Queue dalam Bahasa C

IF2110 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

ADT Queue dengan C – alt-2 (alokasi memori statik)

```c
/* File: queue.h */
#ifndef QUEUE_H
#define QUEUE_H
#include "boolean.h"
#include <stdlib.h>

#define IDX_UNDEF -1
#define CAPACITY 100

/* Definisi elemen dan address */
typedef int ElType;
/* Contoh struktur type Queue: array statik, indeks head dan indeks tail disimpan */
typedef struct {
  ElType buffer[CAPACITY];
  int idxHead;
  int idxTail;
} Queue;
/* Definisi Queue kosong: idxHead = idxTail = IDX_UNDEF. */
```

NOTE: if you want flexibility, buffer should be an ElType* and you will need malloc() in CreateQueue. You will also have to have a destructor, e.g., DestroyQueue(), where you call free(buffer).

ADT Queue dengan C – alt-2 (alokasi memori statik)

```
/********** AKSES (Selektor) **********/
#define IDX_HEAD(q) (q).idxHead
#define IDX_TAIL(q) (q).idxTail
#define     HEAD(q) (q).buffer[(q).idxHead]
#define     TAIL(q) (q).buffer[(q).idxTail]

/********** Konstruktor **********/
void CreateQueue(Queue *q);
/* I.S. Sembarang
   F.S. Membuat sebuah Queue q yang kosong berkapasitas CAPACITY
        jadi indeksnya antara 0..CAPACITY-1
        Ciri Queue kosong: idxHead dan idxTail bernilai IDX_UNDEF */
```

ADT Queue dengan C – alt-2

```
/********** Operasi: pemeriksaan status Queue **********/
/* catatan: fungsi head(q: Queue) diimplementasikan sebagai macro di halaman
            sebelumnya */

boolean isEmpty(Queue q);
/* Mengirim true jika q kosong: lihat definisi di atas */

boolean isFull(Queue q);
/* Mengirim true jika penyimpanan q penuh */

int length(Queue q);
/* Mengirim jumlah elemen q saat ini */
```

ADT Queue dengan C – alt-2

```
/*** Primitif Add/Delete ***/
void enqueue (Queue *q, ElType val);
/* Proses: Menambahkan val sebagai elemen Queue q.
   I.S. queue mungkin kosong, TIDAK penuh */
   F.S. queue bertambah elemen val sebagai tail yang baru, TAIL bergeser ke kanan */
   Jika IDX_TAIL(queue)=CAPACITY-1, maka geser isi tabel, shg IDX_HEAD(queue)=0 */

void dequeue(Queue *q, ElType* val);
/* Menghapus head dari Queue q.
   I.S. queue tidak kosong
   F.S. val berisi nilai head yang lama.
        Jika queue tidak menjadi kosong,
            queue.idxHead berpindah ke elemen berikutnya pada queue.
        Jika queue menjadi kosong,
            queue.idxHead dan queue.idxTail menjadi bernilai IDX_UNDEF. */

#endif
```

ADT Queue dengan C – alt-2

```c
void CreateQueue(Queue *q) {
/* I.S. ... F.S. ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    IDX_HEAD(*q) = IDX_UNDEF;
    IDX_TAIL(*q) = IDX_UNDEF;
}

boolean isEmpty(Queue q) {
/* Mengirim ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    return (IDX_HEAD(q) == IDX_UNDEF) && (IDX_TAIL(q) == IDX_UNDEF);
}
```

ADT Queue dengan C – alt-2

```
boolean isFull(Queue q) {
/* Mengirim ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    return (IDX_HEAD(q) == 0) && (IDX_TAIL(q) == CAPACITY-1);
}

int length(Queue q) {
/* Mengirim ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    if (IDX_HEAD(q)==IDX_UNDEF)
        return 0;
    else
        return (IDX_TAIL(q) - IDX_HEAD(q)) + 1;
}
```

ADT Queue dengan C – alt-2

```
void enqueue(Queue *q, ElType val) {
/* I.S. ... F.S. ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    if (isEmpty(*q)) {
      IDX_HEAD(*q) = 0;
      IDX_TAIL(*q) = 0;
    } else { // *q is not empty
      if (IDX_TAIL(*q)==(CAPACITY-1)) { // elemen mentok kanan, geser dulu
        for (int i=IDX_HEAD(*q); i<=IDX_TAIL(*q); i++) {
          (*q).buffer[i-IDX_HEAD(*q)] = (*q).buffer[i];
        }
        IDX_TAIL(*q) -= IDX_HEAD(*q);
        IDX_HEAD(*q) = 0;
      }
      IDX_TAIL(*q)++;
    }
    TAIL(*q) = val;
}
```

ADT Queue dengan C – alt-2

```
void dequeue(Queue *q, ElType *val) {
/* I.S. ... F.S. ... */
  /* KAMUS LOKAL */
  /* ALGORITMA */
    *val = HEAD(*q);
    if (IDX_HEAD(*q) == IDX_TAIL(*q)) {
      IDX_HEAD(*q) = IDX_UNDEF;
      IDX_TAIL(*q) = IDX_UNDEF;
    } else {
      IDX_HEAD(*q)++;
    }
}
```