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.

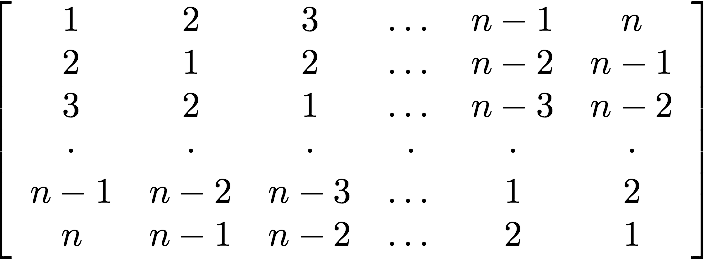
Захищено з оцінкою

« » 2017 р.

Київ – 2017

# Умова завдання

Отримати квадратну матрицю порядка n, заповнену за вказаною закономірністю.



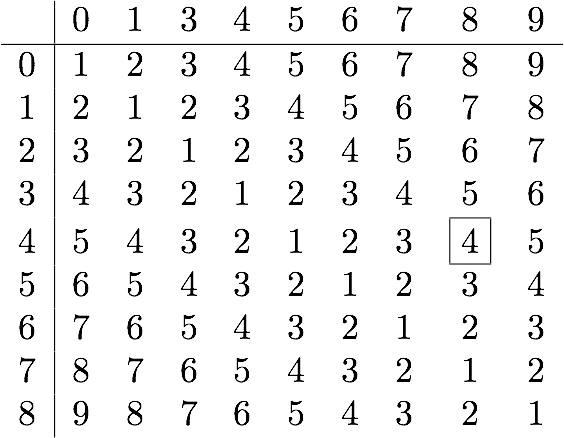
Задачу розв’язати двома способами: використовуючи доступ до елементів масиву через індекси та за допомогою вказівників. Матриці, отримані в результаті виконання програмного продукту, вивести на екран.

# Розвязання задачі

Опис алгоритму заповнення квадратної матриці симетричними значенням:

Оскільки задачею є вивести діагональну симетричну матрицю заповнену в порядку зростання арифметичної прогресії в обидві сторони від центру послідовності (центр послідовності знаходиться на головній діагоналі і починається з цифри 1), ми будемо створювати наші матриці в наступному порядку.

1. Введемо під час компіляції шаблонну заміну що буде вказувати на розмірність матриці – N (і надалі в описі алгоритму я буду звертатись до цієї розмірності як просто до N).
2. Запустимо цикл що буде проходити ряд за рядом - починаючи з першого ряду за індексом нуль закінчуючи N-им (за індексом N-1).
3. Заповнення рядка будемо починати з елементу головної діагоналі, а оскільки матриця діагональна та симетрична цим елементом буде елемент індекс якого в даному ряді буде дорівнювати номеру рядку, тобто для ряду 8 заповнення почнеться з елементу 8, а для ряду 0 з елемента 0. Напрям руху заповнення, вправо від центру до кінця можливого ряду (який буде дорівнювати (N-індекс рядку)-1), надаючи комірці інкрементоване на 1 значення від номера елементу відносно елементу головної діагоналі (тобто 3ій елемент відносно головної діагоналі в рядку 4 буде мати індекс 8 і значення 4 – див малюнок заповненої матриці).
4. Після того як заповнений ряд над головною діагоналлю ми повертаємось на початок ряду і починаю до-заповнюювати його значеннями що знаходять діаметрально дзеркально відносно головної діагоналі (тоюто елемент 1 в рядку 2 пряйме значення елмента 2 в рядку 1). Оскільки попередній ряд вже заповнено ця операція абсолютно безпечна, зуминяємо процес до-заповнення в останньому елменті перед головною діагональю.
5. Ряд заповнено.
6. Переходимо на новий ряд або зупиняємо роботу “заповнювача” в разі якщо ми досягнули обмежувальної межі N.



Особливості імплементації:

При роботі з **індексами** ми можемо абстрагуватись під структури даних що власне і створює масив, і утилізувати на повну доступ по індексам тобто звернення вигляду matrix[2][2] для присвоєння та читання. Цей спосіб дає нам відносну простоту (як щойно було сказано) при присвоєнні або читанні, але робить технічну імплементацію набагато складнішою тому що доводиться наперед конструювати програму так щоб функції сприймали багатомірні масиви тощо.

