

Pointer dan List Berkait



Akmal, S.Si, MT

Mata Kuliah : Struktur Data

Tujuan

- ❑ Mahasiswa dapat menjelaskan pengertian pointer dan bisa menerangkan operasi dasar menggunakan pointer dengan benar
- ❑ Mahasiswa dapat : Mengoperasikan dan membuat program dari semua algoritma primitive list singly (insert, delete, traversal dan searching) dengan benar.

Pokok Bahasan

- ❑ Pengertian Pointer
- ❑ Deklarasi Pointer to Integer dan Alokasi memori
- ❑ Operasi Pada Pointer
- ❑ Pointer to Record
- ❑ Pengertian List berkait
- ❑ Operasi Traversal List
- ❑ Menambah di depan, belakang, dan tengah
- ❑ Menghapus di depan, belakang dan tengah
- ❑ Searching

Pointer

- ❑ Pointer adalah suatu tipe data yang berisi alamat memori, atau disebut juga sebagai penunjuk/pencatat alamat memori.
- ❑ Deklarasi suatu pointer menggunakan notasi * dibelakang tipe

tipe* namaPointer

contoh :

```
int* p;
```

- ❑ Operator awalan & didepan suatu variabel digunakan untuk mengambil alamat dari suatu variabel.

```
p = &x;    // pointer p mencatat alamat x
```

Perhatikan hasilnya !

```
main() {  
    int n=33;  
    cout<<"n = "<<n<<endl;           // isi n  
    cout<<"&n = "<<&n<<endl;         // alamat n  
}
```

Referensi

- ❑ Referensi atau Acuan atau Alias adalah suatu sinonim untuk variable lain. Pendeklarasian referensi yaitu dengan menggunakan operator “&” di belakang suatu tipe data.

tipe& alias = var

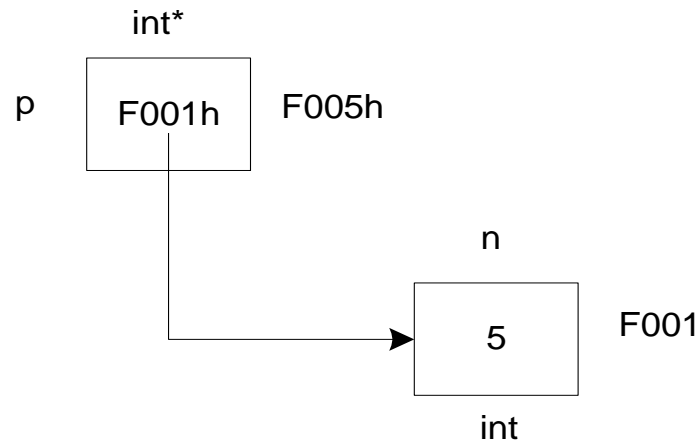
contoh:

int& m=n;

Contoh penggunaan referensi

```
main(){
    int n=3;
    int& r=n;                // r alias dari n
    cout<<"n = "<<n<<" , r = "<<r<<endl; //3,3
    --n;
    cout<<"n = "<<n<<" , r = "<<r<<endl; //2,2
    r*=2;
    cout<<"n = "<<n<<" , r = "<<r<<endl;  //4,4
    cout<<"&n = "<<&n<<" , &r = "<<&r<<endl;
    }                        // alamat juga sama
```

Pointer to Integer



- Gambar diatas menunjukkan bahwa sebuah lokasi dengan nama n yang isinya bernilai 5 dengan tipe integer dicatat alamatnya oleh sebuah pointer p.
- Pointer p sendiri menempati lokasi memori dengan alamat F005h dengan isi yang dicatatnya adalah F001h (alamat dari n).
- Untuk mengakses nilai dari lokasi (variabel) yang ditunjuk oleh *pointer to integer* dengan menggunakan operator `*` (bintang / asterik) di depan variabel pointer

Pointer To Integer

```
main() {  
    int n=5;  
    int* p= &n;  
    cout<<"n = "<<n<<",&n = "<< &n <<endl;    // 5 , F001h  
    cout<<"p = "<<p<<",&p = "<< &p <<endl;    // F001h,F005h  
}
```

```
main() {  
    int n=5;  
    int* p= &n;  
    cout<<"*p = "<< *p <<endl;                // 5  
}                                                  // Contoh nilai variabel yang ditunjuk pointer
```

```
main() {  
    int n=5;  
    int* p= &n;  
    int& m= *p;  
    cout<<"m = "<< m <<endl;                // 5  
}                                                  // Contoh nilai referensi dari variabel yang ditunjuk pointer
```

Operator new

- Ketika pointer sudah dideklarasikan seperti berikut :

```
float* p;    // p adalah pointer untuk suatu lokasi memori bertipe float
```

maka pernyataan di atas hanya mengalokasikan memori untuk pointer itu sendiri.

