Одномерные массивы

Алгоритмы поиска элемента массива

1. Линейный поиск.

<u>Алгоритм.</u>

Последовательно просматриваем массив и сравниваем значение очередного элемента с данным, если значение очередного элемента совпадет с X, то запоминаем его номер в переменной k.

For i := I **to** n **do if** a[i] = x **then** k := i; Недостатки данной реализации алгоритма:

- находим только последнее вхождение элемента
- в любом случае производится п сравнений



Улучшим: будем прерывать поиск, как только найдем элемент:

while $(i \le n)$ and $(a[i] \le x)$ do inc(i);

В результате или найдем нужный элемент, или просмотрим весь массив.

Недостаток данной реализации:

в заголовке цикла сложное условие, что замедляет поиск.



2. Бинарный поиск

Применяется для отсортированных массивов!!!!!!!.

Задача. Дано X и массив A(n), отсортированный по неубыванию Найти і, такой что a[i] = x или сообщить что данного элемента в массиве нет.



Алгоритм

- Является ли X средним элементом массива. Если да, то поиск завершен, иначе переходим к пункту 2.
- Возможно 2 случая:
- а) Х меньше среднего, тогда так как А упорядочен, то из рассмотрения можно исключить все элементы массива, расположенные правее среднего и применить метод к левой половине массива.
- Х больше среднего. Значит, исключаем из рассмотрения левую половину массива и применяем метод к правой части.



```
begin
I := 1; r := n; {на первом шаге рассматриваем весь массив}
 f := false; {признак того, что X не найден}
 while ( | <= r ) and not f do
 begin
   m := (l+r) div 2;
   if a[m] =x then f := true {элемент найден! Поиск прекращаем}
    else if x < a[m] then r := m-1 \{ \text{отбрасываем правую часть} \}
       else I := m + 1 {отбрасываем левую часть}
 end;
```

