Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пензенский Государственный университет»

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №2

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Оценка времени выполнения программ»

Выполнили: ст. гр. 21ВВ1

Сурков А.А

Хорошильцев Е.А

Кривенков И.В

Приняли: Юрова О.В.

Акифьев И.В.

2022

Задание 1:

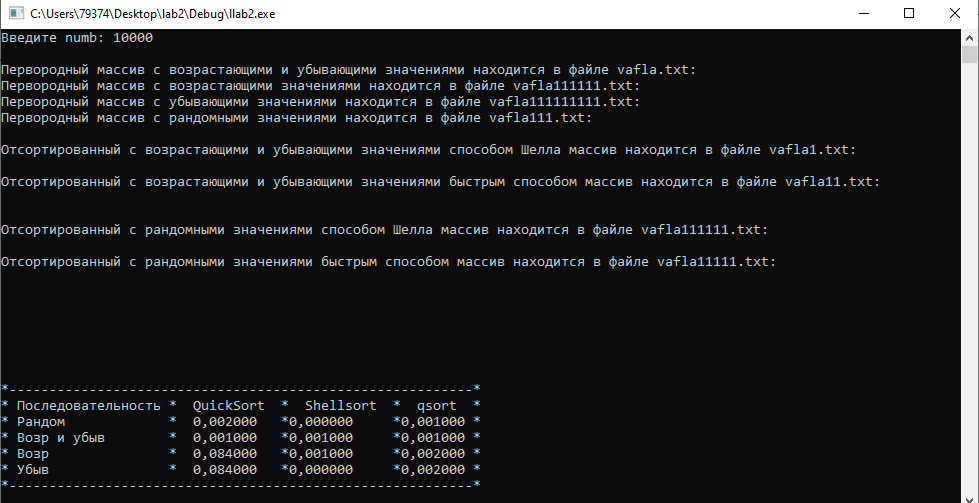
1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от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

