Сортировки

Опять же классическая задача обработки данных.

А любой набор данных можно отсортировать?

Нужно определиться с операцией сравнения.

Для простоты ставим для начала задачу сортировки вещественного массива по возрастанию.

```
void Sort (int n, double *a);
```

Традиционные методы:

— восходящая пузырьковая сортировка
— нисходящая пузырьковая сортировка (просеивание)
— перестановка максимума (сортировка обменом)
— сортировка Шелла
— быстрая сортировка
— сортировка слиянием
— сортировка кучей (пирамидальная, турнирная)
— radix сортировки
— сортировки подсчетом

Оценка производительности и эффективности сортировки:

— трудоемкость
— — гарантированная (для любой перестановки гарантируеся)
— — средняя (на некотором множестве перестановок)
— — “естественная наилучшая”
— дополнительная память
— $O(1)$
— $O(\log N)$
— $O(\sqrt{N})$
— $O(N)$

Сами алгоритмы обсудим чщть позже, а пока пактическая строна вопроса:

1. тестирование быстродействия

генерация массива длины N
замер времени на сортировку
проверка правильности упорядочивания

```
#include <time.h>
..............
int n = ... ;
clock_t t1, t2;
double seconds;

double *a = (double*)malloc(n*sizeof(double));
FillArray (n, a);
t1 = clock();
Sort(n, a);
t2 = clock();
seconds = (double)(t2-t1) / CLOCKS_PER_SEC;
printf("sorting time %f\n", seconds);
printf("%s\n", (TestSort(n, a)) ? "success" : "failure");
------------
```

```
bool TestSort(int n, double *a)
{
   int i;
   for (i=1; i<n; i++) {
      if (a[i-1] > a[i]) return false;
   }
   return true;
}
```

2. универсальность.
Указатель на функцию.

```
int (*f)(int);
bool (*ts)(int, double*);
void (*s)(double);
int (*cmp)(double, double);
int (*cmpv)(const void *, const void *);
```

Важна сигнатура функции, т.е. список типов параметров.

```
double CalculateSomething (double x, double (*f)(double))
{
    return f(x) + f(2*x);
}

double MyFun (double x)
{
    return x*x;
}

z = CalculateSomething(2.5, MyFun);
z = CalculateSomething(1.0, sin);
z = CalculateSomething(5.0, sqrt);
```

Для универсальности процедуры сортировки сравнение элементов можно задавать отдельной функцией

```
int Compare(double a, double b)
{
   return (a<b) ? -1 : (a>b) ? 1 : 0;
}

void Sort (int n, double *a, double (*cmp)(double, double))
{  .....
   if ( cmp(a[i],a[j]) > 0 )  .....  // if (a[i] > a[j]) ....
   .....
}

Sort(n, a, Compare);
```

3. Стандартная функция qsort

```
void qsort (void *array, size_t n_elems, size_t elem_size,
            int (*cmp)(const void *, const void *));
```

Функция сравнения заготовлена на произвольные ситуации. Нам надо ее реализовать в конкретном случае, т.е. для типов элементов нашего массива.

```
double *a;
int n;
.....
qsort (a, n, sizeof(double), CmpDoubleAscending);
.....
int CmpDouble Ascending (const void *a, const void *b)
{
   const double *pa = (const double *)a;
   const double *pb = (const double *)b;
   return (*pa<*pb) ? -1 : (*pa>*pb) ? 1 : 0;
}
```

Теперь подробнее об алгоритмах всех этих сортировках.
Все простые сортировки надо уметь писать сходу и в любом состоянии !!!

```
void swap(double *a, double *b)   //  swap(&a[i], &a[j]);
{                                 //  swap(a+i, a+j);
    double c = *a;
    *a = *b;
    *b = c;
}

void BubbleSort (int n, double *a)
{
    int i, k;
    bool br;   // признак прерывания при досрочной сортировке
    for (k = n; k>1; k--) {
        br = true;
        for (i = 0; i < k-1; i++) {
            if (a[i] > a[i+1]) {
               swap(a+i, a+i+1);
               br = false;
            }
        }
        if (br) return;
    }
}

void SiftSort (int n, double *a)
{
    int i, k;
    for (i = 1, i < n; i++) {
```

```
        for (k = i; k > 0 && a[k] < a[k-1]; k--) {
            swap(a+k, a+k-1);
        }
    }
}

void SwapMaxSort (int n, double *a)
{
    int i, k, imax;
    double amax, c;
    for (k = n-1; k > 1; k--) {
      amax = a[0];
      imax = 0;
      for (i = 1; i <= k; i++) {
         if (amax > a[i]) { amax = a[i]; imax = i; }
      }
      if (imax != k) {
         swap(a+k, a+imax);
      }
   }
}

void ShellSort(int n, double *a)        .// N^2 до N^{4/3} и даже N log^2 N
{
   int i, j, k;
   for (int k = n / 2; k > 0; k /= 2) {
      for (int i = k; i < n; i++) {
         for (int j = i - k; j >= 0 && a[j] > a[j+k]; j -= k) {
             swap(a+j, a+j+k);
         }
      }
   }
}
```