SEARCHING

Struktur Data

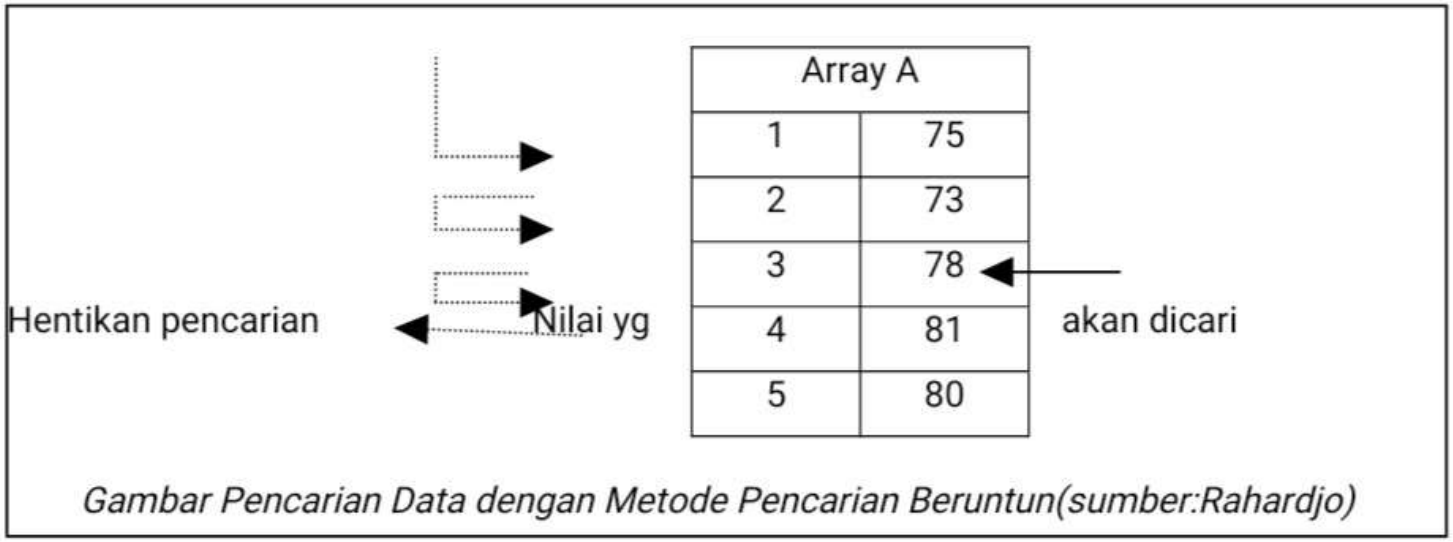
Definisi Pencarian (Searching)

- Pencarian (*searching*) data tertentu yang terkandung di dalam array merupakan hal yang banyak ditemukan dalam kasus-kasus pemrograman. Maka dari itu, pada bagian ini kita akan membahas mengenai cara yang seharusnya digunakan untuk melakukan hal tersebut serte pengimplementasiannya di dalam bahasa Pascal. Dalam ilmu algoritma, metode pencarian data di dalam array diklasifikasikan menjadi dua, yaitu metode pencarian beruntun (*sequential search*) dan metode pencarian bagi dua/pencarian biner (*binary search*).

Metode Pencarian Beruntun (sequential search)

- Metode ini banyak digunakan karena
  - Efektif untuk melakukan pencarian dari sekumpulan data, baik data sudah terurut maupun yang belum terurut atau masih acak.
  - Memiliki cara kerja yang relatif mudah untuk dipahami.
  - Data yang dicari akan dibandingkan dengan seluruh elemen array yang ada. Sebagai contoh, apabila kita memiliki array A yang memiliki indeks 1 sampai n dan kita akan mencari nilai x di dalam array tersebut, maka nilai x tersebut akan dibandingkan dengan nilai A[1] sampai A[n].
  - Diterapkan bahwa apabila data ditemukan pada indeks tertentu, maka proses pencarian akan dihentikan. Hal ini bertujuan agar proses pembandingan nilai tidak dilakukan sampai indeks terakhir karena nilai yang dicari telah

Gambar Pencarian Data dengan Metode Pencarian Beruntun



Gambar Pencarian Data dengan Metode Pencarian Beruntun(sumber:Rahardjo)

- Pada gambar di atas, nilai 1, 2, …5 merupakan indeks array sedangkan nilai 75, 73, … 80 merupakan nilai yang terkandung dalam elemen-elemen array. Sekarang misalkan kita akan melakukan pencarian data 78 di dalam array tersebut, maka salah satu cara yang dapat digunakan adalah dengan membandingkan nilai 78 tersebut dengan seluruh elemen array (dari A[1] sampai A[5]). Mula-mula kita akan membandingkan nilai 78 dengan elemen pertama (A[1]), karena nilainya tidak sama maka pencarian akan dialnjutkan ke elemen berikutnya (A[2]), begitu seterusnya. Pada saat elemen ketiga, nilai A[3] sama dengan nilai yang dicari pencarian akan dihentikan.