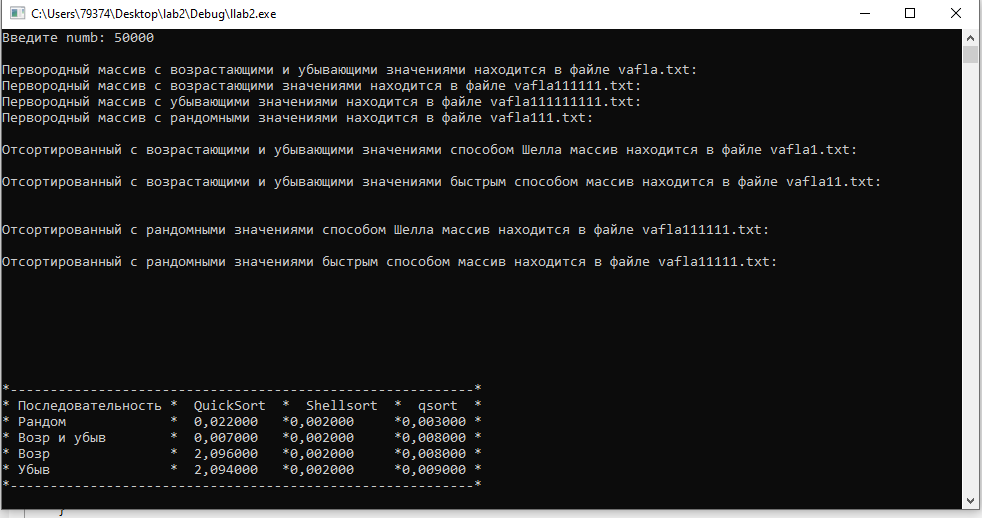
Сложность алгоритма -

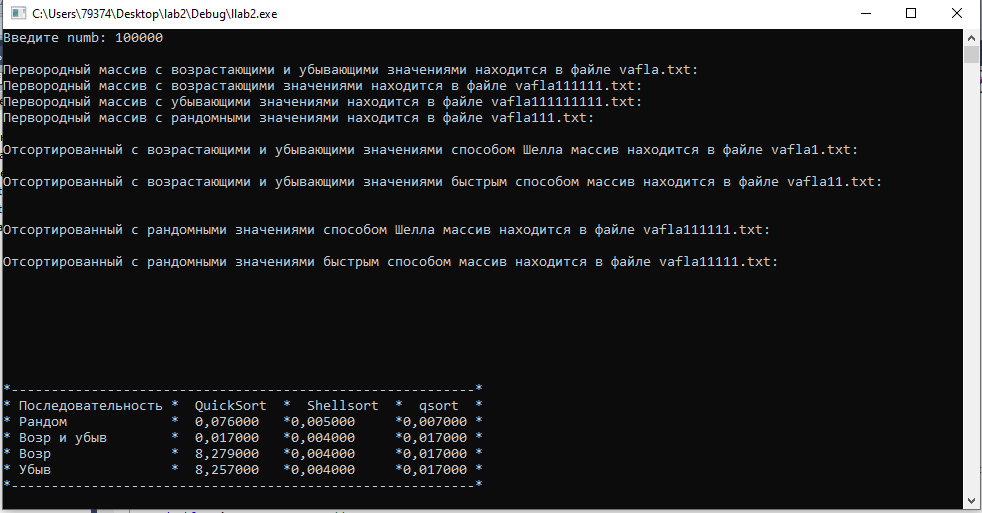
|  |  |  |
| --- | --- | --- |
| № теста | Размер массива | Время выполнения алгоритма в секундах |
| 1 | 100 | 0,00300 |
| 2 | 200 | 0,001600 |
| 3 | 400 | 0,22300 |
| 4 | 1000 | 6,537 |
| 5 | 2000 | 67,611 |
| 6 | 4000 | 1427,623 |
| 7 | 10000 | - |

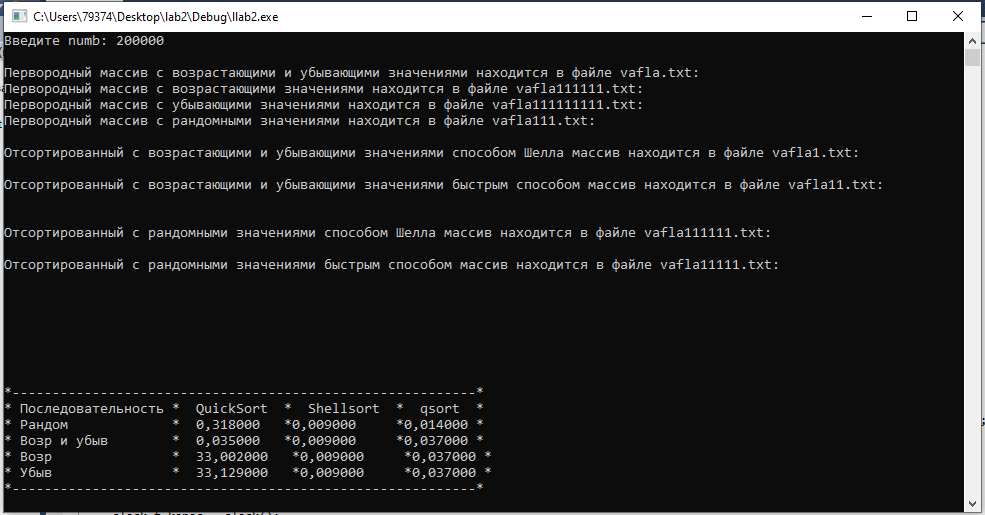
**Задание 2:**

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.









Вывод:

Выполнив данную лабораторную, работу мы укрепили свои знания о том, как правильно измерять время выполнения определенных алгоритмов.

Листинг программы:

Задание 1

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#define N 10000

int main(void)

{

setlocale(LC\_ALL, "Rus");

int i, j, k, elem\_c;

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

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

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

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

A[i] = (int\*)malloc(N \* sizeof(int));

B[i] = (int\*)malloc(N \* sizeof(int));

C[i] = (int\*)malloc(N \* sizeof(int));

}

srand(time(NULL));

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

for (j = 0; j < N; j++){

A[i][j] = rand() % 100;

B[i][j] = rand() % 100;

}

double time\_spent = 0.0; // для хранения времени выполнения кода

clock\_t begin = clock();

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

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

{

C[i][j] = 0;

for(k = 0; k < N; k++)

C[i][j] += A[i][k] \* B[k][j];

}

clock\_t end = clock();

time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

printf("Затраченно время в секундах %f ", time\_spent);

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

free(A[i]);

free(B[i]);

free(C[i]);

}

free(A);

free(B);

free(C);

}

Задание 2

#define \_CRT\_SECURE\_NO\_WARNINGS // для корректной работы scanf()

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <stdlib.h>

#include <windows.h>

#include <time.h>

#pragma comment(linker, "/STACK:1468006400")

void qs(int С[], int left, int right)

