Praktikum 5

Antrian (Queue)

POKOK BAHASAN:

- ✓ Konsep antrian
- ✓ Struktur antrian
- ✓ Implementasi antrian dalam Bahasa C

TUJUAN BELAJAR:

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami stuktur data yang digunakan pada antrian baik yang berupa array maupun *linked list*
- ✓ Memahami karakteristik antrian dan kondisi yang ada yaitu queue dalam keadaan kosong atau penuh
- ✓ Membuat diagram alir dan mengimplementasikan antrian pada suatu permasalahan

DASAR TEORI:

Antrian (*queue*) adalah konsep penyimpanan sekumpulan data yang sering digunakan. Pada antrian data dapat disimpan dalam array atau *linked list*. Pada antrian, penunjuk posisi depan dan belakang harus disimpan. Apabila akan menambah data pada antrian maka menaikkan penunjuk belakang dan meletakkan data pada posisi tersebut. Apabila akan menghapus data pada antrian maka dihapus pada posisi depan dan menaikkan penunjuk depan.

Karakteristik penting dari antrian adalah :

1. Elemen antrian yaitu item-item data yang terdapat di elemen antrian
2. **Front** (penunjuk elemen terdepan dari antrian)

3. **Rear** (penunjuk elemen terakhir dari antrian)
4. **Count** (Jumlah elemen pada antrian)
5. Kondisi antrian

Kondisi antrian yang harus diperhatikan adalah ;

1. Penuh

Bila elemen pada antrian mencapai kapasitas maksimum antrian. Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian. Penambahan elemen menyebabkan kondisi kesalahan Overflow.

2. Kosong

Bila tidak ada elemen pada antrian. Pada kondisi ini, tidak mngkin dilakukan pengambilan elemen dari antrian. Pengambilan elemen menyebabkan kondisi kesalahan Overflow.

1. ANTRIAN DENGAN ARRAY

Pada implementasi antrian dengan array, digunakan sejumlah array `MAX` untuk menyimpan data. Untuk menunjuk bagian depan dan bagian belakang digunakan variable `Front` dan `Rear`. Bila antrian kosong, nilainya diset -1. Untuk operasi penambahan dan penghapusan diimplementasikan dua fungsi yaitu `Tambah()` dan `Hapus()`.

Pada saat menambah elemen baru pada antrian, pertama kali dicek apakah penambahan tersebut dimungkinkan atau tidak. Karena indeks array dimulai dengan 0 maka maksimum data yang dapat disimpan pada antrian adalah `MAX-1`. Jika semua elemen menempati ruang array maka antrian dalam kondisi penuh. Apabila data masih dapat ditambahkan pada antrian maka variable `Rear` dinaikkan satu dan data baru disimpan pada array. Apabila data baru ditambahkan ke antrian untuk pertama kali (dimana variable `Front` bernilai -1) maka variable `Front` diset 0 yang menandakan antrian tidak lagi kosong.

Sebelum menghapus elemen dari antrian harus dipastikan apakah elemen tersedia untuk penghapusan. Jika tidak maka antrian dalam kondisi kosong. Sebaliknya bila tersedia data pada array maka dapat dilakukan penghapusan dan variable `Front`

dinaikkan. Apabila nilai variable `Front` dan `Rear` sama (yang berarti antrian dalam keadaan kosong) maka `Front` dan `Rear` direset -1.

Misalnya akan dilakukan penambahan data pada antrian sampai semua array terisi. Pada kondisi ini nilai `Rear` menjadi `MAX-1`. Misalnya dilakukan penghapusan 5 elemen, maka antrian dikatakan dalam kondisi penuh meskipun 5 array pertama kosong. Untuk mengatasi kondisi ini maka diimplmentasikan antrian sebagai antrian sirkular (*circular queue*). Sehingga selama penambahan, jika sudah mencapai akhir array dan jika awal array kosong (sebagai akibat dari penghapusan) maka elemen baru ditambahkan pada awal array.

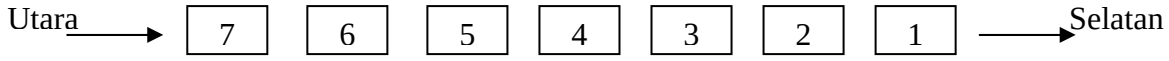
2. ARRAY DENGAN LINKED LIST

Array yang digunakan untuk mengimplementasikan antrian dideklarasikan mempunyai ukuran MAX. Ukuran ini ditentukan pada saat menulis program, tidak dapat diubah pada saat program berjalan. Sehingga pada saat menulis program, harus ditentukan ukuran memori maksimum yang diperlukan untuk membangun array. Hal ini berhugungan dengan deklarasi ruang memori yang tersedia. Jika jumlah elemen yang dapat disimpan dalam antrian kecil, maka banyak ruang memori yang tidak digunakan. Sebaliknya, jika jumlah elemen yang akan disimpan pada antrian terlalu banyak, maka menyebabkan overflow. Untuk menghindari hal tersebut terjadi dan keterbatasan memory maka digunakan struktur data yang disebut *linked list*.

