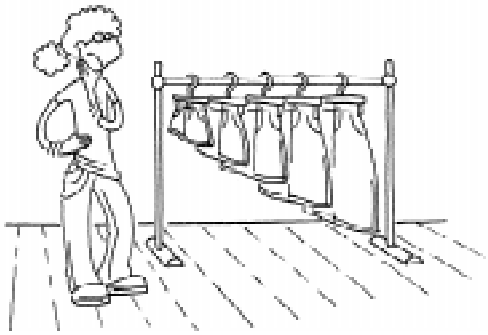
ТЕХНОЛОГИЧЕСКОЕ ОБРАЗОВАНИЕ

Ильин Виктор Павлович,
Цветкова Елена Юрьевна

МЕТОДИКА ОБУЧЕНИЯ ПРОФИЛИРОВАНИЮ ПРОГРАММ НА ПРИМЕРЕ ИЗУЧЕНИЯ ТЕМЫ «СОРТИРОВКА ДАННЫХ»

Тема «Сортировка данных» по причине своей фундаментальности и значимости, многочисленности методов и доступности их исследования дает хорошие возможности для организации как теоретических, так и лабораторно-практических занятий, направленных на формирование у учащихся основных качеств творческой деятельности: видение структуры сложного объекта, умение создавать оригинальный способ исследования на основе известных, умение делать выводы и др. В статье рассматривается методика организации практического занятия по исследованию учащимися временной сложности алгоритмов сортировки с использованием простых средств профилирования программ. Профилирование – это важный элемент цикла создания программ, которому в школе практически не уделяют внимания. Ознакомление учащихся с элементами профилирования программ значительно расширяет методологические возможности преподавателя при проведении занятий по алгоритмизации и программированию. Время, затраченное на ознакомление с профилировщиком, с лихвой окупается за счет повышения качества разрабатываемых программ, выявления участков программы, на выполнение которых затрачивается большое количество времени, и определения мест, в которых необходимо внести улучшения или изменить алгоритм работы.

Под сортировкой в информатике понимается процесс упорядочивания заданной последовательности данных одного типа по значению какого-либо признака (ключа). Основным критерием качества методов сортировки является сложность алгоритма, которая отражает порядок величины требуемого ресурса (времени/дополнительной памяти) в зависимости от размерности задачи. Выделяют временную $T(n)$ и емкостную (объем дополнительной памяти) «сложность» метода, где n – число элементов сортируемой последовательности. Время выполнения ряда алгоритмов зависит не только от размера задачи, но и от начального взаиморасположения данных. Поэтому выделяют максимальную



...процесс упорядочивания заданной последовательности данных одного типа по значению какого-либо признака...

сложность (для самого неблагоприятного случая), среднюю сложность (сложность метода в среднем), минимальную сложность (для наиболее благоприятного случая).

При выборе для изучения в школе конкретных методов сортировки из достаточно большого числа известных [1] будем руководствоваться следующим критерием: изучаемые методы должны быть простыми для изучения и программной реализации, отличаться принципами построения, должны иметь модификации, различаться по временной сложности. Среди простых методов внутренней сортировки, сортирующих элементы практически без использования дополнительной памяти, можно выбрать сортировку обменом, сортировку выбором (выделением), сортировку включениями (вставками). Предлагаемые методы являются простыми как для понимания, так и для программной реализации, и имеются очевидные их модификации, что позволяет организовать исследовательскую работу учащихся за приемлемое время.

