



ИЗ ОПЫТА РАБОТЫ

***М. С. Долинский**, к. т. н., доцент кафедры математических проблем управления Гомельского государственного университета им. Ф. Скорины*

Решение задач по информатике на тему «Сортировка»

Введение

Автор много лет занимается обучением программированию школьников разных возрастов и первокурсников математического факультета (специальности «Программное обеспечение информационных технологий», «Прикладная математика (научно-производственная деятельность)», «Информатика и технологии программирования») [1–23]. Все это время автор занимался созданием литературы для самостоятельного изучения школьниками и студентами, стараясь представить материал в как можно более простой и понятной форме.

В данной статье приводится пример такого материала для обучения решению задач по информатике на тему «Сортировка». В классической постановке задача звучит так: имеется одномерный массив, требуется переставить его элементы по возрастанию/убыванию.

Такой материал может быть интересен для преподавателей как в качестве иллюстрации методики обучения, так и по содержанию. В то же время, автор считает, что этот материал будет полезным и интересным для школьников и студентов, занимающихся

самообучением.

Всем заинтересованным предлагается следующий порядок работы: откладывать статью в сторону и пытаться самостоятельно выполнить предлагаемое задание. Первый раз — после прочтения условия задачи, второй — после прочтения указаний к решению.

Сортировка обменом

Сортировка обменом — это простейший алгоритм, основанный на одном из базовых алгоритмов на одномерном массиве — поиске минимального элемента и его номера:

```
Min:=a[1]; Nom:=1;
for i:=1 to N do
  if a[i]<Min
    then begin Min:=a[i]; Nom:=i; end;
```

То есть мы просматриваем все элементы, и если очередной меньше минимального, то запоминаем его и его номер.

Для выполнения сортировки мы последовательно находим минимальный элемент и его номер, после чего меняем местами первый элемент и найденный минимальный. Затем, уже начиная со второго элемента, снова ищем минимальный и его номер и меняем местами второй элемент с найденным минимальным, и так далее до предпоследнего элемента.

Ниже представлен полный текст программы, которая считывает число N ($N \leq 100$), затем N элементов одномерного массива, а затем сортирует их обменом и выводит результат сортировки.

```
var
  a          : array [1..100] of longint;
  i,j,Min,Nom,N : longint;
begin
  readln(N);
  for i:=1 to N do readln(a[i]);
  for j:=1 to N-1 do
    begin
```

```

Min:=a[j]; Nom:=j;
for i:=j+1 to N do
  if a[i]<Min
    then begin Min:=a[i]; Nom:=i; end;
  a[Nom]:=a[j];
  a[j] :=Min;
end;
for i:=1 to N do writeln(a[i]);
end.

```

Если требуется сортировать массив по убыванию, надо искать каждый раз не минимальный элемент из оставшихся неупорядоченными, а максимальный, для чего достаточно в операторе if изменить знак «<» на знак «>» (а для красоты еще и название Min на Max).

Сортировка с номерами

Иногда при решении задач требуется не только упорядочить элементы массива по возрастанию или убыванию, но еще и запомнить, на какой позиции изначально находился каждый элемент.

Для решения этой задачи достаточно завести специальный массив для хранения изначальных (стартовых) номеров элементов s_nom:

```
for i:=1 to N do s_nom[i]:=i;
```

И затем перемещать элементы в массиве s_top синхронно с перемещением элементов в сортируемом массиве a.

Ниже представлен полный текст программы, решающей такую задачу и выводящей параллельно элементы и их изначальные номера:

```

var
  a,s_nom      : array [1..100] of longint;
  i,j,Min,Nom,N : longint;
begin
  readln(N);
  for i:=1 to N do readln(a[i]);
  for i:=1 to N do s_nom[i]:=i;

```

```

for j:=1 to N-1 do
begin
  Min:=a[j]; Nom:=j;
  for i:=j+1 to N do
    if a[i]<Min
      then begin Min:=a[i]; Nom:=i; end;
  a[Nom]:=a[j]; s_nom[Nom]:=s_nom[j];
  a[j] :=Min; s_nom[j] :=Nom;
end;
for i:=1 to N do writeln(a[i], ' ',s_nom[i]);
end.

