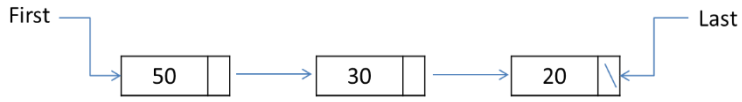
JURNAL PRAKTIKUM 3

Sebuah Single Linked List dengan pointer kepala First dan Last digunakan untuk menyimpan data berupa integer. Berikut ilustrasinya:



Buatlah ADT nya (SLL_First_Last.h, SLL_First_Last.cpp, dan Test_SLL_First_Last.cpp)!

A. Spesifikasi (Silakan ditulis ulang dalam Bahasa C++)

Type infotype : int
Type address : pointer to elmList
Type elmList <
    info : infotype
    next : address
>
Type List <
    First : address
    Last : address
>

Procedure createList (**input/ouput** L : List)
Function createElemen (dataBaru: infotype) → address
Procedure insertFirst (**input/ouput** L : List, **input** P : address)
Procedure insertLast (**input/ouput** L : List, **input** P : address)
Procedure InsertAfter (**input** Prec : address, P : address);
Procedure insertDescending (**input/ouput** L : List, **input** dataBaru : infotype)
Procedure deleteFirst (**input/ouput** L : List, **output** P : address)
Procedure deleteLast (**input/ouput** L : List, **output** P : address)
Procedure deleteAfter (**input** Prec: address, **output** P : address)
Procedure deleteElm (**input/ouput** L : List, **input** dataHapus : infotype)
Procedure printList (**input** L : List);
Function hitungElemen (L: List) → integer
Function median (L: List) → integer

B. Implementasi (Silakan ditulis ulang dalam Bahasa C++)

Procedure createList (**input/ouput** L : List)
{ *IS. –*
FS. Terbentuk sebuah list di mana, first dan last dari L bernilai NIL. }
Kamus
Algoritma
First (L) ← NIL
Last (L) ← NIL

Function createElemen (dataBaru: infotype) → address
{ *Return alamat alokasi memori sebuah elmList yang berisi dataBaru. }*
Kamus
P: address
Algoritma
alokasi (P)
info (P) ← dataBaru
next (P) ← NIL
→ P

Procedure insertFirst (**input/ouput** L : List, **input** P : address)
{ *IS. Terdefinisi pointer P berisi alamat elmList, dan sebuah list L (L mungkin kosong).*
FS. elmList yang ditunjuk oleh P ditambahkan ke dalam list sebagai elemen pertama. }
Kamus
Algoritma
if First (L) = NIL then
First (L) ← P
Last (L) ← P
else
next (P) ← First (L)
First (L) ← P

Procedure InsertAfter (**input** Prec : address, P : address);
{ *IS. Terdefinisi pointer Prec dan P berisi alamat elmList. Prec ≠ Last(L).*
FS. elmList yang ditunjuk oleh P ditambahkan ke dalam list setelah elmList yang ditunjuk oleh Prec. }
Kamus
Algoritma
next (P) ← next (Prec)
next (Prec) ← P

Procedure deleteFirst (**input/ouput** L : List, **output** P : address)
{ IS. Terdefinisi sebuah list L (L tidak kosong dan mungkin berisi satu elemen).
FS. P berisi alamat elmList yang pertama, elmList yang ditunjuk oleh P dihapus dari list }
Kamus
Algoritma
P ← First (L)
if next (First (L) = NIL) then
First (L) ← NIL
Last (L) ← NIL
else
First (L) ← next (First (L))
next (P) ← NIL

Procedure deleteAfter (**input** Prec: address, **output** P : address)
{ IS. Terdefinisi pointer Prec berisi alamat elmList. Prec ≠ Last(L). next(Prec) ≠ Last(L).
FS. P berisi alamat elmList setelah Prec, elmList yang ditunjuk oleh P dihapus dari list }
Kamus
Algoritma
P ← next (Prec)
next (Prec) ← next (P)
next (P) ← NIL