Nilai dari pointer nantinya berupa alamat memori, tetapi memori pada alamat yang akan dicatat oleh suatu pointer belum dialokasikan.

- Perhatikan deklarasi kasus dibawah ini, **p** belum diinisialisasikan (belum mengalokasikan memori). Usaha untuk mengakses **p** mungkin akan mengalami error.

```
*p=3. 1415;           // ERROR : karena tidak ada alokasi memori untuk *p
```

- Jalan terbaik mencegah masalah ini adalah menginisialisasikan pointer ketika dideklarasikan.
 - **Cara 1** : Mencatat lokasi yang sudah ada nama dan pemiliknya
 - **Cara 2** : Mencatat lokasi yang dialokasikan sesuai kebutuhan atau ketika diperlukan) → ALOKASI DINAMIS

Operator new

Cara 1 (Mencatat lokasi yang sudah ada nama dan pemiliknya)

```
float x = 3.1415;           // x berisi nilai 3.1415
float* p = &x;              // p berisi alamat dari x
cout<<"p = "<<*p<<endl;
```

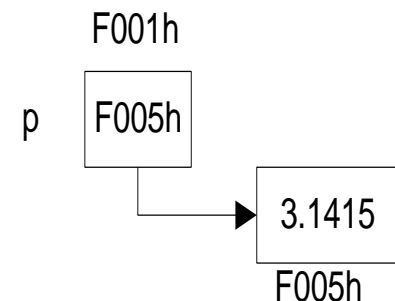
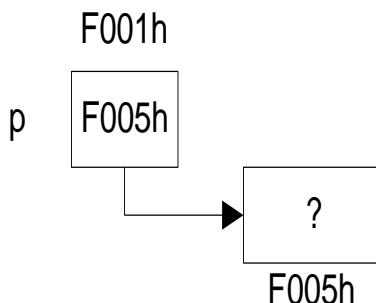
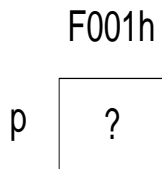
- Dalam kasus ini, mengakses ***p** tidak akan bermasalah karena memori yang digunakan untuk menyimpan **float 3.1415** sudah otomatis dialokasikan ketika **x** dideklarasikan. **p** menunjuk pada alokasi memori yang sama.

Cara 2 (Mencatat lokasi yang dialokasikan sesuai kebutuhan atau ketika diperlukan)

a) `float* p;`

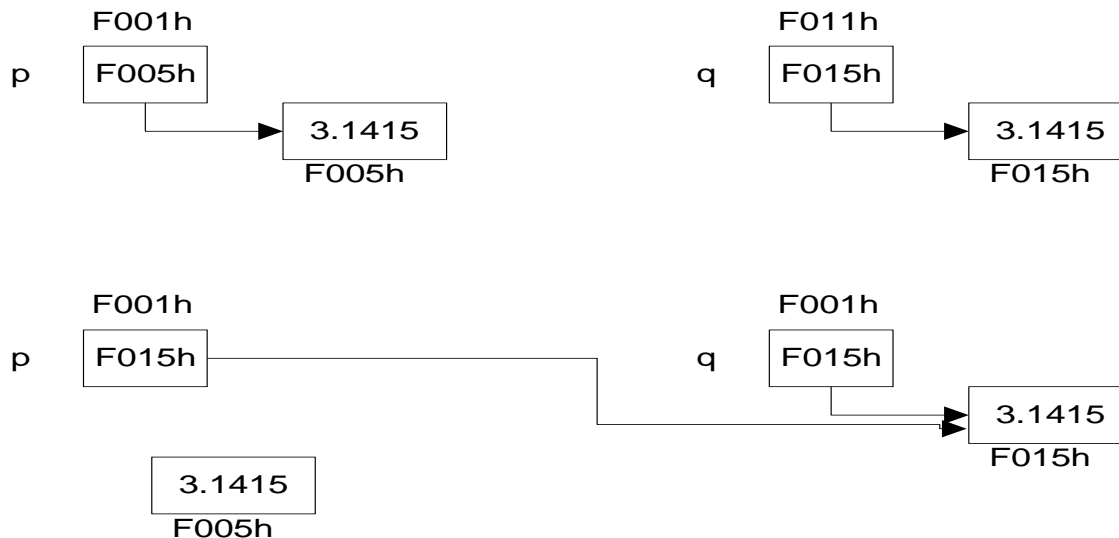
b) **`p = new float;`** // alokasi memori untuk 1 data
// float yang langsung
// dicatat oleh p

