Modul 5 SINGLE LINKED LIST (BAGIAN KEDUA)

TUJUAN PRAKTIKUM
5. Memahami penggunaan *linked list* dengan *pointer* operator- operator dalam program. 6. Memahami operasi-operasi dasar dalam *linked list*. 7. Membuat program dengan menggunakan *linked list* dengan *prototype* yang ada

5.1 *Searching*

Searching merupakan operasi dasar *list* dengan melakukan aktivitas pencarian terhadap *node* tertentu. Proses ini berjalan dengan mengunjungi setiap *node* dan berhenti setelah *node* yang dicari ketemu. Dengan melakukan operasi *searching*, operasi-operasi seperti *insert after*, *delete after*, dan *update* akan lebih mudah.

Semua fungsi dasar diatas merupakan bagian dari ADT dari singgle *linked list*, dan aplikasi pada bahasa pemrograman Cp semua ADT tersebut tersimpan dalam *file* *.c dan *file* *.h.

```
1   /*file : list .h*/
2   /* contoh ADT list berkait dengan representasi fisik pointer*/
3   /* representasi address    dengan pointer*/
4   /* info tipe adalah integer */
5   #ifndef list_H
6   #define list_H
7   #include "boolean.h"
8   #include <stdio.h>
9   #define Nil NULL
10  #define info(P) (P)->info
11  #define next(P) (P)->next
12  #define first(L) ((L).first)
13
14  /*deklarasi record dan struktur data list*/
15  typedef int infotype;
16  typedef struct elmlist *address;
17  struct elmlist{
18      infotype info;
19      address next;
20  };
21
22  /* definisi list : */
23  /* list kosong jika First(L)=Nil */
24  /* setiap elemen address P dapat diacu info(P) atau next(P) */
25  struct list {
26      address first;
27  };
28  /********* pengecekan apakah list kosong ***********/
29  boolean ListEmpty(list L);
30  /*mengembalikan nilai true jika list kosong*/
31
32  /********* pembuatan list kosong *********/
33  void CreateList(list &L);
34  /* I.S. sembarang
35     F.S. terbentuk list kosong*/
36
37  /********* manajemen memori *********/
38  void dealokasi(address P);
39  /* I.S. P terdefinisi
40     F.S. memori yang digunakan P dikembalikan ke sistem */
41
42
43  /********* pencarian sebuah elemen list *********/
44  address findElm(list L, infotype X);
45  /* mencari apakah ada elemen list dengan info(P) = X
```

```
46      jika ada, mengembalikan address elemen tab tsb, dan Nil jika sebaliknya
47  */
48
49  boolean fFindElm(list L, address P);
50  /* mencari apakah ada elemen list dengan alamat P
51     mengembalikan true jika ada dan false jika tidak ada */
52
53  address findBefore(list L, address P);
54  /* mengembalikan address elemen sebelum P
55     jika prec berada pada awal list, maka mengembalikan nilai Nil */
56
57  /********* penambahan elemen **********/
58  void insertFirst(list &L, address P);
59  /* I.S. sembarang, P sudah dialokasikan
60     F.S. menempatkan elemen beralamat P pada awal list */
61
62  void insertAfter(list &L, address P, address Prec);
63  /* I.S. sembarang, P dan Prec alamt salah satu elemen list
64     F.S. menempatkan elemen beralamat P sesudah elemen beralamat Prec */
65
66  void insertLast(list &L, address P);
67  /* I.S. sembarang, P sudah dialokasikan
68     F.S. menempatkan elemen beralamat P pada akhir list */
69
70  /********* penghapusan sebuah elemen *********/
71  void delFirst(list &L, adress &P);
72  /* I.S. list tidak kosong
73     F.S. adalah alamat dari alamat elemen pertama list
74     sebelum elemen pertama list dihapus
75     elemen pertama list hilang dan list mungkin menjadi kosong
76     first elemen yang baru adalah successor first elemen yang lama */
77
78  void delLast(list &L, adress &P);
79  /* I.S. list tidak kosong
80     F.S. adalah alamat dari alamat elemen terakhir list
81     sebelum elemen terakhir list dihapus
82     elemen terakhir list hilang dan list mungkin menjadi kosong
83     last elemen yang baru adalah successor last elemen yang lama */
84
85  void delAfter(list &L, address &P, address Prec);
86  /* I.S. list tidak kosng, Prec alamat salah satu elemen list
87     F.S. P adalah alamatdari next(Prec), menghapus next(Prec) dari list */
88
89  void delP (list &L, infotype X);
90  /* I.S. sembarang
91     F.S. jika ada elemen list dengan alamat P, dimana info(P)=X, maka P
92     dihapus
93     dan P di-dealokasi, jika tidak ada maka list tetap
94     list mungkin akan menjadi kosong karena penghapusan */
95
96  /********** proses semau elemen list *********/
97  void printInfo(list L);
98  /* I.S. list mungkin kosong
99     F.S. jika list tidak kosong menampilkan semua info yang ada pada list */
100
101 int nbList(list L);
102 /* mengembalikan jumlah elemen pada list */
103
104 /********** proses terhadap list **********/
105 void delAll(list &L);
106 /* menghapus semua elemen list dan semua elemen di-dealokasi */
107
108 void invertList(list &L);
109 /* I.S. sembarang
110    F.S. elemen - elemen list dibalik */
111 void copyList(list L1, list &L2)
112 /* I.S. L1 sembarang
```

