**Лабораторна робота № 5. Фізичне подання**

**специфічних масивів**

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

**1 Вимоги**

**1.1 Розробник**

* Макаренко Владислав Олександрович
* Студент 1-го курсу
* Групи КІТ-120а

**1.2 Загальне 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.

**1.2 Загальне завдання**

Нульові елементи розташовані на головній діагоналі та у нижній половині матриці нижче діагоналі

**2 Описи програм**

**Код програми**

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
#define N 8  
int NewIndex(int *x*, int *y*) //пересчет индексов  
{  
 int j=-1;  
 for(int i=0; i<*x*; i++){  
 j+= N-i - 1;  
 }  
 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 > i){  
 *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(){  
 int vec[N\*N/2-N/2];  
 int array[N][N];  
 RandArray(array);  
 clock\_t start = clock();  
 PrintArray(array);  
 clock\_t end = clock();  
 double time1 = (double)(end - start)/CLOCKS\_PER\_SEC;  
 printf("\n Time of access to table elements at traditional representation: %.10f second(s)\n", time1);  
 for(int i = 0; i < N; i++) {  
 for (int j = 0; j < N; j++) {  
 Put(vec, i, j, array[i][j]); //сжатие массив  
 }  
 }  
  
 printf("\n");  
 for(int i = 0; i < N\*N/2-N/2; i++) { //выдача результата сжатия  
 printf(" %3i", vec[i]);  
 }  
 printf("\n\n");  
 clock\_t begin = clock();  
 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");  
 }  
 clock\_t finish = clock();  
 double time2 = (double)(finish - begin)/CLOCKS\_PER\_SEC;  
 printf("\n Time of access to table elements at economical representation: %.10f second(s)\n", time2);  
}

**Результати виконання програми**

За алгоритмом коду дізнаємося значення внутрішнього подання структури з вирівнюванням (див. рис. 1).

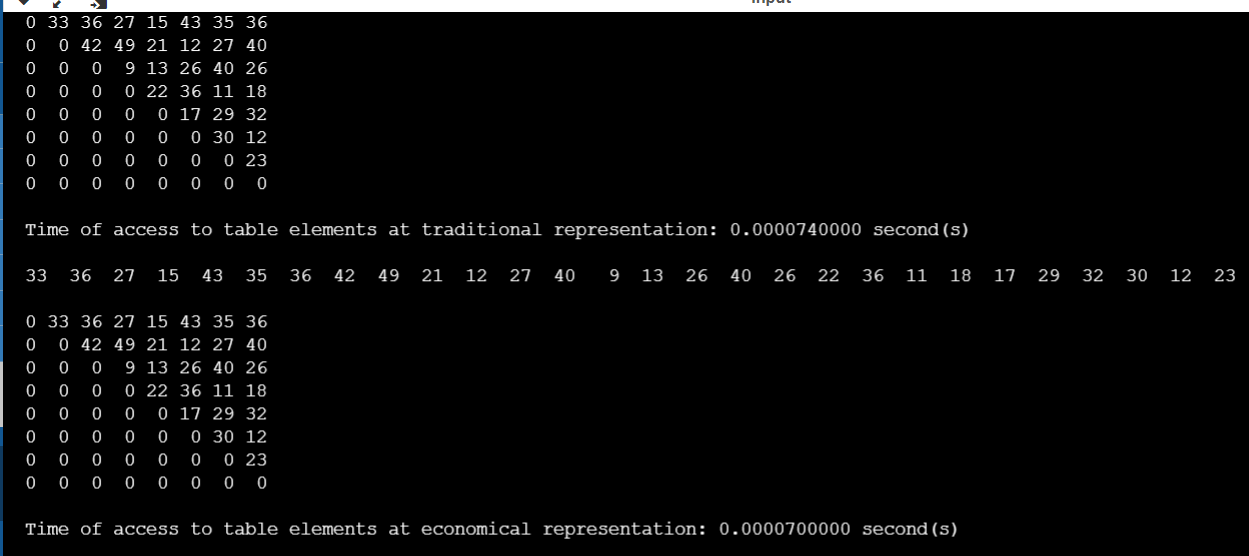


Рисунок 1 – Результати роботи першого кроку

**3 Порівняння часу**

Порівняємо час доступу до елементів у звичайному масиві(зліва) та у розрідженому(справа) за допомогою функції *clock()* (див. рис.2).

|  |  |  |
| --- | --- | --- |
|  | **>** |  |

Рисунок 2 – Порівняння часу

Як бачимо з (рис.2) у розрідженому масиві час доступу до елементів менший.

**Висновок:** на цій лабораторній роботі ми придбали і закріпили навички програмування розміщення в пам’яті специфічних масивів.