НАХОЖДЕНИЕ МАКСИМАЛЬНОГО И МИНИМАЛЬНОГО ЭЛЕМЕНТА В МАССИВЕ

Прусаков И.В.

г. Новочеркасск, Гуманитарно-технический колледж ФГБОУ ВО «Южно-Российский государственный политехнический университет (НПИ) им. М.И. Платова»

Научный руководитель: Растеряев Н.В., г. Новочеркасск, к. т. н., доцент, доцент кафедры «Стандартизация, сертификация и управление качеством», ФГБОУ ВО «Южно-Российский государственный политехнический университет (НПИ) им. М.И. Платова»

Основу современной инженерной деятельности составляет умение ставить задачи, разрабатывать алгоритмы и получать решения, производить анализ полученных данных и делать выводы. Поэтому в своей профессиональной деятельности инженер должен уметь грамотно применять персональный компьютер, современное программное обеспечение для решения научных и инженерных задач.

Типичной задачей при обработке больших массивов данных является задача поиска максимума или минимума. Например, в списке успеваемости учеников класса найти самого прилежного. Иначе говоря, требуется выбрать наибольшее значение среднего балла и указать фамилию ученика. Или, в массиве среднесуточных температур окружающего воздуха за некоторый период наблюдений определить минимальное значение и указать дату самого холодного дня.

Цель работы: разработка программы нахождения максимального и минимального элемента в среде программирования Паскаль-АВС.

Задачи:

1) ознакомиться с алгоритмами поиска в массивах и методами их программной реализации;

2) освоить приемы программирования в интегрированной среде Паскаль-АВС;

3) разработать алгоритм и блок-схему нахождения максимального и минимального элемента в массиве;

4) создать программу нахождения максимального и минимального элемента в массиве и протестировать ее.

Объект исследования: алгоритмы поиска в массивах.

Предмет исследования: Паскаль-программа нахождения максимального и минимального элемента массива в среде Паскаль-АВС.

Алгоритмы поиска информации

Очень часто в реальной жизни нам приходится сталкиваться с задачей поиска информации в объеме данных. Например, поиск фамилии ученика в журнале, поиск нужного слова в словаре. Существует множество алгоритмов поиска, но из всего многообразия алгоритмов рассмотрим два основных и наиболее часто используемых на практике.

В алгоритмах поиска существует два возможных окончания работы: поиск может оказаться удачным – заданный элемент найден в массиве и определено его месторасположение, либо проведенный поиск может оказаться неудачным – необходимого элемента в данном объеме информации нет.

Несмотря на то, что целью поиска является значение элемента, алгоритм поиска в случае удачного окончания выдает так же и местоположение искомого элемента, например его номер в массиве, так как по номеру элемента можно однозначно восстановить и его значение.

Для оценки алгоритмов мы будем использовать такую характеристику, как сложность.

Пусть, например, человек ищет на полке книжку с определенным названием. Книги на полке стоят вразнобой, то есть не по алфавиту. Как будет действовать человек? Он будет сравнивать по порядку название каждой книги на полке с тем названием, которое ему нужно найти. В итоге он или найдет нужную ему книгу, или, просмотрев все книги на полке, не обнаружит нужной книги. Этот пример передает суть алгоритма последовательного поиска в неупорядоченном массиве. Приведем его формальную запись.

Имеется одномерный массив a[1 … n], требуется найти элемент массива, равный P:

Алгоритм последовательного поиска в неупорядоченном массиве

Установить i = 1.

Если a_i = P, алгоритм завершил работу успешно.

Увеличить i на 1.

Если $i \leq n$, то перейти к шагу 2. В противном случае алгоритм завершил работу безуспешно.

Оценим сложность алгоритма последовательного поиска. Естественно оценивать сложность по числу сравнений с искомым элементом. В худшем случае искомый элемент окажется на последнем месте или не будет найден, и тогда необходимо будет проделать n сравнений, то есть сложность алгоритма будет равна n. Такой поиск также называют линейным, так как он решает задачу поиска с линейной скоростью по количеству сравнений.

Усложним задачу. Пусть нам требуется найти минимальный элемент в неупорядоченном массиве. Оказывается, что и эта задача имеет линейную сложность, и для поиска минимального (максимального) элемента в неупорядоченном массиве требуется n – 1 сравнение. Запишем алгоритм поиска максимального элемента в текстовой (вербальной) форме.

Алгоритм поиска максимального элемента в неупорядоченном массиве

Установить счетчик равным 1 (i = 1).

Положим значение текущего максимума равным первому исследуемому элементу ($max = a_1$).

