

TWO

SEARCHING ARRAY

REFRESH ARRAY

- Selama ini kita menggunakan satu variabel untuk menyimpan 1 buah nilai dengan tipe data tertentu.
- Misalnya :
`int a1, a2, a3, a4, a5;`
Deklarasi variabel diatas digunakan untuk menyimpan 5 data integer dimana masing-masing variabel diberi nama a1, a2, a3, a4, dan a5.
- Jika kita memiliki 10 data, 100 data integer bahkan mungkin data yang ingin kita proses tidak kita ketahui atau bersifat dinamis? Kita tidak mungkin menggunakan variabel seperti diatas.
- Di dalam C dan pemrograman yang lain, terdapat suatu fasilitas untuk menyimpan data-data yang bertipe data sama dengan suatu nama tertentu.

DEFINISI ARRAY

- Array adalah suatu tipe data terstruktur yang berupa sejumlah data sejenis (bertipe data sama) yang jumlahnya bisa statis ataupun dinamis dan diberi suatu nama tertentu.
- Elemen-elemen array tersusun secara berderet dan sekuensial di dalam memori sehingga memiliki alamat yang besebelahan/berdampingan.
- Array dapat berupa array 1 dimensi, 2 dimensi, bahkan n-dimensi.
- Elemen-elemen array bertipe data sama tapi bisa bernilai sama atau berbeda-beda.

0	1	2	3	4	5	6	7	indeks
8	10	6	-2	11	7	1	100	value
ffea	ffeb	ffec	ffed	ffef	fffa	fffb	fffc	alamat

CARA PENGAKSESAN ELEMEN ARRAY

- Elemen-elemen array dapat diakses oleh program menggunakan suatu indeks tertentu
- Pengaksesan elemen array dapat dilakukan berurutan atau random berdasarkan indeks tertentu secara langsung.
- Pengisian dan pengambilan nilai pada indeks tertentu dapat dilakukan dengan mengeset nilai atau menampilkan nilai pada indeks yang dimaksud.
- Dalam C, tidak terdapat error handling terhadap batasan nilai indeks, apakah indeks tersebut berada di dalam indeks array yang sudah didefinisikan atau belum. Hal ini merupakan tanggung jawab programmer. Sehingga jika programmer mengakses indeks yang salah, maka nilai yang dihasilkan akan berbeda atau rusak karena mengakses alamat memori yang tidak sesuai.

DEKLARASI ARRAY 1 DIMENSI

```
tipe_data nama_var_array[ukuran];
```

tipe_data : menyatakan jenis tipe data elemen larik (int, char, float, dll)

nama_var_array : menyatakan nama variabel yang dipakai.

ukuran : menunjukkan jumlah maksimal elemen larik.

Contoh:

```
char huruf[9];  
int umur[10];  
int kondisi[2] = {0,1}  
int arr_dinamis[] = {1,2,3}
```

`char huruf[9];` berarti akan memesan tempat di memori komputer sebanyak 9 tempat dengan indeks dari 0-8, dimana semua elemennya bertipe data karakter semuanya. Kalau satu karakter berukuran 1 byte, berarti membutuhkan memori sebesar 9 byte.

`int umur[10];` berarti akan memesan tempat di memori komputer sebanyak 10 tempat dengan indeks dari 0-9, dimana semua elemennya bertipe data integer semuanya. Kalau satu integer berukuran 4 bytes, berarti membutuhkan memori sebesar $4 \times 10 = 20$ bytes.

`int kondisi[2];` berarti akan memesan tempat di memori komputer sebanyak 2 tempat dengan indeks 0-1, dimana semua elemennya bertipe data integer semuanya. Dan pada contoh di atas isi elemen-elemennya yang sebanyak 2 buah diisi sekaligus (diinisialisasi) yaitu pada elemen `kondisi[0]` bernilai 0, dan elemen `kondisi[1]` bernilai 1.

