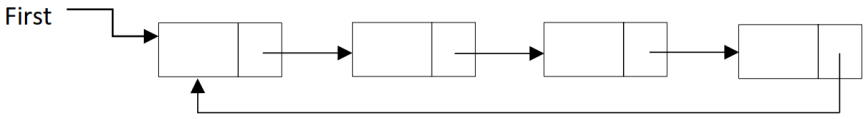
JURNAL PRAKTIKUM MODUL 8

MK STRUKTUR DATA 2021/2022-1

Circular Single Linked-List dengan Pointer First

Sebuah Circular Linked List dengan pointer kepala First digunakan untuk menyimpan data berupa informasi proses/aplikasi. Berikut ilustrasinya:



Buatlah ADT nya (SLL_Circular.h, SLL_Circular.cpp, dan Test_SLL_Circular.cpp/Main.cpp)!

1. Spesifikasi (Silakan ditulis ulang dalam Bahasa C++) (10 Poin)

```
Type infotype <
      nama : string
      prioritas : integer
      sisa_durasi : integer
>
Type address : pointer to elmList
Type elmList <
       info : infotype
      next : address
>
Type List <
      First : 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 P : address)
Procedure printList(input L : List)
Function panjangList(input L : List)⟶integer
Procedure insertAplikasi(input/output L : List, input dataBaru : infotype)
Function eksekusi(input P : address, durasi : integer) ⟶ integer
Procedure eksekusiMulti(input/output current : address, input N,duration : integer)
```

2. Implementasi (Silakan ditulis ulang dalam Bahasa C++) (20 Poin)

```
Procedure createList (input/ouput L : List)
{IS. –
 FS. Terbentuk sebuah list di mana, first dari L bernilai NIL. }
Kamus
Algoritma
    // Sama seperti Single Linked List

Function createElemen (dataBaru: infotype) ⟶ address
{Return alamat alokasi memori sebuah elmList yang berisi dataBaru. }
Kamus
    P: address
Algoritma
    // Sama seperti Single Linked List

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
        next(P) ← First(L)
    else
        next(P) ← First(L)
        First(L) ← P
        // lakukan looping mencari elemen terakhir yaitu next(Q) == next(P)
        next(Q) ← First(L)
    endif

Procedure insertLast (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 terakhir. }
Kamus
Algoritma
    if First (L) == NIL then
        First (L) ← P
        next(P) ← First(L)
    else
        //lakukan looping mencari elemen terakhir yaitu next(Q) == First(L)
        next(Q) ← P
        next(P) ← First(L)
    endif


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
    //Sama seperti Single Linked List
```

```
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)) == First(L) then
        First (L) ← NIL
        next(P) ← NIL
    else
        First (L) ← next (First (L))
        next (P) ← NIL
        //lakukan looping mencari elemen terakhir yaitu next(Q) == P
        next(Q) ← First(L)
    endif

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}
Kamus
Algoritma
Q ← First(L)
// lakukan looping agar pointer Q menunjuk ke elemen kedua terakhir (sebelum terakhir)
P ← next(Q)
next(P) ← NIL
next(Q)← First(L)

Procedure deleteAfter (input Prec: address, output P : address)
{IS. Terdefinisi pointer Prec berisi alamat elmList. Prec bukan elemen terakhir dan
next(Prec) juga bukan elemen terakhir.
FS. P berisi alamat elmList setelah Prec, elmList yang ditunjuk oleh P dihapus dari list}
Kamus
Algoritma
    //Sama seperti Single Linked List


Procedure printList (input L : list);
{   IS. Terdefinisi sebuah list L
    FS. Menampilkan semua info elmList di list. }
Kamus
Algoritma
    P ← First (L)
    if P!=NIL then
        while next(P)!= First(L) do
                output (info (P))
                P ← next (P)
        endwhile
        output(info(P)) //output elemen terakhir
    end
```

3. TUGAS TERBIMBING (40 Poin)

Kali ini anda akan membantu Operating System untuk mengimplementasikan sistem manajemen eksekusi aplikasi/proses. Karena biasanya user menjalankan banyak aplikasi bersamaan di PC, maka perlu diatur sedemikian sehingga ada mekanisme untuk menyimpan beberapa aplikasi/proses tersebut untuk selanjutnya dieksekusi oleh CPU secara bergantian. Anda akan menggunakan Circular Linked List karena anda akan memberikan waktu eksekusi sedikit demi sedikit secara bergantian, terus menerus sampai suatu aplikasi selesai (sisa_durasi <= 0) dan harus **dikeluarkan** dari list.

Selanjutnya anda juga perlu menyiapkan mekanisme untuk menambahkan aplikasi/proses baru ke dalam list. Suatu Aplikasi memiliki info nama, prioritas, dan sisa_durasi. Selain Panjang list yang **belum maksimum**, **prioritas** juga menentukan apakah suatu aplikasi bisa ditambahkan ke list atau tidak. Jika list **sudah maksimum**, namun aplikasi baru memiliki **prioritas lebih tinggi** dari setidaknya 1 aplikasi yang ada di list, maka aplikasi dengan **prioritas terendah** akan **dihapus** dari list, dan aplikasi baru ditambahkan secara **descending**.

