Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №3**

**Програмування, частина 2**

**(Об’єктно - орієнтоване програмування)**

на тему:

“Засоби роботи з динамічною пам’яттю. Динамічні масиви”

Варіант - 25

**Виконала:**

ст. гр. КІ-15

Романішина А. Ю.

**Перевірив:**

асист. каф. ЕОМ

Козак Н. Б.

Львів – 2020

**Мета:** познайомитись із динамічними масивами.

**Теоретичні відомості:**

*Динамічне виділення пам’яті:*

В С++ об‘єкти можна розміщати статично – під час компіляції, або динамічно – під час виконання програми, шляхом виклику функцій зі стандартної бібліотеки. Основна відмінність у використанні даних методів – в їхній ефективності та гнучкості. Статичне розміщення більш ефективне, так як виділення пам‘яті відбувається до виконання програми, проте воно менш гнучке, тому що необхідно наперед знати тип і розмір об‘єкту. Задачі, в яких необхідно зберігати та обробляти наперед не відому кількість елементів, зазвичай потребують динамічного виділення пам‘яті.

Динамічне виділення пам‘яті у мові С++ здійснюється за допомогою оператора new. Оператор new здійснює пошук неперервної області пам‘яті в області пам‘яті, що зветься некерована куча. Некерована куча – це стрктура даних за допомогою якої реалізована пам‘ять, що може бути виділена динамічно в ході виконання програми, а також це область пам‘яті, зарезервована під цю структуру. З іншої сторони куча - це довгий відрізок адрес пам'яті, поділений на блоки різних розмірів, що йдуть підряд. Пам‘ять у кучі поділяється на заняту і вільну. Перед початком роботи програми вся пам‘ять у кучі позначається як вільна. При виклику оператора динамічного виділення пам‘яті у кучі відбувається пошук неперервного сегменту вільної пам‘яті заданого розміру. Час такого пошуку є значним і займає більшу частину часу, що необхідна для виконання операції динамічного виділення пам‘яті. Якщо такий сегмент було знайдено в кучі, то він помічається як занятий і програмі повертається адреса його початку, інакше – програмі повертається ознака відсутності такого сегменту в пам‘яті, найчастіше NULL. Якщо в ході виконання програми значення адреси початку цього сегменту втрачається, то заняту пам‘ять звільнити буде неможливо. Якщо ця ситуація проявляється неодноразово, то це може призвести до вичерпання вільної пам‘яті в системі. Коли динамічно виділена область пам‘яті стає непотрібною, 38 то її потрібно звільнити за допомогою оператора звільнення динамічно виділеної пам‘яті. При його виклику область пам‘яті на яку вказує вказівник і яка була попередньо динамічно виділена з кучі позначається як вільна і її можна буде в подальшому використовувати заново.

Оператор динамічного виділення пам‘яті new може мати дві форми:

* виділення пам’яті під одиничний об’єкт має наступний синтаксис;
* виділення пам’яті під масив заданого розміру має наступний синтаксис.

Для створення динамічного двовимірного масиву використовуються наступні елементи:

1. вказівник на вказівник, який містить адресу початку допоміжного масиву адрес розмір якого рівний висоті двовимірного масиву (кількості рядків);

2. допоміжний масив адрес, що зберігає адреси одновимірних масивів, які власне міститимуть дані; розмір цих масивів рівний розміру ширини двовимірного масиву (кількості стовпців);

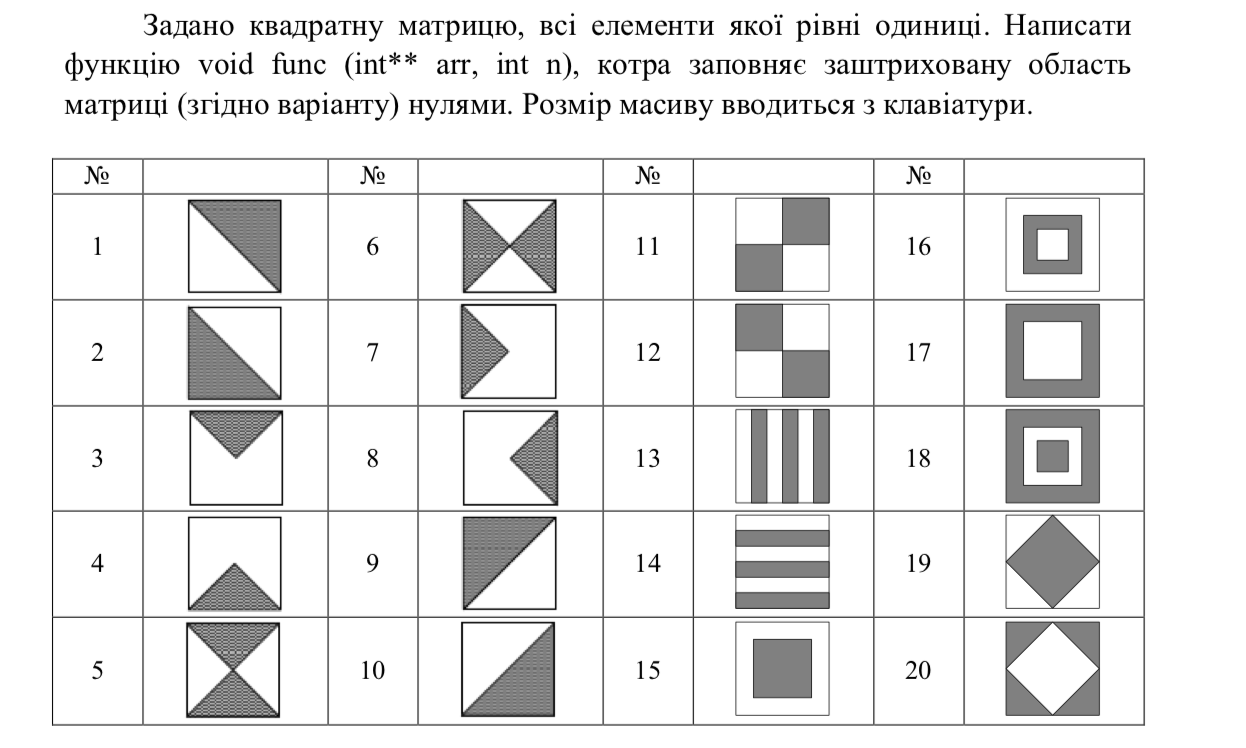
3. множина масивів, що зберігають дані (реалізують рядки масиву).