`int arr_dinamis[];` berarti mendeklarasikan array dengan ukuran maksimum array tidak diketahui, namun ukuran tersebut diketahui berdasarkan inisialisasi yaitu sebanyak 3 elemen, yang isinya 1,2, dan 3.

Kita tidak dapat mendeklarasikan array dinamis tanpa inisialisasi!

PENJELASAN ARRAY 1 DIMENSI

- Tanda `[]` disebut juga “elemen yang ke- ... Misalnya `kondisi[0]` berarti elemen yang ke nol.
- Array yang sudah dipesan, misalnya 10 tempat tidak harus diisi semuanya, bisa saja hanya diisi 5 elemen saja, baik secara berurutan maupun tidak. Namun pada kondisi yang tidak sepenuhnya terisi

tersebut, tempat pemesanan di memori tetap sebanyak 10 tempat, jadi tempat yang tidak terisi tetap akan terpesan dan dibiarkan kosong.

Contoh 1 (variabel array dan variabel biasa)

```
//Dengan variabel biasa:  
int x1=3,x2=5,x3=2,x4=7,x5=9;  
  
//Dengan larik:  
int x[5]={3,5,2,7,9};
```

Bagaimana jika kita ingin menghitung total dari variabel biasa?

```
total = x1 + x2 + x3 + x4 + x5;
```

Contoh 2 (menginputkan dan menampilkan array)

```
#include <stdio.h>  
#include <conio.h>  
  
void main()  
{ int nilai[5], x;  
  clrscr();  
  
  printf("Memasukkan nilai :\n");  
  for(x=0;x<5;x++)  
  {  
    printf("Nilai Angka : "); scanf("%d",&nilai[x]);  
  }  
  printf("\n");  
  
  printf("Membaca nilai :\n");  
  for(x=0;x<5;x++)  
  {  
    printf("Nilai Angka : %d",nilai[x]);  
  }  
  
  getch();  
}
```



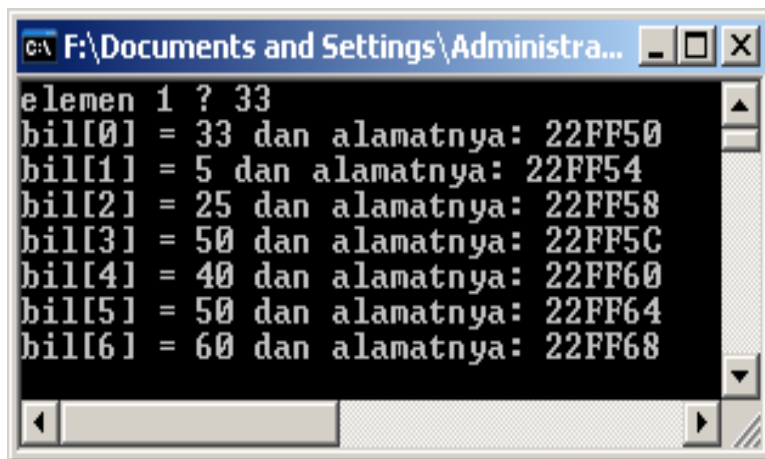
```
C:\ Command Prompt (2) - tc  
Memasukkan nilai :  
Nilai Angka 4  
Nilai Angka 7  
Nilai Angka 3  
Nilai Angka 9  
Nilai Angka 6  
  
Membaca nilai :  
Nilai Angka 4  
Nilai Angka 7  
Nilai Angka 3  
Nilai Angka 9  
Nilai Angka 6
```

Contoh 3 (manipulasi array 1 dimensi)

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7],i;
    printf("elemen 1 ? ");scanf("%d",&bil[0]);
    bil[1] = 5;
    bil[2] = bil[1] + 20;
    for(i=4;i<7;i++) bil[i] = i*10;
    bil[3] = bil[bil[1]];
    for(i=0;i<7;i++) printf("bil[%d] = %d dan alamatnya: %X\n",i,bil[i],&bil[i]);
    getch();
    return 0;
}
```

Hasilnya:



