Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №3**

на тему:

“Засоби роботи з динамічною пам’яттю”

**Виконав:**

Ст. гр. КІ-15

Рикмас О. М.

**Перевірив:**

Викладач

Козак Н.Б.

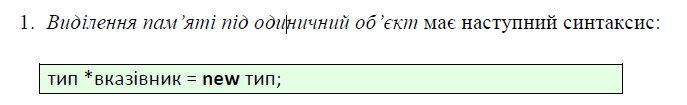
Львів – 2019

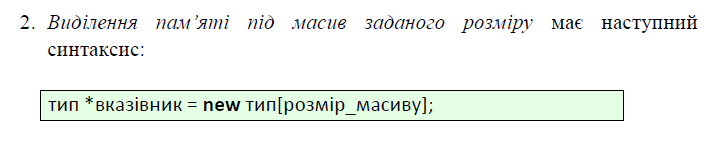
**Мета роботи:** познайомитися із динамічними масивами.

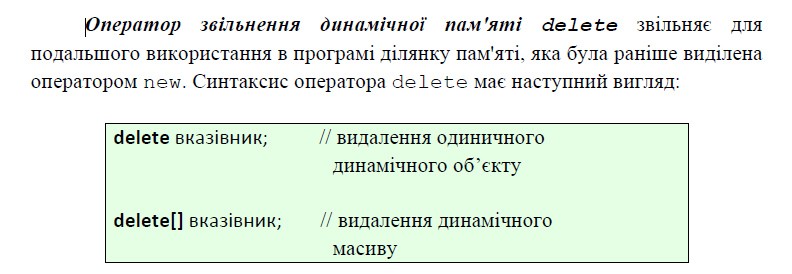
**Теоретичні відомості:**

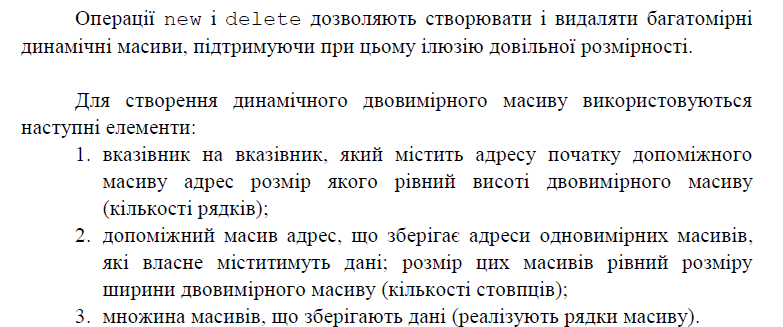
В С++ об‘єкти можна розміщати статично – під час компіляції, або динамічно – під час виконання програми, шляхом виклику функцій зі стандартної бібліотеки. Основна відмінність у використанні даних методів – в їхній ефективності та гнучкості. Статичне розміщення більш ефективне, так як виділення пам‘яті 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.

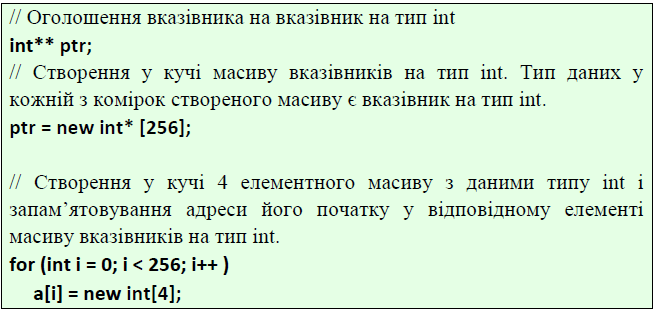




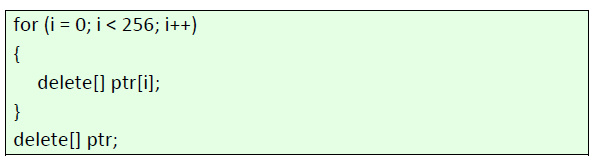




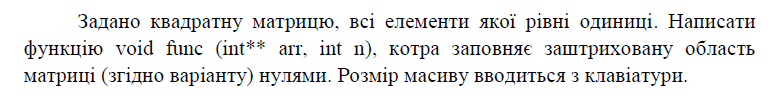
Розглянемо фрагмент коду, виконання якого призведе до створення двовимірного масиву типу int розміром 256\*4, що ми щойно розглянули.

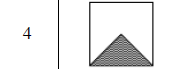


Звільнення пам‘яті для двовимірних динамічних масивів відбувається у зворотному напрямку відносно того, як він створювався – спочатку вивільняється пам‘ять зі всіх масивів з даними, а потім вивільняється пам‘ять масиву вказівників.



***Завдання:***





*Код програми:*

#include <iostream>

#include <iomanip>

using namespace std;

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

{

int ind\_s = 0, ind\_f = size;

for (int i = size-1; i >= 0; i--)

{

for (int j = ind\_s; j < ind\_f; j++)

{

arr[i][j] = 0;

}

ind\_s++;

ind\_f--;

}

return;

}

int main()

{

int size;

cout << "Enter size of matrix:";

cin >> size;

cout << endl << endl;

int\*\* arr;

arr = new int\*[size];

if (arr == NULL)

{

cout << "Error";

return 1;

}

for (int i = 0; i < size; i++) arr[i] = new int[size];

if (arr == NULL)

{

cout << "Error";

return 1;

}

for (int i = 0; i < size; i++) //Започнення всіх елементів матриці 1;

{

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

{

arr[i][j] = 1;

}

}

func(arr, size);

cout << "Your new matrix:" << endl << endl;

for (int i = 0; i < size; i++) //Вивід матриці;

{

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

{

cout.setf(ios::left);

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

}

cout << endl << endl;

}

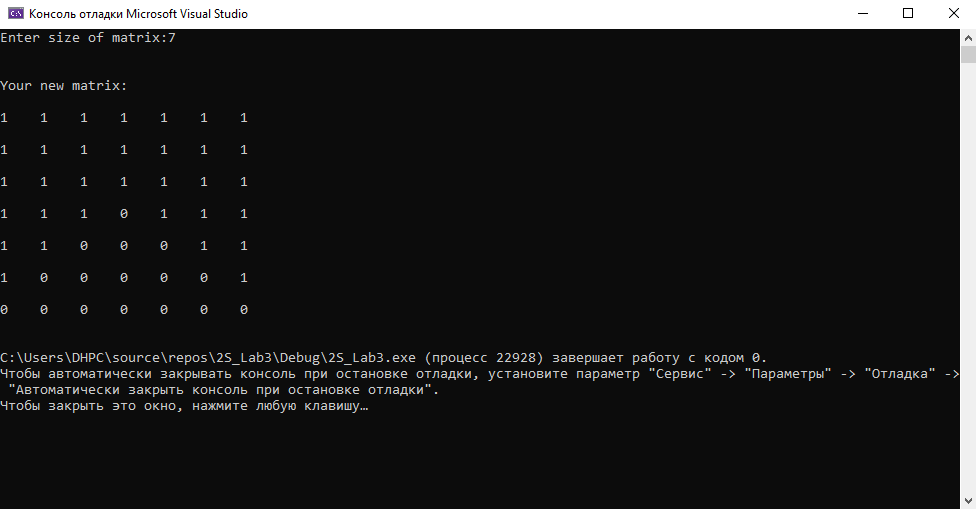
for (int i = 0; i < size; i++) delete[] arr[i];

delete[] arr;

return 0;

}

*Вікно результату:*



*Висновок:* я познайомився з динамічною пам’ятю, засобами виділення, та з динамічними масивами.