Если исследованы еще не все элементы ($i < n$), то перейти к шагу 5, иначе алгоритм окончен (максимальный элемент равен max).

Перейти к следующему элементу (увеличить i на единицу).

Если рассматриваемый элемент больше, чем текущий максимум ($a_i > max$), то значение a_i присвоить max.

Перейти к шагу 4.

Последовательный поиск не является самым эффективным алгоритмом поиска. Например, человеку нужно найти в русско-английском словаре перевод слова на английский язык. Если он будет искать его с помощью алгоритма последовательного поиска (просматривая все слова подряд), то он потратит очень много времени. На самом же деле интуитивно человек действует совсем по-другому.

Поиск слова в словаре наиболее приближен к алгоритму бинарного (двоичного) поиска, который также называют логарифмическим поиском, или методом деления пополам (дихотомией). Этот алгоритм достаточно эффективен, но использовать его можно только в случае, когда данные упорядочены. В этом алгоритме мы используем сравнение искомого элемента с серединным элементом и с помощью результата этого сравнения устанавливаем, в какой части данных находится искомый элемент.

Заметим, что даже если данные упорядочены, то использовать алгоритм бинарного поиска мы можем не всегда. Это касается тех случаев, когда мы не имеем доступа к любому элементу массива. Например, если данные поступают к нам последовательно.

Алгоритмы поиска максимального или минимального элемента массива

Алгоритмизация – это общая последовательность действий, которые необходимо выполнить для построения алгоритма решения задачи, в том числе – выделение конкретных шагов алгоритмического процесса, определение вида формальной записи для каждого шага и установление определенного порядка выполнения каждого шага [1]

Разработка алгоритма решения задачи – это разбиение задачи на последовательно выполняемые этапы, причем результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. При этом должны быть четко указаны как содержание каждого этапа, так и порядок выполнения этапов. Отдельный этап алгоритма представляет собой либо другую, более простую задачу, алгоритм решения которой известен (разработан заранее), либо должен быть достаточно простым и понятным без пояснений.

Пусть поставлена следующая задача: по известным данным среднесуточных температур и дат найти максимальную или минимальную температуру и соответствующую ей дату.

Принцип поиска максимального или минимального элемента массива заключается в следующем. Первое, что необходимо сделать – создать массивы данных и дат. При этом сначала вводится параметр n – число элементов создаваемых массивов. Для ввода элементов массива используется цикл с параметром. Вводятся числовое значение и дата в формате ДД.ММ.ГГ.

Для определения того, будем находить максимальный или минимальный элемент, целочисленной переменной flag будем присваивать значения 1 или 0.

В дополнительную переменную заносится значение первого элемента массиваDan[ ], которое принимается за максимум (минимум); затем организовывается перебор оставшихся элементов массива, каждый из которых сравнивается с максимумом (минимумом); если текущий элемент массива оказывается больше (меньше), чем принятый за максимум (минимум), то этот элемент становится максимальным (минимальным).

