



Array, Struktur dan Pointer

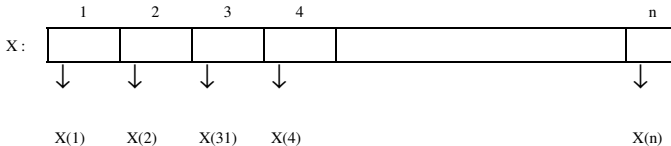
wijanarto



STRUKTUR DATA

LARIK/ARRAY STATIS

- Larik/array digunakan untuk menangani data yang banyak dan bertipe sama.





Algoritma Untuk membaca data dilakukan dengan (1)

```
int x[10];  
scanf("%d", jum);  
for (i=0;i<jum;i++)  
    scanf("%d", x[i]);
```




Algoritma untuk membaca data dengan menghindari data yang sama (2)

```
scanf ("%d", &n);  
k=1;  
do  
    scanf ("%d", &b);  
    ada=0;  
    for (i=1; i<k-1; i++)  
        if (b==x[i] ) ada =1;  
    if (!ada){  
        x[k]=b;  k=k+1;  
    } else printf("data sudah ada\n");  
while (k>n)
```



Algoritma untuk menentukan data terbesar (max) (3)

```
max=x[1]
for (i=2;i<n;i++)
    if (x[i]>max) max=x[i]
```



Algoritma untuk menentukan data terbesar (max) dan terkecil (min) sekaligus (4)

```
max=x[1];  
min=x[1];  
for (i=2;i<n;i++)  
    if (x[i]> max)max=x[i] ;else  
    if (x[i]<min) min=x[i];
```



Catatan

- Jika cara no 3 digunakan juga untuk menentukan data terkecil maka :
- untuk mencari max diperlukan $(n-1)$ kali perbandingan
- untuk mencari nilai min diperlukan $(n-1)$ kali perbandingan
- total perbandingan yang diperlukan untuk menentukan max dan min sekaligus adalah sebanyak $(n-1)+(n-1)=2n-2$ kali.



Catatan (lanj.)

- Sedangkan cara no 4, jumlah perbandingan yang diperlukan adalah:
- $(n-1) \leq \text{jumlah perbandingan} \leq 2n-2$.
- Yang perlu dicatat adalah jika jumlah perbandingan sama dengan $n-1$ maka ini disebut best case atau merupakan urutan terbaik, dan jika $2n-2$ adalah worst case



Struktur data dinamis (Pointer)

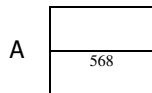
- Variabel **statis** merupakan suatu variable yang kedudukannya di memori bersifat tetap selama program berjalan, dan tidak dapat diubah-ubah sehingga ukuran memori yang diperlukan oleh program tersebut bersifat statis
- Sedangkan variabel **dinamis** adalah suatu variable yang dialokasikan pada saat diperlukan dan dapat dihapus saat program sedang berjalan, dengan demikian ukuran memori yang dibutuhkan oleh program bersifat dinamis.



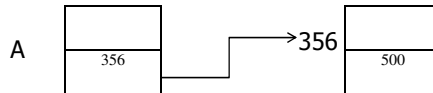
Struktur data dinamis (pointer)

- Pointer merupakan struktur data yang dinamis karena variable yang di deklarasikan menunjuk pada lokasi alamat memori tertentu dalam RAM. Jadi variable pointer tidak berisi suatu nilai tetapi berisi suatu alamat memori tertentu.

Ilustrasi



(a)



(b)

Variabel A Gambar (a) merupakan variabel statis yang berisi suatu nilai statis, sedangkan variabel A pada gambar (b) merupakan variabel dinamis yang menunjuk pada suatu alamat tertentu (356), dimana alamat yang ditunjuk tersebut berisi nilai data 500. Nilai data yang ditunjuk ini biasanya disebut node atau simpul



Memahami pointer

- Variabel biasa : Variabel adalah sesuatu di dalam program yang mempunyai nama dan nilai yang di kandunganya, juga mempunyai tipe serta ukuran atau bobot
- `int k;`
- compiler dan linker menangani setidaknya 2 byte di memori yang menangani nilai integer dia juga mengeset table symbol



Memahami pointer

- Jika $k=5$
- *Ada aturan lvalue=rvalue, 5 di asosiasikan dengan k*
- *Bagaimana dengan $5=k$???*
- *rvalue SELALU di sebelah kanan penugasan*

