Звіт

Автор: Момот Р. КІТ-119а

Дата: 10.04.2020

ЛАБОРАТОРНА РОБОТА № 5. ФІЗИЧНЕ ПОДАННЯ СПЕЦИФІЧНИХ МАСИВІВ

Мета: придбання і закріплення навичок програмування розміщення в пам’яті специфічних масивів.

Індивідуальне завдання

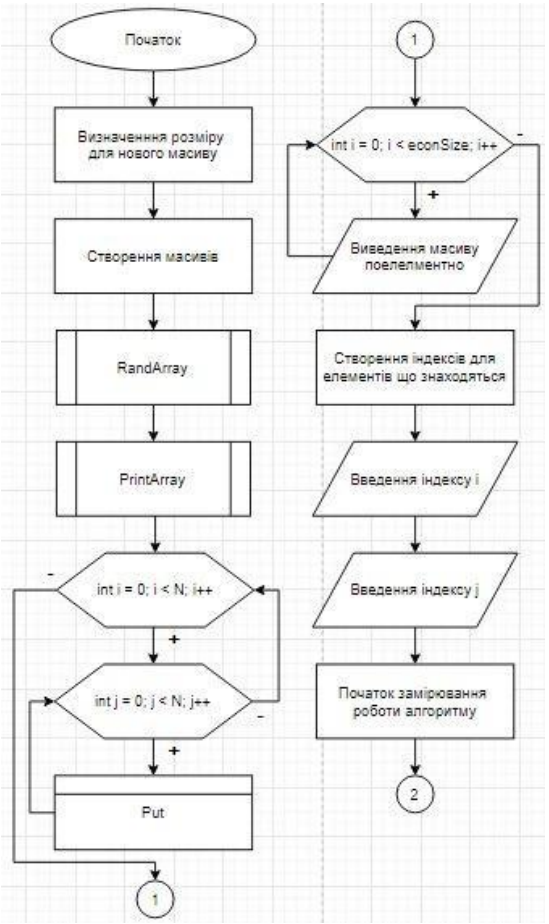
Розробити спосіб ощадливого розміщення в пам’яті заданої розрідженої таблиці, де записані цілі числа. Розробити функції, що забезпечують доступ до елементів таблиці по номерах рядка і стовпця.

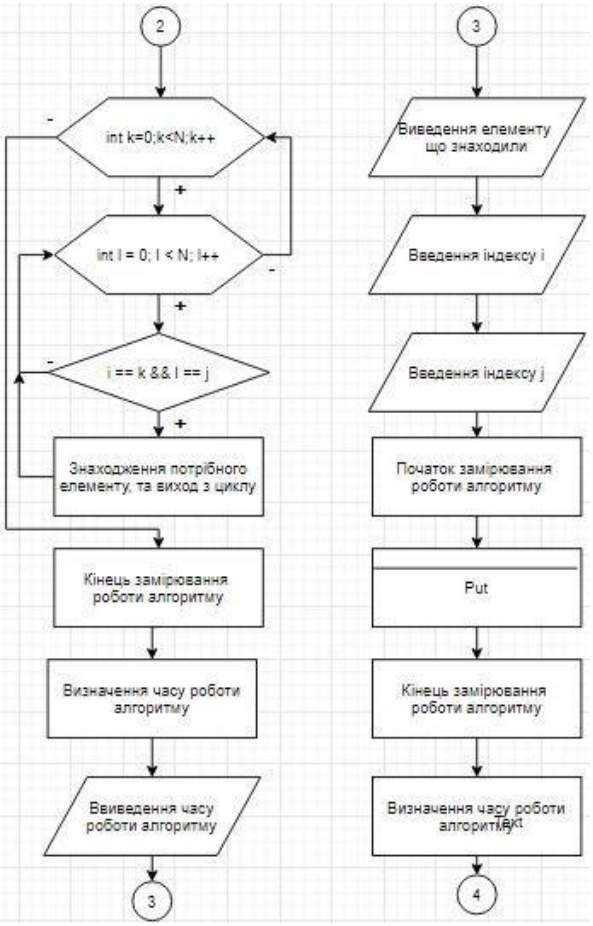
У програмі забезпечити запис і читання всіх елементів таблиці.

