



РГР Анализ сортировок

- Программа сортировки
- Счетчики операций – шаги, сравнения, обмены, рекурсивные вызовы
- Генераторы значений: случайный (диапазон), возрастающий, убывающий
- Генератор массивов – последовательность заданных размерностей
- Собственно эксперимент
- Перенос данных – консоль – блокнот – Excel
- Оценка зависимостей
- Анализ результатов: ожидаемая трудоемкость, реальная, оценка расхождения



РГР Анализ сортировок

```
//-----72-02.cpp
//-----"Быстрая" сортировка
void sort2(int in[], int a, int b, int &cCall, int &cStep, int &cSwap){
    cCall++;
    int i,j,mode;
    if (a>=b) return; // Размер части =0
    for (i=a, j=b, mode=1; i < j; mode >0 ? j-- : i++){
        cStep++;
        if (in[i] > in[j]){ // Перестановка концевой пары
            cSwap++;
            int c = in[i]; in[i] = in[j]; in[j]=c;
            mode = -mode; // со сменой сокращаемого конца
        }
    }
    sort2(in, a, i-1, cCall, cStep, cSwap);
    sort2(in, i+1, b, cCall, cStep, cSwap);
}
```

```
//-----72-01.cpp
//----- Сортировка рекурсивным разделением массива
// В качестве медианы - среднее арифметическое
void sort0(int in[], int a, int b, int &cCall, int &cStep, int &cSwap){
    cCall++;
    int i,j,mode;
    double sr=0;
    if (a>=b) return; // Размер части =0
    for (i=a; i<=b; i++,cStep++) sr+=in[i];
    sr=sr/(b-a+1);
    for (i=a, j=b; i <= j; cStep++){
        if (in[i]< sr) { i++; continue; } // Слева - меньше, пропустить
        if (in[j]>=sr) { j--; continue; } // Справа - больше, пропустить
        cSwap++;
        int c = in[i]; in[i] = in[j]; in[j]=c;
        i++,j--; // Поменять местами и пропустить
    }
    if (i==a) return; // все равны и в правой части
    sort0(in, a, j, cCall, cStep, cSwap);
    sort0(in, i, b, cCall, cStep, cSwap); // рекурсивный вызов для двух частей
}
```



РГР Анализ сортировок

```
int genOrdInc(int idx, int size, int par1, double par2){
    return idx;
}

int genOrdDec(int idx, int size, int par1, double par2){
    return size-idx;
}

int genOrdRandom(int idx, int size, int par1, double par2){
    int vv = rand() | rand()<<15;
    return (int) ((vv%size)*par2);
}

int sizes[]={10000,20000,50000,100000,200000,500000,1000000,2000000,5000000,0};

int *createAndFill(int size, int (*gen)(int,int,int,double), int par1, double par2){
    int *a = new int[size];
    for(int i=0;i<size;i++)
        a[i]=(*gen)(i,size,par1,par2);
    return a;
}
```

```
int testModeBefore=1;
int testModeAfter=0;

void testView(int array[],int size,int step, int enable){
    if (enable && step==0){ // Проверить на самой маленькой
        for(int j=0;j<size;j++){
            printf("%d ",array[j]);
        }
        printf("\n");
    }
}
```



РГР Анализ сортировок

```
void oneExperience(int (*gen)(int,int,int,double), int par1, double par2){
    int cCall,cStep,cSwap;
    for(int i=0;sizes[i]!=0;i++){
        int size = sizes[i];
        int *array = createAndFill(size,gen,par1,par2);
        testView(array,size,i,testModeBefore);
        long t0=clock(); // Начальное значение времени
        cCall=0;
        cSwap=0;
        cStep=0;
        sort0(array,0,size-1,cCall,cStep,cSwap);
        delete []array;
        long tt = clock()-t0;
        testView(array,size,i,testModeAfter);
        printf("%d\t%d\t%d\t%d\t%d\n",size,tt,cCall,cStep,cSwap);
    }
}

int main(){
    srand(time(NULL));
    //oneExperience(genOrdRandom,0,1);
    //oneExperience(genOrdInc,0,1);
    oneExperience(genOrdDec,0,1);
}
```

10000	1	19999	262232	5000
20000	3	39999	564464	10000
50000	10	99999	1543928	25000
100000	19	199999	3287856	50000
200000	29	399999	6975712	100000
500000	71	999999	18701424	250000
1000000	151	1999999	39402848	500000
2000000	314	3999999	82805696	1000000
5000000	839	9999999	220722784	2500000

Process returned 0 (0x0) execution time : 1.640 s
Press any key to continue.



