Міністерство освіти і науки

Національний університет “Львівська політехніка”

**Кафедра ЕОМ**



**Звіт**

з лабораторної роботи № 3

# з дисципліни: “ Програмування, частина 2 (Об'єктно-орієнтоване програмування”

на тему: “ ЗАСОБИ РОБОТИ З ДИНАМІЧНОЮ ПАМ’ЯТТЮ. ДИНАМІЧНІ МАСИВИ ”

Виконав: ст. гр. КІ-15 Гвоздь Ю. А.

Прийняв: Козак Н. Б.

Львів – 2020

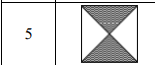
Мета: познайомитися із динамічними масивами.

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.

**Завдання:**

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.



**Виконання завдання:**

Код програми:

#include <iostream>

#include <stdio.h>

#include <iomanip>

using namespace std;

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

{

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

{

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

{

arr[i][j] = 1;

}

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

arr[i][j] = 0;

}

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

arr[i][j] = 0;

}

}

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

{

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

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

cout << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Ukr");

int\*\* arr, n;

cout << "Ведiть розмiр матрицi: ";

cin >> n;

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

arr = new int\* [n];

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

{

arr[i] = new int[n];

}

func(arr, n);

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

{

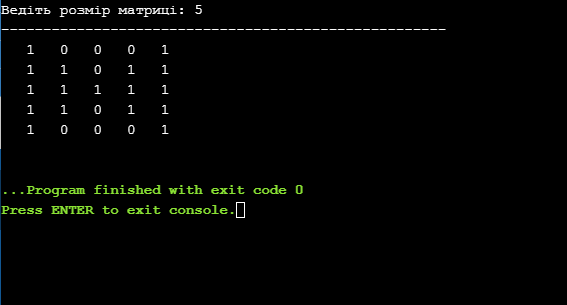
delete[] arr[i];

}

delete[] arr;

return 0;

}Результат роботи програми:



Висновок: на цій лабораторній роботі познайомився із динамічними масивами.