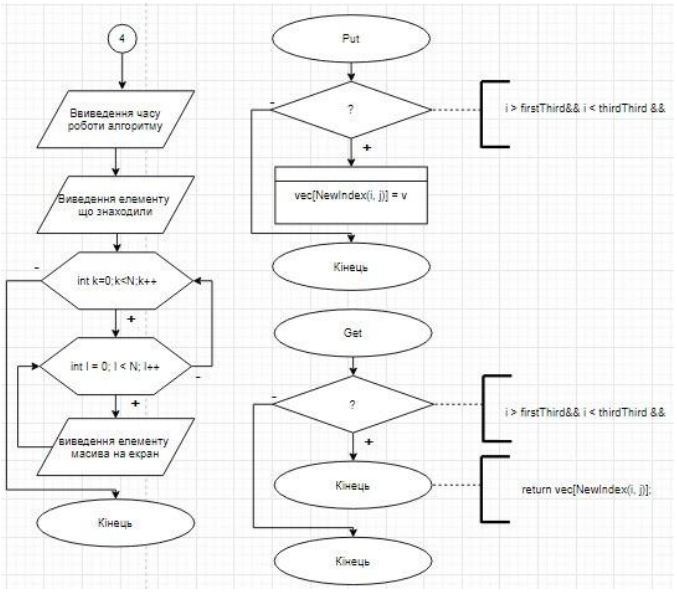
Визначити та порівняти час доступу до елементів таблиці при традиційному та ощадливому поданні її в пам’яті.

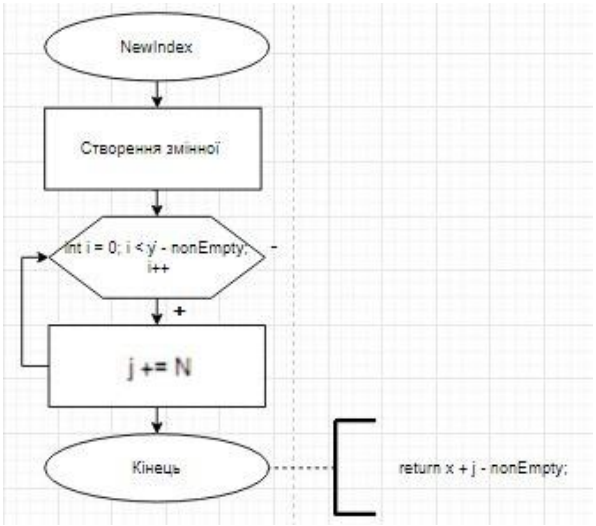
Вміст розрідженої матриці: нульові елементи розташовані у лівій та правій тритинах стовпців матриці.

