**9.3. TÌM KIẾM NHỊ PHÂN (BINARY SEARCH)**

Phép tìm kiếm nhị phân có thể áp dụng trên dãy khoá đã có thứ tự: k1  k2  …  kn.

Giả sử ta cần tìm trong đoạn kinf, kinf+1, …, ksup với khoá tìm kiếm là X, trước hết ta xét khoá nằm giữa dãy kmedian với median = (inf + sup) div 2;

Nếu kmedian < X thì có nghĩa là đoạn từ kinf tới kmedian chỉ chứa toàn khoá < X, ta tiến hành tìm kiếm tiếp với đoạn từ kmedian + 1 tới ksup.

Nếu kmedian > X thì có nghĩa là đoạn từ kmedian tới ksup chỉ chứa toàn khoá > X, ta tiến hành tìm kiếm tiếp với đoạn từ kinf tới kmedian - 1.

Nếu kmedian = X thì việc tìm kiếm thành công (kết thúc quá trình tìm kiếm).

Quá trình tìm kiếm sẽ thất bại nếu đến một bước nào đó, đoạn tìm kiếm là rỗng (inf > sup).

{Tìm kiếm nhị phân trên dãy khoá k1  k2  …  kn; hàm này thử tìm xem trong dãy có khoá nào = X không, nếu thấy nó trả về chỉ số của khoá ấy, nếu không thấy nó trả về 0}

function BinarySearch(X: TKey): Integer;

var

inf, sup, median: Integer;

begin

inf := 1;

sup := n;

while inf  sup do

begin

median := (inf + sup) div 2;

if kmedian = X then

begin

BinarySearch := median;

Exit;

end;

if kmedian < X then inf := median + 1

else sup := median - 1;

end;

BinarySearch := 0;

end;

Người ta đã chứng minh được độ phức tạp tính toán của thuật toán tìm kiếm nhị phân trong trường hợp tốt nhất là O(1), trong trường hợp xấu nhất là O(log2n) và trong trường hợp trung bình cũng là O(log2n). Tuy nhiên, ta không nên quên rằng trước khi sử dụng tìm kiếm nhị phân, dãy khoá phải được sắp xếp rồi, tức là thời gian chi phí cho việc sắp xếp cũng phải tính đến. Nếu dãy khoá luôn luôn biến động bởi phép bổ sung hay loại bớt đi thì lúc đó chi phí cho sắp xếp lại nổi lên rất rõ làm bộc lộ nhược điểm của phương pháp này.