**Информатика 20 марта 11А класс.**

**!!!Задание отмечено жёлтым цветом. Буду проверять после каникул!!!!**

## Поиск элемента в упорядоченном массиве

Чтобы понять идею, положенную в основу способа быстрого поиска элемента в упорядоченном массиве обратимся к простому примеру.

В классе 32 ученика. Учитель загадал одного из них. Как выяснить, кого именно? Конечно, можно взять классный журнал и начать читать по списку фамилии учеников и каждый раз спрашивать учителя тот ли это ученик, которого он загадал. Хорошо, если фамилия ученика находится среди первых фамилий по списку, а если в конце? При таком способе мы можем сделать даже 32 испытания (если фамилия в конце списка). Можно ли уменьшить число испытаний? Можно!

Воспользуемся таким приемом. Допустим, что учитель задумал фамилию ученика под номером 18 (но нам это неизвестно).

Начнем задавать учителю вопросы, а учитель должен нам отвечать только "**да**" или "**нет**".

***Первый вопрос***. Загаданный номер меньше 16? ***Ответ***: нет.

Мы делаем вывод, что задуманная фамилия находится в списке от 17-го номера до 32. И тогда задаем следующий вопрос.

***Второй вопрос***. Загаданный номер меньше 24? ***Ответ***: да.

***Третий вопрос***. Загаданный номер меньше 20? ***Ответ***: да.

***Четвертый вопрос***. Загаданный номер меньше 18? ***Ответ***: нет.

***Пятый вопрос***. Загаданный номер больше 18? ***Ответ***: нет.

**Делаем вывод**. Загаданный номер 18. Таким образом, всего за 5 шагов нам удалось установить загаданный номер. А если бы мы действовали по порядку номеров, тогда пришлось бы выполнить 18 проб. Как видите мы сократили число испытаний более чем в 3 раза!

На этом принципе основан способ быстрого поиска элемента в упорядоченном массиве.

***Пример 1***. Задача поиска элемента в упорядоченном массиве.

Пусть задан целочисленный массив  и задано целое число b. Выяснить входит ли число b в заданный массив и если входит, то каково значение номера элемента равного данному числу, т. е. найти такое p, для которого 

*Первая мысль*, которая возникает - это поочередно сравнивать заданное число **b** с элементами массива, начиная с первого и продолжать это до тех пор, пока не "*наткнемся*" на равный числу **b** элемент.

Если окажется, что после просмотра и сравнения всех элементов массива равного числу **b** не найдется, тогда выдать сообщение, что элемента равного числу **b** в массиве нет.

Но оказывается, что есть способ, который дает возможность значительно быстрее отыскать равный числу **b** элемент (или выяснить, что такого элемента нет).

Этот способ, следуя С.А. Абрамову (см. С.А. Абрамов и Е.В. Зима "Начала информатики", Москва, "Наука", 1990 г.), будем называть "***разделяй и властвуй***".

***Идея*** этого способа очень проста (будем считать, что массив упорядочен по неубыванию, т.е. 

Перед ее изложением, надо выяснить, какой элемент мы будем считать "***средним***" (в дальнейшем без кавычек)?

Если в массиве нечетное число элементов, то это понятно и просто. ***Средний*** он и в самом деле будет находиться посередине массива. Например, если число элементов 17, то ***средним*** будет элемент под номером 9. И действительно он средний, слева от него 8 элементов и справа тоже 8.

А если число элементов - четное? Например, число элементов массива - 20. Тогда в качестве среднего элемента будем считать элемент под номером, который получается от целочисленного деления суммы 1 + 20 на 2, т. е. (1 + 20) **div** 2 = 10. Для массива с числом элементов 124 средним будет элемент под номером 62 (125 делим нацело на 2) и т.п.

Теперь сама ***идея***.

1. Выбирается средний элемент массива и сравнивается с заданным числом **b**.

После чего могут представиться два возможных случая.

***Случай 1***. Выбранный средний элемент меньше числа **b**, тогда нет смысла рассматривать левую часть массива (там равного элемента уже точно не будет), а остается искать равный элемент среди второй, правой половины элементов.