CONTOH PROGRAM BERUNTUN

```
Program CariBeruntun;
Uses crt;
Const
        A : array [1 . . 5] of integer = (75, 73, 78, 81, 80);
Var
        i, x, indeks : integer;
begin
        clrscr;
        write ('Masukkan nilai yang akan dicari :  '); readln(x);
        indeks := 0;
        for i := 1 to 5 do begin
        if A[i] = x  then begin
        indeks := I;
        break;
        end;
        end;
        writeln (x, 'ditemukan pada indeks ke-', indeks);
        readln;
end.
```

Contoh hasil :

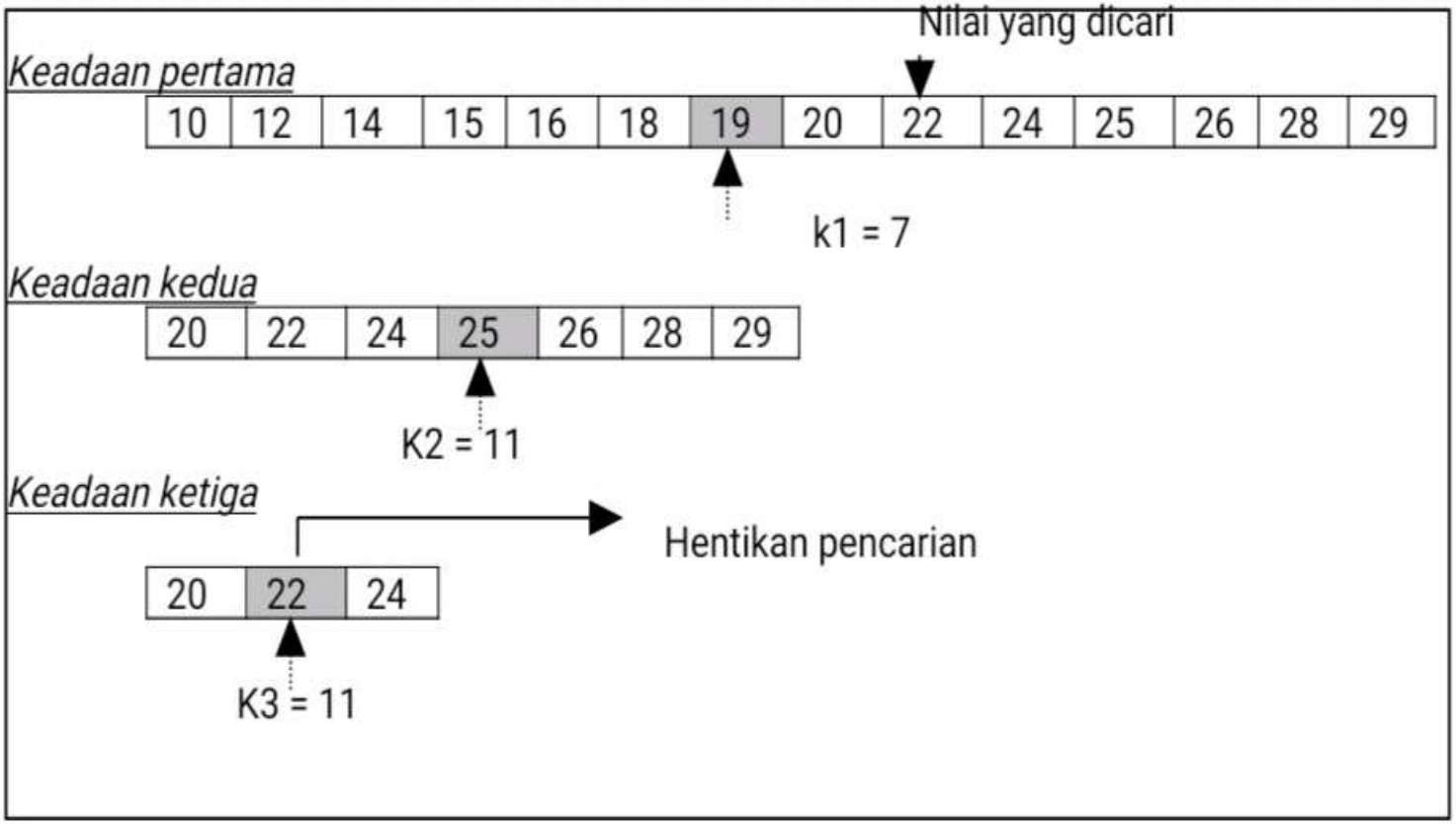
Masukkan nilai yang akan dicari : 78

78 ditemukan pada indeks ke-3

Metode Pencarian Bagi Dua (binary search)

