**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 9

**На тему:** *“Динамічне виділення пам’яті”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст.викл. каф. ПЗ

Муха Т.О.

**Виконав:**

ст. гр. ПЗ-15

Марущак А.С.

**Прийняла:**

асист. каф. ПЗ

Заводовська Н.О.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2021 р.

∑= \_\_\_\_ .

Львів – 2021

**Тема роботи:** динамічне виділення пам’яті.

**Мета роботи:** навчитися використовувати динамічну пам’ять, виділяти та звільняти її засобами мови С.

**Теоретичні відомості**

У мові С існує можливість динамічно розподіляти пам’ять у ході виконання програми, тобто, виділяти стільки пам’яті, скільки її насправді потрібно, і звільнити її тоді, коли вона вже не буде потрібною. Пам’ять виділяється з так званої вільної пам’яті. Розмір вільної пам’яті залежить від операційної системи та моделі пам’яті комп’ютера. Динамічні змінні не мають імен, а доступ до них здійснюється через вказівники. Динамічні об’єкти розміщаються у “кучі” (англомовний термін-відповідник: heap). Для управління вільною пам’яттю у С використовується набір функцій malloc(), calloc(), realloc() та free().

void \*malloc(size\_t size);

виділяє в динамічній пам’яті блок розміром size байт. Виділена пам’ять зайнята довільними даними – “сміттям”. Якщо пам’ять виділена успішно, то malloc повертає нетипізований вказівник на початок виділеного блока пам'яті, а інакше NULL.

void \*calloc(size\_t numb, size\_t size);

виділяє в динамічній пам'яті суцільний масив з numb елементів, кожен з яких займає size байтів. Виділений блок пам'яті заповнюється нулями. Якщо пам'ять виділена успішно, повертається вказівник на початок виділеного блока, а інакше значення NULL.

void \*realloc(void \*ptr, size\_t size);

встановлює у size розмір блока пам'яті, виділеного раніше функціями malloc, calloc або realloc, на початок якого вказує ptr. Якщо ptr == NULL, то функція realloc діє аналогічно функції malloc з параметром size. Якщо зміна розміру блока пам'яті відбулася успішно, то функція realloc повертає вказівник на початок зміненого блока (у разі, якщо адреса блока міняється, дані копіюються в ділянку пам'яті за новою адресою). Адреса блока може і не змінитися. У будь-якому випадку за старою адресою до даних звертатися не можна – потрібно перевірити, чи адреса не змінилася. Якщо ж розмір блока змінити не вдалося, то функція realloc повертає NULL, а до даних можна продовжувати звертатися за старою адресою.

Функція

void free(void \*ptr); звільняє блок пам'яті, на початок якого вказує ptr. Добрим стилем є звільнення пам'яті щоразу, як вона стає непотрібною. Після завершення роботи програми пам'ять буде звільнено, тому в невеликих програмах звільнення пам'яті є питанням доброго стилю. Однак, у великих програмах є небезпека «просочування пам'яті».

**ЛАБОРАТОРНЕ ЗАВДАННЯ**

1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.

2. Одержати індивідуальне завдання.

3. Скласти програму на мові С++ у відповідності з розробленим алгоритмом.

4. Виконати обчислення по програмі.

5. Підготувати та здати звіт про виконання лабораторної роботи.

**Індивідуальне завдання**

4. Ввести num - кількість масивів. Ввести розмірність чергового масиву і його елементи цілого типу, розмістити їх у динамічній пам’яті. Розсортувати масиви по зростанню і вивести на екран.

Код програми:

(lab09\_lib.h):

#ifndef LAB09

#define LAB09

#define RAND\_MAX\_ARRAY\_COUNT 10

#define RAND\_MAX\_ELEMENT\_COUNT 10

#define RAND\_MAX\_ELEMENT 20

#define min(a, b) (((a) < (b)) ? (a) : (b))

typedef struct{

    int\* values;

    int size;

} Array;

Array \*createIntArr(int arrSize);

void printArray(Array \*arr);

void printManyArrays(Array \*\*arrays, int size);

void sortArrayOfArrays(Array \*\*arrays, int size);

void freeSubArrays(Array \*\*arrays, int size);

Array \*\*createArraysRandomly(int arraysCount, int maxElementsCount, int maxElement);

int checkCommandLineArgs(int argc, char \*argv[]);

Array \*\*createArraysFromCommandLine(int argc, char \*argv[]);

Array \*\*createArraysFromConsole(int arraysCount);

#endif

(lab09\_lib.c):

#include "lab09\_lib.h"

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

Array \*createIntArr(int arrSize){

    int \*mem = (int\*)calloc(arrSize, sizeof(int));

    if(!mem) return NULL;

    Array \*structMem = (Array\*)malloc(sizeof(Array));

    if(!structMem) return NULL;

    structMem->values = mem;

    structMem->size = arrSize;

    return structMem;

}

int intArrCmp(Array\* a, Array\* b){

    if(a == NULL || b == NULL) return 0;

    for(int i = 0; i < min(a->size, b->size); i++){

        if(a->values[i] > b->values[i]) return 1;

        else if(a->values[i] < b->values[i]) return -1;

