КІТ-119а

Автор: Чугунов В.Ю

**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. Визначити та порівняти час доступу до елементів таблиці при традиційному та ощадливому поданні її в пам’яті . Усі елементи парних стовпців – нульові

Текст програми

#include <iostream>

#include <stdio.h>

#include <cstdlib>

#include <chrono>

using namespace std;

#define N 6

int NewIndex(int x, int y) //пересчет индексов

{

int j = 0;

for (int i = 0; i < x; i++) j += N - i;

return j + y - x;

}

void Put(int vec[], int x, int y, int v) // Запись в вектор (сжатие)

{

if (y >= x) vec[NewIndex(x, y)] = v;

}

int Get(int vec[], int x, int y) // Чтение из вектора

{

if (y >= x) return vec[NewIndex(x, y)];

else return 0;

}

void RandArray(int a[N][N]) //формирование исходного массива

{

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

{

for (int j = 0; j < N; j++)

{

if (j == 0)

{

a[i][j] = rand() % 50;

continue;

}

if (j%2 ==0)

{

a[i][j] = rand() % 50;

}

else

{

a[i] [j] = 0;

}

}

}

}

void PrintArray(int a[N][N])

{

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

{

for (int j = 0; j < N; j++)

printf("%3i", a[i][j]);

printf("\n");

}

}

void main()

{

setlocale(LC\_ALL, "RUS");

int num1 = 0;

int num2 = 0;

int vec[N \* N / 2 + N / 2];

int array[N][N];

RandArray(array);

PrintArray(array);

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

for (int j = 0; j < N; j++)

Put(vec, i, j, array[i][j]); //сжатие массива

cout << "\n Массив после сжатия \n";

for (int i = 0; i < N \* N / 2 + N / 2; i++) //выдача результата сжатия

printf("%3i", vec[i]);

printf("\n\nПолучение елемента матрицы:\n\n");

printf("Введите i координату\n");

scanf\_s("%d", &num1);

printf("Введите j координату\n");

scanf\_s("%d", &num2);

int elem;

auto begin = chrono::steady\_clock::now();

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

{

for (int j = 0; j < N; j++)

{

if (i == num1 && j == num2)

elem = array[i][j];

}

}

printf("Значение элемента - %d\n\n", elem);

auto end = chrono::steady\_clock::now();

auto elapsed\_ms = chrono::duration\_cast<chrono::nanoseconds>(end - begin);

cout << "Время доступа к элементу без сжатия: " << elapsed\_ms.count() << "ns" << endl << endl << endl;

printf("Получить значение с економного массива\n\n");

printf("Введите i координату\n");

scanf\_s("%d", &num1);

printf("Введите j координату\n");

scanf\_s("%d", &num2);

auto begin2 = chrono::steady\_clock::now();

printf("Значение элемента - %d\n\n", Get(vec, num1, num2));

auto end2 = chrono::steady\_clock::now();

auto elapsed\_ms2 = chrono::duration\_cast<chrono::nanoseconds>(end2 - begin2);

cout << "Время доступа к элементу одномерного массива: " << elapsed\_ms2.count() << "ns" << endl;

printf("\n\n");

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

{

for (int j = 0; j < N; j++) //чтение из сжатого представления

printf("%3i", Get(vec, i, j));

printf("\n");

}

}

Схема алгоритму програми

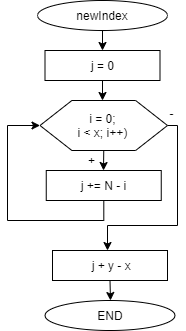
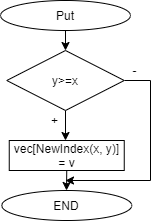
 

Рисунок 1 – Функції Put та newIndex

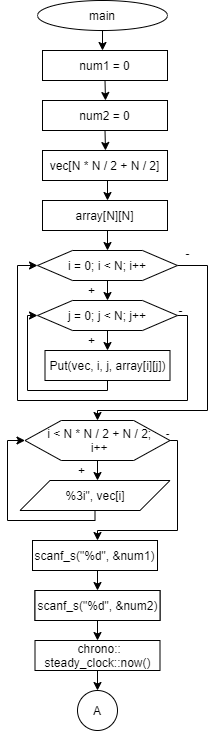
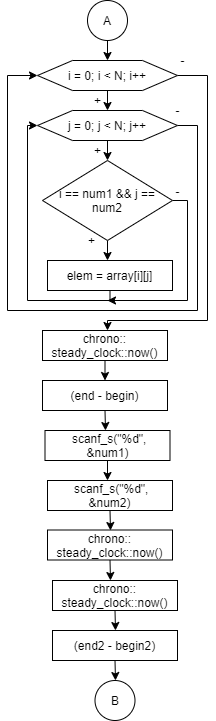
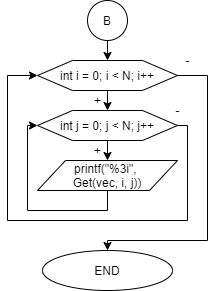