```

Стабильная сортировка (сортировка пузырьком)

В некоторых задачах требуется не менять порядок элементов с одинаковыми ключами сортировки. Например, задача «Таблица» (Гомельская областная олимпиада 2003 г. для учеников 5–8 классов):

После проведения городской олимпиады в главный компьютер залез злобный вирус и перепутал таблицу с результатами. Но к счастью сохранились количество участников (N), места ($A1i$), баллы ($A2i$), фамилия участника ($A3i$) и класс ($A4i$). Помогите заново рассортировать таблицу по местам в порядке возрастания. Если места равны, вывести в том порядке, в котором они идут во вводе.

Формат ввода

```

N
A11 A21 A31 A41
A12 A22 A32 A42
...
A1n A2n A3n A4n

```

Формат вывода

```

B11 B21 B31 B41
B12 B22 B32 B42 — отсортированная таблица.
...
B1n B2n B3n B4n

```

Ограничения:

$1 \leq N \leq 100$

$1 \leq A_1, A_2, A_4 \leq 1000$

Пример ввода:

```
4
2 100 Petrov 7
4 12 Sidorov 9
2 100 Ivanov 11
1 200 Kozlov 7
```

Пример вывода:

```
1 200 Kozlov 7
2 100 Petrov 7
2 100 Ivanov 11
4 12 Sidorov 9
```

Можно заметить, что достаточно вводить исходные данные в два массива — числовой массив *a* (занятые места) и строковый массив *s* (вся остальная информация об участнике с номером *i*), и затем сортировать по местам, синхронно передвигая остальную информацию об участниках:

```
var
  a          : array [1..100] of longint;
  s          : array [1..100] of string;
  i,j,Min,Nom,N : longint;
  t          : string;
begin
  readln(N);
  for i:=1 to N do readln(a[i],s[i]);
  for j:=1 to N-1 do
    begin
      Min:=a[j]; Nom:=j;
      for i:=j+1 to N do
        if a[i]<Min
          then begin Min:=a[i]; Nom:=i; end;
      a[Nom]:=a[j]; t:=s[Nom]; s[Nom]:=s[j];
      a[j] :=Min;          s[j] :=t;
    end;
```

```
for i:=1 to N do writeln(a[i], ' ',s[i]);
end.
```

Однако применение описанной выше сортировки обменом в данном случае выведет ответ:

```
1 200 Kozlov 7
2 100 Ivanov 11
2 100 Petrov 7
4 12 Sidorov 9
```

Вместо требуемого ответа:

```
1 200 Kozlov 7
2 100 Petrov 7
2 100 Ivanov 11
4 12 Sidorov 9
```

Поскольку по условиям задачи в случае равенства мест нужно соблюдать порядок ввода, а сортировка обменом его нарушает уже при переносе строки

```
1 200 Kozlov 7
на первую позицию, а строки
2 100 Petrov 7
на последнюю.
```

В таких задачах требуется применять так называемую стабильную сортировку, которая не меняет порядок следования элементов, имеющих равные ключи сортировки. Одним из таких методов сортировки является сортировка пузырьком:

```
for i:=1 to n-1 do
  for j:=1 to n-i do
    if a[j]>a[j+1]
      then begin
        t:=a[j]; a[j]:=a[j+1]; a[j+1]:=t;
      end;
```

Здесь в цикле по l от 1 до $n-1$, $n-i$ раз сравниваются попарно все соседние элементы, и если текущий элемент массива больше следующего элемента, то они меняются местами. В результате самый большой элемент оказывается на последнем месте (с номером $n-i+1$), то есть при $i = 1$ — на месте n , при $i = 2$ — на месте $n-1$ и т. д. При этом, если элементы имеют одинаковые

значения, они никогда не меняются местами.

