Kode Asisten PJ : MTN A11.4301-U

(Partner)

NIM : A11.2012.06758 NIM : A11.2012.06748

Nama : Muhammad Adhi D Nama : M Azwar Adli.

Modul : PSDA-6

Description :

ADT Kontigu.

Dengan reprerentasi fisik kontigu, satu-satunyastruktur data yang dapat menyimpannya adalah tabel, karena hanya tabel yang mempunyai struktur kontigu. Setiap elemen table mengandung informasi info, sedangkan informasi mengenai Next tidak perlu lagi disimpan secara eksplisit, karena secara implisit sudah tersirat dalam struktur data yang menjadi tempat penyimpanannya.

Elemen terakhir tidak mungkin dikenali dari NEXT, karena NEXT tidak disimpan secara eksplisit. Satu-satunya cara untuk mengetahui elemen terakhir adalah dari alamatnya : P=N, dengan N adalah lokasi pada tabel tempat menyimpan elemen terakhir.

Karena alamat elemen terakhir harus diketahui secara eksplisit, maka representasi list bukan murni seperti di atas, tetapi harus mengandung First(L) dan Last(L), seperti yang pernah dibahas pada Queue.

Pemilihan representasi fisik dari list linierakan sangat mempengaruhi performansi dari algoritma. Pada bagian selanjutnya, akan dipelajari (melalui kasus-kasus), kapan masing-masing representasi cocok untuk dipakai.

**Ringkasan Representasi List**

Representasi Berkait:

|  |  |  |
| --- | --- | --- |
| Representasi lojik  Berkait | Representasi fisik  berkait dengan Pointer | Representasi fisik berkait  dengan Tabel |
| KAMUS UMUM:  L : List  P : address  {kamus belum terdef.} | KAMUS UMUM:  typeinfotype:{terdef}  typeaddress:  pointer toElmt  typeElmtList:  < info:infotype,  Next:address>  typeList:{rep.terdef}  L : List  P : address | KAMUS UMUM:  typeinfotype:{terdef}  typeaddress:integer  typeElmtList:  < info:infotype,  Next:address >  {variabel GLOBAL}  constantNil=0  constantNMAx:address=...  TabMem:array[Nil..NMax]  ofElmtList  FirstAvail:address  typeList:{rep.terdef}  L : List  P : address |
| AKSES:  First(L)  Next(P)  Info(P) | AKSES:  tergantung deklarasi  P↑.Next  P↑.Info | AKSES:  tergantung deklarasi  TabMem(P).Next  TabMem(P).Info |
| PRIMITIF  ALOKASI/DEALOKASI: | PRIMITIF  ALOKASI/DEALOKASI:  (tidak perlu realisasi,  sistem) | PRIMITIF  ALOKASI/DEALOKASI:  (harus direalisasi)  MemFull  InitTab  AllocTab(P)  DeallocTab(P) |
| Pengenal elemen terakhir | Next(P) = Nil |  |

Representasi Kontigu :

|  |  |
| --- | --- |
| Representasi lojik kontigu | Representasi fisik kontigu dengan tabel |
| KAMUS UMUM:  L : List  P : address | KAMUS UMUM:  typeaddress : integer  typeElmtList : < info : infotype >  constantFirst : address = ...  constantLast : address = ...  typeList :  < TabMem : array[First..Last] ofElmtList >  L : List  P : address |
| AKSES:  First(L)  Next(P)  Info(P) | AKSES:  tergantung deklarasi  P ←P + 1  TabELmt(P).Info |
| PRIMITIF  ALOKASI/DEALOKASI: | PRIMITIF ALOKASI/DEALOKASI:  { tidak perlu dilakukan, pada saat pendefinisian  tabel, secara statis sudah ditentukan. Kecuali jika  deklarasi tabel secara dinamis seperti dalam bahasa C } |
| Pengenal elemen terakhir | Last |

Contoh :

void InsVFirst(List \*L, infotype X) {

int i;

address P;

if(ListEmpty(\*L)) {

FirstAdd = IndexMin;

LastAdd = FirstAdd;

(\*L).N = LastAdd;

(\*L).TabMem[FirstAdd].Info = X;

}

else {

P = First(\*L);

InsertFirst(&(\*L),P);

i = Last(\*L);

while(i!=First(\*L)) {

(\*L).TabMem[i].Info = (\*L).TabMem[i-1].Info;

i--;

}

(\*L).TabMem[FirstAdd].Info = X;

}

}

void PrintInfo(List L) {

address P;

if(ListEmpty(L)) {

printf("List kosong\n");

}

else {

P = First(L);

do {

printf("|%d| ", Info(P));

P++;

}

while(P!=Last(L)+1);

}

}

void InsertAfter(List \*L, address P) {

(\*L).N = P;

LastAdd = (\*L).N;

}