3. STUDI KASUS : SIMULASI TEMPAT PARKIR

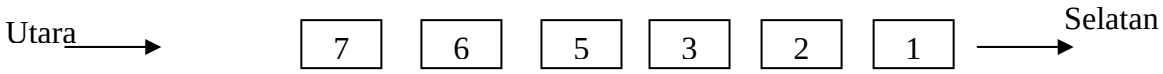
- Ada suatu tempat parkir yang hanya bisa memuat mobil dalam satu baris, jumlah mobil yang bisa masuk dibatasi 10 mobil.
- Mobil masuk lewat pintu Utara (belakang) dan keluar lewat pintu Selatan (depan). Setiap mobil yang masuk diberi nomor.
- Jika mobil yang berada paling depan (di sisi paling Selatan) akan keluar maka mobil tersebut segera bisa keluar. Tetapi jika mobil yang akan keluar adalah mobil yang di tengah, maka mobil yang terletak di depannya (di sebelah Selatan) harus dikeluarkan sementara.

- Setelah mobil yang dimaksud keluar, maka mobil yang dikeluarkan sementara tadi dimasukkan kembali ke tempat parkir dengan susunan seperti semula. Sehingga mobil yang semula berada paling depan tetap berada pada posisinya semula. Mobil-mobil yang terletak di sebelah Utaranya mobil yang keluar tadi digeser maju ke depan, sehingga bagian kosong selalu berada pada pintu Utara (belakang).



Mobil Nomor 4 keluar, maka mobil nomor 1, 2 dan 3 dikeluarkan sementara

Mobil nomor 1, 2 dan 3 diletakkan di tempat semula, mobil nomor 5, 6 dan 7 digeser ke selatan.



Dengan memperhatikan persoalan di atas, sebenarnya simulasi ini selain berisi antrian (*queue*) juga berisi tumpukan (*stack*), yakni pada saat sebuah mobil yang berada di tengah akan dikeluarkan, maka semua mobil yang ada di sebelah Selatannya ditumpuk (di-push) di tempat penampungan sementara. Baru setelah mobil yang dimaksud keluar, maka semua mobil yang berada di tempat penampungan sementara di-pop kembali. Dengan cara ini semua mobil akan mempunyai posisi yang sama seperti sebelum suatu mobil dikeluarkan dari tempat parkir.

TUGAS PENDAHULUAN:

1. Buatlah flowchart untuk operasi penambahan dan penghapusan data pada antrian dengan menggunakan array.
2. Buatlah flowchart untuk operasi penambahan dan penghapusan data pada antrian dengan menggunakan *circular queue*.
3. Buatlah flowchart untuk operasi penambahan dan penghapusan data pada antrian dengan menggunakan linked list.

PERCOBAAN:

1. Buatlah workspace menggunakan Visual C++.
2. Buatlah project untuk praktikum KELIMA.
3. Cobalah untuk masing-masing percobaan di bawah.
4. Implementasikan flowchart simulasi tempat parker yang Anda buat pada Tugas Pendahuluan.

Percobaan 1 : Implementasi antrian dengan array

```
#include <stdio.h>
#define MAX 10

typedef struct {
      int Item[MAX];
      int Front;
      int Rear;
      int Count;
} Queue;

// Inisialisasi antrian
void Inisialisasi(Queue *q)
{
      q->Front = q->Rear = -1;
      q->Count = 0;
}

// Prosedur untuk menyisipkan data pada antrian
void Tambah(Queue *q, int item)
{
      if (q->Rear == MAX-1)
      {
            printf ("\nAntrian Penuh");
            return ;
      }
      q->Rear++;
      q->Item[q->Rear] = item;
      q->Count++;
      if (q->Front == -1) q->Front = 0;
}
```

```
// Prosedur untuk menghapus data dari antrian
int Hapus(Queue *q)
{
      int data ;
      if (q->Front == -1)
      {
            printf ("\nAntrian Kosong");
            return NULL;
      }
      data = q->Item[q->Front];
      q->Count--;
      if (q->Front == q->Rear)
            q->Front = q->Rear = -1;
      else
            q->Front++;
      return data;
}

void Tampil(Queue *q)
{
      for(int i=0; i<q->Count; i++)
            printf("\nData : %d", q->Item[i]);
}

void main()
{
      Queue q;
      int data;
      Inisialisasi(&q);
      Tambah(&q,11);
      Tambah(&q,12);
      Tambah(&q,13);
      Tambah(&q,14);
      Tambah(&q,15);
      Tambah(&q,16);
      Tambah(&q,17);
      Tambah(&q,18);
      Tambah(&q,19);
      Tambah(&q,20);
      Tambah(&q,21);
      Tampil(&q);

      data = Hapus(&q);
      printf("\nHapus Item = %d ", data);

      data = Hapus(&q);
      printf("\nHapus Item = %d ", data);

      data = Hapus(&q);
      printf("\n Hapus Item = %d ", data);
      Tampil(&q);
}
```

