Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

по лабораторной работе №2

по курсу «Л и ОА в ИС»

на тему «Оценка времени выполнения программ»

Выполнили:

студенты группы 20ВВ2:

Тумасов Вадим

Портнов Никита

Сурков Максим

Приняли:

д.т.н Митрохин М.А.

д.т.н Юрова О.В.

Пенза 2021

Цель работы: Создать программу и на её примере оценить время выполнения программы для разного набора данных.

Лабораторные задания:

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.

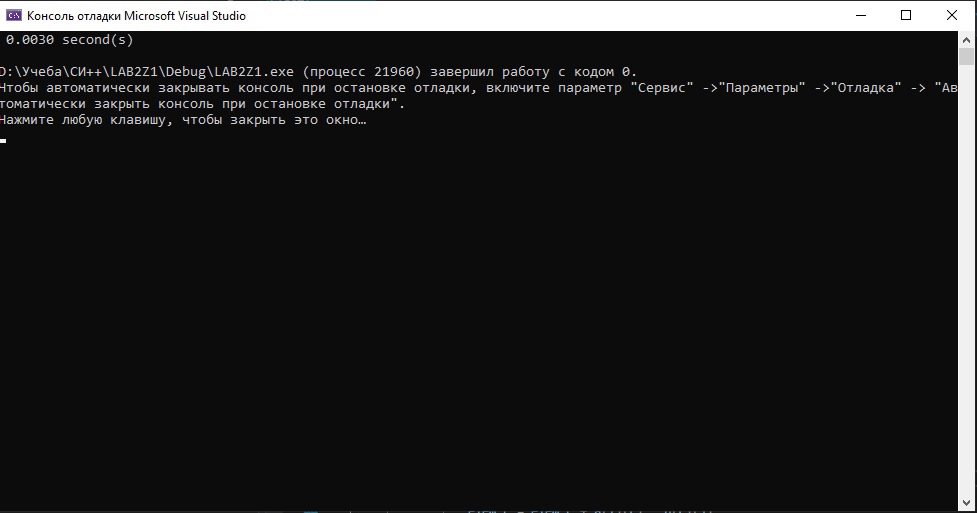
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

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.

Листинг :

[#include](https://vk.com/im?sel=2000000130&st=%23include) <stdio.h> // задание 1  
[#include](https://vk.com/im?sel=2000000130&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=2000000130&st=%23include) <conio.h>  
[#include](https://vk.com/im?sel=2000000130&st=%23include) <time.h>  
  
[#pragma](https://vk.com/im?sel=2000000130&st=%23pragma) comment(linker, "/STACK:19999721")  
  
voidmain()  
{  
setvbuf(stdin, NULL, \_IONBF, 0);  
setvbuf(stdout, NULL, \_IONBF, 0);  
clock\_tstart, end; // объявляем переменные для определения времени выполнения  
int i = 0, j = 0, r;  
int a[1000][1000], b[1000][1000], c[1000][1000], elem\_c;  
start = clock();  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
while (i < 1000)  
{  
while (j < 1000)  
{  
a[i][j] = rand() % 100 + 1; // заполняем массив случайными числами  
j++;  
}  
i++;  
}  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
i = 0; j = 0;  
while (i < 1000)  
{  
while (j < 1000)  
{  
b[i][j] = rand() % 100 + 1; // заполняем массив случайными числами  
j++;  
}  
i++;  
}  
for (i = 0; i < 1000; i++)  
{  
for (j = 0; j < 1000; 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;  
}  
}  
}  
end = clock();  
printf(" %.4f second(s)\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));  
  
  
}

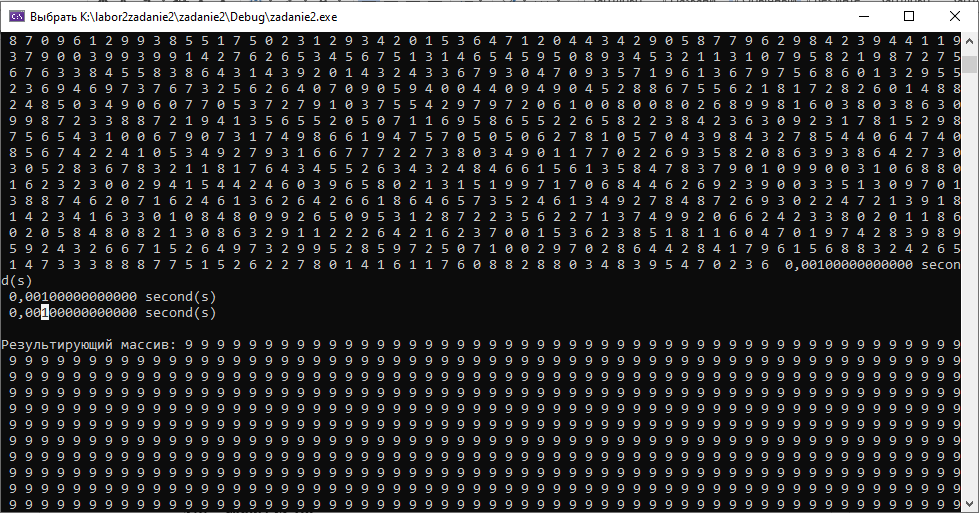
Результат работы программы задание 1

Зависимость времени выполнения от объёма данных:

|  |  |
| --- | --- |
| Объём данных | Время(сек) |
| 100 | 0,004 |
| 200 | 0,023 |
| 1000 | 3,32 |
| 2000 | 34,393 |
| 4000 | 352,517 |
| 10000 | 4 230,204 |
|  |  |
|  |  |