Тогда решение нашей задачи с помощью сортировки пузырьком выглядит так:

```
var
  a      : array [1..100] of longint;
  s      : array [1..100] of string;
  i,j,Min,Nom,N,ta : longint;
  ts     : string;
begin
  readln(N);
  for i:=1 to N do readln(a[i],s[i]);
  for i:=1 to N-1 do
    for j:=1 to N-i do
      if a[j]>a[j+1]
        then begin
          ta:=a[j]; a[j]:=a[j+1]; a[j+1]:=ta;
          ts:=s[j]; s[j]:=s[j+1]; s[j+1]:=ts;
        end;
    for i:=1 to N do writeln(a[i], ' ',s[i]);
end.
```

И это решение выводит правильный ответ:

```
1 200 Kozlov 7
2 100 Petrov 7
2 100 Ivanov 11
4 12 Sidorov 9
```

Сортировка подсчетом

Рассмотрим простейшую задачу сортировки N натуральных чисел $a[i]$ по возрастанию ($N \leq 100000$, $A[i] \leq 100000$).

100 000 чисел ($N \leq 100\,000$) сортировкой обменом, как и сортировкой пузырьком, сортировать недопустимо долго. Поскольку обе сортировки содержат вложенный цикл по N , то они выполняют порядка $N \times N$ итераций. При $N = 100\,000$ это $100\,000 \times 100\,000 = 10$ миллиардов итераций. Такое количество итераций не успевает выполняться за время, которое обычно отводится на тестирование решений. Грубая оценка может быть такой — 10 миллионов итераций всегда успеют, 10 миллиардов итераций — всегда не успеют.

Данная задача может быть решена, основываясь на том факте, что значения чисел находятся в довольно маленьком диапазоне (от 1 до 100 000). Поэтому мы можем завести массив b с таким же количеством элементов (от 1 до 100 000) и в $b[i]$ хранить сколько раз на вводе встретилось число i .

Прежде чем перейти к анализу исходного текста программы сортировки подсчетом, приведем физический аналог этой задачи. Пусть перед нами стоит задача упорядочить монеты, которые лежат в огромной куче. Тогда мы просто разложим эту большую кучу на кучки поменьше, в каждой из которых лежат монеты одного номинала. А затем выложим монетки из этих кучек в порядке возрастания (убывания) номинала монет.

Вернемся к нашей задаче. Программа сортировки подсчетом (по возрастанию) может быть написана так:

```
var
  a      : array [1..100000] of longint;
  n,i,j  : longint;
begin
  for i:=1 to 100000 do a[i]:=0;
  read(n);
  for i:=1 to n do
    begin
      read(j);
      inc(a[j]);
    end;
  for i:=1 to 100000 do
    if a[i]>0
      then for j:=1 to a[i] do writeln(i);
end.
```

Для сортировки по убыванию достаточно заменить строку
for i:=1 to 100000 do
на строку
for i:=100000 downto 1 do

Заметим, что сортировка подсчетом выполняется одиночным циклом, что обеспечивает сложность порядка N (то есть максимум 100 000 итераций).

Быстрая сортировка

Существует также способ сортировки любых чисел со сложностью порядка $N \times \log N$, что вполне приемлемо по времени для $N = 100\,000$. Здесь под $\log N$ понимается двоичный логарифм числа N . $\log_{10} 100\,000 = \log_{10} 10^5 = 5 \times \log_{10} 10 \leq 5 \times 4 = 20$ (поскольку $2^4 = 16 > 10$, то $4 > \log_{10} 10$). Итого потребуется не более чем $20 \times 100\,000 = 2\,000\,000$ итераций, что вполне приемлемо по времени.

Быстрая сортировка участка массива A от p до r происходит следующим образом:

1. Элементы массива A переставляются так, что бы любой из элементов $A[p] \dots A[q]$ был не больше любого из элементов $A[q+1] \dots A[r]$. Эта операция называется разделением. Здесь q — некоторое число в интервале $p \leq q < r$.