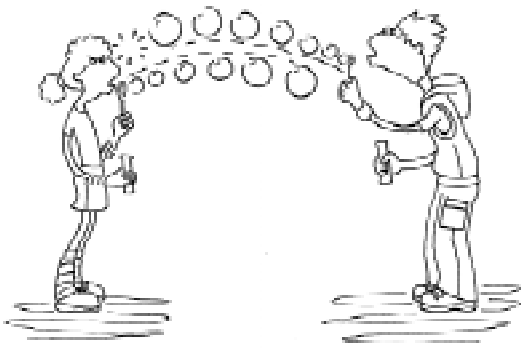
Наиболее часто используемой в школе системой программирования является интегрированная среда разработки *Turbo Pascal*, которая достаточно проста и эффективна для создания программ и включает в себя интерактивный отладчик (подменю *DEBUG*). Наглядным способом изучения алгоритма сортировки является пошаговое исполнение программы, реализующей исследуемый алгоритм сортировки, с просмотром взаиморасположения элементов массива на каждом шаге сортировки в окне просмотра *Watch*. Отладчик позволяет организовать трассировку значений выбранных переменных в процессе отладочного исполнения. Получение экспериментальных временных показателей алгоритмов сортировки вызывает необходимость использования последовательностей больших размерностей, при этом программы, реализующие соответствующие алгоритмы сортировки должны контролироваться набором таймеров и счетчиков. Временные характеристики программы могут быть получены с помощью простой программной системы *Turbo Profiler,* входящей в поставку интегрированной среды разработки. Данная система представляет собой профилировщик – программное средство, позволяющее контролировать каждый шаг выполнения программы и предоставлять подробные статистические сведения на всех этапах ее работы, начиная от подсчета времени и количества выполнений операторов и заканчивая контролем над вызовами прерываний и обращениями к файлам. Сведения, полученные в результате профилирования, позволяют определить целесообразность применения соответствующих методов повышения эффективности программ: уменьшение количества вычислений во внутренних циклах, сокращение числа повторений в цикле, развертывание циклов, изменения порядка в логических выражениях, уменьшение избыточных вычислений, замена медленных операций более быстрыми и т. д. Профилирование – это одна из очень полезных частей процесса создания качественного программного обеспечения, которая, наряду с отладкой, должна стать неотъемлемым элементом цикла создания программ. В настоящее время имеются мощные профилировщики, сочетающие возможности как профилирования, так и оптимизации программ. В условиях школы для предлагаемых задач вполне достаточно системы *Turbo Profiler.* Основные принципы работы и возможности профилировщика при рассмотрении конкретных примеров усваиваются учащимися достаточно быстро. Для проведения занятий целесообразно предоставить учащимся краткую инструкцию по работе с профилировщиком и, возможно, с целью уменьшения времени на проведение исследований, файлы программ методов сортировки (исходный текст и исполняемая программа), хотя, в силу их несложности, программы могут быть заранее написаны и отлажены самими учащимися.

Рассмотрим возможный вариант проведения исследований на примере анализа простейшего алгоритма сортировки обменами (пузырьковая сортировка) одномерного массива. Это самый известный из учебной литературы алгоритм сортировки. Первый вариант алгоритма представляет

собой два вложенных цикла. На каждом шаге внутреннего цикла сравниваются попарно соседние элементы. Если они не упорядочены, то элементы меняются местами. При сортировке по возрастанию за один проход внутреннего цикла путем последовательных обменов наибольший элемент перемещается на последнее место, при следующем проходе на предпоследнее и т. д. (вариант 1). В улучшенном варианте (вариант 2) для устранения просмотра уже отсортированной части массива правая граница внутреннего цикла при очередном проходе уменьшается на единицу. Эти алгоритмы имеют среднюю, максимальную и минимальную временные сложности $T(n^2)$ и требуют дополнительную память только для вспомогательных переменных при организации циклов. Следующее улучшение алгоритма (вариант 3) может быть достигнуто путем учета того факта, что если за очередной проход не было сделано ни одного обмена, то массив уже отсортирован и следующие проходы можно пропустить. Средняя и максимальная временные сложности улучшенного алгоритма остаются теми же, но в случае уже отсортированного массива минимальная временная сложность – $T(n)$.