Percobaan 2 : Implementasi *circular queue*

```
#include <stdio.h>
#define MAX 10

typedef struct {
      int Item[MAX];
      int Front;
      int Rear;
      int Count;
} Queue;

// Inisialisasi antrian
void Inisialisasi(Queue *q)
{
      q->Front = q->Rear = -1;
      q->Count = 0;
}

// Prosedur untuk menyisipkan data pada antrian
void Tambah(Queue *q, int item)
{
      if ((q->Rear == MAX-1 && q->Front == 0) || (q->Rear + 1 == q-
>Front))
      {
            printf ("\nAntrian Penuh");
            return ;
      }
      if (q->Rear == MAX - 1)
            q->Rear = 0;
      else
            q->Rear++;
      q->Item[q->Rear] = item;
      q->Count++;
      if (q->Front == -1) q->Front = 0;
}
```

```
// Prosedur untuk menghapus data dari antrian
int Hapus(Queue *q)
{
      int data ;
      if (q->Front == -1)
      {
            printf ("\nAntrian Kosong");
            return NULL;
      }
      else
      {
            data = q->Item[q->Front];
            q->Count--;
            if (q->Front == q->Rear)
                  q->Front = q->Rear = -1;
            else
            {
                  if (q->Front == MAX-1)
                        q->Front = 0;
                  else
                        q->Front++;
            }
      }
      return data;
}

void Tampil(Queue *q)
{
      for(int i=0; i<q->Count; i++)
            printf("\nData : %d", q->Item[i]);
}

void main()
{
      Queue q;
      int data;
      Inisialisasi(&q);
      Tambah(&q,11);
      Tambah(&q,12);
      Tambah(&q,13);
      Tambah(&q,14);
      Tambah(&q,15);
      Tambah(&q,16);
      Tambah(&q,17);
      Tambah(&q,18);
      Tambah(&q,19);
      Tambah(&q,20);
      Tambah(&q,21);
      Tampil(&q);
      data = Hapus(&q);
      printf("\nHapus Item = %d ", data);
      data = Hapus(&q);
      printf("\nHapus Item = %d ", data);
      data = Hapus(&q);
      printf("\nHapus Item = %d ", data);
      Tampil(&q);
}
```

Percobaan 3 : Implementasi antrian dengan linked list

```
#include<stdio.h>
#include<malloc.h>

typedef struct node
{
      int Item;
      struct node *link;
}Node;

typedef struct {
      Node *Front;
      Node *Rear;
}Queue;

// Prosedur untuk menyisipkan data pada antrian
void Tambah(Queue *Q, int y)
{
      Node *ptr;
      ptr=(Node *) malloc(sizeof(Node));
      ptr->Item=y;
      ptr->link=NULL;
      if(Q->Front ==NULL)
      {
            Q->Front = Q->Rear = ptr;
      }
      else
      {
            Q->Rear->link=ptr;
            Q->Rear=ptr;
      }
}

// Prosedur untuk menghapus data dari antrian
int Hapus(Queue *Q)
{
      int num;
      if(Q->Front==NULL)
      {
            printf("\n\n\t\tAntrian Kosong\n\n");
            return(0);
      }
      else
      {
            num=Q->Front->Item;
            Q->Front = Q->Front->link;
            if(Q->Front == NULL)
                  Q->Rear = NULL;
            //printf("\nNilai yang dihapus : %d \n",num);
            return(num);
      }
}
```

```
void Tampil(Node *N)
{
      printf ("\nFront -> ");

      while (N != NULL)
      {
            if(N->link == NULL)
            {
                  printf("%5d", N->Item);
                  printf("   <- Rear");
            }
            else
                  printf("%5d", N->Item);
            N = N->link;
      }
}

int Count(Node *N)
{
      int c=0;
      while (N != NULL)
      {
            N = N->link;
            c++;
      }
      return c;
}

void main()
{
      int item;
      Queue Q;

      Q.Front = Q.Rear = NULL;
      Tambah(&Q, 11);
      Tambah(&Q, 12);
      Tambah(&Q, 13);
      Tambah(&Q, 14);
      Tambah(&Q, 15);
      Tambah(&Q, 16);
      Tambah(&Q, 17);

      Tampil(Q.Front);
      printf("\nJumlah data pada antrian : %d", Count(Q.Front));

      item=Hapus(&Q);
      printf("\nNilai yang dihapus : %d",item);
      item=Hapus(&Q);
      printf("\nNilai yang dihapus : %d",item);
      item=Hapus(&Q);
      printf("\nNilai yang dihapus : %d",item);

      Tampil(Q.Front);
      printf("\nJumlah data pada antrian : %d", Count(Q.Front));
}
```

LATIHAN:

1. Implementasikan studi kasus simulasi tempat parkir dengan ketentuan sebagai berikut :
   a. Gunakan struktur data dengan elemen data berupa Nomor mobil.
   b. Gunakan struktur data array dan linked list.
   c. Buatlah sebuah flowchart yang meliputi proses
      - Mobil masuk.
      - Mobil paling depan keluar.
      - Mobil tengah keluar disertai proses penampungan mobil sementara, memasukkan mobil dari penampungan sementara dan menyusun tempat parkir kembali.
2. Berikan kesimpulan dari percobaan dan latihan yang telah Anda lakukan.