{

int pivot; // разрешающий элемент

int l\_hold = left; //левая граница

int r\_hold = right; // правая граница

pivot = С[left];

while (left < right) // пока границы не сомкнутся

{

while ((С[right] >= pivot) && (left < right))

right--; // сдвигаем правую границу пока элемент [right] больше [pivot]

if (left != right) // если границы не сомкнулись

{

С[left] = С[right]; // перемещаем элемент [right] на место разрешающего

left++; // сдвигаем левую границу вправо

}

while ((С[left] <= pivot) && (left < right))

left++; // сдвигаем левую границу пока элемент [left] меньше [pivot]

if (left != right) // если границы не сомкнулись

{

С[right] = С[left]; // перемещаем элемент [left] на место [right]

right--; // сдвигаем правую границу влево

}

}

С[left] = pivot; // ставим разрешающий элемент на место

pivot = left;

left = l\_hold;

right = r\_hold;

if (left < pivot) // Рекурсивно вызываем сортировку для левой и правой части массива

qs(С, left, pivot - 1);

if (right > pivot)

qs(С, pivot + 1, right);

}

// Функция сортировки Шелла

void ShellSort(int n, int С[])

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2) //пока шаг больше нуля будет выполняться код ниже

for (i = step; i < n; i++) //текущее значение шага записывается в i, пока i< значений массива будет выполняться код ниже

{

tmp = С[i]; //в tmp записывается первое значение массива

//перебираем элементы, отстоящие от j

for (j = i; j >= step; j=j-step) // текущее значение i записывается в j, пока j больше или равно step выполняется код ниже

{

if (tmp < С[j - step]) //Если первый элемент массива меньше, чем отстоящий, то выполняется код ниже

С[j] = С[j - step];//переходим к следующему отстающему элементу

else

break;

}

С[j] = tmp;//сюда помещаем сохраненный элемент

}

}

int compare (const void \*a, const void \*b)

{

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

}

int main()

{

srand(time(NULL)); //функция для генерации рандомных чисел

setlocale(LC\_ALL, "Rus");//русская локализация

FILE \*F = fopen ( "vafla.txt", "wt" );//открытие файла для записи (A,B)

FILE \*Ff = fopen ( "vafla1.txt", "wt" );//открытие файла для записи (Shellsort C)

FILE \*Fff = fopen ( "vafla11.txt", "wt" );//открытие файла для записи (QuickSort C)

FILE \*Ffff = fopen ( "vafla111.txt", "wt" );//открытие файла для записи (Random(D))

FILE \*Fffff = fopen ( "vafla1111.txt", "wt" );//открытие файла для записи (A,B = C)

FILE \*Ffffff = fopen ( "vafla11111.txt", "wt" );//открытие файла для записи (QuickSort D)

FILE \*Fffffff = fopen ( "vafla111111.txt", "wt" );//открытие файла для записи (Shellsort D)

FILE \*Ffffffff = fopen ( "vafla1111111.txt", "wt" );//открытие файла для записи (G(ubiv))

FILE \*Fffffffff = fopen ( "vafla11111111.txt", "wt" );//открытие файла для записи (H(vozr))

FILE \*Ffffffffff = fopen ( "vafla111111111.txt", "wt" );//открытие файла для записи (QuickSort H)

FILE \*Fffffffffff = fopen ( "vafla1111111111.txt", "wt" );//открытие файла для записи (Shellsort H)

FILE \*Fa = fopen ( "vafla3.txt", "wt" );//открытие файла для записи (QuickSort G)

FILE \*Faa = fopen ( "vafla33.txt", "wt" );//открытие файла для записи (ShellSort G)

FILE \*Ft = fopen ( "vafla2.txt", "wt" );//открытие файла для записи (qsort C)

FILE \*Ftt = fopen ( "vafla22.txt", "wt" );//открытие файла для записи (qsort D)

FILE \*Fttt = fopen ( "vafla222.txt", "wt" );//открытие файла для записи (qsort G)

FILE \*Ftttt = fopen ( "vafla2222.txt", "wt" );//открытие файла для записи (qsort H)

int numb, i, j, n, b, k, d, numb1, numb2;

//ввод количества значений массива

printf("Введите numb: ");

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

printf("\n");

numb1 = numb/2;

numb2 = numb/2;

//создание массива

int\* A;

A = (int \*)malloc(numb1 \* sizeof(int));//выделение памяти под массив

int\* B;

B = (int \*)malloc(numb2 \* sizeof(int));//выделение памяти под массив

int\* C;

C = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* D;

D = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* G;

G = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* H;

H = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

fprintf(F, "Первородный с возрастающими и убывающими значениями массив:\n");//запись фразы в файл

printf("Первородный массив с возрастающими и убывающими значениями находится в файле vafla.txt:\n");