c) `*p = 3.1415;`



Operasi Pointer

- ❑ Operasi yang umum dilakukan terhadap pointer adalah **operasi Assignment**.
- ❑ Misalkan ada 2 pointer p dan q, lalu dilakukan operasi assignment sbb :
 $p = q;$
Artinya p akan mencatat alamat yang sama dengan yang dicatat oleh q;
- ❑ Gambaran keadaan pointer adalah sbb



Dealokasi

- ❑ Jika suatu tempat yang dicatat oleh suatu pointer ingin dibebaskan maka haruslah dilakukan proses dealokasi dengan tujuan lokasi tersebut bisa ditempati oleh data jenis lain.

Perintah yang digunakan adalah :

`delete(p)`

- ❑ Dengan bentuk pengelolaan seperti ini maka dilakukan pengelolaan memori secara dinamis yang artinya tidak perlu mengalokasikan memori lebih awal terlebih dahulu secara tetap (fixed).
- ❑ Jika diperlukan maka bisa dilakukan alokasi dan jika sudah tidak diperlukan maka lokasi memori tersebut bisa dilakukan dealokasi / dibebaskan

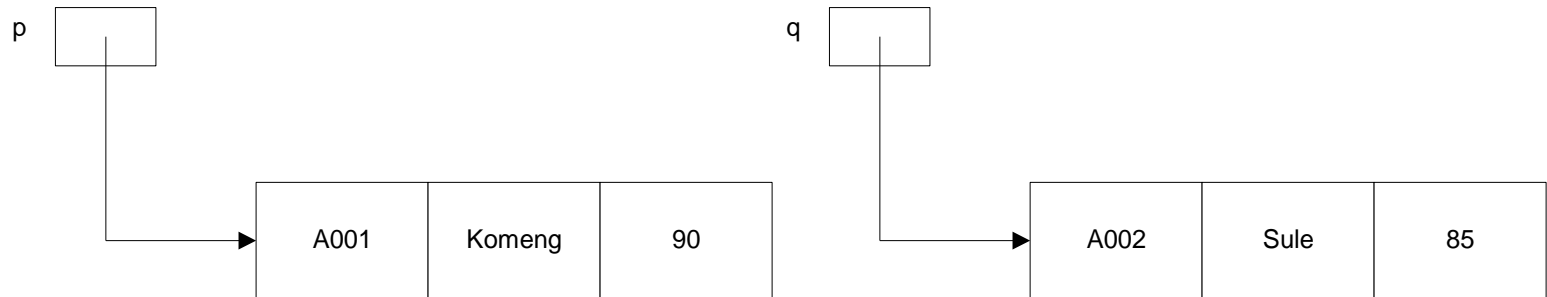
```

main() {
    int n = 1;
    int* p = &n;
    int* q;
    int* r;
    int* s;
    cout <<" n = "<< n << "      &n      = "<< &n << endl ;
    cout <<" p = "<< p << "      *p      = "<< *p <<endl;
    cout <<"&p = "<< &p <<"    &(*p)= "<< &(*p)<<endl;
    q = new int;                // alokasi
    *q = 2;
    cout <<" q = "<< q <<"      *q      = "<< *q <<endl;
    cout <<" &q = "<< &q <<"    &(*q)= "<< &(*q)<<endl;
    // delete(q)                // apa yg terjadi ??
    r = new int;
    *r = 3;
    cout <<" r = "<< r <<"      *r      = "<< *r <<endl;
    cout <<" &r = "<< &r <<"    &(*r) = "<< &(*r)<<endl;
    s=q;
    *s = 4;
    cout <<" q = "<< q <<"      *q      = "<< *q <<endl;
    cout <<" s = "<< s <<"      *s      = "<< *s <<endl;
    cout <<" &q = "<< &q <<"    &s = "<< &s<<endl;
}