Fakultas Informatika
Telkom University
informatics lab

```
113     F.S. L1 = L2, L1 dan L2 menunjuk pada elemen yang sama */
114
115 list fCopyList(list L);
116 /* mengembalikan list yang merupakan salinan dari L */
117 #endif
```

5.2 *Latihan*

2. Buatlah ADT *Single Linked list* sebagai berikut di dalam *file* **"singlelist.h"**:

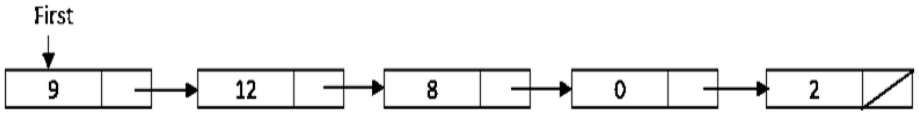
```
Type infotype : int
Type address  : pointer to ElmList

Type ElmList <
    info : infotype
    next : address
>

Type List : < First : address >

prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertFirst( in/out L : List, in P : address )
```

Kemudian buat implementasi ADT *Single Linked list* pada *file* **"singlelist.cpp".**
Adapun isi data



Gambar 5-1 Ilustrasi elemen

Cobalah hasil implementasi ADT pada *file* "**main.cpp**"

```
int main()
{
    List L;
    address P1, P2, P3, P4, P5 = NULL;
    createList(L);

    P1 = alokasi(2);
    insertFirst(L,P1);

    P2 = alokasi(0);
    insertFirst(L,P2);

    P3 = alokasi(8);
    insertFirst(L,P3);

    P4 = alokasi(12);
    insertFirst(L,P4);

    P5 = alokasi(9);
    insertFirst(L,P5);

    printInfo(L)
    return 0;
}
```

Fakultas Informatika School of Computing Telkom University informatics lab

```
9 12 8 0 2

Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```

Gambar 5-2 *Output* singlelist

3. Carilah elemen dengan info 8 dengan membuat fungsi baru.
   `fungsi findElm( L : List, x : infotype ) : address`

```
8 ditemukan dalam list
Process returned 0 (0x0)   execution time : 0.020 s
Press any key to continue.
```

Gambar 5-3 *Output* pencarian 8

4. Hitunglah jumlah total info seluruh elemen (9+12+8+0+2=31).

```
Total info dari kelima elemen adalah 31
Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```

Gambar 5-4 *Output* total info elemen

Fakultas Informatika
School of Computing
Telkom University
informatics lab