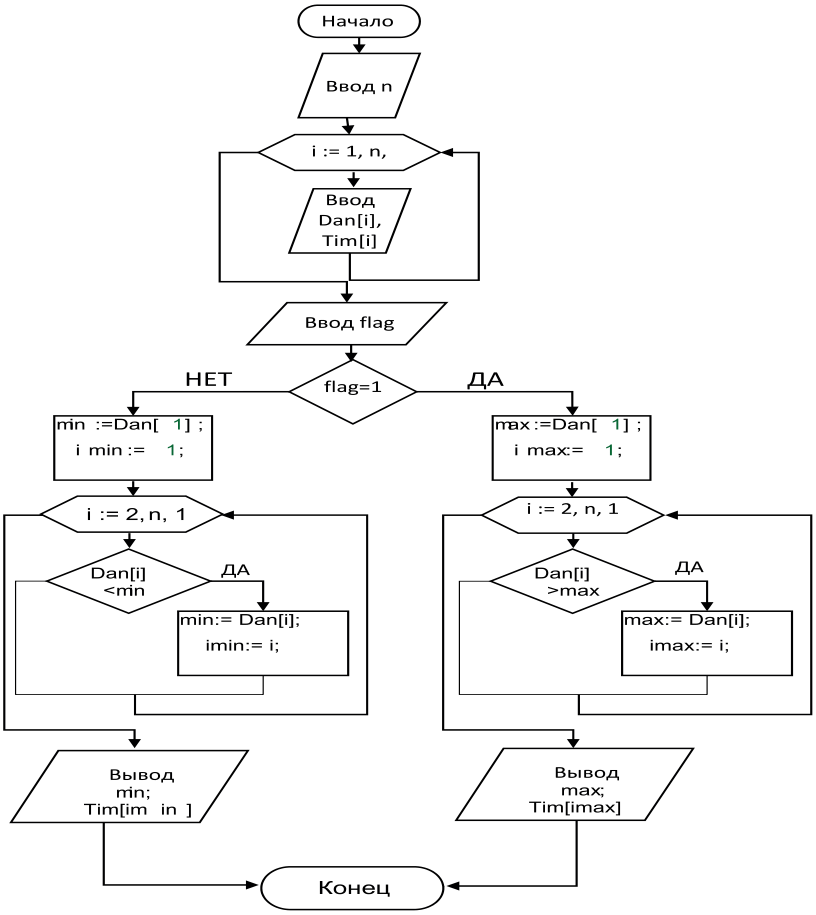

Рис. 1. Блок-схема алгоритма нахождения максимального и минимального элемента

```
Program Max_element_massiva_2;
     Var i, n, imax, imin, flag : integer;
         max, min : real;
         Dan : array [1..100] of real;
         Tim : array [1..100] of string[10];
    BEGIN    { Mein }
     Writeln('   Введите количество элементов массива') ;
     Readln(n) ;
     for i:=1 to n do
        begin
          write('  Введите ',i,'-е числовое значение  ');
          readln(Dan[i]);
          write('  Введите дату замера в формате ДД.ММ.ГГ  ');
          readln(Tim[i]);
        end;
     Writeln('   Какой элемент ищем?') ;
     Writeln('   Введите 1, если максимальный') ;
     Writeln('   Введите 0, если минимальный') ;
     Readln(flag) ;
  { Ищем максимальный элемент}
     If flag = 1 then
                  begin
     max:= Dan[1] ;  imax:= 1 ;
     For i := 2 to n do
        if Dan[i] > max then   begin
           max:= Dan[i]; imax:= i; end ;
     Writeln(' Максимальный элемент массива = ',max:6:3) ;
     Writeln(' Время = ',Tim[imax]) ;
                  end ;
  { Ищем минимальный элемент}
     If flag = 0 then
                  begin
     min:= Dan[1] ;  imin:= 1 ;
     For i := 2 to n do
        if Dan[i] < min then   begin
           min:= Dan[i]; imin:= i; end ;
     Writeln(' Минимальный элемент массива = ',min:6:3) ;
     Writeln(' Время = ',Tim[imin]) ;
                  end ;
        END.  { Mein }
```