Якщо вимірів більше, то використовується більша кількість допоміжних масивів до яких приєднуватимуться інші масиви, завдяки чому власне і утворюватимуться нові виміри. Загалом можна сказати: скільки зірочок при оголошенні базового вказівника на багатовимірний масив, стільки вимірів міститиме цей масив.

Якщо нам необхідно прочитати скажімо 3 комірку 2-го масиву даних (комірку 3-го стовпця у другому рядку за умови нумерації рядків і стовпців починаючи з нуля), то нам слід записати наступний рядок коду \*(\*(ptr+2)+3). Цей рядок коду аналогічний запису ptr[2][3]. При роботі з масивами форми з дужками і вказівниками є еквівалентними, тому можна застосовувати будь-яку з них, яка є зручнішою для конкретного випадку. Проте слід зазначити що форма з вказівниками є низькорівневішою і значно потужнішою, ніж дужкова форма, проте часто вона є джерелом помилок. Для роботи з багатовимірними масивами допускається і змішана форма: \*(ptr[2]+3).

Розглянемо фрагмент коду, виконання якого призведе до створення двовимірного масиву типу int розміром 256\*4, що ми щойно розглянули.

**Індивідуальне завдання:**

****

Мій варіант – 5

**Код програми:**

#include <iostream>

#include <stdio.h>

#include <iomanip>

using namespace std;

void func(int\*\* arr, int n)

{

int a, b, c, d;

c = (n / 2) + 1;

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

{

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

{

arr[i][j] = 1; // fill the matrix with the symbol "1"

}

}

for (int i = 0; i < c; i++)

{

for (int j = 0; j <= i; j++)

{

a = n - j - 1;

arr[i][j] = 0; // fill in the "0" with the edge

arr[i][a] = 0; // of the diagonals at the top

}

}

for (int i = c; i < n; i++)

{

a = n - i;

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

{

b = n - j - 1;

arr[i][j] = 0; // fill in the "0" with the edge

arr[i][b] = 0; // of the diagonals at the bottom

}

}

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

{

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

{

if (arr[i][j] != 0) { arr[i][j] = 1; }

cout << setw(4) << arr[i][j]; // output matrix

}

cout << endl;

}

}

int main()

{

int\*\* arr, n;

cout << "Enter order of matrix:";

cin >> n;

cout << "-----------------------------------------------------" << endl;

arr = new int\* [n];

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

{

arr[i] = new int[n]; // create a dynamic two-dimensional array

}

func(arr, n);

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

{

delete[] arr[i]; // clear memory

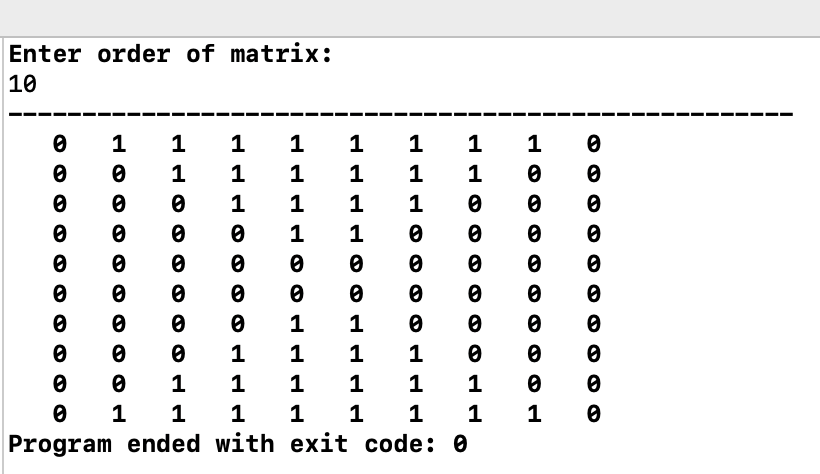
}

delete[] arr;

return 0;

}

**Результат виконання функції :**

****

**Висновок:** на цій лабораторній роботі я познайомилась із динамічними масивами.