2. Процедура сортировки рекурсивно вызывается для массивов $A[p \dots q]$ и $A[q+1, r]$. В качестве «граничного элемента» принимается $X = A[p]$.

3. Далее заводим два указателя: i — при просмотре элементов с начала сортируемого участка и j — при просмотре элементов с конца сортируемого участка. И пока $i < j$ выполняется следующее: находим $A[j]$ — первый с конца элемент меньше X и $A[i]$ — первый с начала элемент больше X . Если $i < j$, то меняем местами элементы $A[i]$ и $A[j]$:

$t = A[i]; A[i] = A[j]; A[j] = t;$

По завершению цикла «пока $i < j$ » мы и получаем границу «разделения» j .

Понятно, что вызывать алгоритм быстрой сортировки нужно передавая ему вначале номера начального и конечного элементов массива, например, `QuickSortA(1,N)`.

Ниже приведено полное решение задачи сортировки по возрастанию N целых чисел со значениями в диапазоне `longint` (примерно от -2 миллиардов до $+2$ миллиардов):

```
const
  MaxN = 100000;
var
  x      : array [1..MaxN] of longint;
  i,N,K,Ans : longint;
  procedure QSort_X_Up(p,r:longint);
```

```

var
  i,j,a,t : longint;
begin
  if p>=r then exit;
  a := X[p]; i:=p-1; j:=r+1;
  while i<j do
    begin
      repeat j:=j-1 until X[j]<=a;
      repeat i:=i+1 until X[i]>=a;
      if i<j
        then begin
          t:= X[i]; X[i]:= X[j]; X[j]:=t;
          end;
        end;
      QSort_X_Up(p,j);
      QSort_X_Up(j+1,r);
    end;
begin
  readln(N);
  for i:=1 to N do readln(X[i]);
  QSort_X_Up(1,N);
  for i:=1 to N do writeln(X[i]);
end.

```

Для сортировки по убыванию достаточно изменить знаки неравенства на противоположные в строках:

```

      repeat j:=j-1 until X[j]<=a;
      repeat i:=i+1 until X[i]>=a;

```

А также для красоты заменить слово Up на слово Down в определении и вызове рекурсивной процедуры.

Заключение

В данной статье приведен материал для обучения решению задач по информатике на тему «Сортировка». Технической основой методики является разработанная инструментальная система дистанционного обучения (Distance Learning Belarus — <http://dl.gsu.by>). Все задачи, приведенные в статье, могут быть сданы в курсе «Методы алгоритмизации».

Литература

1. Долинский, М. С. Об опыте подготовки школьников Гомельской области к республиканским и международным олимпиадам по информатике / М. С. Долинский // Информатизация образования. – 2009. – № 1 (54). – С. 29–40.
2. Долинский, М. С. Система интернет-курсов дифференцированного обучения программированию школьников и студентов / М. С. Долинский, М. А. Кугейко // Информатизация образования. – 2010. – № 1 (58). – С. 58–68.
3. Долинский, М. С. Как учить думать школьников и студентов? / М. С. Долинский, М. А. Кугейко // Информатизация образования. – 2010. – № 2 (59). – С. 62–72.
4. Долинский, М. С. Технология развивающего дифференцированного обучения программированию младших школьников «с чистого листа» / М. С. Долинский, М. А. Кугейко // Информатизация образования. – 2010. – № 3 (60). – С. 12–20.
5. Долинский, М. С. Интернет-курс «Базовое программирование» как средство подготовки к областным олимпиадам по информатике / М. С. Долинский, М. А. Кугейко // Информатизация образования. – 2010. – № 4 (61). – С. 3–15.
6. Долинский, М. С. Развитие мышления младших школьников на основе флеш-заданий на рисование, раскраску и конструирование в системе DL.GSU.BY / М. С. Долинский, Ю. В. Решетько, М. А. Кугейко // Информатизация образования. – 2011. – № 1 (62). – С. 24–35.
7. Долинский, М. С. Какими должны быть задачи на олимпиадах по информатике / М. С. Долинский, М. А. Кугейко // Информатизация образования. – 2011. – № 1 (62). – С. 68–76.
8. Долинский, М. С. Флеш-шаблоны для создания заданий развивающего обучения / М. С. Долинский, Ю. В. Решетько, М. А. Долинская // Информатизация образования. – 2011. – № 2 (63). – С. 14–28.
9. Долинский, М. С. Конструирование интерактивных флеш-заданий на развитие мышления / М. С. Долинский, Ю. В. Решетько, М. А. Долинская // Информатизация образования. – 2011. – № 3 (64). – С. 21–33.
10. Долинский, М. С. Конструирование интерактивных флеш-заданий на развитие мышления на базе произвольных картинок / М. С. Долинский, Ю. В. Решетько, М. А. Долинская //