### [#include](https://vk.com/im?sel=160893546&st=%23include) <stdio.h> // задание 2 [#include](https://vk.com/im?sel=160893546&st=%23include) <malloc.h> [#include](https://vk.com/im?sel=160893546&st=%23include) <conio.h> [#include](https://vk.com/im?sel=160893546&st=%23include) <ctime> [#include](https://vk.com/im?sel=160893546&st=%23include) <stdlib.h> [#include](https://vk.com/im?sel=160893546&st=%23include) <iostream> using namespace std; intcompareubivan(const void\* x1, const void\* x2) { return (\*(int\*)x2 - \*(int\*)x1); } intcomparevozr(const void\* x1, const void\* x2) { return (\*(int\*)x1 - \*(int\*)x2); } void qsubiv(int\* items, int left, int right) { clock\_t start, end; start = clock(); inti, j; int x, y; i = left; j = right; /\* выборкомпаранда \*/ 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) qsubiv(items, left, j); if (i<right) qsubiv(items, i, right); } void qsvozr(int\* items, int left, int right) { clock\_t start, end; start = clock(); inti, j; int x, y; i = left; j = right; /\* выборкомпаранда \*/ 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) qsvozr(items, left, j); if (i< right) qsvozr(items, i, right); } void main() { clock\_t start, end; intn,left,right; setlocale(LC\_ALL, "Rus"); cout « "Введитеколичествоэлементовмассива" « endl; cin » n ; start = clock(); int\* A = new int[n]; srand(time(NULL)); cout « "Исходныймассив: "; for (inti = 0; i< n; i++) { A[i] = rand() % 10; cout « A[i] « " "; } left = 0; right = n - 1; qsubiv(A, left, right); //qsort(A, n/2, sizeof(int), compareubivan); //left = n/2; right = n - 1; //qsvozr(A, left, right); /\*if (n % 2) { qsort(A + (n / 2), n / 2 + 1, sizeof(int), comparevozr); } else { qsort(A + (n / 2), n / 2, sizeof(int), comparevozr); } \*/ cout « endl « "Результирующиймассив: "; for (inti = 0; i< n; i++) cout « A[i] « " "; delete[]A; end = clock(); printf(" %.4f second(s)\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC)); system("pause»void"); }

### Результат работы программызадание 2(2) (сортировка “по убыванию”)



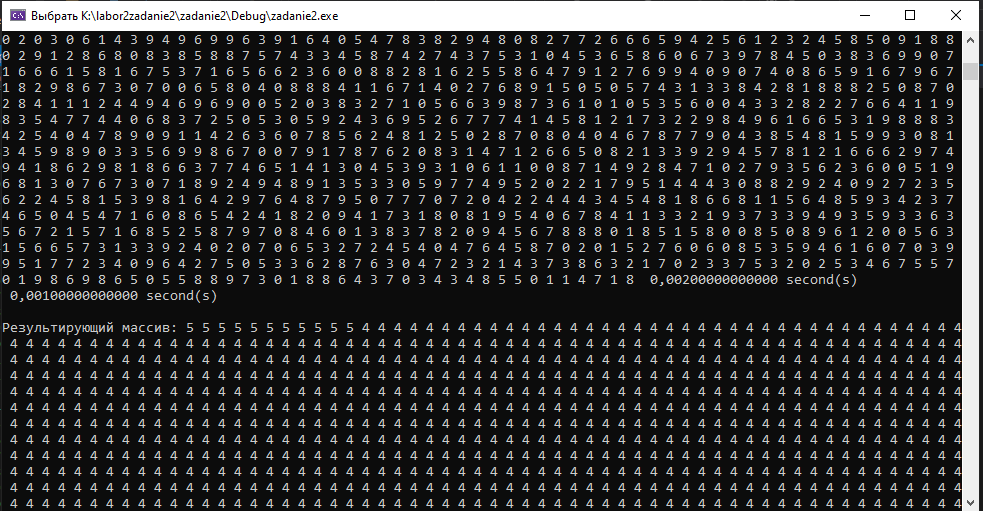
|  |  |  |
| --- | --- | --- |
| qsort | 10000 | 0,001 |
| Быстрая сортировка | 10000 | 0,001 |
| shell | 10000 | 0,001 |
| qsort | 40000 | 0,003 |
| Быстрая сортировка | 40000 | 0,003 |
| shell | 40000 | 0,006 |

### Результат работы программызадание 2(3)(сортировка “по возрастанию”)

### 

|  |  |  |
| --- | --- | --- |
| qsort | 10000 | 0,001 |
| Быстрая Сортировка | 10000 | 0,001 |
| shell | 10000 | 0,001 |
| qsort | 40000 | 0,004 |
| Быстрая Сортировка | 40000 | 0,001 |
| shell | 40000 | 0,004 |

### Результат работы программы задание 2(4)(сортировка “по убыванию”до половины и “по возрастанию до конца”)



|  |  |  |
| --- | --- | --- |
| qsort | 10000 | 0,002 |
| Быстрая Сортировка | 10000 | 0,002 |
| shell | 10000 | 0,001 |
| qsort | 40000 | 0,01 |
| Быстрая Сортировка | 40000 | 0,009 |
| shell | 40000 | 0,004 |

### Вывод:

### В ходе выполнения задания 1 мы вычислили сложность программы равную “n^3” и подтвердили ее полученным графиком,созданного с помощью таблицы на странице 4.В задании 2 нам требовалось проанализировать каждый алгоритм сортировки на его быстродействие и выявить самый быстрый.Исходя из полученных таблиц (страницы 8,9,10) мы сделали вывод-самым быстрым алгоритмом является “Быстрая сортировка”.Но рекомендовать использовать только этот алгоритм мы не можем,так как в ходе проведения анологичных исследований на другом ПК, результаты изменились и самым быстрым алгоритмом стал “Сортировка Шелла”.