```

Pointer to Record

- Pointer juga bisa digunakan untuk mencatat alamat dari suatu record / structure
Caranya : 1. Buat tipe nama record/structure
2. Buat nama alias tipe pointer yang menunjuk ke tipe nama record
3. Deklarasikan variable-variabel yang mengacu pada alias tipe pointer tsb.



Mendeklarasikan pointer to record

```
struct mahasiswa {
    char NPM[8] ;
    char nama[20];
    int nilai;
};
typedef mahasiswa* pointerToMhs;    //pointer menunjuk address record mahasiswa

main() {
    pointerToMhs p,q;                //p,q : pointer to record
    p=new mahasiswa;                // alokasi melalui p
    cin>>p->NPM;
    cin>>p->nama;
    cin>>p->nilai;

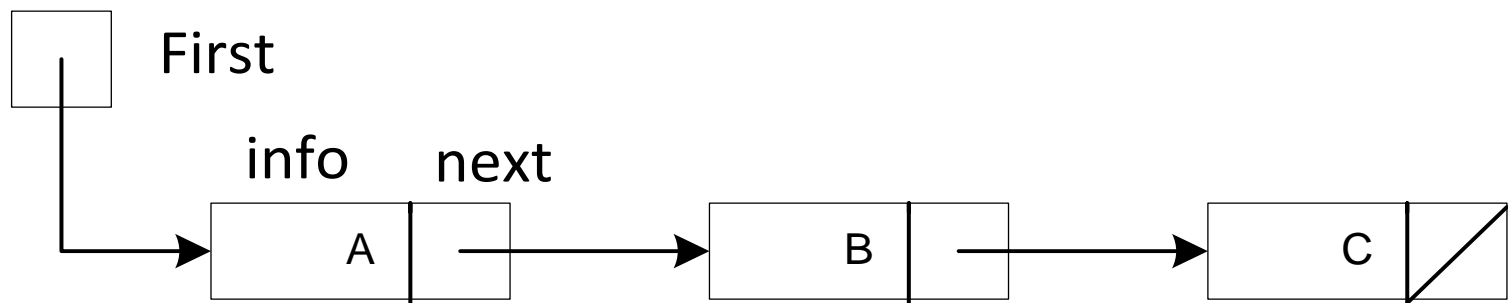
    q=new mahasiswa;                // alokasi melalui q
    cin>>q->NPM;
    cin>>q->nama;
    cin>>q->nilai;

    cout<<p->NPM<<p->nama<<p->nilai;
    cout<<q->NPM<<q->nama<<q->nilai;
}
```

- ❑ Pengaksesan atribut suatu record melalui suatu pointer menggunakan operator " → "
Contoh :
p→NPM

Singly Linked List

- Linked List atau List berkait adalah suatu bentuk tipe data abstrak yang merupakan kumpulan dari data berbentuk record dengan ciri setiap elemen data (record) memiliki karakteristik:
 - Ada informasi (*info*)
 - Ada pengait (pointer) antara satu elemen dengan elemen yang lain (*next*)