Function eksekusi(**input** P : address, durasi : **integer**) ⟶ **integer**
{IS. Terdefinisi pointer P berisi alamat elmList, dan durasi eksekusi CPU (>0)
FS. sisa_durasi P berkurang sebanyak durasi, sisa_durasi dikembaikan oleh fungsi}

Function panjangList(**input** L : List) ⟶ **integer**
{IS. Terdefinisi sebuah list L (mungkin kosong)
FS. Dikembaikan jumlah elemen di dalam list L}

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) berdasarkan prioritas.
Note: Gunakan procedure insertFirst, insertLast, dan insertAfter yang sudah dibuat sebelumnya. }

Procedure insertAplikasi(**input/output** L : List, **input** dataBaru : infotype)
{IS. Terdefinisi sebuah list L (mungkin kosong), mungkin penuh (jumlah elemen = max_applications). Terdefinisi sebuah data.
FS. jika memungkinkan (list belum maksimum atau prioritas dataBaru > prioritas terendah di array), dataBaru ditambahkan ke list. Jika ada data prioritas terendah yang harus dihapus, maka info data yang dihapus dari list diprint ke layar.
Note: Gunakan panjangList, deleteLast, dan insertDescending.}

Procedure deleteElm(**input/ouput** L : List, **input** P : address)
{IS. Terdefinisi sebuah list L (L tidak kosong), dan address elemen yang akan dihapus.
FS. Elemen yang ditunjuk oleh P dihapus dari list.
Note: Gunakan procedure deleteFirst, deleteLast, dan deleteAfter yang sudah dibuat sebelumnya.}

Procedure eksekusiMulti(**input/output** current : address, **input** N, duration : integer)
{IS : terdefinisi posisi aplikasi (elemenList) yang akan mulai di eksekusi. Terdefinisi jumlah aplikasi yang akan dieksekusi CPU (N). Terdefinisi durasi eksekusi untuk setiap aplikasi (durasi).
FS : sisa_durasi setiap aplikasi yang dieksekusi akan berkurang (mungkin dihapus jika telah habis), posisi pointer current menunjuk ke posisi terakhir aplikasi yang masih ada sisa durasi (tidak dihapus dari list). }
Note : Gunakan function/procedure eksekusi, deleteElm}

JURNAL PRAKTIKUM MODUL 8

MK STRUKTUR DATA 2021/2022-1

Circular Single Linked-List dengan Pointer First

Program Utama / **Main**

// Set constant max_applications (list) = 5

// Insert 5 aplikasi(elemenList) [<'word', 5, 10>, <'excel', 4, 20>, <'chrome', 8, 5>, <'wa', 7, 10>, <'dota', 10, 30>]terurut descending sesuai dengan prioritas. Prioritas tertinggi ada di depan (first).

// PrintInfo

// Insert sebuah aplikasi baru <'power point', 6, 15>

// PrintInfo

// Lakukan eksekusiMulti misal untuk N=3, dan durasi=10, current = First(L)

// PrintInfo

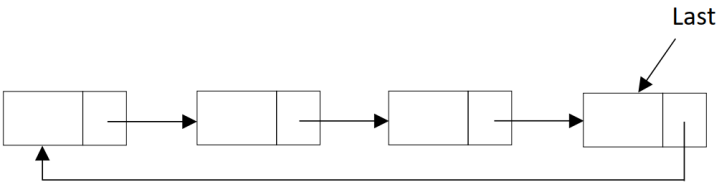
// Lakukan eksekusiMulti misal untuk N=3, dan durasi=10, current = melanjutkan dari yang sebelumnya

// PrintInfo

4. TUGAS MANDIRI (50 Menit) (30 Poin)

Modifikasi struktur list menjadi seperti ini (List < Last : Address >). Lalu modifikasi Procedure insertFirst dan insertLast. Jelaskan apa perbedaannya implementasinya!

Hint : First bisa didapatkan dari next(Last(L)).




```
Procedure createList (input/ouput L : List)
Kamus
Algoritma
        Last(L) <- NULL

Procedure printList (input L : List)
Kamus
Algoritma
        if Last(L) = NULL
                print("List kosong")
        else
                address P <- next(Last(L))
                print(info(P).nama)
                print(info(P).prioritas)
                print(info(P).sisa_durasi)
                P <- next(P)
                While (P != next(Last(L)))
                        print(info(P).nama)
                        print(info(P).prioritas)
                        print(info(P).sisa_durasi)
                        P <- next(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. }

Procedure insertLast (**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 terakhir. }

Jelaskan apa perbedaannya yang paling terlihat pada implementasi insertFirst dan insertLast antara menggunakan pointer First dan menggunakan pointer Last dalam struktur List!