//заполнение массива возрастающими и убывающими значениями

for (int i = 0; i <= numb1; i++) {

for(i = numb1; i != 0; i--) {

A[i] = i - 1;

fprintf(F, "%d\t", A[i]);

}

for(i = 0; i != numb2; i++) {

B[i] = i + 1;

fprintf(F, "%d\t", B[i]);

}

}

k = -1; i = 0; j = 0;

for(int i = 0; i <= numb1; i++) {

C[k++] = A[i];

}

for(int i = 0; i <= numb2 ; i++) {

C[k++] = B[i];

}

for(int i = 0; i <= numb1+numb2-1; i++) {

fprintf(Fffff, "%d\t", C[i]);

}

fprintf(Fffffffff, "Первородный с возрастающими значениями массив:\n");//запись фразы в файл

printf("Первородный массив с возрастающими значениями находится в файле vafla111111.txt:\n");

fprintf(Ffffffff, "Первородный с убывающими значениями массив:\n");//запись фразы в файл

printf("Первородный массив с убывающими значениями находится в файле vafla111111111.txt:\n");

for(i = numb; i != -1; i--) {

G[i] = i - 1;

fprintf(Ffffffff, "%d\t", G[i]);

}

for(i = 0; i != numb; i++) {

H[i] = i + 1;

fprintf(Fffffffff, "%d\t", H[i]);

}

fprintf(Ffff, "Первородный с рандомными значениями массив:\n");//запись фразы в файл

printf("Первородный массив с рандомными значениями находится в файле vafla111.txt:\n");

srand(time(NULL));

for (i = 0; i < numb; i++){

D[i] = rand() % 100;

fprintf(Ffff, "%d\t", D[i]);

}

printf("\nОтсортированный с возрастающими и убывающими значениями способом Шелла массив находится в файле vafla1.txt:\n");

printf("\nОтсортированный с возрастающими и убывающими значениями быстрым способом массив находится в файле vafla11.txt:\n");

fprintf(Fff, "\nОтсортированный массив быстрым способом с возрастающими и убывающими значениями :\n");//запись фразы в файл

double time\_potr = 0.0; // для хранения времени выполнения кода

clock\_t nach = clock();

qs(C, 0, numb-1);

clock\_t konec = clock();

for (int i = 0; i < numb; i++){

fprintf(Fff, "%d\t ", C[i]);

}

time\_potr += (double)(konec - nach) / CLOCKS\_PER\_SEC;

//printf("Затраченно время в секундах на quick sort %f ", time\_potr);

printf("\n");

//сортировка методом Шелла для возрастающего и убывающего

fprintf(Ff, "\nОтсортированный массив способом Шелла с возрастающими и убывающими значениями :\n");//запись фразы в файл

double time\_spent = 0.0; // для хранения времени выполнения кода

clock\_t begin = clock();

ShellSort(numb-1, C);//вызов функции метода шелла

clock\_t end = clock();

time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

//printf("Затраченно время в секундах на Shell's sort %f ", time\_spent);

//вывод отсортированного массива на экран

for (int i = 0; i < numb; i++){

fprintf(Ff, "%d\t ", C[i]);

}

printf("\nОтсортированный с рандомными значениями способом Шелла массив находится в файле vafla111111.txt:\n");

printf("\nОтсортированный с рандомными значениями быстрым способом массив находится в файле vafla11111.txt:\n");

printf("\n");

fprintf(Ffffff, "Отсортированный с рандомными значениями быстрым способом массив:\n");//запись фразы в файл

double time\_p = 0.0; // для хранения времени выполнения кода

clock\_t na = clock();

qs(D, 0, numb-1);

clock\_t ko = clock();

for (int i = 0; i < numb; i++){

fprintf(Ffffff, "%d\t ", D[i]);

}

time\_p += (double)(ko- na) / CLOCKS\_PER\_SEC;

//printf("Затраченно время в секундах на quick sort %f ", time\_p);

printf("\n");

//сортировка методом Шелла для рандомного массива

fprintf(Fffffff, "Отсортированный с рандомными значениями способом Шелла массив:\n");//запись фразы в файл

double time\_s = 0.0; // для хранения времени выполнения кода

clock\_t be = clock();

ShellSort(numb-1, D);//вызов функции метода шелла

clock\_t en = clock();

time\_s += (double)(en - be) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

//printf("Затраченно время в секундах на Shell's sort %f ", time\_s);

printf("\n");

//вывод отсортированного массива с возрастающей и убывающей последовательностями на экран

for (int i = 0; i < numb; i++){

fprintf(Fffffff, "%d\t ", D[i]);

