§9. Поиск элементов с заданными свойствами

При обработке информации постоянно приходится сталкиваться с задачами поиска данных. Эти задачи весьма разнообразны: от поиска телефонного номера или справочных данных до проверки правильного ответа в тестах или угадывание числа в играх. Алгоритмы поиска являются одними из наиболее часто выполняемых алгоритмов.

Пусть, например, в массиве **h** хранится рост учеников 9 класса (рис 2.6).

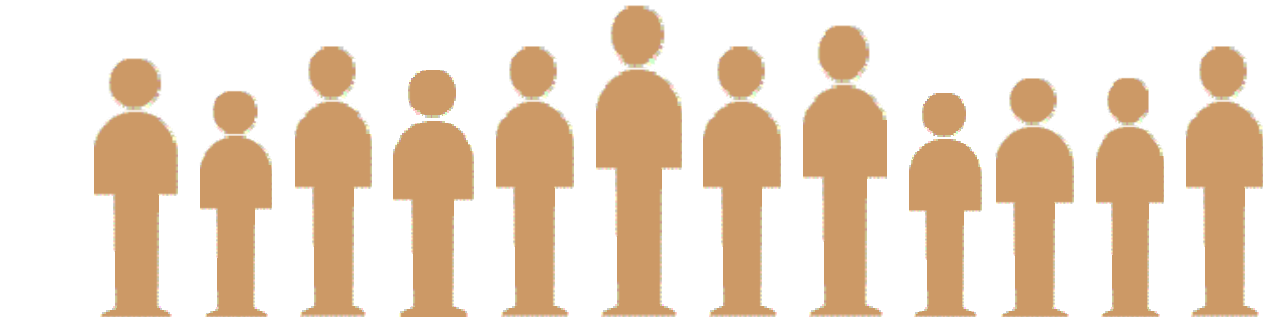


Рис 2.6

Для этого массива можно, например, сформулировать такие задачи поиска:

- Определить, **есть ли хотя бы один** ученик, рост которого равен 172 см (больше 180 см, меньше 160 см).
- Найти номер по списку (**индекс**) ученика, рост которого 172 см.
- Определить, **сколько** учеников в классе имеют рост 170 см (меньше 160 см, больше роста первого по списку ученика).
- Найти номера (**индексы**) всех учеников ростом 170 см (больше 175 см).
- Найти ученика максимального (минимального) роста, т.е. определить его номер и рост.

Обобщая эти задачи можно сказать, что цель поиска заключается в нахождении значений, индексов и количества элементов массива, удовлетворяющих заданным условиям.

В качестве простых условий поиска чаще всего используется сравнение значений элементов массива *X[i]* с заданными числом *B* (например: равенство *X[i] = B)* или неравенства, например: *X[I] < B* или *X[i] >= B.*

Известны различные алгоритмы поиска, каждый из которых эффективен для решения определенного круга задач.

Самый простой способ поиска элементов массива с заданными свойствами – это последовательный просмотр всех элементов и проверка выполнения условий поиска. Такой алгоритм поиска называется **линейным** или **последовательным**.

Рассмотрим примеры решения некоторых задач поиска.

Пример 1. В массиве хранится рост 12 учеников 9 класса (рис 2.6). Составить программу, которая определяет, есть ли в классе хотя бы один ученик с ростом 172 см и выводит его номер по списку.

Это простейшая задача поиска элемента массива, значение которого равно заданному. Алгоритм ее решения сводится к последовательному сравнению значений всех элементов массива с заданным числом.

Массив данных о росте учеников зададим в виде целочисленных констант:

```
const  h: array[1..12] of integer= (160, 154, 170, 160, 172, 181, 170, 175, 152, 157, 155, 168);
```

Результаты поиска будем помещать в переменную *k*. Если условие не выполнено ни разу (элемент не найден), *k:=0*. Если условие выполнено (элемент найден), то переменой *k* присваивается индекс найденного элемента *k:= i* .

Рассмотрим алгоритм поиска.

