15. Алгоритмы поиска

Цель поиска состоит в нахождении элемента, имеющего заданное значение ключевого поля.

15.1. Линейный поиск

Линейный поиск используется в случае, когда нет никакой дополнительной информации о местоположении разыскиваемых данных. Он представляет собой последовательный перебор массива до обнаружения требуемого ключа или до конца, если ключ не обнаружен:

```
int p_lin1(tmas a[], int  n, int x)
{
for(int i=0; i < n; i++)
      if (a[i].key == x) return i;
  return -1;
}
```

В данном алгоритме на каждом шаге делается две проверки: проверка на равенство ключевого поля и искомого ключа и проверка условия продолжения циклического алгоритма. Для исключения проверки условия продолжения циклического алгоритма вводится вспомогательный элемент – *барьер*, который предохраняет от выхода за пределы массива:

```
int p_lin2(tmas a[], int n, int x)
{
 a[n+1].key = x;
  int i = 0;
 while (a[i].key != x) i++;
      if (i == n+1) return -1;
            else return i;
}
```

Эффективность такого алгоритма почти в два раза выше предыдущего.

15.2. Поиск делением пополам

Поиск деления пополам используется, когда данные упорядочены, например, по неубыванию ключевого поля. Алгоритм состоит в последовательном исключении той части массива, в которой искомого элемента быть не может. Для этого берется средний элемент, и если значение ключевого поля этого элемента больше, чем значение искомого ключа, то можно исключить из рассмотрения правую половину массива, иначе исключается левая половина мас-

сива. Процесс продолжается до тех пор, пока в рассматриваемой части массива не останется один элемент.

```
int p_dv(tmas a[], int n, int x)
{
 int i=0, j=n-1, m;
while(i < j)
{
m=(i + j)/2;
if (x > a[m].key) i = m+1; else j = m;
}
if (a[i].key == x) return i;
else return -1;
}
```

15.3. Интерполяционный поиск

Для массивов с равномерным распределением элементов можно использовать формулу, позволяющую определить примерное местоположение элемента:

$$m = i + \frac{(i - j)(x - a[i].key)}{a[i].key - a[j].key},$$

где i, j − начало и конец интервала; x − искомое значение ключевого поля.

```
int p_dv(tmas a[], int n, int x)
{
 int i = 0, j = n-1, m;
while(i < j)
{
 if (a[i].key == a[j].key) // предотвращение деления на нуль
    if (a[i].key == x) return i;
                  else return -1;
      m=i + (j - i) * (x - a[i]) / (a[j] - a[i]);
if (a[m].key == x) return m;
  else
    if (x > a[m].key) i = m+1; else j = m-1;
}
 return -1;
}
```

Данный поиск быстрее двоичного в 3–4 раза, однако вблизи ключевого поля может вести себя неустойчиво. Поэтому обычно несколько шагов делают с использованием интерполяционного поиска, а затем используют двоичный поиск.