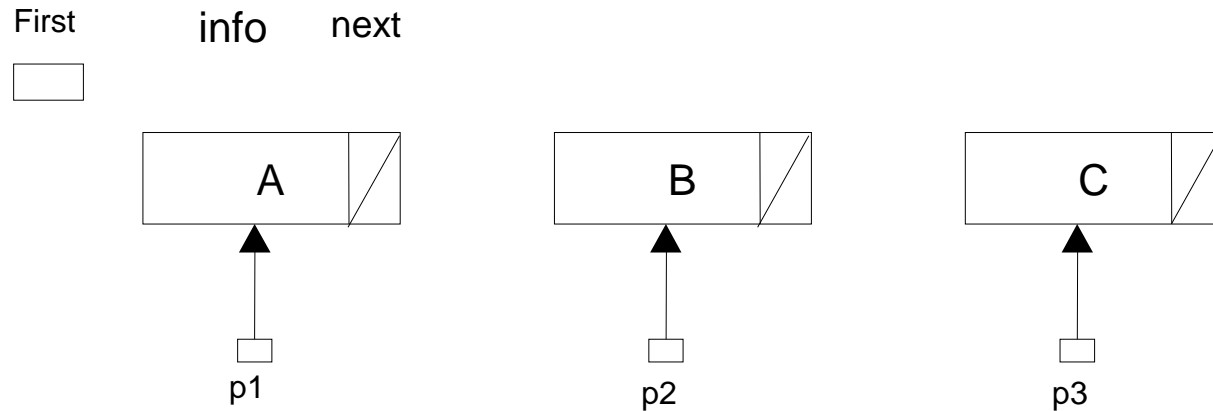
Pendeklarasian tipe struktur list berkait

```
struct namaRecord {  
    tipe1 info1;           // info  
    .....  
    namaRecord* namaPointer; // namaPointer sbg pengait  
};  
typedef namaRecord* pointer; // membuat alias pointer  
typedef pointer List;        // membuat alias List
```

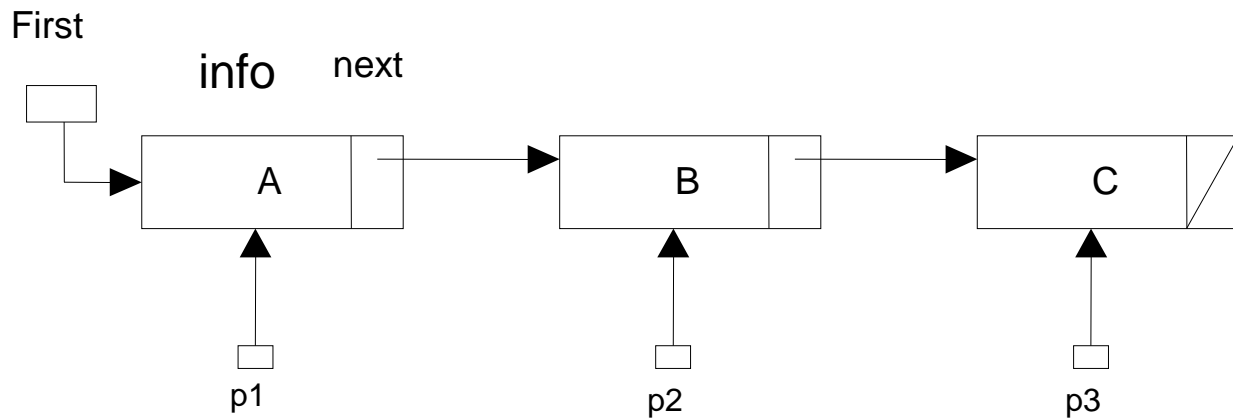
Contoh:

```
struct ElmtList {  
    char info;  
    ElmtList* next;  
};  
typedef ElmtList* pointerToElmtList;  
  
typedef pointerToElmtList List;
```

Contoh bagaimana membentuk list



- ▣ $P1 \rightarrow next = p2;$
- ▣ $P2 \rightarrow next = p3;$
- ▣ $First = p1;$



```

struct ElmtList {
    char info;                //info
    ElmtList* next;          // pointer next sbg pengait
};
typedef ElmtList* pointerToElmtList;    // membuat tipe alias pointer to record
typedef pointerToElmtList List;

main() {
    pointerToElmtList p1, p2, p3;
    List First;

    cout<<"Input  : "<<endl;
    p1=new ElmtList;
    cout<<"info  = "; cin>>p1->info;          // A
    p1->next=NULL;

    p2=new ElmtList;
    cout<<"info  = "; cin>>p2->info;          // B
    p2->next=NULL;

    p3=new ElmtList;
    cout<<"info  = "; cin>>p3->info;          // C
    p3->next=NULL;

    p1->next=p2;                // kaitkan
    p2->next=p3;                // kaitkan
    First = p1;                // First menunjuk elemen pertama

    cout<<"Output  : "<<endl;
    cout<<"info    : "<<p1->info<<endl;        // A
    cout<<"info    : "<<p2->info<<endl;        // B
    cout<<"info    : "<<p1->next->info<<endl;    // B
    cout<<"info    : "<<p3->info<<endl;        // C
}

```