    }

    if(a->size > b->size) return 1;

    else if(b->size > a->size) return -1;

    else return 0;

}

void printArray(Array \*arr){

    if(arr == NULL) return;

    printf("[");

    for(int i = 0; i < arr->size - 1; i++){

        printf("%d, ", arr->values[i]);

    }

    printf("%d", arr->values[arr->size - 1]);

    printf("]\n");

}

void printManyArrays(Array \*\*arrays, int size){

    for(int i = 0; i < size; i++){

        printArray(\*(arrays + i));

    }

}

void swap(Array \*\*a, Array \*\*b){

    Array \*temp = \*a;

    \*a = \*b;

    \*b = temp;

}

void sortArrayOfArrays(Array \*\*arrays, int size){

    if(arrays == NULL) return;

    for(int i = 0; i < size; i++){

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

            if(intArrCmp(\*(arrays + j - 1), \*(arrays + j)) > 0){

                swap(arrays + j - 1, arrays + j);

            }

        }

    }

}

void freeSubArrays(Array \*\*arrays, int size){

    if(arrays == NULL) return;

    for(int i = 0; i < size; i++){

        if(arrays + i == NULL) continue;

        if((\*(arrays + i))->values != NULL) free((\*(arrays + i))->values);

        free(\*(arrays + i));

    }

}

void randomFill(Array \*array, int maxElem){

    for(int i = 0; i < array->size; i++){

        array->values[i] = rand() % maxElem + 1;

    }

}

Array \*\*createArraysRandomly(int arraysCount, int maxElementsCount, int maxElement){

    if(arraysCount <= 0 || maxElementsCount <= 0 || maxElement <= 0) return NULL;

    srand(time(NULL));

    int size = 0;

    Array \*\*arrays = (Array\*\*)calloc(arraysCount, sizeof(Array\*));

    if(arrays == NULL) return NULL;

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

        size = rand() % maxElementsCount + 1;

        \*(arrays+ i) = createIntArr(size);

        if(\*(arrays + i) == NULL) return NULL;

        randomFill(\*(arrays + i), maxElement);

    }

    return arrays;

}

void fillArrayFromConsole(Array \*array){

    for(int i = 0; i < array->size; i++){

        scanf("%d", array->values + i);

    }

}

int checkCommandLineArgs(int argc, char \*argv[]){

    int num = 0;

    if(argc > 1 && atoi(argv[1]) == argc - 2){

        for(int i = 2; i < argc; i++){

            if(atoi(argv[i]) < 1) return 0;

        }

        return 1;

    }

    return 0;

}

Array \*\*createArraysFromCommandLine(int argc, char \*argv[]){

    Array \*\*arrays;

    int arraysCount = atoi(argv[1]);

    arrays = (Array\*\*)calloc(arraysCount, sizeof(Array\*));

    int arraySize = 0;

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

        arraySize = atoi(argv[i + 2]);

        \*(arrays + i) = createIntArr(arraySize);

        printf("Enter elements of %d%s array(size: %d): ", i+1, i + 1 == 1 ? "st" : i + 1 == 2 ? "nd" : i + 1 == 3 ? "rd" : "th", arraySize);

        fillArrayFromConsole(\*(arrays + i));

    }

    return arrays;

}

Array \*\*createArraysFromConsole(int arraysCount){

    Array \*\*arrays;

    arrays = (Array\*\*)calloc(arraysCount, sizeof(Array\*));

    int arraySize = 0;

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

        printf("\nEnter size of %d%s array: ", i+1, i + 1 == 1 ? "st" : i + 1 == 2 ? "nd" : i + 1 == 3 ? "rd" : "th");

        scanf("%d", &arraySize);

        \*(arrays + i) = createIntArr(arraySize);

        printf("Enter elements of %d%s array: ", i+1, i + 1 == 1 ? "st" : i + 1 == 2 ? "nd" : i + 1 == 3 ? "rd" : "th");

        fillArrayFromConsole(\*(arrays + i));

    }

    return arrays;

}

(lab09.c):

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include "lab09\_lib.h"

int main(int argc, char \*argv[]){

    int num = 0;

    Array \*\*arrays;

    if(argc > 1 && strcmp(argv[1], "-r") == 0){

        srand(time(NULL));

        num = rand() % RAND\_MAX\_ARRAY\_COUNT + 1;

        arrays = createArraysRandomly(num, RAND\_MAX\_ELEMENT\_COUNT, RAND\_MAX\_ELEMENT);

    }

    else if(checkCommandLineArgs(argc, argv)){

        num = atoi(argv[1]);

        arrays = createArraysFromCommandLine(argc, argv);

    }

    else{

        printf("Enter count of arrays: ");

        scanf("%d", &num);

        printf("\n");

        arrays = createArraysFromConsole(num);

    }

    printf("\nCount of arrays: %d", num);

    printf("\n\nUnsorted list of arrays:\n");

    printManyArrays(arrays, num);

    sortArrayOfArrays(arrays, num);

    printf("\n\nSorted list:\n");

    printManyArrays(arrays, num);

    freeSubArrays(arrays,num);

    free(arrays);

    return 0;