Рассмотрим два варианта программной реализации метода (вариант 1, вариант 3). Тексты профилированных программ, реализующих данный метод, приведены в табл. 1. Так как профилировщику необходимы исходный текст и откомпилированная программа и для исследования требуются разные исходные данные как по способу задания, так и размерности массива, а механизм задания исходных данных все равно исключается из анализа метода, то можно необходимые изменения проводить непосредственно в тексте программы (случайная или фиксированная процедуры ввода, размерность и т. д.). Целесообразно сравнивать алгоритмы на одной и той же случайной последовательности. Для генерации исходных данных для крайних случаев (исходный массив уже упорядочен по возрастанию или убыванию) можно воспользоваться процедурой ввода для случайной последовательности, затем отсортировать при помощи процедуры, реализующей данный метод, и полученный упорядоченный массив использовать для исследования метода. Для проведения экспериментального сравнительного анализа различных методов необходимо испытания проводить на одинаковых наборах входных данных, упорядоченных в прямом порядке, упорядоченных в обратном порядке и случайных, для коротких, средних и длинных последовательностей. Для анализа программы с помощью *Turbo Profiler* требуются как исходный, так и выполняемый файлы программы сортировки. Поэтому при компиляции программы сортировки в интегрированной среде *Turbo Pascal* необходимо установить флажок в пункте меню *Options|Debugger...* в категории *Debugging|Browsing Stand-Alone Debugging* в состояние *On*. После запуска профилировщика необходимо загрузить исполняемую программу (расширение файла .exe) в систему Turbo Profiler. Откроются два окна: *Module*, в котором находится исходный текст, и *Execution Profile* – для отображения статистики процесса выполнения программы (см. рис. 1).

Перед началом профилирования программы необходимо пометить участки программы («области»), для которых собираются статистические данные (строка, оператор или вся программа). Комбинация клавиш <Alt-F10> активирует локальное меню окна *Module*. Пункт *Add Areas* предлагает список возможных границ «области» (для небольших программ можно



...пузырьковая сортировка...

воспользоваться опцией *Every Line in Module* – каждая строка в модуле.). По клавише F9 производится сбор статистических данных (для больших массивов на это потребуется определенное время, для уменьшения которого можно ограничить набор анализируемых областей). В каждой строке нижней панели содержится: название «области», количество секунд, затраченных на выполнение данной «области», процентное отношение этого времени к общему времени выполнения программы, горизонтальная диаграмма, пропорциональная этому отношению. По команде *Display* (для доступа комбинация клавиш <Alt-F10>) для установки параметров изображения предоставляется пять способов изображения данных в окне *Execution Profile:*

Time – изображение полного времени, затраченного на выполнение каждой из помеченных «областей»;

Count – изображение числа, показывающего, сколько раз за время выполнения программы управление передавалось помеченной «области»;

Both – одновременный показ времени выполнения и количества вызовов, для каждой помеченной «области»;

Per Call – изображение среднего времени, затраченного на одно выполнение помеченной «области»;

Longest – показ максимального времени затраченного на выполнение данной «области».

Для того чтобы вывести листинг профилируемой программы в файл, выберите в меню команду *Print|Options, File,* в окне *Destination File*, наберите на клавиатуре имя файла и нажмите <ENTER>, выберите в меню команду *Print|Module*, выберите название модуля и нажмите <ENTER>.

Задавая разную размерность и последовательность (несколько реализаций случайной последовательности, упорядоченные последовательности для наилучшего и наихудшего случая) исходных данных в массиве, учащиеся проводят исследование программы, реализующей метод сортировки, анализируют характеристики программы, выводимые в окнах профилировщика и в тексте профилированной программы. Анализируя полученные частотные и временные показатели выполнения операторов программы, учащийся определяет временные сложности алгоритма. Могут быть построены графики зависимости времени выполнения и числа итераций от длины входной последовательности при разной ее упорядоченности. В приведенных примерах наиболее убедительным для учащихся является результат сравнения алгоритмов сортировки для практически упорядоченных последовательностей (например, единственный элемент последовательности находится не на соответствующем месте, см. табл. 1), при этом временная сложность первого варианта алгоритма является квадратичной, а третьего варианта линейной. В зависимости от времени, выделенного для изучения темы, учащиеся исследуют один метод сортировки (и его простейшие модификации) или несколько методов. При проведении исследований нужно учесть, что при очень большом числе элементов в исходной последовательно-

