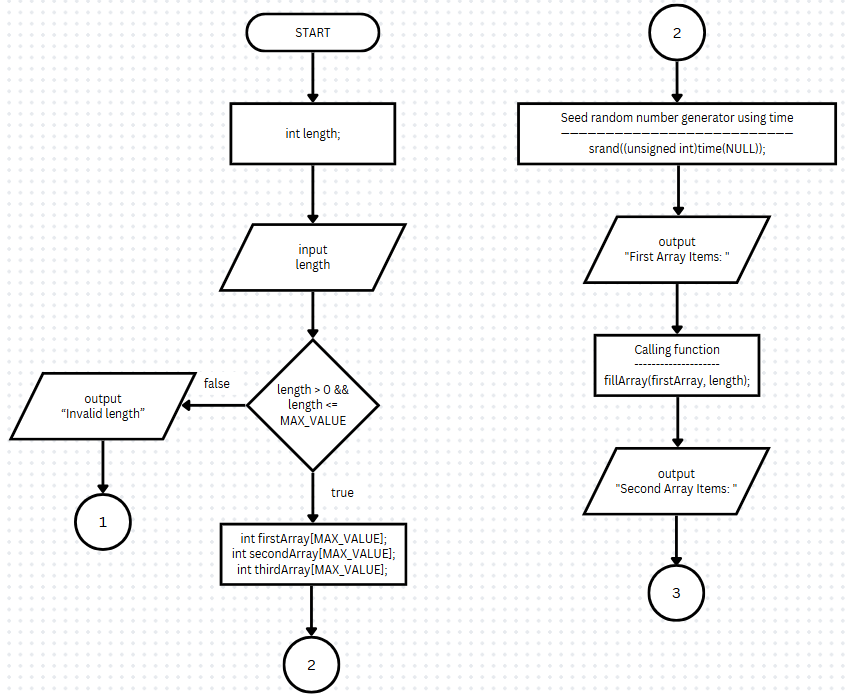
* 1. **Линк към GitHub Repository на курсовата работа**

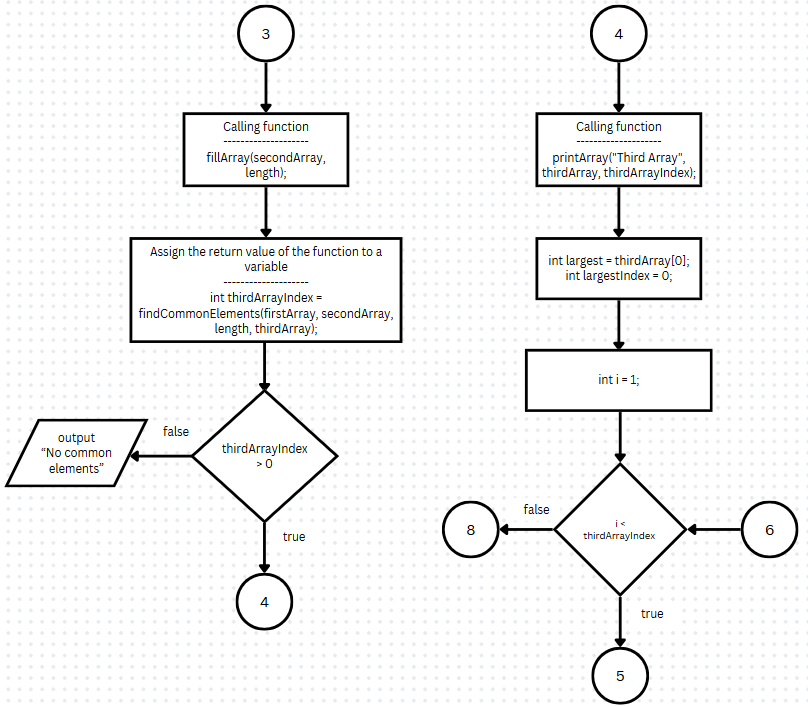
<https://github.com/vaniavaneva/Course-01-SCE/tree/main/Course%20Work>

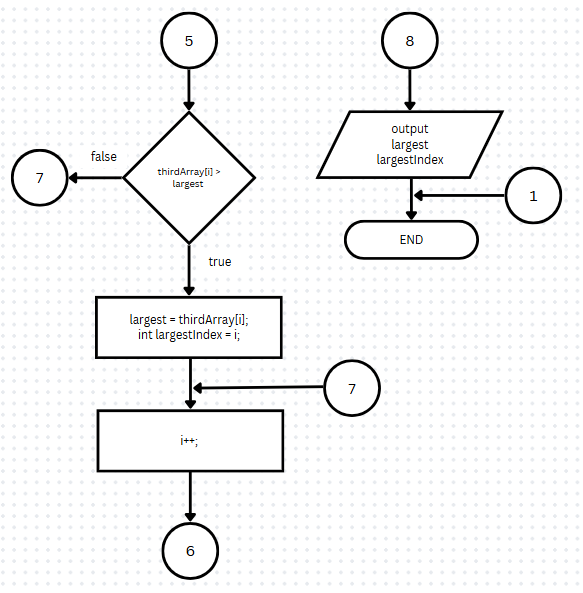
1. **Обем на чертежите (Блок-схема)**
   1. **Линк към блок-схемата**

<https://www.canva.com/design/DAGX83PSsA4/2vZwVGyCzFCG8xE91NxZkw/edit?utm_content=DAGX83PSsA4&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

* 1. **Блок-схема**

****

****

****