```
elemen 1 ? 33
bil[0] = 33 dan alamatnya: 22FF50
bil[1] = 5 dan alamatnya: 22FF54
bil[2] = 25 dan alamatnya: 22FF58
bil[3] = 50 dan alamatnya: 22FF5C
bil[4] = 40 dan alamatnya: 22FF60
bil[5] = 50 dan alamatnya: 22FF64
bil[6] = 60 dan alamatnya: 22FF68
```

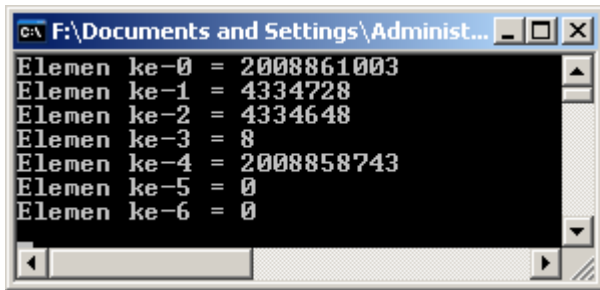
Terlihat bahwa alamat array berurutan dengan jarak antar alamat adalah 4 bytes (integer berukuran 4 bytes)

Contoh 4 (tanpa inisialisasi langsung ditampilkan)

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7]; //tanpa inisialisasi
    for(int i=0;i<7;i++){
        printf("Elemen ke-%i = %d\n",i,bil[i]);
    }
    getch();
    return 0;
}
```

Hasilnya:



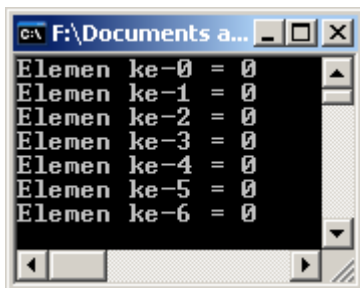
```
Elemen ke-0 = 2008861003
Elemen ke-1 = 4334728
Elemen ke-2 = 4334648
Elemen ke-3 = 8
Elemen ke-4 = 2008858743
Elemen ke-5 = 0
Elemen ke-6 = 0
```

Contoh 5 (inisialisasi dengan 0)

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7] = {0};    //inisialisasi 0
    for(int i=0;i<7;i++){
        printf("Elemen ke-%i = %d\n",i,bil[i]);
    }
    getch();
    return 0;
}
```

Hasilnya:



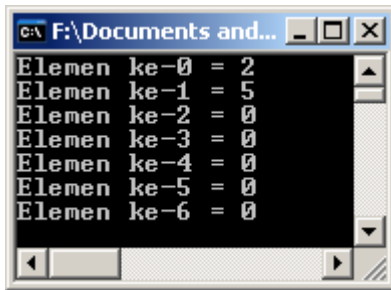
```
Elemen ke-0 = 0
Elemen ke-1 = 0
Elemen ke-2 = 0
Elemen ke-3 = 0
Elemen ke-4 = 0
Elemen ke-5 = 0
Elemen ke-6 = 0
```

Contoh 6 (inisialisasi hanya 2 elemen pertama)

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7] = {2,5};
    for(int i=0;i<7;i++){
        printf("Elemen ke-%i = %d\n",i,bil[i]);
    }
    getch();
    return 0;
}
```

Hasilnya:



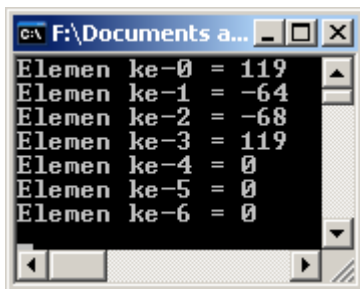
```
Elemen ke-0 = 2
Elemen ke-1 = 5
Elemen ke-2 = 0
Elemen ke-3 = 0
Elemen ke-4 = 0
Elemen ke-5 = 0
Elemen ke-6 = 0
```