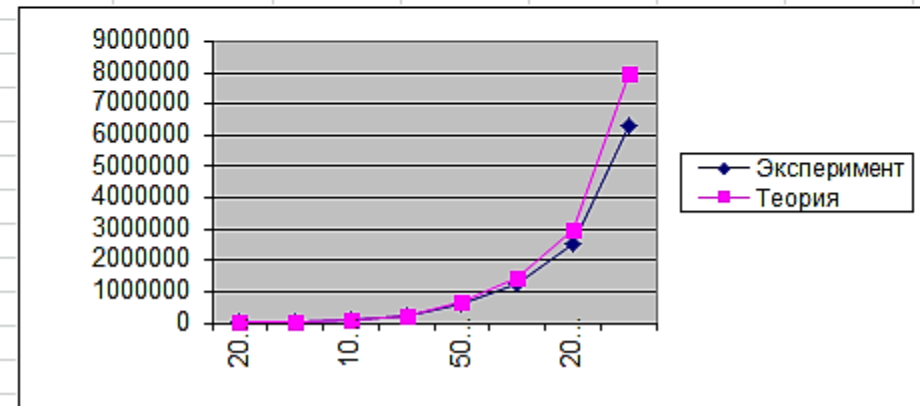
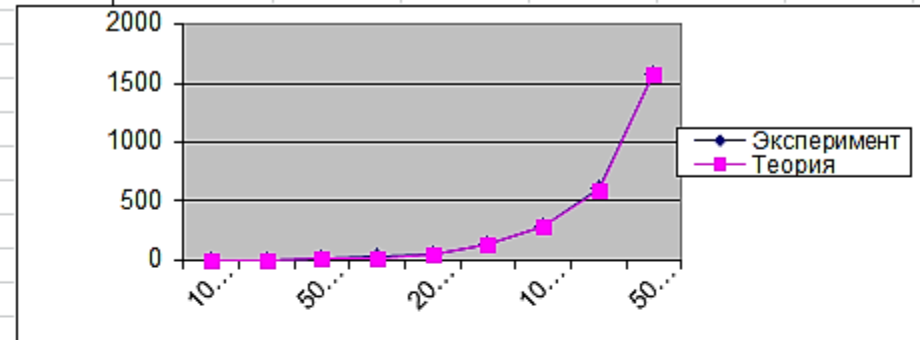
- [illegible]



РГР Анализ сортировок

- A: размерность, B: измеряемый параметр,
- C: функция трудоемкости –
 $=\$E\$2*A2*LOG(A2;2)$ – линейно-логарифмическая
- E2: коэффициент, вычисляемый для равенства значений в B6-C6
 $=B6/(A6*LOG(A6;2))$
- D – относительный процент отклонения $=(B2-C2)/C2$

	A	B	C	D	E
1	N	T1(эскп)	T1*(матем)	отклонение	коэф по N1
2	10000	3	1,9	59%	0,0000141968
3	20000	5	4,1	23%	
4	50000	12	11,1	8%	
5	100000	26	23,6	10%	
6	200000	50	50,0	0%	
7	500000	136	134,4	1%	
8	1000000	286	283,0	1%	
9	2000000	602	594,3	1%	
10	5000000	1585	1579,6	0%	
11					
12	N	calls(эскп)	calls(матем)	отклонение	коэф по N1
13	10000	12613	9528	32%	0,0717075969
14	20000	25375	20491	24%	
15	50000	62901	55966	12%	
16	100000	126411	119104	6%	
17	200000	252549	252549	0%	
18	500000	632469	678769	-7%	
19	1000000	1264011	1429245	-12%	
20	2000000	2531263	3001905	-16%	
21	5000000	6330109	7978724	-21%	
22					





РГР Анализ сортировок

Задания и методические материалы

[Все материалы также на vk.com/cprog_cs](#)

[Информатика \(семестр 1\)](#)

[Языки программирования \(семестр 2\)](#)

[ООП \(семестр 3\)](#)

[Оценка производительности программ](#)

[Программа построения графиков](#)

(формат данных программы)

[РГР, КР и КП](#) (все задания)

Лабораторные работы (все)

[4.2. Арифметические задачи](#)

[4.3. Итерационные циклы и приближенные вычисления](#)

[4.4. Символы. Строки. Текст](#)

[4.5. Последовательные текстовые файлы](#)

[4.6. Сортировка и поиск](#)

[5.2. Указатели и ссылки](#)

[5.4. Иерархия типов данных и функций](#)

[6.2. Массивы указателей](#)

[6.3. Линейные списки](#)

[7.4. Рекурсия и поисковые задачи](#)

[8.5. Деревья](#)

[9.1. Биты. Байты. Машинные слова](#)

[9.2. Работа с памятью на низком уровне](#)

[9.2. Функции с переменным числом параметров](#)

[9.3. Указатель на функцию](#)

[9.4. Позиционирование в текстовых файлах](#)

[9.5. Структуры данных в двоичном файле](#)

[10.1. Объекты и классы](#)

[10.3. Переопределение операций](#)

[10.5. Шаблоны. Классы структур данных](#)

Тестовые задания (все)

(вопросы без ответов)

[2. Анализ программ](#)

[4.2. Арифметические задачи](#)

[4.3. Итерационные циклы](#)

[4.4. Символы. Строки. Текст](#)

Измерение производительности программ

Зачем, что и как измерять?

Измерение времени работы программы произвольных данных, какой практический «потолок» использования данных. При этом нужно учитывать следующие важные моменты:

- можно измерять время работы программы с помощью системного времени (**clock**), то мы получим, что это время зависит от аппаратной реализации.
- в 4.1. было введено понятие трудоемкости (сложности) входных данных N . При измерении трудоемкости $T(N)$, что справедливо при линейной сложности; невозможно;
- для получения трудоемкости в программе должна присутствовать соответствующая операция.
- время выполнения программы может зависеть от трудоемкости иногда бывает достаточно сложным в конкретном случае мы будем иметь некий коэффициент.
- измерения можно проводить разными способами, а можно получать значения измеряемых величин.

В качестве примера посмотрим, как выглядит динамический массив указателей с сохранением порядка при вставке прямо упорядоченного текста, $T=N^2/2$, текст литературного произведения, имеющий достаточную длину.

//-----62-06.cpp

CPRG

[Главная](#)

[Задания](#)

[Учебник](#)

[Программы](#)

[Анимации и модели](#)

[Литература и глоссарий](#)

[Метод.материалы](#)

[Программистский юмор](#)