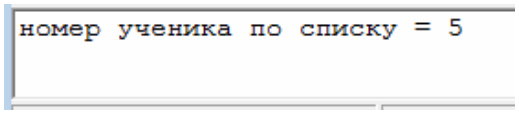
- Присваиваем начальное значение *k:=0*.
- Проверку выполнения условий поиска (сравнение значений) производим в цикле *for*. На каждом шаге цикла от 1 до 12 сравниваем значение очередного элемента *h[i]* и заданного числа (в данном примере 172).
- На первых четырех шагах условие *h[i] = 172* не выполняется, значение переменной *k* остается равным 0. Это условие будет выполнено на пятом шаге. Элемент найден, переменной *k* присваивается значение его индекса *k:=5*. Далее на шагах 6 – 12 это значение *k* сохраняется, поскольку в данном примере искомый элемент единственный.
- По завершению цикла выводится индекс элемента (номер ученика по списку) либо сообщение ‘*ученика с таким ростом нет*’.

Программа выглядит так:

```
program Primer9_1;
const h: array[1..12] of integer= (160, 154, 170, 160, 172, 181, 170, 175,
                                   152, 157, 155, 168);
var i, k :  integer ;
begin
```

```
k:= 0;                                          { начальное значение  k }
for i :=1 to 12 do                              { просмотр массива  }
        if h[i] = 172 then k := i ;
if k>0 then writeln('номер ученика по списку = ', k)        { вывод }
       else writeln('ученика с таким ростом нет');
end.
```

Результат работы будет выглядеть так:



В рассмотренном примере искомый элемент был единственным. Если элементов, удовлетворяющих условиям поиска несколько, то возможны два варианта последующих действий:

1) продолжить выполнение цикла до конца, тогда в *k* будет помещен индекс последнего из найденных элементов;
2) прервать цикл командой *break*, тогда в *k* останется индекс первого из найденных элементов.

Пример 2. В массиве хранится рост учеников 9 класса (рис 2.6). Составить программу, которая подсчитывает, сколько учеников имеют рост больше 170 см.

Массив данных о росте учеников зададим в виде целочисленных констант.

Условием поиска в этой задаче является неравенство *h[i] > 170.* Результат поиска будем помещать в переменную *k* (счетчик найденных элементов). При выполнении условия поиска значение этой переменной увеличивается на 1.

Рассмотрим алгоритм поиска.

- Присваиваем начальное значение *k:=0*.
- Проверку выполнения условий поиска производим в цикле *for*. На каждом шаге цикла от 1 до 12 сравниваем значение очередного элемента *A[i]* и заданного числа (в данном примере 170).
- На четырех шагах условие *A[i] > 170* не выполняется, значение переменной *k* остается равным 0. Это условие будет выполнено на пятом шаге. Первый искомый элемент обнаружен, значение счетчика стало *k:=1.* В дальнейшем условия поиска выполняются еще дважды: на 6 и на 8 шагах. В результате *k := 3.* Далее это значение остается неизменным.
- По завершению цикла выводим значение *k*.

Программа будет выглядеть так:

```
program Primer9_2;
const h: array[1..12] of integer= (160, 154, 170, 160, 172, 181, 170, 175,
                                   152, 157, 155, 168);
var i, k :  integer ;
begin
     k:= 0;                                  { начальное значение  k }
     for i :=1 to 12 do                      { просмотр массива  }
          if h[i] > 170 then k:=k+1;         { подсчет k }
     writeln(' таких учеников ', k );        { вывод k }
end.
```

Протестируем программу. В окне вывода получим сообщение 'таких учеников 3’

Приведем примеры некоторых условий, которые часто используют при поиске элементов:

A[i] > 0; положительных чисел; *A[i] < 0;* отрицательных чисел;

A[i] <> 0; чисел, не равных нулю;

A[i] mod 2 = 0 четных чисел ;

X[I] mod 2 = 1 нечетных чисел;

A[i] mod B = 0 чисел, кратных заданному целому числу *B.*

Пример 3. Составить программу, которая формирует массив из 15 случайных целых чисел от 0 до 50 и выводит на экран нечетные элементы массива и их индексы.