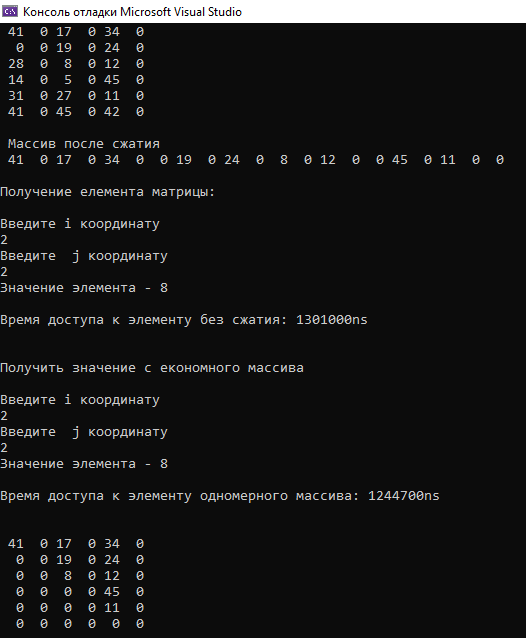
Рисунок 2 – Функція main





Продовження рис. 2

Результат роботи програми



Висновок

Під час виконання лабораторної роботи були здобуті навички розроблення способу ощадливого розміщення в пам’яті заданої розрідженої таблиці. Порівняно доступ до елементів таблиці по номерах рядка і стовпця при звичайному та ощадливому доступі.

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

Контрольні 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.

1. Як масиви представляються в пам’яті?

Елементи масиву в пам'яті машини фізично розташовуються послідовно, згідно з описом. При цьому кожен елемент займає в пам'яті кількість байт, що відповідає його розміру.

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

Його можна визначити за формулою:

S \* N1 \* … \* Nn, де S - розмір одного елемента, N1 ... Nn - розміри діапазонів індексів (тобто кількість значень, які може приймати відповідний індекс).

1. Як визначається адреса елемента масива?

При зверненні до елементу масиву a[i1, i2, i3, ..., in] адреса відповідного елемента обчислюється як:

**@a [i1, i2, i3, ..., in] = a0 + S \* ((... (i1p \* N1 + i2p) \* N2 + ... + i (n-1) p) \* Nn-1 + inp)**, де A0 - базовий адреса (адреса початку виділеного блоку пам'яті для масиву), ikp - значення k-го індексу, наведене до цілого з нульовим початковим зміщенням.

1. Чи залежить час доступу до елемента масива від мірності цього масива і 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.

Тобто, час доступу до елементу масиву тим більше, чим більше розмірність масиву.

1. Чим динамічні масиви відрізняються від статичних?

У динамічних масивах розмір може змінюватися під час виконання програми, а у статичних – ні.

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

У масивах змінної довжини не фіксується розмір при їх компіляції, а задається під час створення чи 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.

1. Що таке «гетерогенні масиви»?

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

1. 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.

1. Які масиви вважаються симметричними, як іх 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 N2, а лише ***N\*(N+1)/2*** її елементів. Іншими словами, в пам'яті необхідно подати тільки верхній (включаючи і діагональ) трикутник квадратної логічної структури.

1. 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.

1. 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 підвідрізку 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.

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

V-список - структура даних, розроблена Філом Багвелом 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 списку. Для того, щоб знайти елемент у V-списку, треба знати всього лише адресу масиву, в якому знаходиться шуканий елемент і його індекс в цьому масиві.

1. 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 мала, індекси масиву займають істотно менше простору, ніж повноцінні покажчики, особливо на архітектурі з великою розрядністю.

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

1. 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.

Існує два основних види хеш-таблиць: з ланцюжками і відкритою адресацією, які вирішують задачу вирішення колізій, тобто способи розміщення в хеш-таблиці пар ключ-значення, які дають однаковий хеш-код. Хеш-таблиця з відкритою адресацією - це масив пар ключ-значення. Хеш-таблиця з ланцюжками - це масив покажчиків на списки пар ключ-значення.