Praktikum StrukturData
Modul PSDA

PSDA-02. ADT QUEUE

Bagian 1. Representasi Tabel Kontigu Alokasi Dinamik - Alternatif 1

1. Buatlah **ADT Queue** sesuai spesifikasi di bawah ini yang diimplementasikan dengan tabel kontigu dengan **alokasi dinamik** dalam bahasa C dengan representasi alternatif I yaitu dengan TAIL adalah indeks elemen terakhir dan HEAD selalu diset sama dengan 1 jika queue tidak kosong (lihat kembali diktat "Struktur Data"). Jika queue kosong, HEAD dan TAIL menunjuk indeks 0. Untuk type boolean, gunakan file berisi definisi type boolean secara terpisah (**boolean.h**).
2. Buatlah driver untuk memeriksa apakah seluruh primitif yang didefinisikan telah berjalan dengan baik.

```
{ Modul ADT Queue – Alternatif I }
{ *** Deklarasi Queue yang diimplementasi dengan tabel kontigu *** }
{ *** HEAD dan TAIL adalah alamat elemen pertama dan terakhir  *** }
{ *** Queue mampu menampung MaxEl buah elemen *** }

{ *** Konstanta *** }
constant Nil : integer = 0

{ *** Definisi elemen dan address *** }
type infotype : integer
type address  : integer   { indeks tabel }

{ *** Definisi Type Queue *** }
type Queue : < T     : ...,     { tabel penyimpan elemen, tergantung bahasa }
               HEAD  : address, { alamat penghapusan }
               TAIL  : address, { alamat penambahan  }
               MaxEl : integer  { maksimum banyaknya elemen queue }
             >
{ Definisi Queue kosong: Head = Nil; TAIL = Nil. }
{ Catatan implementasi: T[0] tidak pernah dipakai }

{ Definisi akses dengan Selektor : Isilah dengan selektor yang tepat }

{ *** Predikat Pemeriksaan Kondisi Queue *** }
function IsEmpty (Q : Queue) → boolean
{ Mengirim true jika Q kosong }
function IsFull (Q : Queue) → boolean
{ Mengirim true jika tabel penampung elemen Q sudah penuh yaitu mengandung MaxEl
elemen }
function NBElmt (Q : Queue) → integer
{ Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong. }

{ *** Konstruktor *** }
procedure CreateEmpty (output Q : Queue, input Max : integer > 0)
{ I.S. Max terdefinisi }
{ F.S. Sebuah Q kosong terbentuk dan salah satu kondisi sbb :       }
{      Jika alokasi berhasil, tabel memori dialokasi berukuran Max  }
{      atau : jika alokasi gagal, Q kosong dg Maksimum elemen=0     }
{ Proses : Melakukan alokasi memori dan membuat sebuah Q kosong     }
```

```
{ *** Destruktor *** }
procedure DeAlokasi (input/output Q : Queue)
{ Proses : Mengembalikan memori Q }
{ I.S. Q pernah dialokasi }
{ F.S. Q menjadi tidak terdefinisi lagi, MaxEl(Q) diset 0 }
{ *** Operator-Operator Dasar Queue *** }
procedure Add (input/output Q : Queue, input X : infotype)
{ Proses : Menambahkan  X pada Q dengan aturan FIFO }
{ I.S. Q mungkin kosong, tabel penampung elemen Q TIDAK penuh }
{ F.S. X menjadi TAIL yang baru, TAIL "maju" }
procedure Del (input/output Q : Queue, output X : infotype)
{ Proses: Menghapus elemen pertama pada Q dengan aturan FIFO }
{ I.S. Q tidak kosong }
{ F.S. X = nilai elemen HEAD pada I.S.,
       Jika Queue masih isi : HEAD diset tetap = 1, elemen-elemen setelah HEAD yang
       lama digeser ke "kiri", TAIL = TAIL - 1;
       Jika Queue menjadi kosong, HEAD = TAIL = Nil. }
```

