Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

Лабораторная работа №4

“Методы доступа к элементам массива”

Проверил: М.Кроитор

Выполнила:Журба Дарья

Группа: IA2102

Кишинев 2023

Задача:

Индивидуальный вариант:

Вектор Б

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr | l1..h1 | l2..h2 | l3..h3 | l4..h5 |
| 4 | -9..-7 | 1..2 | 0..3 | 2..5 |

Программа:

#include <iostream>

#include <vector>

#include <unistd.h>

using namespace std;

template<class element>

class MultidimensionalVector

{

privatr:

element \*vector;

int \*vector\_;

element \*\*\*\*iliffe;

int l1, h1;

int l2, h2;

int l3, h3;

int l4, h4;

public

MultidimensionalVector<element>(int l1, int h1, int l2, int h2, int l3, int h3, int l4, int h4)

{

this->l1 = l1;

this->h1 = h1;

this->l2 = l2;

this->h2 = h2;

this->l3 = l3;

this->h3 = h3;

this->l4 = l4;

this->h4 = h4;

int countElements = (this->h1 - this->l1 + 1) \* (this->h2 - this->l2 + 1) \* (this->h3 - this->l3 + 1) \* (this->h4 - this->l4 + 1);

vector = new element[countElements];

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

{

vector[i] = i;

}

}

vector\_ = new int[15];

vector\_[0] = 4;

vector\_[1] = l1;

vector\_[2] = h1;

vector\_[3] = l2;

vector\_[4] = h2;

vector\_[5] = l3;

vector\_[6] = h3;

vector\_[7] = l4;

vector\_[8] = h4;

vector\_[9] = (h1 - l1 +1) \* (h2 - l2 +1) \* (h3 - l3 +1) \* (h4 - l4 +1);

vector\_[13] = 1;

vector\_[12] = (h4 - l4 + 1) \* vector\_[13];

vector\_[11] = (h3 - l3 + 1) \* vector\_[12];

vector\_[10] = (h2 - l2 + 1) \* vector\_[11];

vector\_[14] = l1 \* vector\_[10] + l2 \* vector\_[11] + l3 \* vector\_[12] + l4 \* vector\_[13];

int i1, i2, i3, d, step;

d = -l4;

step = h4 - l4 + 1;

iliffe = new element \*\*\*[h1 - l1 + 1] - l1;

for (i1 = l1; i1 <= h1; i1++)

{

\*(iliffe + i1) = new elemnt \*\*[h2 - l2 + 1] - l2;

for (i2 = l2; i2 <= h2; i2++)

{

\*(\*(iliffe + i1) + i2) = new element \*[h3 - l3 + 1] - l3;

for (i3 = l3; i3 <= h3; i3++, d += step)

{

\*(\*(\*(iliffe + i1) + i2) + i3) = vector + d;

}

}

}

virtual element &byRow(int i1, int i2, int i3, int i4)

{

int D1, D2, D3, D4;

D4 = 1;

D3 = (this->h4 - this->l4 + 1) \* D4;

D2 = (this->h3 - this->l3 + 1) \* D3;

D1 = (this->h2 - this->l2 + 1) \* D2;

return vector[(i1 - this->l1) \* D1 + (i2 - this->l2) \* D2 + (i3 - this->l3) \* D3 +(i4 - this->l4) \* D4];

}

element &definingVector(int i1, int i2, int i3, int i4)

{

return vector[i1 \* vector\_[10] + i2 \* vector\_[11] + i3 \* vector\_[12] + i4 \* vector\_[13] - vector\_[14]];

}

element %Iliffe(int i1, int i2, int i3, int i4)

{

return \*(\*(\*(\*(iliffe + i1) + i2) + i3) + i4);

}

};

int main()

{

MultidimensionalVector<int> vector(-9, -7, 1, 2, 0, 3, 2, 5);

int count = 0;

double result1 = 0, result2 = 0, result3 = 0, result4 = 0;

for (int i1 = -9; i1<= -7; i1++){

for (int i2 = 1; i2<= 2; i2++){

for (int i3 = 0; i3<= 3; i3++){

for (int i4 = 2; i4<= 5; i4++){

auto start1 = chrono::steady\_clock::now();

vector.byRow(i1, i2, i3, i4);

auto end1 = chrono::steady\_clock::now();

result1 += chrono::duration\_cast<<chrono::nanoseconds>(end1 - start1).count();

auto start2 = chrono::steady\_clock::now();

vector.byCol(i1, i2, i3, i4);

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

result2 += chrono::duration\_cast<<chrono::nanoseconds>(end2 - start2).count();

auto start3 = chrono::steady\_clock::now();

vector.definingVector(i1, i2, i3, i4);

auto end3 = chrono::steady\_clock::now();

result3 += chrono::duration\_cast<<chrono::nanoseconds>(end3 - start3).count();

auto start4 = chrono::steady\_clock::now();

vector.Iliffe(i1, i2, i3, i4);

auto end4 = chrono::steady\_clock::now();

result4 += chrono::duration\_cast<<chrono::nanoseconds>(end4 - start4).count();

count++;

}

}

}

}

return0;

}

Это программа на C++, которая определяет класс под названием MultidimensionalVector, который позволяет создавать многомерные векторы. Класс включает в себя три метода для доступа к элементам вектора: byRow, definingVector и Iliffe.

Метод byRow возвращает элемент вектора, соответствующий заданным четырем индексам. Метод definingVector возвращает элемент вектора, вычисленный с использованием формулы, которая включает в себя четыре индекса и некоторые свойства вектора. Метод Iliffe возвращает элемент вектора, соответствующий заданным четырем индексам, используя последовательность Iliffe, которая представляет собой метод хранения многомерных массивов в линейном блоке памяти.

Программа также включает в себя основную функцию, которая создает объект MultidimensionalVector и вызывает метод byRow для всех элементов вектора для измерения времени выполнения. Программа вычисляет сумму времени выполнения и выводит ее в стандартный вывод.

Из результата мы видим, что метод доступа с использованием векторов Iliffe является самым быстрым и самым менее затратным по использованию памяти.