- Сформируем массив случайных чисел *A[i] := random(51).*
- В цикле просмотра массива зададим условие нечетности значения элемента *if A[i] mod 2 = 1* (остаток от целочисленного деления на 2 равен 1).
- Организуем вывод нечетных значений элементов и их индексов:
  writeln(' найден элемент A[‘, i, ‘] = ‘, A[i]);

  Если элемент не найден, никакого сообщения не выводится.

  Для сравнения выведем также все значения сформированного массива.

  Программа будет выглядеть так:

```
program Primer9_3;
var A:  array[1..15] of integer;
    i :  integer ;
begin
     for i := 1 to 15 do A[i] := random(51);        { формирование массива }
     for i :=1 to 15 do                             { просмотр массива}
          if A[i] mod 2 = 1 then                    { проверка условия }
               writeln('найден элемент A[‘ , i , ‘] = ‘, A[i]);   {вывод нечетных}
```

```
    for i:=1 to 15 do write(a[i]:4 );                    { вывод всех чисел }
end.
```

Результат работы может выглядеть так:

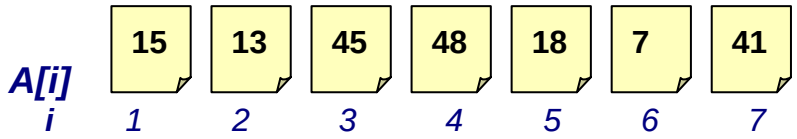
```
найден элемент A[1] = 27
найден элемент A[2] = 23
найден элемент A[5] = 39
найден элемент A[8] = 15
найден элемент A[10] = 9
  27  23  18  20  39   4   2  15  16   9  44  38  44  30  32
```

Очень часто для решения задачи требуется находить максимальный (наибольший) или минимальный (наименьший) элемент массива. Например, определить рост самого высокого ученика в классе, определить какого числа был самый холодный день в декабре.

Пример 4. Составить программу, которая формирует массив из 7 случайных целых чисел от 0 до 50 и осуществляет поиск максимального элемента.

Рассмотрим алгоритм поиска максимального элемента на примере сформированного случайным образом массива *A* из 7 элементов (рис 2.7)

Рис 2.7



Значение максимального элемента будем хранить в переменной *max*, а его индекс в переменной *k*.

- Вначале полагаем, что максимальным является первый элемент: *max:= A[1], k:=1*.
- Простейший алгоритм поиска сводится к последовательному сравнению со значением *max* всех элементов массива *A[i]*, начиная со второго.
- Организуем цикл *for* с параметром *i* , изменяющимся от 2 до 7.
  На каждом шаге цикла проверяем условие *A[i] > max*. Если оно выполняется (очередной элемент больше *max*), то значение этого элемента принимаем в качестве максимального *max := A[i]*, а его индекс *k:=i.*
  В нашем примере новое значение *max* присваивается на третьем шаге (*A[3]>15, max:=45, k:=3*) и на четвертом (*A[4]>45, max:=4, k:=4*).
- После перебора всех элементов массива переменная *max* будет содержать значение максимального элемента массива, а переменная *k* – его индекс.

Программа может выглядеть так:

```
program Primer9_4;
var A:  array[1..7] of integer;
      max, k, i :  integer ;
begin
      for I := 1 to 7 do A[i] := random(51);        { формирование массива }
      max:=A[1];  k:=1;                             { начальные значения  max и  k }
      for  i :=2 to 7 do                            { просмотр массива }
        if A[i] > max then                          { условие поиска максимума }
              begin
                  max:=A[i];   k:=i;                { присвоение значений max и k }
              end;
      writeln(' максимальный элемент A[' , k , ' ] = ', max);    { вывод  max }
      for i:=1 to 7 do write(a[i]:4 );              { вывод всех чисел }
end.
```

Результат работы может выглядеть так:

```
максимальный элемент A[4] = 48
 15  13  45  48  18   7  41
```

Если в массиве несколько элементов имеют максимальное значение, то в переменной *k* будет запоминаться индекс первого из них. Для запоминания индекса последнего из максимальных элементов необходимо использовать условие *A[i] >= max*.