Praktikum StrukturData
Modul PSDA

Bagian 2. Representasi Tabel Kontigu Alokasi Dinamik - Alternatif II

1. Buatlah **ADT Queue** yang diimplementasikan dengan tabel kontigu dengan **alokasi dinamik** dalam bahasa C dengan representasi alternatif II yaitu dengan HEAD dan TAIL yang "bergerak". Setelah TAIL mencapai indeks terakhir, jika terjadi penambahan elemen baru dan tabel belum penuh, semua elemen dimundurkan kembali sampai HEAD = 1 (lihat kembali diktat "Struktur Data"). Jika queue kosong, HEAD dan TAIL menunjuk indeks 0. Untuk type boolean, gunakan file berisi definisi type boolean secara terpisah (**boolean.h**).
   Definisi konstanta, struktur data queue, primitif-primitif yang terkait dengan pemeriksaan kondisi queue, konstruktor, destruktor, dan selektor sama dengan PSDA-02. Bagian 1.
2. Buatlah driver untuk memeriksa apakah seluruh primitif yang didefinisikan telah berjalan dengan baik.

```
{ Modul ADT Queue - Alternatif II }

{ Definisi konstanta, struktur data queue, primitif-primitif yang terkait dengan
pemeriksaan kondisi queue, konstruktor, destruktor, dan selektor sama dengan P-09.
Bagian 1. }

{ *** Operator-Operator Dasar Queue *** }
procedure Add (input/output Q : Queue, input X : infotype)
{ Proses : Menambahkan  X pada Q dengan aturan FIFO }
{ I.S. Q mungkin kosong, tabel penampung elemen Q TIDAK penuh }
{ F.S. X menjadi TAIL yang baru, TAIL "maju". }
{       Jika TAIL = MaxEl, semua elemen digeser mundur terlebih dahulu sampai
        HEAD = 1, baru X ditambahkan pada TAIL }
procedure Del (input/output Q : Queue, output X : infotype);
{ Proses : Menghapus elemen pertama pada Q dengan aturan FIFO }
{ I.S. Q tidak kosong }
{ F.S. X = nilai elemen HEAD pada I.S.,
        Jika Queue masih isi : HEAD "maju".
        Jika Queue menjadi kosong, HEAD = TAIL = Nil. }
```

Praktikum StrukturData
Modul PSDA

Bagian 3. Representasi Tabel Kontigu Alokasi Dinamik - Alternatif III

1. Buatlah **ADT Queue** yang diimplementasikan dengan tabel kontigu dengan **alokasi dinamik** dalam bahasa C dengan representasi alternatif III yaitu dengan HEAD dan TAIL yang "berputar"mengelilingi indeks tabel (lihat kembali diktat "Struktur Data"). Jika queue kosong, HEAD dan TAIL menunjuk indeks 0. Untuk type boolean, gunakan file berisi definisi type boolean secara terpisah (**boolean.h**).
   Definisi konstanta, struktur data queue, primitif-primitif yang terkait dengan pemeriksaan kondisi queue, konstruktor, destruktor, dan selektor sama dengan PSDA-02. Bagian 1.
2. Buatlah driver untuk memeriksa apakah seluruh primitif yang didefinisikan telah berjalan dengan baik.

```
{ Modul ADT Queue - Alternatif III }

{ Definisi konstanta, struktur data queue, primitif-primitif yang terkait dengan
pemeriksaan kondisi queue, konstruktor, destruktor, dan selektor sama dengan PSDA-
02. Bagian 1. }

{ *** Operator-Operator Dasar Queue *** }
procedure Add (input/output Q : Queue, input X : infotype)
{ Proses : Menambahkan  X pada Q dengan aturan FIFO }
{ I.S. Q mungkin kosong, tabel penampung elemen Q TIDAK penuh }
{ F.S. X menjadi TAIL yang baru, TAIL "maju". }
{      Jika TAIL baru = MaxEl + 1, maka TAIL diset = 1. }
procedure Del (input/output Q : Queue, output X : infotype);
{ Proses : Menghapus elemen pertama pada Q dengan aturan FIFO }
{ I.S. Q tidak kosong }
{ F.S. X = nilai elemen HEAD pada I.S.,
       Jika Queue masih isi : HEAD "maju".
       Jika HEAD baru menjadi MaxEl + 1, maka HEAD diset = 1;
       Jika Queue menjadi kosong, HEAD = TAIL = Nil. }
```