Contoh 7 (karakter yang tidak diinisialisasi)

```
#include <stdio.h>
#include <conio.h>

int main(){
    char h[5];
    for(int i=0;i<5;i++){
        printf("Elemen ke-%i = %c\n",i,h[i]);
    }
    getch();
    return 0;
}
```

Hasilnya:



```
Elemen ke-0 = 119
Elemen ke-1 = -64
Elemen ke-2 = -68
Elemen ke-3 = 119
Elemen ke-4 = 0
Elemen ke-5 = 0
Elemen ke-6 = 0
```

Operasi Elemen-elemen Array

- Penambahan elemen array
- Menampilkan elemen array
- Pencarian elemen array
 - o Jika ditemukan, katakan KETEMU!
- Penghapusan elemen array
 - o Cari, jika ditemukan baru dihapus.

- Penghapusan elemen array pada dasarnya tidak ada!
- Yang ada adalah *pe-replace-an* isi elemen array yang akan dihapus oleh elemen-elemen berikutnya.

Ilustrasi penghapusan data 11 (elemen ke-4), $n = 8$:

0	1	2	3	4	5	6	7	indeks
8	10	6	-2	11	7	1	100	value
ffea	ffeb	ffec	ffed	ffef	fffa	fffb	fffc	alamat

Penggeseran elemen, $n = 7$:

0	1	2	3	4	5	6	7	indeks
								value
ffea	ffeb	ffec	ffed	ffef	fffa	fffb	fffc	alamat

- Pengeditan elemen array
 - Cari, jika ditemukan baru diedit dengan elemen yang baru.
 - Pengisian elemen baru sama saja melakukan *pe-replace-an* data pada indeks elemen yang diedit.

Pengertian Searching

- Pada suatu data seringkali dibutuhkan pembacaan kembali informasi (retrieval information) dengan cara searching.
- Searching adalah pencarian data dengan menelusuri tempat pencarian data tersebut.
- Tempat pencarian data tersebut dapat berupa array dalam memori, bisa juga pada file pada external storage.

Teknik-teknik Searching

1. Sequential Search

2. Binary Search
3. Interpolation Search
4. Quick Search

Sequential Search

- Adalah suatu teknik pencarian data dalam array (1 dimensi) yang akan menelusuri semua elemen-elemen array dari awal sampai akhir, dimana data-data tidak perlu diurutkan terlebih dahulu.
- Kemungkinan terbaik (best case) adalah jika data yang dicari terletak di indeks array terdepan (elemen array pertama) sehingga waktu yang dibutuhkan untuk pencarian data sangat sebentar (minimal).
- Kemungkinan terburuk (worst case) adalah jika data yang dicari terletak di indeks array terakhir (elemen array terakhir) sehingga waktu yang dibutuhkan untuk pencarian data sangat lama (maksimal).

Misalnya terdapat array satu dimensi sebagai berikut:

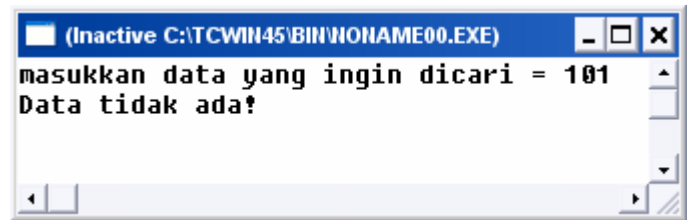
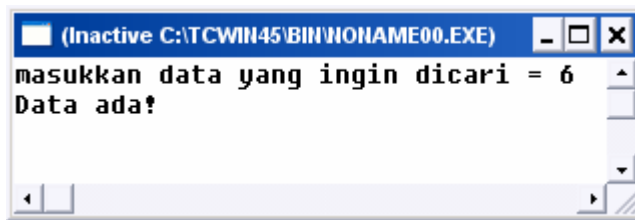
0	1	2	3	4	5	6	7	indeks
8	10	6	-2	11	7	1	100	value
ffea	ffeb	ffec	ffed	ffef	fffa	fffb	fffc	alamat

Kemudian program akan meminta data yang akan dicari, misalnya 6.