Для поиска минимального элемента необходимо заменить знак “>” больше в условии оператора ветвления на знак “<” меньше.

Заметим, что в алгоритме нахождения максимального или минимального элемента можно искать индекс этого элемента и не использовать промежуточную переменную.

Пример 5. В массив с клавиатуры вводятся результаты соревнований по бегу (значения времени от 50 до 100 с). Составить программу, которая определяет номер и время победителя.

Результаты 12 участников соревнований будем вводить с клавиатуры в массив *A[i]*. Номер победителя (индекс участника, показавшего минимальное время) будем хранить в переменной *k*. Тип всех переменных *integer*.

- До начала просмотра массива считаем, что минимальное время показал участник номер 1. Присвоим *k:=1.*
- В цикле просмотра массива зададим условие поиска индекса участника с минимальным временем: *if A[i] < A[k] then k:=i;*.
- По окончанию цикла выведем результаты всех участников, номер и время победителя.

Программа будет выглядеть так:

```
program Primer9_5;
var A:  array[1..12] of integer;
    i, k :  integer ;
begin
    for i:=1 to 12  do read(A[i]);          { ввод результатов в массив  }
    k:=1;                                   { начальное значение  k }
    for  i :=2 to 12 do                     { просмотр массива }
      if A[i] < A[k] then k:=i;             { индекс минимального элемента}
    writeln('победил участник номер ', k,  ' с результатом ',A[k]);
end.
```

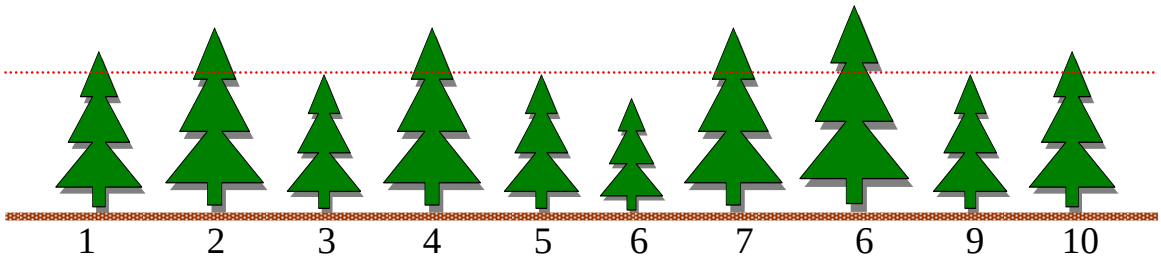
Результат работы может выглядеть так:

```
68 74 93 67 87 84 79 65 72 92 75 82
победил участник номер 8 с результатом 65
```

?

1. Какой алгоритм поиска называют последовательным?

1. На елочном базаре случайным образом берут несколько елок. Их высота в метрах заносится в массив вещественных чисел. Какие задачи поиска могут быть сформулированы для этих данных?.



Упражнения

1. Составьте программу, которая формирует массив из 16 случайных целых чисел от -20 до 20, и выполняет одну из следующих операций поиска:

а) определяет, есть ли в этом массиве положительные числа;

б) находит элементы с нулевыми значениями;

в) находит индексы четных чисел;

г) подсчитывает количество отрицательных чисел;

д) определяет, есть ли в этом массиве хотя бы одно число, кратное трем;

е) находит в этом массиве индексы элементов, значения которых кратны трем;

На экран выводятся исходный массив и найденные элементы.

2. В массиве хранятся сведения о количестве осадков (в мм), выпадавших ежедневно в сентябре. Составьте программу, которая подсчитывает количество дождливых дней.

3. В массиве хранится информация о среднедневной температуре декабря. Составьте программу, которая подсчитывает сколько в декабре было дней с нулевой, отрицательной и положительной температурой.

4. В массиве хранится информация о стоимости товаров. Составьте программу, которая определяет стоимость самого дешевого (дорогого) товара и его индекс.