Информатизация образования. – 2011. – № 4 (65). – С. 3–14.

11. Долинский, М. С. Конструирование интерактивных флеш-заданий на базе собственных танов / М. С. Долинский, Ю. В. Решетько, Н. С. Лебедевко // Информатизация образования. – 2012. – № 1 (66). – С. 24–34.

12. Долинский, М. С. Конструктор интерактивных флеш-заданий как открытая система для создания электронных учебных пособий / М. С. Долинский, Ю. В. Решетько, М. А. Долинская, Н. С. Лебедевко // Информатизация образования. – 2012. – № 2 (67). – С. 35–45.

13. Долинский, М. С. Электронное учебное пособие «Математика. Начальная школа» / М. С. Долинский, Ю. В. Решетько, Н. С. Лебедевко // Информатизация образования. – 2012. – № 3 (68). – С. 30–42.

14. Долинский, М. С. Создание электронных учебных пособий для вузовских дисциплин с помощью конструктора флеш-заданий / М. С. Долинский, Ю. В. Решетько // Информатизация образования. – 2012. – № 4 (69). – С. 34–45.

15. Долинский, М. С. Интерактивная анимация в электронных учебных пособиях, создаваемых с помощью конструктора флеш-заданий / М. С. Долинский, Ю. В. Решетько, М. А. Долинская // Информатизация образования. – 2013. – № 1 (70). – С. 30–38.

16. Долинский, М. С. Учебный интернет-курс и перманентный интернет-конкурс «Математика 1–8 классов» / М. С. Долинский, Ю. В. Решетько, М. А. Долинская // Информатизация образования, 2013. – № 2 (71). – С. 38–47.

17. Долинский, М. С. Концептуальные основы и практика сквозного развивающего обучения информатике и программированию от детского сада до вуза / М. С. Долинский, Ю. В. Решетько, М. А. Долинская // Информатизация образования. – 2013. – № 3 (72). – С. 16–25.

18. Долинский, М. С. Об одном подходе к обучению программированию на первом курсе / М. С. Долинский, М. А. Долинская // Информатизация образования. – 2014. – № 1 (73). – С. 32–41.

19. Долинский, М. С. Использование форума при обучении программированию первокурсников / М. С. Долинский // Информатизация образования. – 2014. – № 2 (74). – С. 22–34.

20. Долинский, М. С. Элементы теории чисел: системы счисления / Долинский, М. С. // Информатизация образования. –

2015. – № 1 (75). – С. 14–28.

21. Долинский, М. С. Элементы теории чисел: битовая обработка / Долинский, М. С. // Информатизация образования. – 2015. – № 2 (76). – С. 3–15.

22. Долинский, М. С. Элементы теории чисел: решето Эратосфена / Долинский, М. С. // Информатизация образования. – 2016. – № 1 (77). – С. 11–20.

23. Долинский, М. С. Элементы теории чисел: длинная арифметика / Долинский, М. С. // Информатизация образования. – 2016. – № 2 (78). – С. 12–21.

Статья поступила 04.01.2017