- Berbeda dengan metode pencarian beruntun yang dapat digunakan untuk data belum terurut, metode pencarian bagi dua ini hanya dapat digunakan untuk data-data yang telah terurut, baik secara menaik maupun menurun.
- Dalam metode ini elemen array akan dibagi menjadi dua, sehingga banyaknya proses pembandingan nilai dapat dikurangi. Sebagai contoh, apabila terdapat array A yang memiliki n buah elemen dengan indeks 1 sampai n dan datanya telah terurut secara menaik, maka array tersebut akan dipenggal pada indeks ke-k, dimana k = n + 1 div 2. hal tersebut mengakibatkan array A terbagi menjadi dua bagian, yaitu dari A[1] . . A[k-1] dan A[k+1] . . A[n], sedangkan a[k] menjadi pemenggal atau pembatas antara dua bagian tersebut. Apabila x (nilai yang dicari) sama dengan nilai A[k] maka hentikan pencarian, sedangkan bila tidak, periksa apakah nilai A[k] > x ataukah A[k] < x. Bila A[k] lebih besar dari x, maka ulangi metode pencarian tersebut untuk A[1] sampai A[k-1]. Sebaliknya, apabila A[k] lebih kecil dari x, maka ulangi metode pencarian tersebut untuk A[k+1] sampai A[n].

GAMBAR PENCARIAN DATA DENGAN METODE PENCARIAN BAGI DUA



Gambar Pencarian Data dengan Metode Pencarian Bagi Dua(sumber:Rahardjo)

- Pada gambar di atas, array terdiri dari 14 buah elemen yang sudah terurut secara menaik dengan indeks 1 sampai 14. Mula-mula (keadaan pertama), array akan dibagi menjadi dua bagian. Pembatasnya adalah indeks ke-7. Nilai 7 didapat dari (1+14) div 2. Karena nilai pada indeks ke-7 (nilai 19) lebih kecil dari nilai yang dicari (nilai 22), maka proses pencarian akan diulang untuk indeks ke-8 sampai ke-14. Pada keadaan kedua ini array tersebut juga akan dibagi menjadi dua. Kali ini pembatasnya adalah indeks ke-11, yang berasal dari (8+4) div 2. Karena nilai pada indeks ke-11 (nilai 25) lebih besar dari nilai yang dicari (niali 22), maka proses pencarian akan dilakukan lagi untuk indeks ke-8 sampai ke-10. Pada keadaan ini (keadaan ketiga), array akan dibagi menjadi dua pada indeks ke-9, yang berasal dari (8+10) div 2. Karena nilai pada indeks ke-9 sama dengan nilai yang dicari, maka proses pencarian pun dihentikan.

CONTOH PROGRAM BAGI DUA

- Program CariBagiDua;
- Uses crt;
- Const
- A : array [1 . . 14] of integer = (10,12,14,15,16,18,19,20,22,24,25,26,28,29);
- Var
- idxAwal, *{indeks array awal}*
- idxAkhir, *{indeks array akhir}*
- k, *{indeks pemenggal/pembatas}*
- x : integer; *{nilai yang dicari}*
- ketemu : boolean;*{variabel status, ditemukan atau tdak ?}*
- begin
- clrscr;
- write (‘masukkan nilai yang akan dicari : ‘); readln(x);
-
- *{melakukan pencarian}*
- idxAwal := 1;
- idxAkhir := 14; *{14 adalah jumlah elemen array A}*
- ketemu := false;
- while (not ketemu) and (idxAwal<= idxAkhir) do begin

```
k:= (idxAwal + idxAkhir) div 2;
if A[k] = x then begin
ketemu := true;
end else begin
if A[k] < x then begin
idxAwal := k + 1;          {mencari di bagian kanan}
end else begin
idxAkhir := k – 1;         {mencari di bagian kiri}
end;
end;
end;

 {memeriksa, ketemu atau tidak}
if ketemu then begin
writeln (x, ‘ ditemukan pada indeks ke-‘, k);
end else begin
writeln(x, ‘ tidak ditemukan’);
end;
readln;
end.
```

Contoh hasil :

Masukkan nilai yang akan dicari : 22

22 ditemukan pada indeks ke-9

- **Latihan :**

  Lakukan pencarian nilai 25, dengan menggunakan metode bagi dua untuk data 5,8,10,15,19, 22,25,30,33,38,42,45,50