}

fprintf(Fa, "Отсортированный с возр значениями быстрым способом массив:\n");//запись фразы в файл

double time\_pot = 0.0; // для хранения времени выполнения кода

clock\_t nac = clock();

qs(H, 0, numb-1);

clock\_t kon = clock();

for (int i = 0; i < numb; i++){

fprintf(Fa, "%d\t ", H[i]);

}

time\_pot += (double)(kon- nac) / CLOCKS\_PER\_SEC;

//printf("Затраченно время в секундах на quick sort %f ", time\_p);

printf("\n");

fprintf(Faa, "Отсортированный с возр значениями способом Шелла массив:\n");//запись фразы в файл

double time\_st = 0.0; // для хранения времени выполнения кода

clock\_t beg = clock();

ShellSort(numb-1, H);//вызов функции метода шелла

clock\_t eni = clock();

time\_st += (double)(eni - beg) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

printf("\n");

for (int i = 0; i < numb; i++){

fprintf(Faa, "%d\t ", H[i]);

}

fprintf(Ffffffffff, "Отсортированный с убыв значениями быстрым способом массив:\n");//запись фразы в файл

double time\_pota = 0.0; // для хранения времени выполнения кода

clock\_t naca = clock();

qs(G, 0, numb-1);

clock\_t konc = clock();

for (int i = 0; i < numb; i++){

fprintf(Ffffffffff, "%d\t ", G[i]);

}

time\_pota += (double)(konc- naca) / CLOCKS\_PER\_SEC;

printf("\n");

fprintf(Fffffffffff, "Отсортированный с убыв значениями способом Шелла массив:\n");//запись фразы в файл

double time\_sta = 0.0; // для хранения времени выполнения кода

clock\_t begi = clock();

ShellSort(numb-1, G);//вызов функции метода шелла

clock\_t enid = clock();

time\_sta += (double)(enid - begi) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

printf("\n");

for (int i = 0; i < numb; i++){

fprintf(Fffffffffff, "%d\t ", G[i]);

}

double time\_potan = 0.0; // для хранения времени выполнения кода

clock\_t nacan = clock();

qsort(C, numb, sizeof(int), compare);

clock\_t koncn = clock();

for (int i = 0; i < numb; i++){

fprintf(Ft, "%d\t ", C[i]);

}

time\_potan += (double)(koncn- nacan) / CLOCKS\_PER\_SEC;

double time\_potani = 0.0; // для хранения времени выполнения кода

clock\_t nacani = clock();

qsort(D, numb, sizeof(int),compare);

clock\_t koncni = clock();

for (int i = 0; i < numb; i++){

fprintf(Ftt, "%d\t ", D[i]);

}

time\_potani += (double)(koncni- nacani) / CLOCKS\_PER\_SEC;

double time\_potanic = 0.0; // для хранения времени выполнения кода

clock\_t nacanic = clock();

qsort(G, numb, sizeof(int),compare);

clock\_t koncnic = clock();

for (int i = 0; i < numb; i++){

fprintf(Fttt, "%d\t ", H[i]);

}

time\_potanic += (double)(koncnic- nacanic) / CLOCKS\_PER\_SEC;

double time\_potanica = 0.0; // для хранения времени выполнения кода

clock\_t nacanica = clock();

qsort(H, numb, sizeof(int),compare);

clock\_t koncnica = clock();

for (int i = 0; i < numb; i++){

fprintf(Ftttt, "%d\t ", G[i]);

}

time\_potanica += (double)(koncnica- nacanica) / CLOCKS\_PER\_SEC;

printf("\*----------------------------------------------------------\*");

printf("\n");

printf("\* Последовательность \* QuickSort \* Shellsort \* qsort \*");

printf("\n");

printf("\* Рандом \* %f",time\_p);

printf(" \*%f", time\_s);

printf(" \*%f", time\_potani);

printf(" \*");

printf("\n");

printf("\* Возр и убыв \* %f",time\_potr);

printf(" \*%f", time\_spent);

printf(" \*%f", time\_potan);

printf(" \*");

printf("\n");

printf("\* Возр \* %f",time\_pot);

printf(" \*%f", time\_st);

printf(" \*%f", time\_potanic);

printf(" \*");

printf("\n");

printf("\* Убыв \* %f",time\_pota);

printf(" \*%f", time\_sta);

printf(" \*%f", time\_potanica);

printf(" \*");

printf("\n");

printf("\*----------------------------------------------------------\*");

getchar();getchar();

}