Блок-схема алгоритму програми







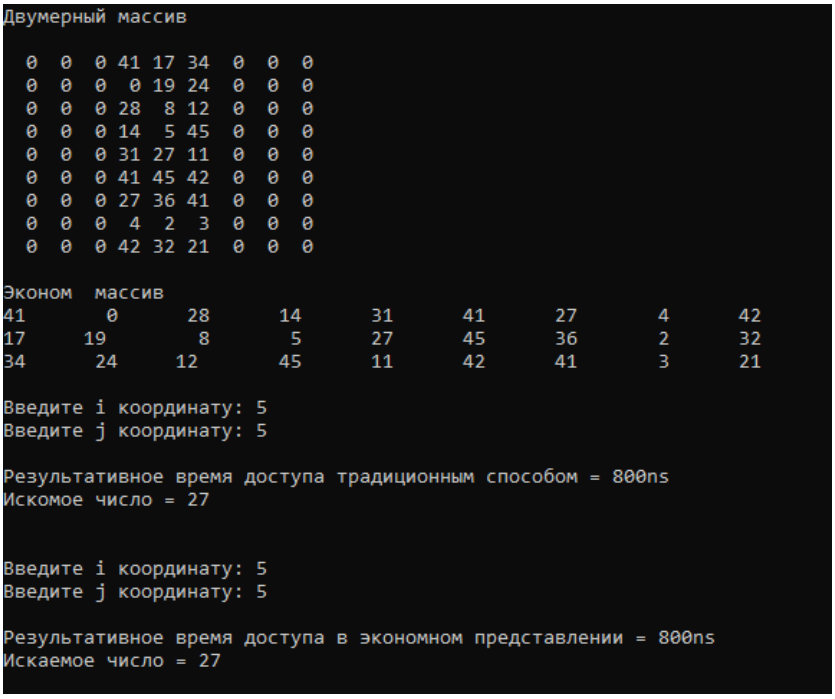


Текст програми

```
#define N 9
#include <iostream>
#include <chrono>
#include <locale>
using namespace std;

static int firstThird = N / 3 - 1;
static int thirdThird = N / 3 * 2;
static int nonEmpty = N / 3;

int NewIndex(int, int);
void Put(int[], int, int, int);
int Get(int[], int, int);
void RandArray(int[N][N]);
void PrintArray(int[N][N]);

void main()
{
    setlocale(LC_ALL, "Rus");
    const int econSizeX = N, econSizeY = N / 3;
    const int econSize = econSizeX * econSizeY;
    int arr[N][N];
    int econArr[econSize];
    int i, j, element;

    RandArray(arr);
    PrintArray(arr);

    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            Put(econArr, i, j, arr[i][j]);

    printf("\nЭконом  массив\n");
    for (int i = 0; i < econSize; i++)
        printf("%2i\t", econArr[i]);
    printf("\n");

    cout << endl << "Введите i координату: ";
    cin >> i;
    cout << "Введите j координату: ";
    cin >> j;

    auto begin = chrono::steady_clock::now();
    for (int k = 0; k < N; k++)
        for (int l = 0; l < N; l++)
            if ((i - 1) == k && l == (j - 1))
            {
                element = arr[k][l];
                l = N;
                k = N;
            }
    auto end = chrono::steady_clock::now();
    auto elapsed_ms = chrono::duration_cast<chrono::nanoseconds>(end - begin);

    cout << "\nРезультативное время доступа традиционным способом = " << elapsed_ms.count() <<
"ns" << endl; // время работы программы
    printf("Искомое число = %d\n\n", element);

    cout << endl << "Введите i координату: ";
    cin >> i;
    cout << "Введите j координату: ";
    cin >> j;
```

```
    auto begin2 = chrono::steady_clock::now();
    element = Get(econArr, i-1, j-1);
    auto end2 = chrono::steady_clock::now();
    auto elapsed_ms2 = chrono::duration_cast<chrono::nanoseconds>(end2 - begin2);

    cout << "\nРезультативное время доступа в экономном представлении = " <<
elapsed_ms2.count() << "ns" << endl; // время работы программы
    printf("Искаемое число = %d\n", element);
}

int NewIndex(int x, int y)                  //пересчёт индексов
{
    int j = 0;
    for (int i = 0; i < y - nonEmpty; i++)
        j += N;
    return x + j;
}

void Put(int vec[], int i, int j, int v)    //запись из двумерного массива в эконом
{
    if (j > firstThird&& j < thirdThird) vec[NewIndex(i, j)] = v;
}

int Get(int vec[], int i, int j)            //запись данных в эконом. массив
{
    if (j > firstThird&& j < thirdThird) return vec[NewIndex(i, j)];
    else return 0;
}

void RandArray(int a[N][N])                 //заполнение массива нужными элементами
{
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            if (j > firstThird&& j < thirdThird) a[i][j] = rand() % 50;
            else a[i][j] = 0;
}

void PrintArray(int a[N][N])
{
    printf("Двумерный массив\n\n");
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            printf("%3i", a[i][j]);
        printf("\n");
    }
}
```