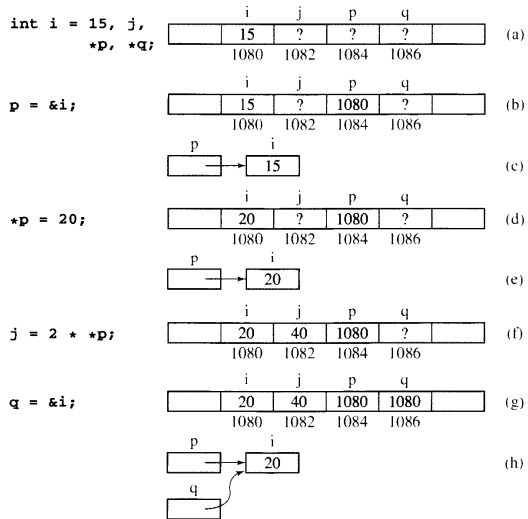


Memahami Pointer

& operator dereference
Mengambil alamat suatu variabel

* Operator reference
Mengambil nilai dari
alamat suatu variabel

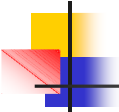
sumber





Struktur

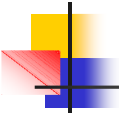
- *Structures* digunakan saat kita perlu untuk memproses data dengan tipe data yang beragam, namun kita tetap mengingnkannya untuk mereferensi data tersebut sebagai entitas tunggal.



Struktur Statis

```
struct student {
    char name[30];
    float marks;
} student1, student2;

int main() {
    struct student student3;
    char s1[30]; float f;
    scanf ("%s", name);
    scanf (" %f", & f);
    student1.name = s1;
    student2.marks = f;
    printf ("Nama : %s \n", student1.name);
    printf (" Marks are %f \n", student2.marks);
    return 0;
}
```

Struktur Dinamis

```
typedef struct node *stack;  
struct node {  
    int data;  
    stack next;  
};  
stack top;
```

Tipe data baru
pointer stack
bertipe struct
node

Record node,
berisi info data
dan pointer ke
node selanjutnya

Variabel top
bertipe pointer
stack



Alokasi Struktur Dinamis

```
typedef struct telm *addr;  
struct telm {  
    int data;  
    addr next;  
}elm;  
typedef addr stack;
```

```
Contoh Alokasi addr t :  
t=(addr)malloc(sizeof(elm));
```

```
Contoh DeAlokasi addr t :  
Free(t) ;
```



Implementasi Pada Stack

```
typedef struct node *stack;  
struct node {  
    int data;  
    stack next;  
};  
stack top, bottom;  
Int A,B,C;
```



Implementasi Pada Stack

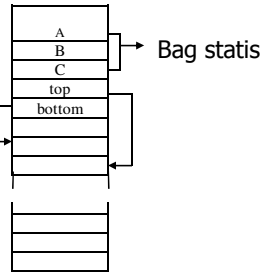
```
typedef struct node *stack;  
struct node {  
    int data;  
    stack next;  
};  
stack top, bottom;  
Int A, B, C;
```

top dan bottom merupakan variable pointer yang belum menunjuk kesuatu simpul dalam hal ini diberi nilai **NULL**, untuk mengalokasikan node (simpul) pada variable pointer digunakan keyword **malloc**, sedang untuk membebaskan pointer digunakan **free**.

```
top=(struct node*)malloc(sizeof(struct node));  
bottom=(struct node*)malloc(sizeof(struct node));  
free(top); free(bottom)
```

Ilustrasi

```
typedef struct node *stack;
struct node {
    int data;
    stack next;
};
stack top, bottom;
int A, B, C;
```



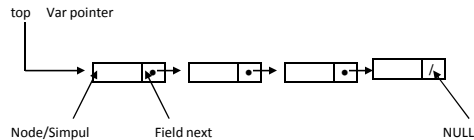


Implementasi Pada Stack

```
typedef struct node *stack;
struct node {
    int data;
    stack next;
};
stack top, bottom;
int A, B, C;
```

next merupakan field bertipe pointer yang menunjuk pada record node, ini disebut suatu linked list atau daftar berkait atau senarai berantai, dimana variable next menunjuk pada node selanjutnya begitu seterusnya.

Ilustrasi



```
typedef struct node *stack;  
struct node {  
    int data;  
    stack next;  
};  
stack top, bottom;  
int A,B,C;
```

Ilustrasi

```
top=(struct node*)malloc(sizeof(struct node));  
bottom=(struct node*)malloc(sizeof(struct node));
```

