МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний авіаційний університет

Кафедра прикладної інформатики

Лабараторна робота №10

З дисципліни “Теорія алгоритмів”

.

Виконав:

Студент 1 курсу ІКІТ

Групи УС-112

Лазоренко Максим Олександрович

Перевірив:Ходаков Д.В

Київ 2016

# ТЕМА: Двовимірні масиви

Теоретичні відомості:

До цього моменту, ми розглядали тільки одномірні масиви, тобто, до елементу масиву ми зверталися через один індекс. Однак, масиви можуть бути і двовимірними і тривимірними і, навіть, н-мірними. Багатовимірні масиви – це масиви, у яких є більше одного індексу. Замість одного рядка елементів, багатовимірні масиви можна розглядати як сукупність елементів, які розподілені по двох або більше вимірам. Ось так, наприклад, можна візуалізувати двовимірний масив:

|  |  |
| --- | --- |
| 1  2  3 | [][][][][]  [][][][][]  [][][][][] |

У цьому прикладі зображений двовимірний масив розміром 3\*5, 3 – рядка і 5 стовпців. Оголошення двовимірного масиву майже нічим не відрізняється від оголошення одновимірного, за винятком того, що при оголошенні двовимірного масиву, потрібно вказувати розмір кожного вимірювання в квадратних дужках. Наприклад, давайте оголосимо двовимірний масив розміром 8\*8, це розмір поля для стандартних шашок – 8 строк и 8 стовпців:

|  |  |
| --- | --- |
| 1 | int checkers[8][8]; // двумерный массив |

Тобто, двовимірний масив добре підходить для зберігання інформації на шашковому поле. Також двовимірний масив можна легко використовувати для зберігання інформації про будь-який інший грі – шахи, хрестики нулики, сапер і т. д. Щоб отримати доступ до будь-якого елементу такого масиву, потрібно скористатися двома значеннями – індексами, перший індекс – це номер рядка, а другий – номер стовпчика. Все вище сказане відноситься і до n-мірним масивам. Хоча, вже 4-х мірні масиви важкувато візуалізувати. Присвоювати значення елементів масиву дуже просто, ось приклад:

|  |  |
| --- | --- |
| 1  2 | // присваиваем первому элементу массива значение - 5  myArray[0] = 5; |

Приклад привласнення значення елемента двовимірного масиву:

|  |  |
| --- | --- |
| 1  2 | // присваиваем первому элементу массива значение - 5  myArray[1][3] = 10; |

У цьому прикладі ми присвоїли значення 10 елементу двовимірного масиву myArray, який знаходиться в другому рядку і в 4-му стовпці. Візуально це виглядає так:

|  |  |
| --- | --- |
| 1  2  3 | [\_\_][\_\_][\_\_][\_\_][\_\_]  [\_\_][\_\_][\_\_][10][\_\_]  [\_\_][\_\_][\_\_][\_\_][\_\_] |

Як бачите, все просто, головне пам'ятайте, що нумерація рядків і стовпців завжди починається з 0. Тобто, я ще раз хочу вам нагадати, що ви ніколи не повинні намагатися записати дані після останнього елемента масиву, наприклад, коли у вас є масив розміром – 10 елементів і ви намагаєтеся привласнити значення елементу з індексом [10]. Пам'ять для масиву була виділена тільки для десяти елементів, (індекси від 0 до 9), тому елемента з індексом 10 просто не існує. В такому випадку, запис в оперативній пам'яті може призвести до непередбачуваних наслідків – наприклад, ви можете в кінцевому підсумку зіпсувати роботу паралельно запущеної програми. Однак, як правило, операційна система не дозволить такого роду безрозсудну поведінку і призведе до краху програми, якщо та спробує отримати доступ до нерозподіленого пам'яті.

Давайте розглянемо практичний приклад використання масивів у програмах:

# Завдання :

# Варіант №13

|  |  |
| --- | --- |
| Заповнити матрицю, від лівого нижнього кута по діагоналі: ліворуч - вгору. | Описание: Описание: l09_e009 |



Текст програми на с++

#include "stdafx.h"

#include <iostream>

using namespace std;

int main()

{

const int S = 15;

int A[S][S];

int s;

cin >> s;

short i, j, x = 1;

int n;

int i0;

for (i = 0; i < s; i++)

{

for (j = 0; j < s; ++j)

{

n = (i + j + 1)// рахуємо число елементів діагналі, якій належить A[i][j]

if (n > s)// якщо умова виконується, то елемент A[i][j] знаходиться під побічною діагоналлю

{

n = 2 \* s - (j + i + 1); // правильно перераховуємо кількість елементів в діагоналі

A[i][j] = i + (s - 1)\*(s - 2) / 2 - n\*(n - 1) / 2 + s\*(s + 1) / 2;

/\*(s - 1)\*(s - 2) / 2 - n\*(n - 1) / 2 - кількість елементів між побічною діагоналлю и теперішньою діагоналлю

s\*(s + 1) / 2 - кількість елементів головної діагоналі і над нею\*/}

else

{

A[i][j] = n - j + n\*(n - 1) / 2;

/\*n - i - номер элемента в діагоналі

n\*(n - 1) / 2 - кількість елементів в трикутнику обмеженому діагоналлю в якій знаходиться елемент\*/

}

cout<<A[i][j]<<"\t";}

cout <<endl;}

for (int m = s-1; m >= 0; m--)

{

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

{

cout << A[m][n] << "\t";

}cout << endl;

}

system("pause");

return 0;}

**Результат програми:**