***Случай 2***. Выбранный средний элемент больше или равен заданному числу **b** и, тогда нет смысла рассматривать правую часть массива, а поиск продолжается среди элементов левой части.

Таким образом сразу сокращается число испытаний вдвое.

2. С той половиной массива, в которой установлено, что может быть равный элемент, проделывается та же процедура деления.

3. Процесс такого деления продолжается. Наконец наступает такой момент, что в выбранной части массива остается один элемент (т.е. номера элементов слева и справа станут равными), тогда он является искомым, либо (если он не равен заданному числу) такого элемента вообще нет в массиве.

Рассмотрим несколько *частных примеров*, чтобы уразуметь математику этого вопроса.

***Пример 1***. Пусть задан массив из 10 упорядоченных по неубыванию чисел:

23 34 45 48 56 63 67 72 78 89

 1 2 3 4 5 6 7 8 9 10

Надо найти элемент этого массива, равный числу 78.

Пусть **p** - номер элемента левой границы массива, **q** - номер элемента правой границы массива, **s** - номер *среднего* элемента.

Устанавливаем средний элемент. Им будет элемент под номером  (знак "\" - знак целочисленного деления), т. е. 56.

Сравниваем этот элемент с числом 78: 56 < 78.

Значит искомый элемент находится справа от 56.

Рассматриваем правую половину заданного массива, т.е. элементы, находящиеся справа от 56:

 63 67 72 78 89

 6 7 8 9 10

Тогда, p = 6, q = 10.

Выбираем и в нём средний элемент. Им будет элемент под номером

s = (p + q)\2 = (6 + 10)=8, равный 72.

Сравниваем: 72 < 78

Значит, снова искомый элемент будет находится справа от 72, т.е. в следующей части массива:

 78 89

 9 10

p = 9, q = 10.

Находим среди них средний элемент. Им будет элемент под номером

s = (p + q)\2 = (9 + 10)\2 = 9.

Сравниваем: 78 < 78.

Условие не выполняется. Значит надо брать элементы массива слева от числа 78.

Тогда, p = 9, q = 9. Номера левой и правой границ стали равны, цикл заканчивается, искомым элементом будет элемент под номером 9, равный 78.

В самом деле: 78 = 78.

Еще один пример. Задан массив из 15 элементов:

7 9 11 12 15 18 20 24 34 42 45 67 78 89 98

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Надо найти элемент массива, равный числу 9.

p = 1, q = 15, s = (p + q)\2 = (1 + 15)\2 = 8.

Средним является элемент под номером 8, равный 24.

Сравниваем: 24 < 9.

Условие не выполняется, значит будем искать элемент в левой части массива, т.е. ищем его среди элементов:

7 9 11 12 15 18 20 24

1 2 3 4 5 6 7 8

p = 1, q = 8, s =(p + q)\2 = (1 + 8)\2 = 4.

Средним элементом будет элемент под номером 4, равный 12.

Сравниваем: 12 < 9.

Условие не выполняется. Снова надо рассматривать левую часть оставшихся элементов, т.е. элементы, расположенные слева от 12 (включая 12):

7 9 11 12

1 2 3 4

p = 1, q = 4, s = (p + q)\2 = (1 + 4)\2 = 2.

Средним элементом будет элемент под номером 2, равный 9.

Сравниваем: 9 < 9.

Условие не выполняется.

Рассматриваем элементы слева от 9 (включая 9):

7 9

1 2

p = 1, q = 2, s = (p + q)\2 = (1 + 2)\2 = 1.

Средним является элемент под номером 1, равный 7.

Сравниваем: 7 < 9.

Условие выполняется, значит надо рассматривать элемент справа от 7, т.е. 9.

9

2

p = 2, q = 2, p = q, цикл заканчивается, искомый элемент под номером 2, равный 9.

Вы, конечно, обратили внимание, рассматривая эти два примера, что при движении вправо, т. е., когда рассматриваем правую часть массива, то граничный элемент, он же бывший средний (с которым сравниваем), в эту часть массива не включается, а при движении влево, средний элемент включается в оставшуюся часть.