Jika ada maka akan ditampilkan tulisan “ADA”, sedangkan jika tidak ada maka akan ditampilkan tulisan “TIDAK ADA”.

```
#include <stdio.h>
#include <conio.h>
```

```
void main(){
    clrscr();
    int data[8] = {8,10,6,-2,11,7,1,100};
    int cari;
    int flag=0;
    printf("masukkan data yang ingin dicari = ");scanf("%d",&cari);
    for(int i=0;i<8;i++){
        if(data[i] == cari) flag=1;
    }
    if(flag==1) printf("Data ada!\n");
    else printf("Data tidak ada!\n");
}
```



- Dari program diatas, terlihat bahwa dilakukan perulangan untuk mengakses semua elemen array data satu persatu berdasarkan indeksny.
- Program menggunakan sebuah variabel flag yang berguna untuk menandai ada atau tidaknya data yang dicari dalam array data. Hanya bernilai 0 atau 1.
- Flag pertama kali diinisialisasi dengan nilai 0.
- Jika ditemukan, maka flag akan diset menjadi 1, jika tidak ada maka flag akan tetap bernilai 0.
- Semua elemen array data akan dibandingkan satu persatu dengan data yang dicari dan diinputkan oleh user.

Question: Bagaimana jika data yang dicari ditemukan maka data tersebut ditampilkan terletak di indeks ke berapa?

Hint: Gunakan indeks array!

Problem: Apakah cara di atas efisien? Jika datanya ada 10000?

Solution: Untuk meningkatkan efisiensi, seharusnya jika data yang dicari sudah ditemukan maka perulangan harus dihentikan!

Hint: Gunakan break!

Question: Bagaimana cara menghitung ada berapa data dalam array yang nilainya sama dengan data yang dicari oleh user?

Hint: Gunakan variabel counter yang nilainya akan selalu bertambah jika ada data yang ditemukan!

Penggunaan Sentinel (Penjaga)

Perhatikan array data berikut ini:

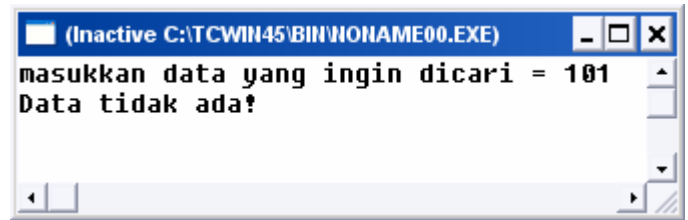
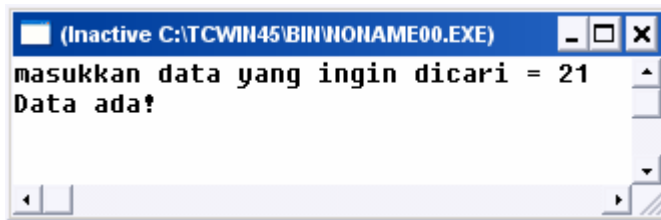
0	1	2	3	4	5	6	indeks
3	12	9	-4	21	6		value
ffea	ffeb	ffec	ffed	ffef	fffa	ffffb	alamat

- Terdapat 6 buah data dalam array (dari indeks 0 s/d 5) dan terdapat 1 indeks array tambahan (indeks ke 6) yang belum berisi data (disebut sentinel)
- Array pada indeks ke 6 berguna untuk menjaga agar indeks data berada pada indeks 0 s/d 5 saja. Bila pencarian data sudah mencapai array indeks yang ke-6 maka berarti data TIDAK ADA, sedangkan jika pencarian tidak mencapai indeks ke-6, maka data ADA.

```
#include <stdio.h>
#include <conio.h>
```

```
void main(){
    clrscr();
    int data[7] = {3,12,9,-4,21,6};
    int cari,i;
    printf("masukkan data yang ingin dicari = ");scanf("%d",&cari);
```

```
data[6] = cari;
i=0;
while(data[i] != cari) i++;
if(i<6) printf("Data ada!\n"); else printf("Data tidak ada!\n");
}
```



KESIMPULAN: sangat efisien!