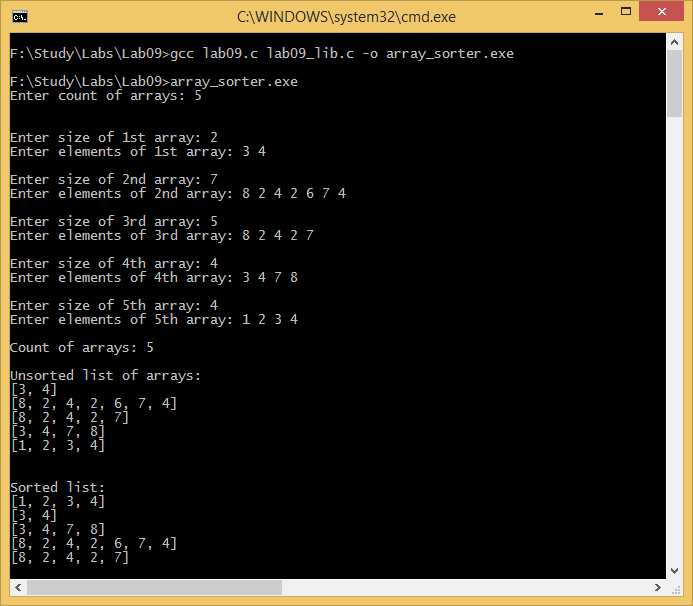
}

**Протокол роботи:**

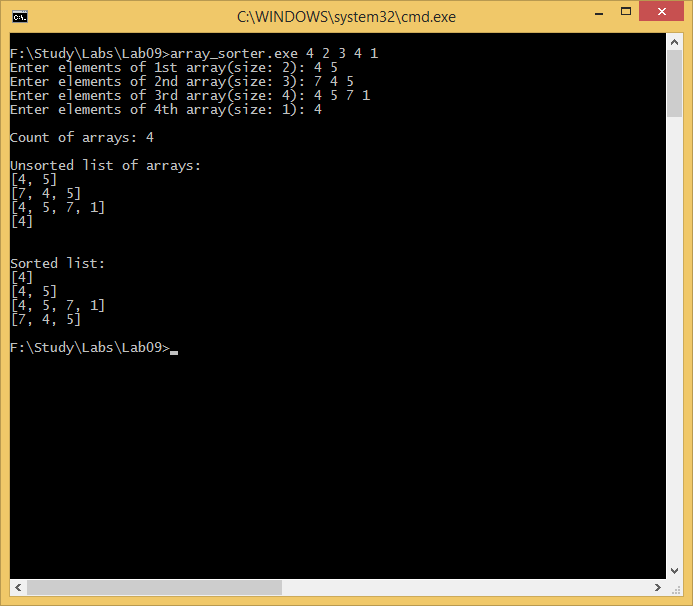
Програма може працювати в 3 режимах:

* У першому вона просто запускається і отримує всю необхідну інформацію про розміри масивів, їх кількість і елементи з консолі.
* У другому вона отримує інформацію про к-ть масивів та їх розміри з аргументів командного рядка.
* У третьому програма працює в повністю випадковому режимі – всі розміри і елементи задаються випадково.

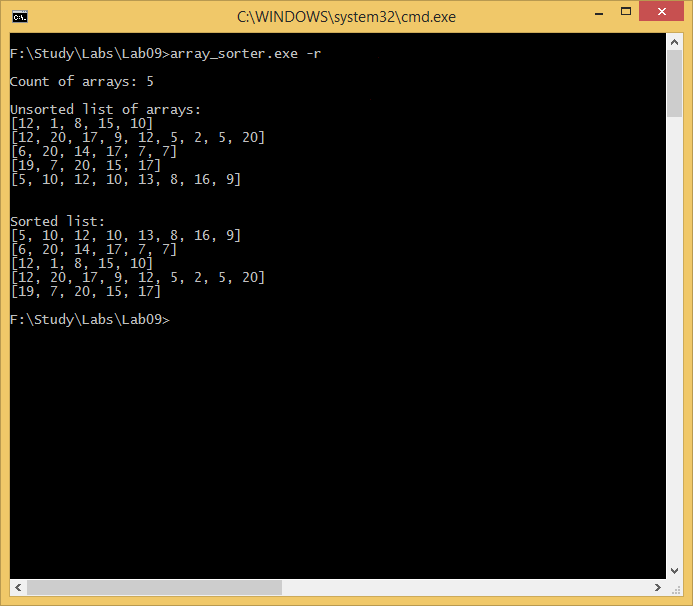
Потім програма сортує отримані масиви в лексикографічному порядку і виводить відсортований список.



Розміри з командної стрічки:



Випадкове заповнення(для швикдого тестування):



**Висновок:**

Виконання цієї лабараторної роботи допомогло нам освоїти роботу з динамічною пам’яттю, з методами для роботи з нею. Це допоможе нам більш економно використовувати пам’ять, а також динамічно змінювати її розміри.