МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт до лабораторних робіт

з ООП

Виконав:

ст. групи КІ-15

Ковальський П.А.

Прийняв:

викл. Козак Н.Б.

Львів-2020

ЛАБОРАТОРНА РОБОТА № 3.

ЗАСОБИ РОБОТИ З ДИНАМІЧНОЮ ПАМ’ЯТТЮ.

ДИНАМІЧНІ МАСИВИ

**Мета:** познайомитися із динамічними масивами.

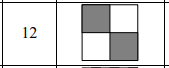
КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

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, 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.

**Варіант-12**

Задано квадратну матрицю, всі елементи якої рівні одиниці. Написати функцію void func (int\*\* arr, int n), котра заповняє заштриховану область матриці (згідно варіанту) нулями. Розмір масиву вводиться з клавіатури.



**Код програми:**

#include <iostream>

#include <iomanip>

using namespace std;

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

{

int n2, i, j;

if (n % 2 == 0)

n2 = n / 2 - 1;

else

n2 = (n - 1) / 2;

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

if (i <= n2) {

for (j = n2 + 1; j < n; j++)

arr[i][j] = 0;

}

else {

for (j = 0; j <= n2; j++)

arr[i][j] = 0;

}

}

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

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

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

cout << endl << endl;

}

}

int main() {

int \*\*arr, n;

cout << "Matrix size = ";

cin >> n;

arr = new int\*[n];

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

arr[i] = new int[n];

}

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

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

arr[i][j] = 1;

}

}

func(arr, n);

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

{

delete[] arr[i];

}

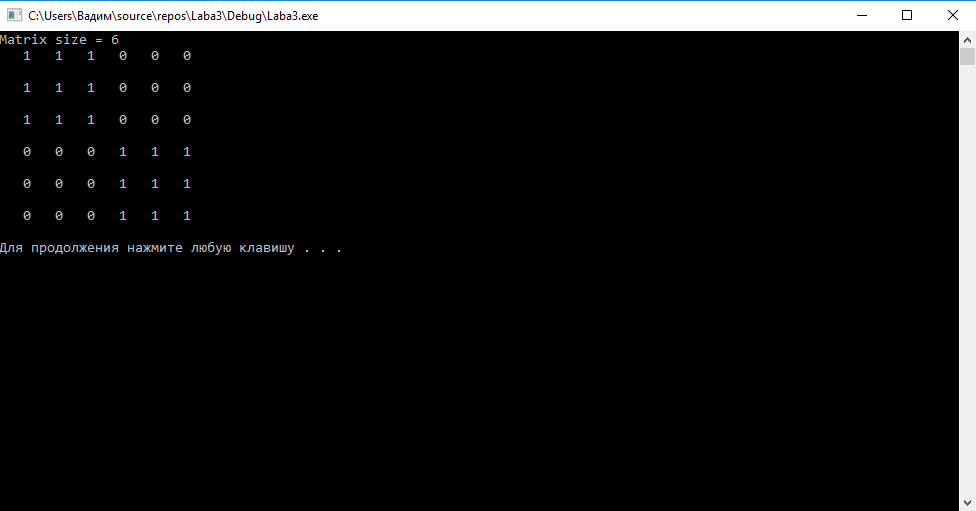
delete[] arr;

system("pause");

return 0;

}

**Результат:**



**Висновок:**

На даній лабораторній роботі я ознайомився із динамічними масивами.