Лабораторна робота № 9

Використання масивів вказівників на функції для організації командного інтерфейсу роботи (меню) програм

Мета роботи: *оволодіння практичними навичками роботи з вказівниками, динамічними масивами, функціями та вказівниками на функції.*

Завдання

Скласти програму, що працює з послідовністю цілих чисел, яка міститься в динамічному одновимірному масиві. Кількість елементів та вказівник на динамічно виділену для масиву пам'ять об'єднати в окрему структуру (`struct`). Програму реалізувати у вигляді набору функцій з однотипним заголовком. Роботу програми організувати у вигляді діалогу з користувачем, в якому той обирає дію, яку бажає виконати з масивом, після чого бачить результат роботи відповідної функції на екрані. Вибір функції користувачем реалізувати за допомогою масиву вказівників на функції. В програмі передбачити функції для: **1)** виводу масиву на екран; **2)** вводу елементів масиву з клавіатури; **3)** три функції відповідно до завдання варіанту.

1. **а)** заповнити масив квадратами послідовних цілих чисел із вказаного користувачем відрізка; **б)** знайти середнє значення парних елементів масиву; **в)** знайти добуток елементів масиву, менших за вказане користувачем число.

2. **а)** заповнити масив вказаною кількістю послідовних простих чисел починаючи з 2; **б)** знайти середнє значення усіх елементів масиву; **в)** знайти в масиві елемент, найближчий за величиною до заданого користувачем числа.

3. **а)** заповнити масив послідовністю простих чисел, менших за вказане користувачем число розташованих в порядку спадання; **б)** знайти середнє значення непарних елементів масиву; **в)** знайти в масиві різницю між найбільшим та найменшим елементами.

4. **а)** заповнити масив вказаною кількістю послідовних степенів числа 2, починаючи з нульового; **б)** знайти різницю середнього значення парних елементів масиву та середнього значення непарних; **в)** знайти в масиві елемент, найближчий за величиною до числа 0.

5. **а)** заповнити масив вказаною кількістю послідовних степенів числа 2, що належать заданому користувачем діапазонові; **б)** знайти суму перших за порядком двох непарних елементів; **в)** знайти в масиві елемент, квадрат якого найближчий за величиною до заданого користувачем числа.

6. **а)** заповнити масив вказаною кількістю степенів числа 2, заданих випадково від 1 до 10; **б)** знайти добуток середнього значення парних

елементів масиву та середнього значення непарних; **в)** знайти в масиві елемент, квадрат якого найближчий за величиною до числа 0.

7. **а)** заповнити масив вказаною кількістю послідовних простих чисел, що належать заданому користувачем діапазонові; **б)** знайти суму перших за порядком двох парних елементів; **в)** знайти в масиві другий за величиною елемент.

8. **а)** заповнити масив членами арифметичної прогресії з вказаними користувачем різницею та першим членом; **б)** знайти середнє значення парних елементів масиву; **в)** знайти добуток елементів масиву, менших за вказане користувачем число.

9. **а)** заповнити масив вказаною кількістю послідовних простих чисел починаючи з 7; **б)** знайти середнє значення непарних елементів масиву; **в)** знайти в масиві елемент, квадрат якого найближчий за величиною до заданого користувачем числа.

10. **а)** заповнити масив послідовністю чисел, кратних заданому користувачем числу, розмір масиву також задає користувач; **б)** знайти середнє геометричне непарних елементів масиву; **в)** знайти в масиві різницю між двома найбільшими елементами.

11. **а)** заповнити масив вказаною кількістю степенів числа 2, заданих випадково від 1 до 10; **б)** знайти різницю добутку і суми елементів масиву; **в)** знайти в масиві найбільшу кількість однакових елементів.

12. **а)** заповнити масив вказаною кількістю послідовних степенів числа 3, починаючи з нульового; **б)** знайти добуток середнього значення парних елементів масиву та середнього значення непарних; **в)** знайти в масиві елемент, квадрат якого найближчий за величиною до числа 0.

13. **а)** заповнити масив вказаною кількістю послідовних простих чисел, що належать заданому користувачем діапазонові; **б)** знайти суму перших за порядком двох парних елементів; **в)** знайти в масиві елемент, куб якого найближчий за величиною до заданого користувачем числа.