Рис. 2. Скриншот программы

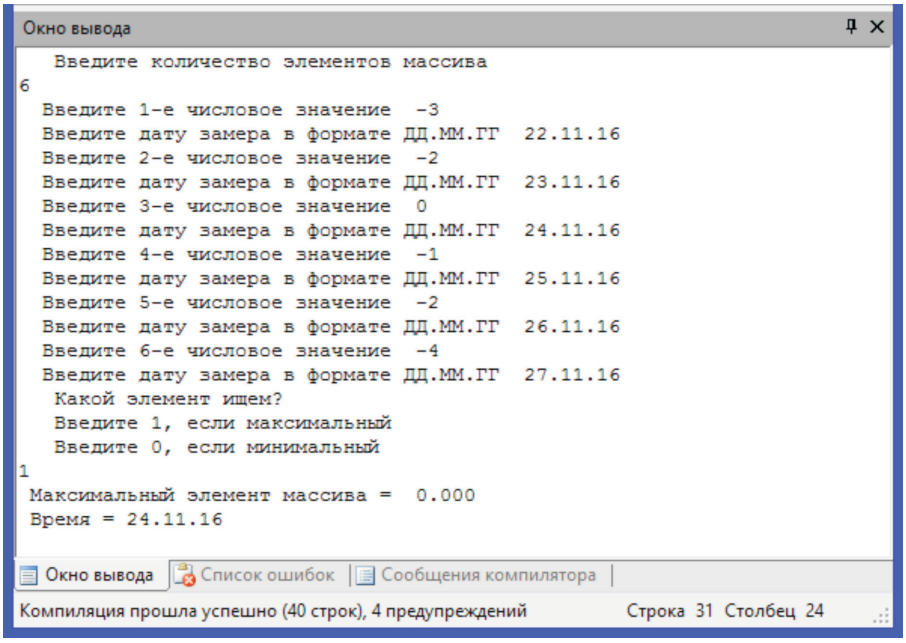

Рис. 3. Скриншот результатов нахождения максимального элемента

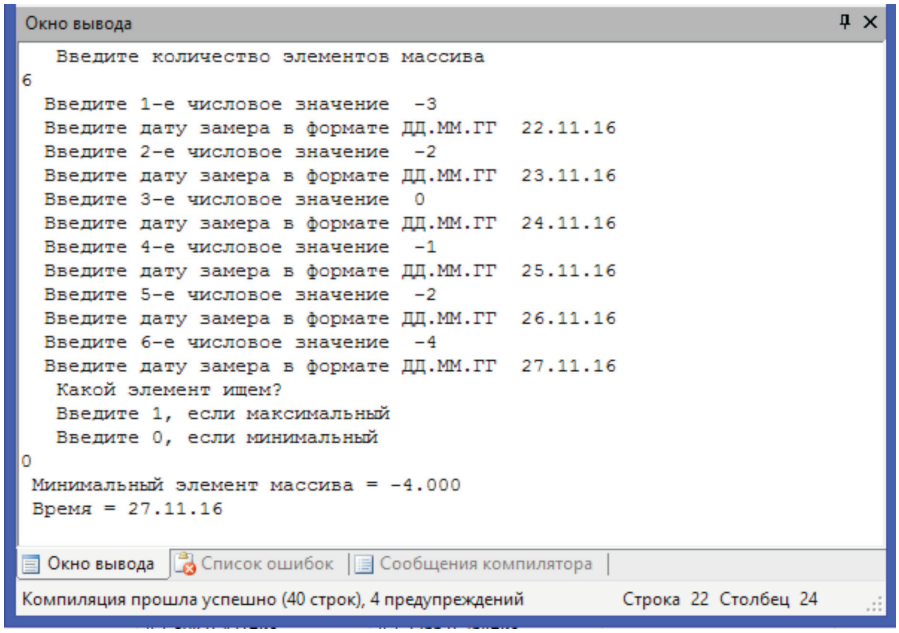

Рис. 4. Скриншот результатов нахождения минимального элемента массива

Таким образом, после завершения перебора элементов массива в дополнительной переменной окажется максимальное (минимальное) значение среди элементов массива.

Кроме этого введены еще две переменные imax и imin, которые будут использоваться для хранения номеров максимального и минимального элементов массива. После выхода из цикла будут найдены значения максимального или минимального элементов массива Dan[ ], а также их номера, по которым будут найдены соответствующие даты в массиве Tim[ ].

Представим разработанный выше алгоритм в виде блок-схемы.

Блок-схемой называется наглядное графическое изображение алгоритма, когда отдельные его этапы изображаются при помощи различных геометрических фигур – блоков, а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок. Типичные действия алгоритма изображаются геометрическими фигурами согласно ГОСТ 19.701-90.

Блок-схема алгоритма представлена на рис. 1.

Интегрированная среда программирования Паскаль-АВС. Разработка и тестирование программы

На рис. 2 представлен скриншот разработанной программы в интегрированной среде программирования Паскаль-АВС.

Система программирования Паскаль-АВС представляет собой диалект стандартного языка Паскаль. Система создавалась на факультете математики, механики и компьютерных наук ЮФУ как учебная среда программирования (автор – кандидат физико-математических наук, доцент кафедры алгебры и дискретной математики С.С. Михалкович) [2]. По мнению разработчиков этой системы, первоначальное обучение программированию должно проходить в достаточно простых и дружественных средах, в то же время эти среды должны быть близки к стандартным и иметь богатые и современные библиотеки подпрограмм.

На рис. 3 представлен скриншот результатов нахождения максимального значения температуры и соответствующей ей дате.

На рис. 4 представлен скриншот результатов нахождения минимального значения температуры в массиве и соответствующей ей дате.

Выводы

В результате выполнения моей научно-исследовательской работы достигнута цель исследования – разработана программа нахождения максимального и минимального элемента в среде программирования Паскаль-АВС. Программа протестирована по двум ветвям вычислительного процесса: нахождение максимального и нахождение минимального элемента. При этом я ознакомился с алгоритмами поиска информации в больших объемах данных. Освоил некоторые приемы программирования в интегрированной среде Паскаль-АВС. Разработал блок-схему и программу нахождения максимального и минимального элемента одномерного массива. Надеюсь, что полученные знания и навыки помогут мне успешно сдать экзамен по дисциплине информатика.

Список литературы

1. Логинов В.И., Шемагина Л.Н. Основы алгоритмизации : учеб.-метод. пособие для студ. оч. заоч. обуч. техн. специальностей. – Н. Новгород : Изд-во ФГОУ ВПО «ВГАВТ», 2010. – 81 с.

2. Михалкович С.С., Логинов В.И., Шемагина Л.Н. Основы программирования : учеб.-метод. пособие для студ. 1 курса. – Ростов н/Д. : Изд-во ЮФУ, 2007. – 40 с.