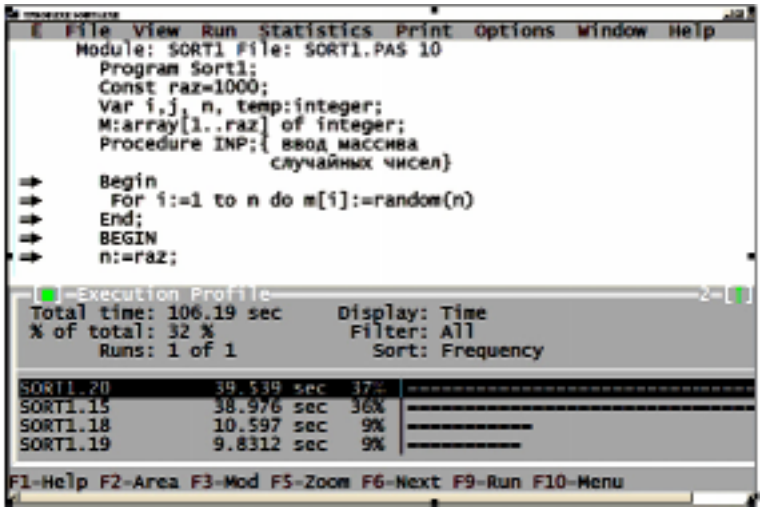


Рис. 1. Интерфейсные окна *Turbo Profiler*

Табл. 1. Тексты профилированных программ сортировки

Сортировка обменами (1)

```
Program: D:\BP\BIN\TEST\SORT1.EXE
File SORT1.PAS
Time  Counts
Program Sort1;
Uses Vvod;{генерация исходных данных}
Const raz=1000;
Var i, j, n, temp:integer;
M:array[1..raz] of integer;
0.0014  1      BEGIN
                n:=raz;
                Inp;{генерация
                исходных данных}
0.0000  1       for i:= 1 to n-1 do
0.1169  999      for j := 1 to n-1 do
80.980  998001    if M[j]>M[j+1] then
                   begin {обмен}
                    Temp := M[j];
                    M[j] := M[j+1];
                    M[j+1] := Temp
85.552  998001     end
0.0000  1      END.
```

Улучшенная сортировка обменами (3)

```
Program: D:\BP\BIN\TEST\SORT3.EXE
File SORT3.PAS
Time  Counts
Program Sort3;
Uses Vvod; {генерация исходных данных}
Const raz=1000;
Var i, j, n, temp:integer;
F:boolean;{фиксация факта обмена}
M:array[1..raz] of integer;
0.0014  1    BEGIN
              n:=raz;
              Inp; {генерация
              исходных данных}
              i:=1;
              f:=true;
0.0000  3     while (i<= n-1)and
                          F=true do
              begin
0.0000  2      F := false;
0.0000  2      for j := 1 to n-i do
0.0568  1997   if M[j]>M[j+1] then
               begin {обмен}
                 Temp := M[j];
                 M[j] := M[j+1];
                 M[j+1] := Temp;
                 F:= true;
0.0020  1997   end;
               i:=i+1;
               end
0.0012  1     END.
```

сти профилирование занимает большое время (можно уменьшить за счет ограничения при выборе числа контролируемых «областей» программы), а при малом числе элементов профилировщик оказывается нечувствительным к малым временным характеристикам (для устранения этого недостатка можно зациклить процесс создания и сортировки массива).

При проведении занятий с учащимися средних классов можно обойтись изучением и исследованием простых методов сортировки, для старших классов (особенно физико-математических) целесообразно показать и более совершенные методы. Проведенные учащимися самостоятельные исследования временной сложности алгоритмов могут помочь в восприятии программирования не только как процесса кодирования, но и, что достаточно важно, как творческого процесса конструирования и исследования программ.

Литература

1. *Кнут Д.* Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. М.: «Вильямс», 2000.

Ильин Виктор Павлович,
кандидат технических наук,
доцент СПбГЭТУ «ЛЭТИ»,
Цветкова Елена Юрьевна,
учитель информатики
ГОУ гимназия № 56,
г. Санкт-Петербург.