Результати роботи програми



Висновок

У результаті роботи програми було розроблено спосіб ощадливого розміщення в пам’яті заданої розрідженої таблиці, де записані цілі числа. Також було порівняно доступ до елементів таблиці по номерах рядка і стовпця при звичайному (двомірний масив) та ощадливому (вектор) доступі.

Доступ при ощадливому способі швидший, бо при такому пошуку не враховуються нульові елементи, які і сповільнюють пошук при звичайному доступі.

Відповіді на питання

1. Що таке «дескриптор» масива, яке його призначення?

Дескриптор (заголовок) містить загальні відомості про фізичну структуру. Дескриптор необхідний, наприклад, в тому випадку, коли граничні значення індексів елементів масиву невідомі на етапі компіляції, і, отже, виділення пам'яті для масиву може бути виконано тільки на етапі виконання програми.

2. Як масиви представляються в пам’яті?

Елементи масиву в пам’яті машини фізично розташовуються послідовно. Кожен елемент займає в пам’яті кількість байт, що відповідає його розміру.

3. Як визначити обсяг пам’яті, що необхідний для запису масива?

Одним із способів реалізації статичних масивів з одним типом елементів є наступний: під масив виділяється безперервний блок пам'яті об'ємом
$S * N_1 * \ldots * N_n$, де S - розмір одного елемента, N_1 ... N_n - розміри діапазонів індексів (тобто кількість значень, які може приймати відповідний індекс).

4. Як визначається адреса єлемента масива?

При зверненні до елементу масиву $a[i_1, i_2, i_3, ..., i_n]$ адреса відповідного елемента обчислюється як

$@a\ [i_1, i_2, i_3, ..., i_n] = a_0 + S * ((... (i_{1p} * N_1 + i_{2p}) * N_2 + ... + i\ (n-1)\ p) * N_{n-1} + i_{np})$, де A_0 - базовий адреса (адреса початку виділеного блоку пам'яті для масиву), i_{kp} - значення k-го індексу, наведене до цілого з нульовим початковим зміщенням.

5. Чи залежить час доступу до елемента масива від мірності цього масива и чому?

Час звернень до елементу масиву залежить від рівномірності масиву і становить:

1-мірний масив - 2 операції, 2-мірний масив - 4 операції,
3-мірний масив - 6 операції, n-мірний масив - 2 * n операції.

Висновок: час доступу до елементу масиву тим більше, чим більше розмірність масиву.

6. Чим динамічні масиви відрізняються від статичних?

Динамічними називаються масиви, розмір яких може змінюватися під час виконання програми. Статичні масиви такої можливості не мають.

7. Чим характеризуються масиви змінної довжини, в чому їх відмінність від динамічних?

При роботі з масивами змінної довжини їх розмір не фіксується при компіляції, а задається при створенні або ініціалізації масиву під час виконання програми. Від динамічних масивів вони відрізняються тим, що для них не надаються засоби автоматичної зміни розміру зі збереженням вмісту, так що при необхідності програміст повинен реалізувати такі засоби самостійно.

8. Що таке «гетерогенні масиви»?

Гетерогенним називається масив, в різні елементи якого можуть бути безпосередньо записані значення, що відносяться до різних типів даних.

9. На логічному рівні як представляються ассоциативні масиви?

Асоціативний масив - абстрактний тип даних, що дозволяє зберігати пари виду «(ключ, значення)» - і підтримує операції додавання пари, а також пошуку і видалення пари по ключу.

Асоціативний масив з точки зору інтерфейсу зручно розглядати як звичайний масив, в якому в якості індексів можна використовувати не тільки цілі числа, а й значення інших типів - наприклад, рядки.