Binary Search

- Data yang ada harus diurutkan terlebih dahulu berdasarkan suatu urutan tertentu yang dijadikan kunci pencarian.
- Adalah teknik pencarian data dalam dengan cara membagi data menjadi dua bagian setiap kali terjadi proses pengurutan.
- Prinsip pencarian biner adalah:
 - o Data diambil dari posisi 1 sampai posisi akhir N
 - o Kemudian cari posisi data tengah dengan rumus $(\text{posisi awal} + \text{posisi akhir}) / 2$
 - o Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
 - o Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
 - o Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah - 1
 - o Jika data sama, berarti ketemu.

Contoh Data:

Misalnya data yang dicari **17**

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
A				B				C

Karena $17 > 15$ (data tengah), maka: awal = tengah + 1

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
					A	B		C

Karena $17 < 23$ (data tengah), maka: akhir = tengah – 1

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
					A=B=C			

Karena $17 = 17$ (data tengah), maka KETEMU!

Programnya:

```
int binary_search(int cari){
    int l,r,m;
    l = 0;
    r = n-1;
    int ktm = 0;

    while(l<=r && ktm==0){
        m = (l+r)/2;
        if(data[m] == cari) ktm=1;
        else if (cari < data[m]) r=m-1;
        else l=m+1;
    }

    if(ktm==1) return 1; else return 0;
}
```

Interpolation Search

- Teknik ini dilakukan pada data yang sudah terurut berdasarkan kunci tertentu
- Teknik searching ini dilakukan dengan perkiraan letak data.

- Contoh ilustrasi: jika kita hendak mencari suatu nama di dalam buku telepon, misal yang berawalan dengan huruf T, maka kita tidak akan mencarinya dari awal buku, tapi kita langsung membukanya pada 2/3 atau 3/4 dari tebal buku.
- Jadi kita mencari data secara relatif terhadap jumlah data.
- Rumus posisi relatif kunci pencarian dihitung dengan rumus:

$$Posisi = \frac{kunci - data [low]}{data [high] - data [low]} \times (high - low) + low$$

Misal terdapat data sebagai berikut:

Kode	Judul Buku	Pengarang
025	The C++ Programming	James Wood
034	Mastering Delphi 6	Marcopolo
041	Professional C#	Simon Webe
056	Pure JavaScript v2	Michael Bolton
063	Advanced JSP & Servlet	David Dunn
072	Calculus Make it Easy	Gunner Christian
088	Visual Basic 2005 Express	Antonie
096	Artificial Life : Volume 1	Gloria Virginia

Kunci Pencarian ? **088**

Low ? **0**

High ? **7**

$$Posisi = (088 - 025) / (096 - 025) * (7 - 0) + 0 = [6]$$

Kunci[6] = kunci pencarian, data ditemukan : **Visual Basic 2005**

Kunci Pencarian ? **060**

Low ? **0**

High ? **7**

$$\text{Posisi} = (060 - 025) / (096 - 025) * (7 - 0) + 0 = [3]$$

Kunci[3] < kunci pencarian, maka teruskan

$$\text{Low} = 3 + 1 = 4$$

$$\text{High} = 7$$

Ternyata Kunci[4] adalah **063** yang lebih besar daripada **060**.

Berarti tidak ada kunci **060**.

Programnya:

```
int interpolationsearch(int key,int n){
    int low,high,pos,i;
    low=0;
    high=n-1;
    do{
        pos = (key - data[low]) * (high - low) / data[high] -
data[low] + low;
        if (data[pos] == key) return pos;
        if (data[pos] > key) high = pos-1;
        else
            if (data[pos] < key) low = pos + 1;
    } while(key >= data[low] && key <= data[high]);
    return -1
}
```

SOAL-SOAL:

Buatlah paper tentang cara penggunaan dan teknologi dari website-website pencari yang ada di Internet!

NEXT

Sorting Array