Queue
Dalam Bahasa C

ADT Queue dengan C - Alternatif II (Alokasi Memori Dinamis)

```c
/* File: queue.h */
#ifndef queue_H
#define queue_H
#include "boolean.h"
#include <stdlib.h>
#define Nil -1
/* Definisi elemen dan address */
typedef int infotype;
typedef int address; /* indeks tabel */
/* Contoh deklarasi variabel bertype Queue: */
/* Versi I: tabel dinamik, Head dan Tail eksplisit,
   ukuran disimpan */
typedef struct {
    infotype *T;  /* tabel penyimpan elemen */
    address HEAD; /* alamat penghapusan */
    address TAIL; /* alamat penambahan */
    int MaxEl;    /* Max elemen queue */
} Queue;
/* Definisi Queue kosong: HEAD=Nil; TAIL=Nil. */
```

ADT Queue dengan C - Alternatif II

```
/********** AKSES (Selektor) **********/
/* Jika Q adalah Queue, maka akses elemen : */
#define Head(Q) (Q).HEAD
#define Tail(Q) (Q).TAIL
#define InfoHead(Q) (Q).T[(Q).HEAD]
#define InfoTail(Q) (Q).T[(Q).TAIL]
#define MaxEl(Q) (Q).MaxEl

/********** Prototype **********/
boolean IsEmpty (Queue Q);
/* Mengirim true jika Q kosong: lihat definisi di atas */
boolean IsFull(Queue Q);
/* Mengirim true jika tabel penampung elemen Q sudah penuh */
/* yaitu mengandung elemen sebanyak MaxEl */
int NBElmt(Queue Q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q
   kosong */
```

ADT Queue dengan C - Alternatif II

```
/*** Kreator ***/
void CreateEmpty(Queue *Q, int Max);
/* I.S. sembarang */
/* 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 MaxEl=0 */
/* Proses : Melakukan alokasi,Membuat sebuah Q kosong */

/*** Destruktor ***/
void DeAlokasi(Queue *Q);
/* Proses: Mengembalikan memori Q */
/* I.S. Q pernah dialokasi */
/* F.S. Q menjadi tidak terdefinisi lagi, MaxEl(Q) diset 0 */
```

ADT Queue dengan C - Alternatif II

```
/*** Primitif Add/Delete ***/
void Enqueue (Queue * Q, infotype X);
/* 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 bergeser ke kanan */
/* Jika Tail(Q)=MaxEl-1, maka geser isi tabel, shg Head(Q)=0 */

void Dequeue(Queue * Q, infotype* X);
/* Proses: Menghapus X pada Q dengan aturan FIFO */
/* I.S. Q tidak mungkin kosong */
/* F.S. X = nilai elemen HEAD pd I.S., HEAD bergeser ke kanan;
        Q mungkin kosong */

#endif
```

ADT Queue dengan C - Alternatif II

```
boolean IsEmpty (Queue Q) { /* Mengirim true jika Q kosong: ... */
  /* Kamus Lokal */

  /* Algoritma */
  return ((Head(Q)==Nil) && (Tail(Q)==Nil));
}
boolean IsFull (Queue Q) { /* Mengirim true jika tabel ... */
  /* Kamus Lokal */

  /* Algoritma */
  return ((Head(Q)==0) && (Tail(Q)==MaxEl(Q)-1));
}
int NBElmt(Queue Q) { /* Mengirimkan banyaknya elemen queue. ... */
  /* Kamus Lokal */

  /* Algoritma */
  return (Tail(Q)-Head(Q)+1);
}
```

ADT Queue dengan C - Alternatif II

```
void CreateEmpty(Queue *Q, int Max) {
/* I.S. sembarang */
/* 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 MaxEl=0 */
/* Proses : Melakukan alokasi, Membuat sebuah Q kosong */

    /* Kamus Lokal */

    /* Algoritma */
    (*Q).T = (infotype *) malloc (Max * sizeof(infotype));
    if ((*Q).T != NULL) {
        MaxEl(*Q) = Max;
        Head(*Q) = Nil;
        Tail(*Q) = Nil;
    } else /* alokasi gagal */ {
        MaxEl(*Q) = 0;
    }
}
```

ADT Queue dengan C - Alternatif II

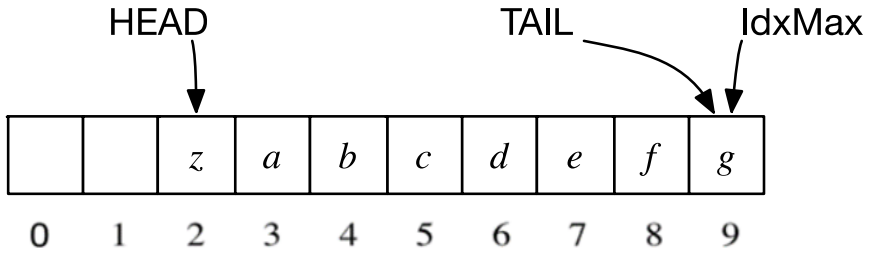
```
void DeAlokasi(Queue *Q) {
/* Proses: Mengembalikan memori Q */
/* I.S. Q pernah dialokasi */
/* F.S. Q menjadi tidak terdefinisi lagi, MaxEl(Q) diset 0 */

    /* Kamus Lokal */

    /* Algoritma */
    MaxEl(*Q) = 0;
    free((*Q).T);
}
```

ADT Queue dengan C - Alternatif II

```
void Enqueue(Queue *Q, infotype X) {
/* Proses: ... I.S.: ... F.S.: ... */
    /* Kamus Lokal */
    address i, j;
    /* Algoritma */
    if (IsEmpty(*Q)) {
        Head(*Q)=0;
    } else /* Q tidak kosong */ {
        if (Tail(*Q)==MaxEl(*Q)-1) { /* Geser elemen smp Head(Q)=0 */
            i = Head(*Q); j = 0;
            do {
                *((*Q).T+j) = *((*Q).T+i);
                i++; j++;
            } while (i<=Tail(*Q));
        }
        Tail(*Q) = NBElmt(*Q)-1; Head(*Q) = 0;
    }
    Tail(*Q)++;
    InfoTail(*Q)=X;
}
```



ADT Queue dengan C - Alternatif II

```
void Dequeue(Queue * Q, infotype* X) {
/* Proses: Menghapus X pada Q dengan aturan FIFO */
/* I.S. Q tidak mungkin kosong */
/* F.S. X = nilai elemen HEAD pd I.S.,HEAD bergeser ke kanan;
   Q mungkin kosong */

    /* Kamus Lokal */

    /* Algoritma */
    *X = InfoHead(*Q);
    if (Head(*Q)==Tail(*Q)) { /* Set mjd queue kosong */
        Head(*Q)=Nil;
        Tail(*Q)=Nil;
    } else {
        Head(*Q)++;
    }
}
```