|  |
| --- |
| Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №2  по курсу «Л и ОА в ИЗ»  на тему «Оценка времени выполнения программ» |
|  |
|  |
| Выполнил:  студент группы 20ВВ4  Кривцов Н.А.  Слабинский Н.А.  Приняли:  Юрова О.В.  Акифьев И.В. |
| Пенза 2021 |

**Цель работы**: Оценить время работы программ.

**Задание 1:**

1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

**Задание 2**:

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

**Теоретическая часть:** Для оценки времени выполнения программ языка Си или их частей могут использоваться средства, предоставляемые библиотекой **time.h**. Данная библиотека содержит описания типов и прототипы функций для работы с датой и временем.

**Типы данных:**

1. clock\_t - возвращается функцией clock(). Обычно определён как int или long int.

2. time\_t - возвращается функцией time(). Обычно определён как int или long int.

3. struct tm - нелинейное, дискретное календарное представление времени.

**Задание 1:**

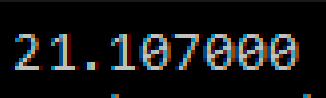
1. O(n3)
2. 100 элементов: 

200 элементов: 

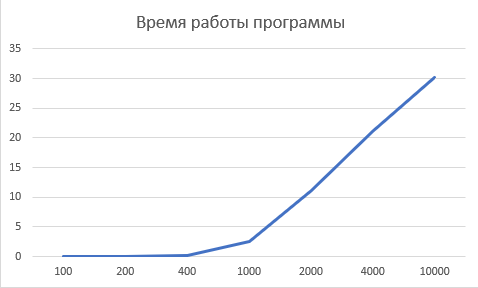
400 элементов: 

1000 элементов: 

2000 элементов: 

4000 элементов: 

10000 элементов: 



**Листинг задание 1:**

#include "stdio.h";

#include "stdlib.h";

#include "time.h";

unsigned short int a[1000][1000], b[1000][1000], c[1000][1000];

int main(void)

{

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

int i = 0, j = 0, r;

int schet = 1000;

srand(time(NULL));

short int start\_time = clock();

int elem\_c;

while (i < schet)

{

while (j < schet)

{

a[i][j] = rand() % 100 + 1

j++;

}

i++;

}

srand(time(NULL));

i = 0; j = 0;

while (i < schet)

{

while (j < schet)

{

b[i][j] = rand() % 100 + 1;

j++;

}

i++;

}

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

{

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

{

elem\_c = 0;

for (r = 0; r < 1000; r++)

{

elem\_c = elem\_c + a[i][r] \* b[r][j];

c[i][j] = elem\_c;

}

}

}

short int end\_time = clock();

short int search\_time = end\_time - start\_time;

printf("%f", search\_time / 1000.0);

return(search\_time);

}

**Задание 2:**

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

int m[10000];

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

x = items[(left + right) / 2];

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

items[j + gap] = items[j];

items[j + gap] = x;

}

}

}

int cmp(const void\* a, const void\* b) {

{

return \*(int\*)a - \*(int\*)b;

}

}

int main()

{

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

//for (int i = 0; i < 10000; i++) {

// m[i] = rand() % 100 + 1;

//}

//for (int i = 1; i < 10000; i++) { //Возрастающая

// m[i] = m[i-1]+1;

//}

// for (int i = 0; i < 10000; i++) { //Убывающая

// m[i] = 10000-i;

// }

// for (int i = 0; i < 5000; i++) { //Возрастающе-убывающая

// m[i] = m[i - 1] + 1;

// }

// for (int i = 5; i < 10000; i++) {

// m[i] = 10000 - i;

// }

int start\_time = clock();

//shell(m, 10000);

//qs(m, 0, 10000 - 1);

qsort(m, 10000, sizeof(int), cmp);

int end\_time = clock();

int search\_time = end\_time - start\_time;

printf("%d", search\_time);

return (0);

}

|  |  |  |  |
| --- | --- | --- | --- |
|  | Сортировка Шелла | Быстрая сортировка | Ф-я быстрой сортировки |
| Случайны набор | 0,13с | 0,012с | 0,14с |
| Возрастающая | 0,002с | 0,001c | 0,004c |
| Убывающая | 0,20с | 0,015c | 0,04c |
| Убываюшая-возрастающая | 0,20с | 0,011c | 0,10c |

**Вывод:**

**Задание 1:** Получив данные о скорости работы программы и показав их графически, мы убедились, что наше заключение о сложности программы было верно. O(n3)

**Задание 2:** На практике выяснили скорость обработки массивов различными способами. Сделав несколько тестов, мы можем выделить Быструю сортировку, как самый эффективный способ сортировки.