



# 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);
}
```



# 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){  
        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\n",size,tt,cCall,cStep,cSwap);  
    }  
}  
  
int main(){  
    srand(time(NULL));  
    //oneExperience(genOrdRandom,0,1);  
    //oneExperience(genOrdInc,0,1);  
    oneExperience (genOrdDec,0,1);  
}
```

D:\Temp\LectureExamples\bin\Debug\LectureExamples.exe

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.



# РГР Анализ сортировок

- Консоль: правая кнопка, верхний левый угол: выделить всё, копировать
- Блокнот: вставить, сохранить
- Excel: данные, из текста, выбор файла, с разделителями, разделитель-пробел

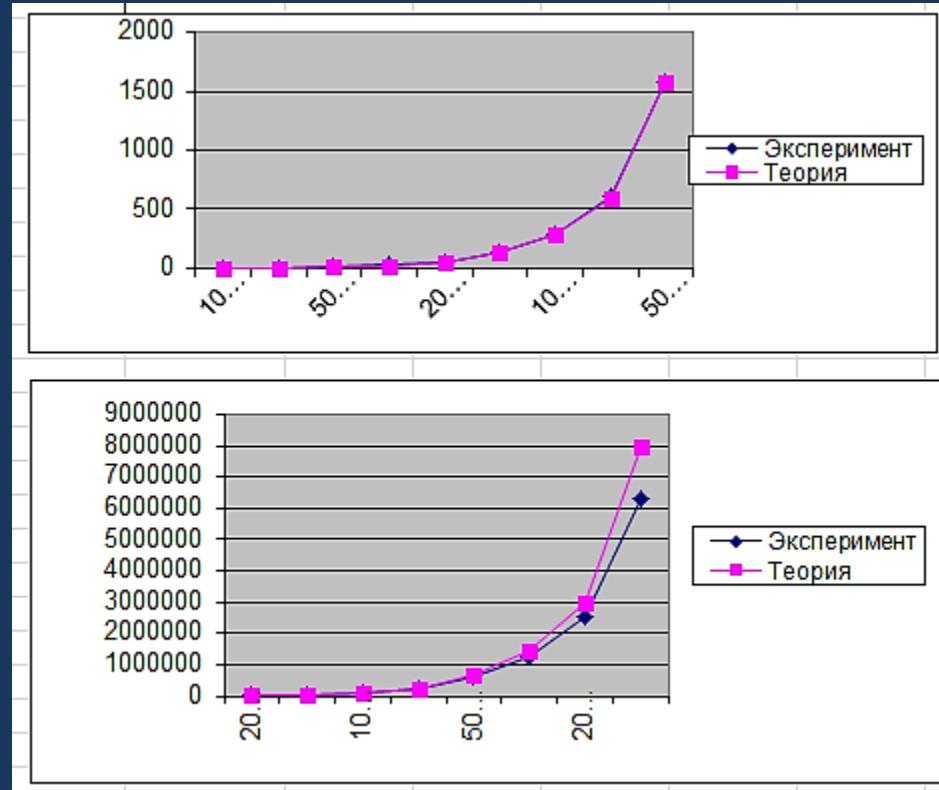
A	B	C	D	E	F	G	H	I	J	K
1	Среднее арифметическое					Быстрая				
2	Random					Random				
3	N	time	calls	steps	swaps	N	time	calls	steps	swaps
4	10000	3	12613	234032	31080	10000	1	13925	149699	51421
5	20000	5	25375	503454	66988	20000	3	27823	318783	113172
6	50000	12	62901	1 373 197	184087	50000	7	69631	941778	312485
7	100000	26	126411	2 922 441	392719	100000	15	138995	2030246	663989
8	200000	50	252549	6 193 854	835988	200000	31	277639	4258380	1417147
9	500000	136	632469	16642677	2256315	500000	84	694711	11899025	3839063
10	1000000	286	1264011	35034435	4762319	1000000	173	1389883	25980143	8077904
11	2000000	602	2531263	73570755	1E+07	2000000	363	2779835	51545778	17357476
12	5000000	1585	6330109	195497657	2,7E+07	5000000	970	6950043	139325969	46413055
13										
14	Dec					Dec				
15	10000	1	19999	262232	5000	10000	123	19999	49995000	5000
16	20000	2	39999	564464	10000	20000	482	39999	199990000	10000
17	50000	6	99999	1543928	25000					
18	100000	12	199999	3287856	50000					
19	200000	27	399999	6975712	100000					
20	500000	77	999999	18701424	250000					
21	1000000	150	1999999	39402848	500000					
22	2000000	315	3999999	82805696	1000000					
23	5000000	843	9999999	220722784	2500000					
24	2N-1				N/2					
25										



# РГР Анализ сортировок

- А: размерность, В: измеряемый параметр,
- С: функция трудоемкости –  
 $=$E$2*A2*LOG(A2;2)$  – линейно-логарифмическая
- Е2: коэффициент, вычисляемый для равенства значений в В6-С6  
 $=B6/(A6*LOG(A6;2))$
- D – относительный процент отклонения  $=(B2-C2)/C2$

	A	B	C	D	E
1	N	T1(эспл)	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(эспл)	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](#)

**C<sub>PROG</sub>**

[Главная](#)  
[Задания](#)

[Учебник](#)  
[Программы](#)

[Анимации и модели](#)  
[Литература и глоссарий](#)

[Метод.материалы](#)  
[Программистский юмор](#)

## Измерение производительности программ

Зачем, что и как измерять?

Измерение времени работы программы прои-  
данных, какой практический «потолок» использо-  
ванных. При этом нужно учитывать следующие важ-

- можно измерять время работы программы в системного времени (`clock`), то мы пол-  
того, это время зависит от аппаратной и со-  
фтверной среды;
- в 4.1. было введено понятие трудоемкости в  
размерности входных данных  $N$ . Приводилось  
трудоемкости  $T(N)$ , что справедливо при  
невозможно;
- для получения трудоемкости в про-  
цессорное время необходимо, чтобы в про-  
цессоре присутствует соответствующая опера-  
ция;
- время выполнения программы может не  
иметь линейной зависимости от размерности  
трудоемкости иногда бывает достаточ-  
но, чтобы в конкретном случае мы будем иметь неко-  
торую зависимость;
- измерения можно проводить разными  
способами, но в результате измерений  
мы можем получать значения измеряемых  
параметров.

В качестве примера посмотрим, как выглядят  
изменения в работе динамический массив указателей с сохранением по-  
зиций при вставке прямо упорядоченного текста,  $T=N^2/2$ ,  
где  $N$  – количество символов в тексте, а  $T$  – время выполнения программы.  
Для этого воспользуемся текстом литературного произведения, имеющий доста-

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