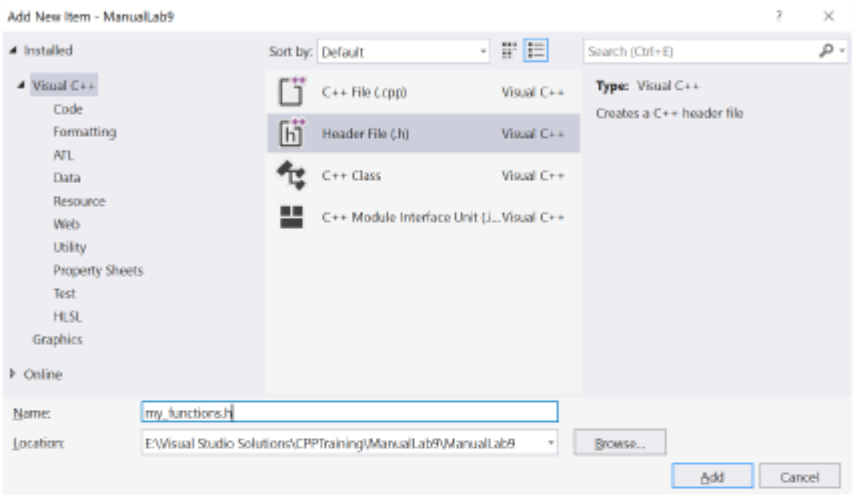
14. **а)** заповнити масив вказаною кількістю послідовних степенів числа 3, починаючи з нульового; **б)** знайти добуток середнього значення парних елементів масиву та середнього значення непарних; **в)** знайти в масиві другий за величиною елемент.

15. **а)** заповнити масив членами арифметичної прогресії з вказаними користувачем різницею та першим членом; **б)** знайти добуток парних елементів масиву; **в)** знайти в масиві парний елемент, найближчий за величиною до заданого користувачем числа.

Приклад

16. **а)** заповнити масив випадковими цілими числами з вказаного користувачем відрізка; **б)** заповнити масив вказаною користувачем кількістю послідовних чисел Фібоначчі; **в)** знайти в масиві суму від'ємних елементів; **г)** знайти добуток усіх елементів масиву.

Проект міститиме багато функцій, тому структуруємо його, використовуючи заголовковий файл (***header***). Він додається до проекту аналогічно до файлів з кодом.



У заголовкових файлах, зазвичай записують лише оголошення типів та функцій(прототипи). Назвемо наш заголовковий файл my_fuctions.h, він може виглядати так:

```
#pragma once
#include <iostream>
#include <Windows.h>
using namespace std;

//оголошення структури з масивом
struct my_array {
    int size;
    int* data;
};
```

```
// прототипи функцій, реалізація в my_functions.cpp
void array_generate_p(my_array&);
void array_print(my_array&);
void array_input(my_array&);
void array_generate_f(my_array&);
void array_get_sum_negative(my_array&);
void array_get_product(my_array&);
```

Директива ***#pragma once*** вказує, що код файлу потрібно включати в результуючий код для компіляції лише один раз. Оскільки директива ***#include "my_fuctions.h"*** просто включає код вказаного файла у місце, де вона записана, то якщо цей заголовковий файл підключений у різних файлах, можемо отримати декілька описів однієї і тієї ж функції у різних місцях, – компілятор вважатиме це за помилку. Усі функції повинні мати однотипну сигнатуру, оскільки вказівники на них будуть елементами одного і того ж масиву в головній функції.

