Міністерство освіти і науки України

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 ”

Варіант 18

Виконав: ст. гр. КІ-15

Марков Л.Ю.

Прийняв:

Козак Н.Б.

Львів – 2020

Мета: 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.

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).

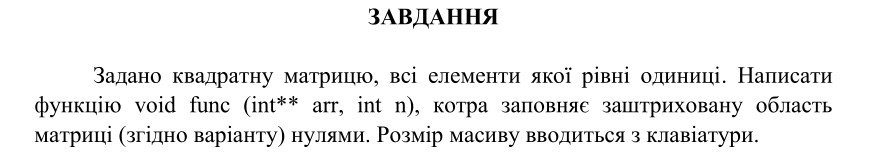
Звільнення пам‘яті для двовимірних динамічних масивів відбувається

у зворотному напрямку відносно того, як він створювався – спочатку

вивільняється пам‘ять зі всіх масивів з даними, а потім вивільняється пам‘ять

масиву вказівників.

Варіант 18:





Виконання:

#include <iostream>

using namespace std;

void Clean(int\*\* arr, int n) {

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

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

arr[i][j] = 1;

}

}

}

void Show(int\*\* arr, int n) {

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

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

cout << arr[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

void Func(int\*\* arr, int n) {

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

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

arr[i][j] = 0;

}

}

for (int i = 0.2 \* n; i < n \* 0.8; i++) {

for (int j = 0.2 \* n; j < n \* 0.8; j++) {

arr[i][j] = 1;

}

}

for (int i = 0.35 \* n; i < n \* 0.65; i++) {

for (int j = 0.35 \* n; j < n \* 0.65; j++) {

arr[i][j] = 0;

}

}

}

int main() {

while (true) {

int n = 0;

cout << "Enter length of array: ";

cin >> n;

if (n < 0) {

break;

}

int\*\* arr = new int\* [n];

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

arr[i] = new int[n];

}

Clean(arr, n);

Show(arr, n);

Func(arr, n);

Show(arr, n);

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

delete[] arr[i];

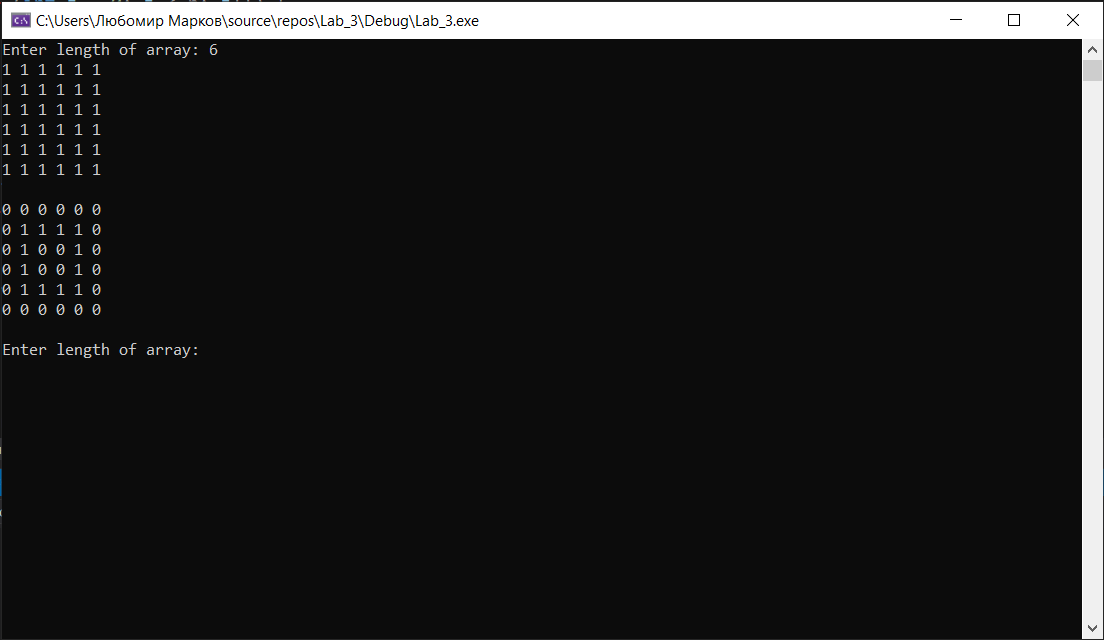
}

delete[] arr;

}

return 0;

}

**Результат:**

Висновок: на цій лабораторній роботі я познайомився із динамічними масивами.

**Контрольні питання:**

**1.Що таке динамічне 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.У цьому випадку використовується комбінація операторів new і delete. Оператор new виділяє пам‘ять для змінної (масиву) в спеціальній області пам‘яті, яка називається “купа” (heap). Оператор delete звільняє виділену пам‘ять. Кожному оператору new має відповідати свій оператор delete.

**2. Як динамічно виділити пам‘ять для змінної?**

Загальна форма виділення пам‘яті для одиночної змінної оператором new має вигляд:

ptrName = new type; де

ptrName – ім‘я змінної (покажчика), яка буде вказувати на виділену пам‘ять;

type – тип змінної. Розмір пам‘яті виділяється достатній для поміщення в неї значення змінної даного типу type.

**3.Як динамічно звільнити пам‘ять пам‘ять для змінної?**

Якщо пам‘ять для змінної виділена оператором new, то після завершення використання змінної, цю пам‘ять потрібно звільнити оператором delete. У мові C++ це є обов‘язковою умовою. Якщо не звільнити пам‘ять, то пам‘ять залишиться виділеною (зайнятою), але використовувати її не зможе жодна програма. У даному випадку відбудеться “***витік пам‘яті***” (memory leak).

У мовах програмування Java, C# звільняти пам‘ять після виділення не потрібно. Цим займається “*збирач сміття*” (garbage collector).

Загальна форма оператора delete для одиночної змінної:

delete ptrName;

де *ptrName* – ім‘я покажчика, для якого було раніше виділено пам‘ять оператором new. Після виконання оператора delete покажчик ptrName вказує на довільний фрагмент пам‘яті, який не є зарезервований (виділений).

**4. Скільки байт в пам‘яті займає вказівник?**

На першому етапі оголошується вказівник на вказівник, який міститиме адресу початку допоміжного масиву адрес розмір якого рівний висоті двовимірного масиву (кількості рядків).

Кожен елемент масиву даних типу int займає у пам‘яті стільки

місця, скільки необхідно для зберігання одного значення типу int, а саме 4

байти, тому відстань між елементами масивів з даними рівна 4.

**5. Що таке зубчатий масив?**

Створюючи масиви з даними різного розміру у другому вимірі

динамічних двовимірних масивів можна створювати так звані зубчаті масиви, які в окремих випадках можуть значно зекономити пам‘ять в порівнянні з використанням двовимірних масивів. Приклад зубчатого масиву зображено на рис. 3.3. Принцип створення зубчатих масивів відрізняється від принципу створення двовимірних динамічних масивів лише етапом створення одновимірних масивів з даними, де на відміну від двовимріних динамічних масивів масиви з даними можуть мати різний розмір утворюючи не матрицю, а зубці, завдяки чому даний вид масивів і одержав свою назву.