Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2021

**ОТЧЁТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

Выполнили:

студентки группы 20ВВ4

Кожевникова А. В.

Мамелина Ю. В.

Приняла:

д.т.н Юрова О. В.

Пенза 2021

**Цель работы:** изучить и научиться применять средства, предоставляемые библиотекой time.h.

**Задание 1:**

1. Вычислили порядок сложности программы (О-символику).

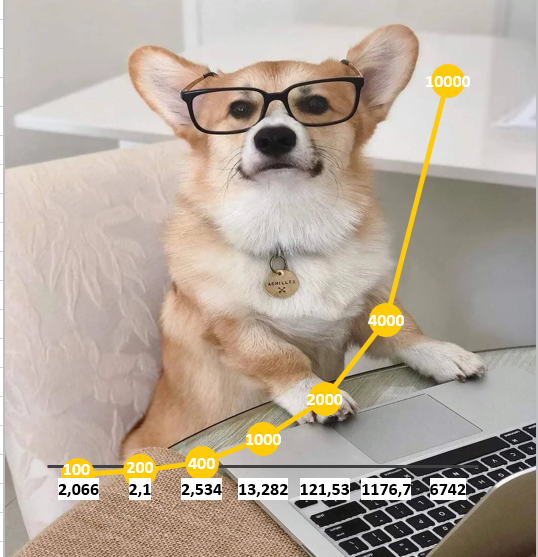
О(n3).

1. 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество элементов | 100 | 200 | 400 | 1000 | 2000 | 4000 | 10000 |
| Время | 2,066 | 2,1 | 2,534 | 13,282 | 121,53 | 1176,7 | 6742 |

3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

**График зависимости времени от количества элементов массива:**



**Листинг:**  
[#include](https://vk.com/im?q=%23include) <stdlib.h>  
[#include](https://vk.com/im?q=%23include) <ctime>  
[#include](https://vk.com/im?q=%23include) <iostream>  
using namespace std;  
int main(){  
setlocale(LC\_ALL, "rus");  
clock\_t start, end; // объявляем переменные для определения времени выполнения  
int i = 0, j = 0, r, n = 0;  
cout << "Введите n:" << endl;  
cin >> n;  
int\*\* a = new int\*[n];  
for (int i = 0; i < n; ++i) {  
a[i] = new int[n];  
}  
int\*\* b = new int\* [n];  
for (int i = 0; i < n; ++i) {  
b[i] = new int[n];  
}  
int\*\* c = new int\* [n];  
for (int i = 0; i < n; ++i) {  
c[i] = new int[n];  
}  
int elem\_c;  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
while (i < n)  
{  
while (j < n)  
{  
a[i][j] = rand() % 100; // заполняем массив случайными числами  
j++;  
}  
i++;  
}  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
i = 0; j = 0;  
while (i < n)  
{  
while (j < n)  
{  
b[i][j] = rand() % 100; // заполняем массив случайными числами  
j++;  
}  
i++;  
}  
for (i = 0; i < n; i++)  
{  
for (j = 0; j < n; j++)  
{  
elem\_c = 0;  
for (r = 0; r < n; r++)  
{  
elem\_c = elem\_c + a[i][r] \* b[r][j];  
c[i][j] = elem\_c;  
}  
}  
}  
for (int i = 0; i < n; ++i) {  
delete[] a[i];  
}  
delete [] a;  
for (int i = 0; i < n; ++i) {  
delete[] b[i];  
}  
delete[] b;  
for (int i = 0; i < n; ++i) {  
delete[] c[i];  
}  
delete[] c;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
return 0;  
}

**Задание 2:**

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборезначений массива.

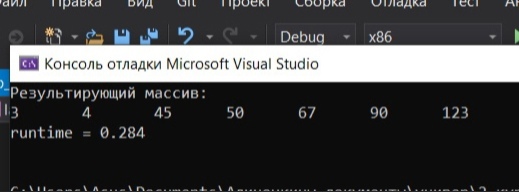


Рисунок 1 – время работы алгоритма Шелла.

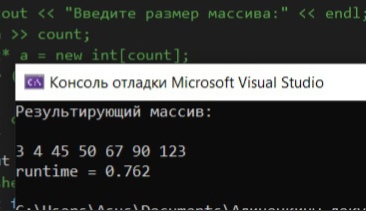


Рисунок 2 – время работы алгоритма быстрой сортировки.

Листинг:

[#include](https://vk.com/im?sel=558284884&st=%23include) <iostream>  
using namespace std;  
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;  
}  
}  
for (int i = 0; i < count; i++)  
{  
cout << items[i] << "\t";  
}  
}  
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);  
  
}  
int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
int a[] = { 45,67,3,123,4,50,90 };  
cout << "Результирующий массив:" << endl;  
shell(a, count);  
int first = 0, last = count - 1;  
qs(a, first, last);  
cout << endl;  
for (int i = 0; i < count; i++) cout << a[i] << " ";  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
return 0;  
}

2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.

