**Министерство науки и высшего образования   
Пензенский государственный университет  
Кафедра “Вычислительная техника”  
  
  
  
  
  
Отчёт**по лабораторной работе №2  
по курсу “Логика и основы алгоритмизации в ИЗ”   
тема:  Оценка времени выполнения программ

Выполнили студенты группы 22ВВС1:  
 Краснорылов М.А.  
 Сайганов Д.В.  
 Приняли: Акифьев И.В.  
 Юрова О.В.

Пенза 2023

**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.

**Ход работы:**

1. Сложность программы –O(N^3)

2. Время выполнения программы матриц размерами 100



Время выполнения программы матриц размерами 200



Время выполнения программы матриц размерами 400



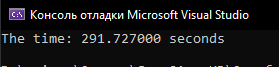
Время выполнения программы матриц размерами 1000



Время выполнения программы матриц размерами 2000



Время выполнения программы матриц размерами 4000



Время выполнения программы матриц размерами 10000



3. График зависимости времени от размера матрицы



**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 <stdio.h>

#include <malloc.h>

#include <conio.h>

#include <locale.h>

#include <time.h>

#include <stdlib.h>

#include <cstdlib>

int compare(const void\* x1, const void\* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}

void qs(int items[], int left, int right) //вызов функции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

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);

}

//сортировка методом Шелла

void ShellSort(int n, int mass[])

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++)

{

tmp = mass[i];

for (j = i; j >= step; j -= step)

{

if (tmp < mass[j - step])

mass[j] = mass[j - step];

else

break;

}

mass[j] = tmp;

}

}

int main()

{

clock\_t start, end;

int maxcount = 20000000;

setlocale(LC\_ALL, "Rus");

//ввод N

int N;

printf("Введите количество элементов массива:");

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

//выделение памяти под массив

int\* mass;

int\* mass1;

mass = (int\*)malloc(N \* sizeof(int));

mass1 = (int\*)malloc(N \* sizeof(int));

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

{

mass[i] = rand() % 100;

}

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

{

mass1[i] = mass[i];

}

//qsort(mass, N, sizeof(int), compare);

//сортировка методом Шелла

start = clock();

ShellSort(N, mass);

end = clock();

double diff = (end - start);

printf("\n |Рандом |Возрастающая|Убывающая | | Домик |\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\nШелла| %lf |", diff);

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount + 5;

}

maxcount = 20000000;

start = clock();

ShellSort(N, mass);

end = clock();

diff = (end - start);

printf("% lf|", diff);

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount - 2;

}

maxcount = 20000000;

start = clock();

ShellSort(N, mass);

end = clock();

diff = (end - start);

printf("% lf |", diff);

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

{

mass[i] = mass1[i];

}

for (int i = 0; i < N/2; i++)

{

mass[i] = maxcount + 2;

}

maxcount = 20000000;

for (int i = N/2; i < N; i++)

{

mass[i] = maxcount - 2;

}

maxcount = 20000000;

start = clock();

ShellSort(N, mass);

end = clock();

diff = (end - start);

printf("% lf|", diff);

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = rand() % 100;

}

start = clock();

qs(mass, 0, N - 1);

end = clock();

diff = (end - start);

printf("\nQS |");

printf(" %lf |", diff);

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount + 5;

}

start = clock();

qs(mass, 0, N - 1);

end = clock();

diff = (end - start);

printf("% lf |", diff);

maxcount = 20000000;

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount + 2;

}

start = clock();

qs(mass, 0, N - 1);

end = clock();

diff = (end - start);

printf("% lf |", diff);

maxcount = 20000000;

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

{

mass[i] = mass1[i];

}

for (int i = 0; i < N/2; i++)

{

mass[i] = maxcount + 5;

}

for (int i = N/2; i < N; i++)

{

mass[i] = maxcount - 2;

}

start = clock();

qs(mass, 0, N - 1);

end = clock();

diff = (end - start);

printf("% lf |", diff);

///////////////////////

//QSORT

maxcount = 20000000;

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = rand() % 100;

}

printf("\nQsort|");

start = clock();

qsort(mass, N, sizeof(int), compare);

end = clock();

diff = (end - start);

printf("% lf |", diff);

////////////

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount + 5;

}

start = clock();

qsort(mass, N, sizeof(int), compare);

end = clock();

diff = (end - start);

printf("% lf |", diff);

////////////////

maxcount = 20000000;

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

{

mass[i] = mass1[i];

}

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

{

mass[i] = maxcount - 2;

}

start = clock();

qsort(mass, N, sizeof(int), compare);

end = clock();

diff = (end - start);

printf("% lf |", diff);

///////////////////////

maxcount = 20000000;

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

{

mass[i] = mass1[i];

}

for (int i = 0; i < N/2; i++)

{

mass[i] = maxcount + 5;

}

maxcount = 20000000;

for (int i = N/2; i < N ; i++)

{

mass[i] = maxcount - 2;

}

start = clock();

qsort(mass, N, sizeof(int), compare);

end = clock();

diff = (end - start);

printf("% lf |", diff);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

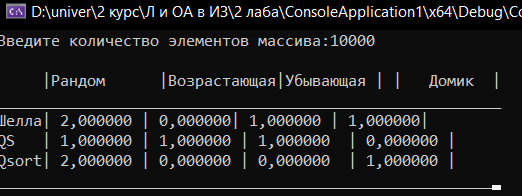
free(mass);

\_getch();

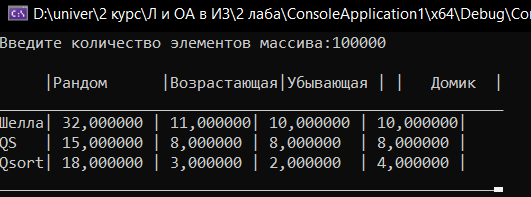
return 0;

}

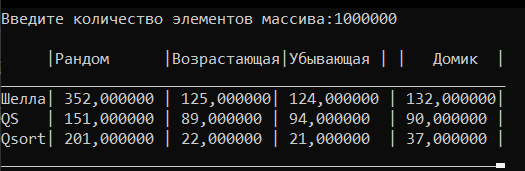
Результат работы при массиве размером 10000.



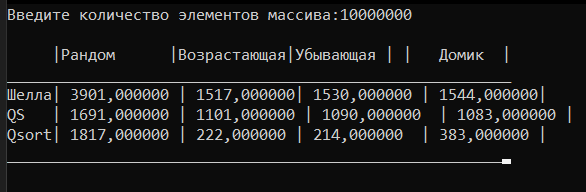
Результат работы при массиве размером 100000.



Результат работы при массиве размером 1000000.



Результат работы при массиве размером 10000000.



Вывод: в ходе выполнения лабораторной работы были разработаны

программы, реализующие работу с разными видами сортировок. При анализе полученных результатов делаем вывод, что сортировка Шелла уступает при любых размерах массива и методах его заполнения. Наиболее производительной и оптимальной является сортировка Qsort, она выдает наилучшее время везде, кроме рандомного заполнения массива, в этом случае производительнее оказывается сортировка QS.