Для реалізації наших функцій створимо ще один C++-файл з кодом, назвемо його "***my_fuctions.cpp***". Розміщена на початка директива ***#include "my_fuctions.h"*** додасть до нього увесь код заголовкового файлу і в ньому будуть доступні оголошення наших функцій, тип структури для зберігання а також усі функції і типи з системних заголовкових файлів ***iostream.h*** та ***Windows.h*** та простір імен ***std***. Помістимо в цьому файлі повний опис потрібних нам функцій:

```
#include "my_fuctions.h"
//реалізація функцій
void array_generate_p(my_array& a)
{
     if (a.data != 0)
     {//перевіряємо, чи масив не містить даних
     //Якщо для масиву вже було виділено
     //пам'ять, то звільняємо її
          delete[] a.data;
     }
     cout << "Введіть розмір масиву";
```

```
    cin >> a.size;//читаємо розміо масиву
    //та виділяємо для нього пам'ять
    a.data = new int[a.size];
    cout << "Введіть діапазон для "
        <<"елементів  масиву";
    int a1, a2; cin >> a1 >> a2;
    for (int i = 0; i < a.size; i++)
    {
        a.data[i] = rand() % (a2 - a1 + 1) + a1;
    }
    cout << "Масив згенеровано\n";
    system("pause");
}
void array_print(my_array& a)
{
    if (a.data == 0)
    {//перевіряємо, чи масив створено
    //Якщо ні,  - завершуємо роботу функції
        cout << "спочатку потрібно створити масив";
        return;
    }
    cout << "Елементи масиву:\n";
    for (int i = 0; i < a.size; i++)
    {
        cout<<a.data[i]<<'\t';
    }
    cout << endl;
    //призупиняємо виконання програми
    system("pause");
}
```

```
void array_input(my_array& a)
{
     if (a.data != 0)
     {//перевіряємо, чи масив не містить даних
          delete[] a.data;
     }
     cout << "Введіть розмір масиву";
     cin >> a.size;//читаємо розмір масиву
     //та виділяємо для нього пам'ять
     a.data = new int[a.size];
     cout << "Введіть діапазон елементи масиву";
     //вводимо елементи масиву з клавіатури
     for (int i = 0; i < a.size; i++)
     {
          cin>>a.data[i];
     }
     cout << "Масив заповнено\n";
     system("pause");
}

void array_generate_f(my_array& a)
{
     if (a.data != 0)
     {
          delete[] a.data;
          a.size = 0; a.data = 0;
     }
     cout << "Введіть розмір масиву > 2:\n";
     cin >> a.size;//читаємо розмір масиву
     //та виділяємо для нього пам'ять
```

```
    a.data = new int[a.size];
    a.data[0] = 1; a.data[1] = 1;
    for (int i = 2; i < a.size; i++)
    {
        a.data[i] = a.data[i-1] + a.data[i-2];
    }
    cout << "Масив заповнено\n";
    system("pause");
}

void array_get_sum_negative(my_array& a)
{
    if (a.data == 0)
    {
        cout << "спочатку потрібно створити масив\n";
        system("pause");
        return;
    }
    int s = 0;
    for (int i = 0; i < a.size; i++)
    {
        if(a.data[i]<0)s+= a.data[i];
    }
    cout << "Сума від'ємних елементів:\n";
    cout << s <<endl;
    system("pause");
}
void array_get_product(my_array& a)
{
    if (a.data == 0)
```

```
    {
        cout << "спочатку потрібно створити масив\n";
        system("pause");
        return;
    }
    int d = 1;
    for (int i = 0; i < a.size; i++)
    {
        d *= a.data[i];
    }
    cout << "Добуток елементів масиву:\n";
    cout << d << endl;
    system("pause");
}
```

У файлі Source.cpp спочатку реалізуємо функцію main для перевірки роботи усіх реалізованих нами функцій. Наприклад викличемо їх послідовно одна за одною:

```
#include "my_fuctions.h"
void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    srand(time(0));
    my_array m;
    m.data = 0; m.size = 0;
    array_generate_f(m);
    array_print(m);
    array_generate_p(m);
    array_print(m);
    array_get_sum_negative(m);
```

```
    array_input(m);
    array_print(m);
    array_get_product(m);
}
```

Перетворимо нашу функцію main на програму, що викликає функції в режимі діалогу з користувачем. Для цього спочатку оголосимо масив вказівників на функції, та ініціалізуємо його вказівниками на створені нами функції:

```
void(*my_commands[])(my_array&) = { array_input,
    array_generate_f, array_generate_p, array_print,
        array_get_sum_negative, array_get_product };
```

Далі організуємо вічний цикл, в якому запитуватимемо користувача номер команди меню і виконуватимемо функцію з масиву з відповідним номером. Орієнтовний код функції ***main*** :

```
#include "my_fuctions.h"
void main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    srand(time(0));
    my_array m;
    m.data = 0; m.size = 0;
    void(*my_commands[])(my_array&) = { array_input,
        array_generate_f, array_generate_p,
            array_print, array_get_sum_negative,
                array_get_product };
    int user_choise = 0;
    while (true)//вічний цикл
    {
        system("cls");//очистка екрану
```

```
//від результатів попередньої роботи
    cout << "Оберіть потрібну вам дію:\n"
         << "0 - ввести масив з клавіатури\n"
         << "1 - утворити послідовність Фібоначчі\n"
         << "2 - заповнити масив випадково\n"
         << "3 - роздрукувати масив\n"
         << "4 - знайти суму від'ємних елементів\n"
         << "5 - знайти добуток\n6 - завершити\n>";
    cin >> user_choise;
    //вихід з циклу
    if (user_choise == 6)break;
    //контролюємо вихід індекса за межі масиву
    if (user_choise >= 0 && user_choise < 6)
    {
        //викликаємо обрану функцію з масиву
        my_commands[user_choise](m);
    }
}
```