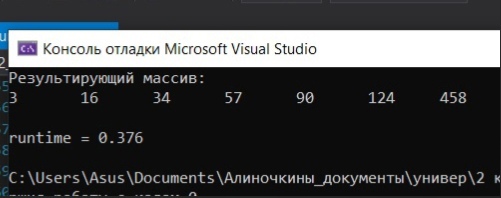


Рисунок 3 – время работы алгоритма Шелла на примере массива с возрастающей последовательностью.

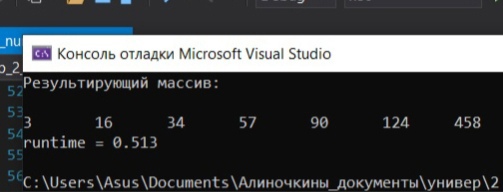


Рисунок 4 – время работы алгоритма быстрой сортировки на примере массива с возрастающей последовательностью.

Листинг:

int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
int a[] = { 3,16,34,57,90,124,458};  
cout << "Результирующий массив:" << endl;  
shell(a, count);  
int first = 0, last = count - 1;  
qs(a, first, last);  
cout << endl;  
for (int i = 0; i < count; i++) cout << a[i]«"\t";  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
return 0;  
}

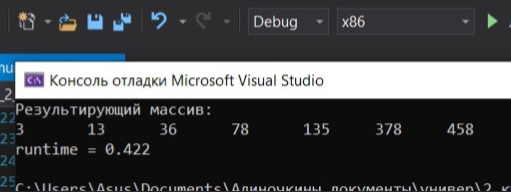
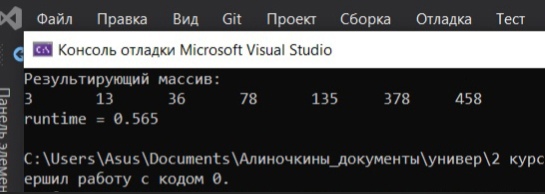
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

Рисунок 5 – время работы алгоритма Шелла на примере массива с убывающей последовательностью.

  
Рисунок 6 – время работы алгоритма быстрой сортировки на примере массива с убывающей последовательностью.

Листинг:

int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
int a[] = { 458, 378,135,78,36,13,3};  
cout << "Результирующий массив:" << endl;  
shell(a, count);  
int first = 0, last = count - 1;  
qs(a, first, last);  
for (int i = 0; i < count; i++) cout << a[i]«"\t";  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
return 0;  
}

4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

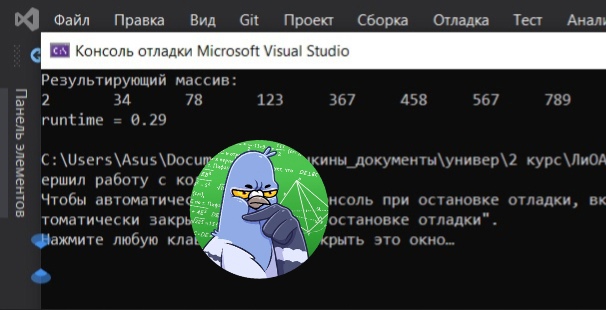


Рисунок 7 – время работы алгоритма Шелла на примере массива одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Рисунок 7.пояснение.ехе. – мы вставили голубя, чтобы собачке корги не было так одиноко в отчёте.

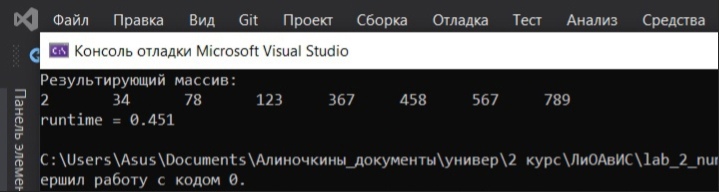


Рисунок 8 – время работы алгоритма быстрой сортировки на примере массива одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Листинг:

int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 8;  
int a[] = { 78,123,367,567,789,458,34,2 };  
  
cout << "Результирующий массив:" << endl;  
shell(a, count);  
int first = 0, last = count - 1;  
qs(a, first, last);  
for (int i = 0; i < count; i++) cout << a[i] <<"\t";  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
  
return 0;  
}

5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

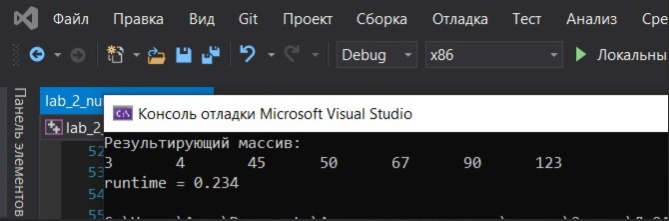


Рисунок 9 – время работы стандартной функции qsort на случайном наборезначений массива.