Альтернативно, роботу з масивом можна проводити і напряму за вказівниками, не використовуючи “цукор” у вигляді доступу за індексом. Вказівник це спеціальний об’єкт чиє значення це адреса пам’яті комп’ютера (власне усі масиви являють собою вказівники на певні ділянки пам’яті попередньо задекларованим розміром). Це не так зручно, оскільки доступ треба проводити звертаючись до але має свої переваги:

* Приберається рівень абстракції (так званий синтаксичний цукор, більш відомий як доступ по індексу), завдяки чому виникає розуміння як же саме організована ця структура данних в памяті і як з нею можна працювати.
* Розуміння чого в свою чергу призводить до розуміння не важливості “багатомірності” маивів, і досягнення таких же результатів при операції з одномірною структурою (див імплементація прикладу з вказівниками).

В обох випадках можна використовувати доступ як за вказівником так і за індексом.

# Лістинг програми

1. Лістинг программи - problem\_2\_23\_indeces.cpp

#include <iostream>

#define N 9

// це приклад программи що використовує індекси для доступу до елементів

// масиву. В данному разі ми працюємо з двомірним масивом (або масивом вказівників на вказівники типу інт)

void matrix\_fill(int matrix[][N]){

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

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

matrix[row][row+col] = col+1;

}

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

matrix[row][col] = matrix[col][row];

}

}

}

void matrix\_print(int matrix[][N]){

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 matrix[N][N];

matrix\_fill(matrix);

matrix\_print(matrix);

return 0;

}

1. Лістинг программи - problem\_2\_23\_pointers.cpp

#include <iostream>

#define N 9

// використовує той же самий алгоритм заповнення

// що і попередній приклад – але доступ надається напряму

// через вказівник

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

matrix\_fill(matrix);

matrix\_print(matrix);

return 0;

}

1. Лістинг программи - problem\_2\_23\_alt\_pointers.cpp

#include <iostream>

#define N 9

// цей приклад заповнення використовує вказівники так само як і попередній

// але ткаож альтернативний спосіб заповнення де значення вираховується

// за формулою = |CurrentRowDiagonalElementIndex – CurrentElementIndex |+ 1

void matrix\_fill(int\* matrix){

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

\*( matrix + i ) = 1 + abs(

( ( ( i / N ) \* N ) + i / N ) - i

) ;

}

}

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

matrix\_fill(matrix);

matrix\_print(matrix);

return 0;

}

# Результати роботи програми

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

Результати роботи программи однакові для усіх трьох вищеприведених 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 |
| 4. | N=9 | 1 2 3 4 5 6 7 8 9  2 1 2 3 4 5 6 7 8  3 2 1 2 3 4 5 6 7  4 3 2 1 2 3 4 5 6  5 4 3 2 1 2 3 4 5  6 5 4 3 2 1 2 3 4  7 6 5 4 3 2 1 2 3  8 7 6 5 4 3 2 1 2  9 8 7 6 5 4 3 2 1 |

# Висновки

Массив є однією з базовых структур данных мови програмування C++, він реалізований як послідовність вказівників, що в свою чергу вказують на пам'ять що зарезервована (аллокована) для певного типу даних.

* Можливо використовувати інтуїтивно зрозумілу модель доступу до елементів масиву через індекси, цей шлях накладає певні обмеження на дизайн програмних продуктів.
* Іншим варіантом є користування вказівників напряму, що не є інтуїтивно зрозумілим процесом (оскільки потрібно розрахувати попередньо позицію елемента відносно початку масиву).

Обива процеси є (майже) однаковими з точки зору программи (під «капотом» скомпільованого коду) оскільки відбуваються операції з вказівниками. З усього вищесказаного слідує що не треба забувати про модель доступу для масиву і те що доступ до данних надається через вказівники, що означає, що такі данні треба опрацьовувати обережно і не доспускати зміни данних де не треба а ще краще по можливості використовувати вбудовані або альтернативні C++ структури даних що можуть пропонувати програмісту значно ширші можливості ніж простий голий масив вказівників.

Висновки щодо конкретного завдання – було реалізовано дві різні варіанти звернення до масиву, в першому варіанті це був двомірний масив (масив вказівників на вказівники на тип даних ціле число), у другому був використаний одномірний масив розміром N квадрат. Алгоритм заповнення був визначений на користь інтуїтивно зрозумілого для людини (під інтуїтивно-зрозумілим я маю такий алгоритм що простіше “уявити”), альтернативно було використано просто формулу заповнення де значення елементу масиву (одномірного) дорівнювало б модулю різниці порядкового індекса елемента та індексу елемента головної діагоналі цього ряду (реалізація problem\_2\_23\_alt\_pointers.cpp)