Так, в первом примере, после первого шага, когда был определен средний элемент 56, сравнив его с числом 78 (56 < 78), мы

убеждаемся, что неравенство выполняется и надо рассмотреть правую часть массива. И далее рассматриваем ее, но без среднего элемента 56:

 63 67 72 78 89

 6 7 8 9 10

Во втором примере после нахождения среднего элемента (20) и сравнения его с заданным числом - 9 (20 < 9), мы убеждаемся, что неравенство не выполняется и рассматриваем левую часть массива вместе со средним элементом 20:

7 9 11 12 15 18 20

1 2 3 4 5 6 7

Почему такая разница? Почему в первом случае средний элемент не включается в рассматриваемую часть массива, а во втором случае - включается?

Для выяснения этого вопроса попробуем выполнить второй пример, не включая граничный элемент в левую часть массива.

После первого шага придется рассматривать часть массива:

7 9 11 12 15 18 20

1 2 3 4 5 6 7

После второго деления часть массива:

7 9 11

1 2 3

После третьего деления:

7

1

И тогда окажется, что нужный элемент пропущен. Отчего это происходит?

Понятно почему. По результатам любого сравнения  мы сразу массив разбиваем на две части: от 1 до s и от s + 1 до n. Исключив элемент массива под номером s, т. е. средний элемент, мы его вообще выбрасываем из массива. Он не попадает ни в левую, ни в правую части.

***Алгоритм***

Первоначально в качестве границ - левой и правой берутся значения 1 и n. Пусть p и q обозначают левую и правую границы, тогда p := 1, а q := n; далее производится деление и шаг за шагом границы сближаются. Процесс деления должен продолжаться до совпадения границ, когда p = q (т. е. цикл выполняется пока Границы сближаются следующим образом: пусть s - номер среднего элемента, значение s - это целая часть среднего арифметического границ p и q 

**если  тогда** надо изменить прежнюю нижнюю границу на   а верхняя граница (q) остается без изменения (выбирается правая часть массива), **иначе**, оставить без изменения нижнюю границу (p), а верхнюю заменить на s 

Теперь можно записать этот алгоритм в виде последовательности операторов:

p := 1; q := n;

**while** p < q **do**

 **begin**

 s := (p + q) **div** 2;

 **if** a[s] < b **then** p := s + 1 **else** q := s

**end**;

Может случиться, что элемента массива, равного заданному числу b в массиве нет. Это обстоятельство надо учесть и добавить следующие операторы

**if** a[p] = b **then** writeln('Число ', b, ' входит в масс. и равно ', p, '-му элементу')

 **else** writeln('Такого элемента в массиве нет')

***Программа***

**Program** Problem1;

 **uses** WinCrt;

 **const**

 n = 20;

 **type**

 t = **array**[1..n] **of** integer;

 **var**

 a : t;

 p, q, s, b, i : integer;

 **begin**

 writeln('Вводите элементы упорядоченного по неубыванию массива');

 **for** i := 1 **to** n **do**

 **begin**

 write('Введите ', i, '-й элемент '); readln(a[i])

 **end**;

 writeln;

 write('Введите число b '); readln(b);

 p := 1; q := n;

 **while** p < q **do**

 **begin**

 s := (p + q) **div** 2;

 **if** a[s] < b **then** p := s + 1

 **else** q := s

 **end**;

 **if** a[p ]= b

 **then** writeln('Число ', b, ' равно ', p, '-му элементу')

 **else** writeln('Такого элемента в массиве нет')

 **end**.

***Задание 1***

**1**. Измените программу поиска места элемента, предполагая, что числа  упорядочены по убыванию  и выполните ее несколько раз, задавая различные значения для числа **b**.

**2**. Идея деления пополам ("***разделяй и властвуй***") может быть положена в основу алгоритма решения еще одной задачи, похожей на задачу поиска элемента. Пусть  и пусть **b** - некоторое число. Для числа **b** имеется n + 1 возможность: 

Требуется определить, какая из возможностей имеет место. Ответом может быть одно из чисел 1, 2, ..., n, n + 1 (порядковый номер возможности).