Листинг:

int a[] = { 45,67,3,123,4,50,90 };  
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  
}  
int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
  
cout << "Результирующий массив:" << endl;  
int first = 0, last = count - 1;  
qsort(a, count,sizeof(int),compare);  
for (int i = 0; i < count; i++)  
{  
cout«a[i]<<"\t";  
}  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
  
return 0;  
}

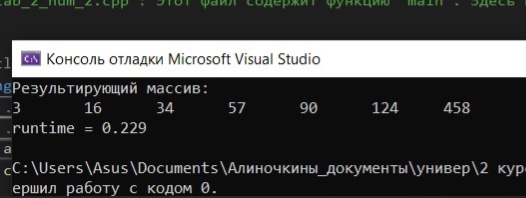


Рисунок 10 – время работы стандартной функции qsort на примере массива с возрастающей последовательностью.

Листинг:

int a[] = {  3,16,34,57,90,124,458 };  
int compare(const void\* x1, const void\* x2) // функция сравнения элементов массива  
{  
return (\*(int\*)x1 - \*(int\*)x2);  
}  
int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
  
cout << "Результирующий массив:" << endl;  
int first = 0, last = count - 1;  
qsort(a, count,sizeof(int),compare);  
for (int i = 0; i < count; i++)  
{  
cout«a[i]<<"\t";  
}  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
  
return 0;  
}

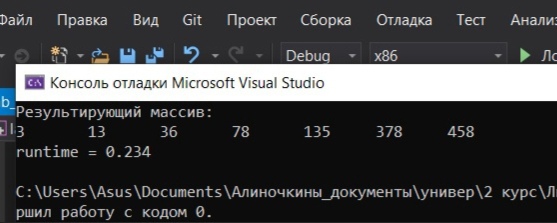


Рисунок 11 – время работы стандартной функции qsort на примере массива представляющем собой убывающую последовательность чисел.

Листинг:

int a[] = { 458, 378,135,78,36,13,3 };  
int compare(const void\* x1, const void\* x2) // функция сравнения элементов массива  
{  
return (\*(int\*)x1 - \*(int\*)x2);  
}  
int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 7;  
  
cout << "Результирующий массив:" << endl;  
int first = 0, last = count - 1;  
qsort(a, count,sizeof(int),compare);  
for (int i = 0; i < count; i++)  
{  
cout<<a[i]<<"\t";  
}  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
  
return 0;  
}

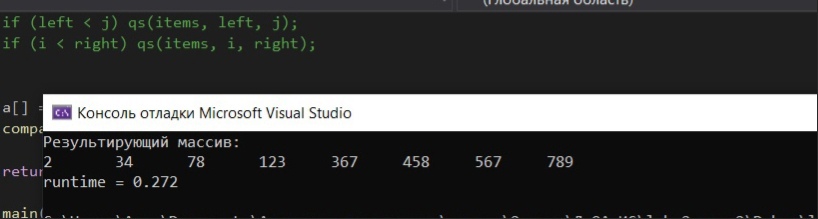


Рисунок 12 – время работы стандартной функции qsort на примере массива одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Листинг:

int a[] = { 78,123,367,567,789,458,34,2 };  
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  
}  
int main()  
{  
setlocale(LC\_ALL, "rus");  
srand(time(NULL)); // инициализируем параметры генератора случайных чисел  
int count = 8;  
  
cout << "Результирующий массив:" << endl;  
int first = 0, last = count - 1;  
qsort(a, count,sizeof(int),compare);  
for (int i = 0; i < count; i++)  
{  
cout << a[i]<<"\t";  
}  
cout << endl;  
cout << "runtime = " << clock() / 1000.0 << endl; // время работы программы  
  
return 0;  
}

Приложение к заданию 2:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Время | | |
| Алгоритм | Возрастающая последовательность | Убывающая последовательность | Первая половина – возрастает, вторая – убывает |
| Шелла | 0,376 | 0,422 | 0,29 |
| Быстрой сортировки | 0,513 | 0,565 | 0,451 |
| qsort | 0,234 | 0,229 | 0,272 |

**Вывод:** научились определять сложность алгоритма, изучили зависимость времени от количества элементов в массиве с помощью графика.

Изучили средства работы библиотеки time.h, научились использовать её в профессиональной деятельности. Пронаблюдали (в ходе сравнения), что алгоритм быстрой сортировки показал себя наименее эффективным в быстроте выполнения процессов. Самым же быстрым в выполнении себя проявила стандартная функция qsort.