10. Які масиви вважаються симметричними, як їх подають на фізичному рівні?

Двовимірний масив, в якому кількість рядків дорівнює кількості стовпців називається квадратною матрицею. Квадратна матриця, у якій елементи, розташовані симетрично щодо головної діагоналі, попарно рівні один одному, називається симетричною.

Якщо матриця порядку n симетрична, то в її фізичну структуру досить відобразити не N^2, а лише ***N*(N+1)/2*** її елементів. Іншими словами, в пам'яті необхідно подати тільки верхній (включаючи і діагональ) трикутник квадратної логічної структури.

11. За яких умов масиви вважаються розрідженими?

Розріджений масив, або розріджена матриця (sparse array), - це масив, в якому не всі елементи використовуються, є в наявності або потрібні в даний момент. Розріджена матриця - це матриця з переважно нульовими елементами. В іншому випадку, якщо більша частина елементів матриці ненульові, матриця вважається щільною.

12. Яке призначення дерева відрізків і яке відношення вони мають до масивів?

Дерево відрізків - це структура даних, яка дозволяє ефективно реалізувати операції такого вигляду: знаходження суми/мінімуму елементів масиву в заданому відрізку ***a[l...r]***, де ***l*** і ***r*** надходять на вхід алгоритму, при цьому додатково можлива зміна елементів масиву: як зміна значення одного елемента, так і зміна елементів на цілому подотрезке масиву (тобто дозволяється привласнити всім елементам **a[l ...r]** будь-яке значення, або додати до всіх елементів масиву якесь число). Цими операціями дерево відрізків не обмежується, існують та інші більш складні операції.

Дерево відрізків має лінійний розмір.

Підраховують і запам'ятовують суму елементів всього масиву, тобто відрізка ***a [0...n-1]***, а також суму на двох половинках цього масиву: ***a[0...n/2]***, і ***a[n/2+1...n-1]***. Кожну з цих двох половинок в свою чергу розбивають навпіл, вважають і зберігають суму на них, потім знову розбивають навпіл, і так далі, поки поточний відрізок не досягне довжини 1.

13. Що таке V-список, в чому його переваги?

V-список - структура даних, розроблена Філом Багвелл в 2002 році. V-список складається зі звичайного списку масивів, розміри яких утворюють геометричну прогресію.

V-список об'єднує в собі швидкий доступ до випадкових елементів і швидке розширення списка. Для того, щоб знайти елемент у V-списку, треба знати всього лише адреса масиву, в якому знаходиться шуканий елемент і його індекс в цьому масиві.

14. Коли необхідні паралельні масиви?

У паралельних масивів є ряд практичних переваг в порівнянні з класичним підходом:

• Вони можуть бути використані в мовах, які підтримують тільки масиви примітивних типів, але не підтримують масиви записів, або не підтримують записи зовсім.

• Вони можуть зберегти відчутний обсяг пам'яті в деяких випадках, тому що більш ефективно вирішують питання вирівнювання.

• Якщо кількість елементів мало, індекси масиву займають істотно менше простору, ніж повноцінні покажчики, особливо на архітектурі з великою розрядністю.

• Послідовне читання єдиного поля кожного запису в масиві дуже швидко на сучасних комп'ютерах, тому що це рівноцінно лінійному проходу по єдиному масиву, що дає ідеальні локальність і поведінку кешу.

15. Яке призначення хеш-таблиць, як вони створюються?

Хеш-таблиця (hash table) - це спеціальна структура даних для зберігання пар ключів і їх значень, це звичайний масив з незвичайною адресацією, що задається хеш-функцією. По суті це асоціативний масив. Головна властивість хеш-таблиць полягає в тому, що всі три операції вставка, пошук і видалення в середньому виконуються за час O(1), середній час пошуку по ній також одно O(1) і O(n) в гіршому випадку.