Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»

Відділення другої вищої та післядипломної освіти

**ЛАБОРАТОРНА РОБОТА №2**

(варіант 23)

з курсу «Програмування»

на тему «Масиви та вказівники»

Виконав: студент 3-го курсу

групи ІС-зп71

Бутузов О.В.

Прийняв: викл. Древаль М.М.

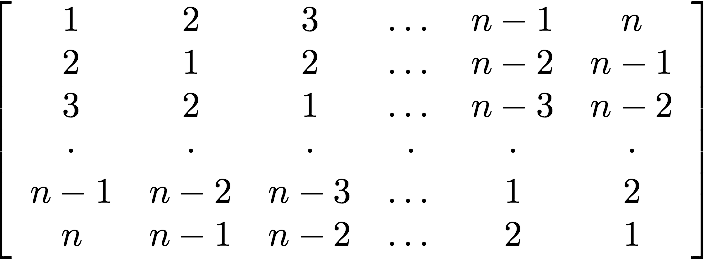
Захищено з оцінкою

« » 2017 р.

Київ – 2017

# Умова завдання

Отримати квадратну матрицю порядка n, заповнену за вказаною закономірністю.



Задачу розв’язати двома способами: використовуючи доступ до елементів масиву через індекси та за допомогою вказівників. Матриці, отримані в результаті виконання програмного продукту, вивести на екран.

# Розвязання задачі

Задача була імплементована використовуючи два методи заповнння масиву:

1. Наївний:
   1. Коли заповення тривало рядок за рядком, від елменту розташованого на головній діагоналі.
   2. Решта елементів рядку заповнювалась шляхом дзеркалнього копіювання вже заповнений рядків.

Цей спосіб легкий для розуміння оскільки таким чином прицює людська уява.

1. За формулою – коли значення рядку було число різниці між індексом елементу головної діагоналі (данного рядка) і порядкового індексу елементу + 1 (оскільки ми почанаємо рахувати з 1 а не з 0).

Щодо вигляду заповнення масиву то ми використали (як задано умовою задачі) два різних типи звертання до елементів масиву, і щоб взагалі зрозуміти природу імплементації масиву в мові програмування C/C++ я такою використав два різних типа мисивів – так званий одномірний та двомірний.

Для наглядної демонстрації – динамічне резервування памяті проходило за допомогою функції calloc, на противагу статичному декларуванню масива (коли “під капотом”) використовувається malloc.

Звертання за індексом та звертання за вказівником взаємно замінні, хоч використання доступу за індексом більш приємне для фінального користувача.

# Лістинг програм

Оскільки результатом лабораторної роботи є не 1 програма а 8, я приводжу код лише 2 із них (з коментарями) – всі інші джерельні файли програм доступні в директорії з лабораторною роботою.

// problem\_2\_v23\_2d\_dynamic\_9\_formula.cpp

#include <iostream>

#define N 9

// Данна программа працює з динамічно створеним двомірним масивом інтів

// шляхом заповнення його за формулою де значення елементу буде модуль різниці

// індексу елемента головної діагоналі та індексом елементу + 1.

void matrix\_fill(int\*\* matrix){

for ( int i = 0, total=N\*N; i < total; i++ ) {

int row = i / N;

int col = i % N;

int index = row \* N + col ;

\*( \*( matrix + row ) + (col) ) = 1 + abs( (row \* N + row) - index );

}

}

void matrix\_print(int\*\* matrix){

for (int row = 0; row < N; row++){

for (int col = 0; col < N; col++){

std::cout

<< \*( \*( matrix + row ) + (col) )

<< (col+1 == N ? "":" ");

}

std::cout << std::endl;

}

}

int main( int argc, char \*argv[] ){

int\*\* matrix = (int \*\*) calloc( N, sizeof( int \* ) );

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

\*(matrix+i) = (int \*) calloc( N, sizeof( int ) );

}

matrix\_fill(matrix);

matrix\_print(matrix);

free(matrix);

return 0;

}

// problem\_2\_v23\_1d\_static\_9\_simple.cpp

#include <iostream>

#define N 9

// Программа скомпільована з цього коду працює з одномірним масивом інтів   
// (що фінально буде показаний як матриця) шляхом заповнення його   
// наївним способом вздовж центральної діагоналі а пустуючі елементи рядку  
// будуть дзеркально заповнюватись. До ступ в данному впадку проходить   
// за індексами.

void matrix\_fill(int matrix[]){

for (int row = 0; row < N; row++){

for (int col=0,stop=N-row; col < stop; col++){

matrix[((row\*N) + (row+col))] = col+1;

}

for (int col=0; col < row; col++){

matrix[((row\*N) + (col))] = matrix[((col\*N) + (row))];

}}

}

void matrix\_print(int matrix[]){

for (int row = 0; row < N; row++){

for (int col = 0; col < N; col++){

std::cout

<< matrix[( row\*N + col )]

<< (col+1 == N ? "":" ");

}

std::cout << std::endl;

}

}

int main( int argc, char \*argv[] ){

int matrix[N\*N];

matrix\_fill(matrix);

matrix\_print(matrix);

return 0;

}

# Результати роботи програми

Результати роботи програми подаємо у вигляді таблиці контрольних значень (табл. 1).

Результати роботи програми однакові для усіх (8) імплементацій.

Таблиця 1. Таблиця контрольних значень

|  |  |  |
| --- | --- | --- |
| № тесту | Вхідні дані | Результати |
| 1. | N = 1 | 1 |
| 2. | N = 2 | 1 2  2 1 |
| 3. | N = 5 | 1 2 3 4 5  2 1 2 3 4  3 2 1 2 3  4 3 2 1 2  5 4 3 2 1 |

# Висновки

Масив є однієї із базових впорядкованих структур данних мови програмування C/С++ (та і інших мов) що реалізована за допомогою розтяшованих під ряд вказівників, спеціальної конструкції в мп що дозволяє звертатись напряму до комірки памяті. Масив як такий зберігає лише вказівники, які вже в свою чергу зсилається на регіони памяті що призначені для (попередньо) задекларованих типів данних.

Ми можемо задекларувати масив статично (наприклад int matrix[5]) або динамічно (використовуючи malloc або calloc), з моєї точки зору динамічна декларація масиву є більш гнучкою оскільки дозволяє звільняти пам'ять під час виконання программи, чим статично декларовані масиви не можуть похизуватись.

З точки зору який тип масиву використовувати я схиляюсь до думки що з одномірними масивами легше працювати, вони простіші для імплементацій, (наприклад не потрібно декларувати другий вимір масиву (розмір суб масиву)), з іншої сторони зникає абстракція звертання до індексу рядку і колонки, і не обхідно попередньо обчислювати індекс елементу для звертання за вказівником. Щодо вказівників і масивів – ім’я змінної масиву є вказівником на перший його елемент (за індексом нуль), і наступні індекси рахуються простим додаванням індексу елементу до імені змінної елементу (якщо прослідкувати нумерацію комірок пам’яті то кожен наступний відрізок пам’яті має розмір задекларованої змінної).

Вказівники є вкрай потужними інстурментами мови програмування C/C++, але з яким потрібно бути вкарй обережним.