Procedure printList (**input** L : list);
{ IS. Terdefinisi sebuah list L
FS. Menampilkan semua info elmList di list. }
Kamus
Algoritma
P ← First (L)
while P≠NIL do
output (info (P))
P ← next (P)

C. Program Utama (Silakan ditulis ulang dalam Bahasa C++)

Kamus

L: List

Algoritma

createList (L)	
printList (L)	{}
insertDescending (L, 40)	
printList (L)	{40}
insertDescending (L, 10)	
printList (L)	{40 10}
insertDescending (L, 50)	
printList (L)	{50 40 10}
insertDescending (L, 42)	
printList (L)	{50 42 40 10}
insertDescending (L, 16)	
printList (L)	{50 42 40 16 10}
output (median (L))	{40}
deleteElm (L, 10)	
printList (L)	{50 42 40 16}
output (median (L))	{41}
deleteElm (L, 40)	
printList (L)	{50 42 16}
deleteElm (L, 42)	
printList (L)	{50 16}
deleteElm (L, 50)	
printList (L)	{16}
deleteElm (L, 16)	
printList (L)	{}

D. TUGAS TERBIMBING

Buat implementasi procedure berikut ini:

Procedure insertLast (**input/ouput** L : List, **input** P : address)
{ IS. Terdefinisi pointer P berisi alamat elmList, dan sebuah list L (L tidak kosong).
FS. elmList yang ditunjuk oleh P ditambahkan ke dalam list sebagai elemen terakhir. }

Procedure insertDescending (**input/ouput** L : List, **input** dataBaru : infotype)
{ IS. Terdefinisi sebuah data, dan sebuah list L (L mungkin kosong).
FS. dataBaru ditambahkan ke dalam list dengan aturan: data di dalam list harus selalu terurut secara menurun (descending).
Note: Gunakan procedure insertFirst, insertLast, dan insertAfter yang sudah dibuat sebelumnya. }

Procedure deleteLast (**input/ouput** L : List, **output** P : address)
{ IS. Terdefinisi sebuah list L (L tidak kosong dan berisi lebih dari satu elemen).
FS. P berisi alamat elmList yang terakhir, elmList yang ditunjuk oleh P dihapus dari list }

Procedure deleteElm (**input/ouput** L : List, **input** dataHapus : infotype)
{ IS. Terdefinisi sebuah list L (L mungkin kosong).
FS. Elemen dengan info = dataHapus, dihapus dari list.
Note: Gunakan procedure deleteFirst, deleteLast, dan deleteAfter yang sudah dibuat sebelumnya.}

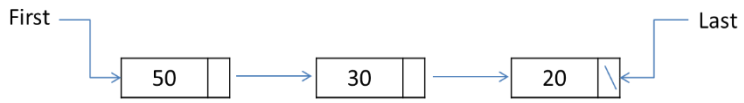
E. TUGAS MANDIRI (50 Menit)

Buat implementasi function berikut ini:

Function hitungElemen (L: List) → integer

{ Mengembalikan banyaknya elemen dalam list L.

Ilustrasi:



Banyaknya elemen = 3

}

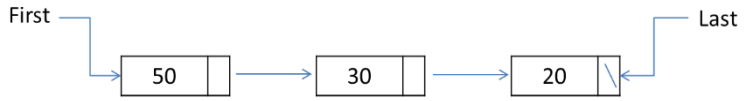
Function median (L: List) → integer

{ IS. List mungkin kosong. Jika list tidak kosong, maka data dalam list sudah terurut.

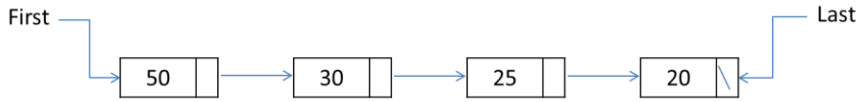
FS. Mengembalikan nilai median dari list, jika list tidak kosong. Jika list kosong, maka kembalikan 0.

Note: Gunakan function hitungElemen yang sudah dibuat sebelumnya.

Ilustrasi:



Median = 30



Median = (30 + 25) / 2 = 27.5

}