Pertemuan 10

Priority QUEUE

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- mendemonstrasikan TDA Queue .
- menerapkan TDA Queue pada program aplikasi komputer

Outline Materi

- Basis Operasi
- Representasi
  - Set
  - List
- Implementasi
  - Array
  - Linked List

Basis Operasi

Basis HPIFO (Highest Priority In First Out), elemen yang diambil / dihapus adalah elemen yang mempunyai prioritas tertinggi (waktu kedatangan tidak menjadi penentu).

Priority QUEUE dibedakan atas 2 tipe :

- **Ascending Priority : Queue diurutkan dengan prioritas yang menaik.**
- **Descending Priority : Queue diurutkan dengan prioritas yang menurun.**

Representasi

Representasi Priority QUEUE

- **SET**
  - **EnQueue tergantung kepada waktu kedatangan.**
  - **EnQUEUE mudah, 1 step**
  - **DeQUEUE susah, lama karena harus dilakukan pencarian terhadap data yang mempunyai prioritas tertinggi.**
- **LIST**
  - **EnQueue berdasarkan prioritasnya.**
  - **EnQUEUE susah, lama karena harus mencari posisi yang tepat untuk meletakkan data, sehingga Queue tetap urut berdasarkan prioritasnya.**
  - **DeQUEUE mudah, karena elemen yang berada paling depan adalah elemen yang mempunyai prioritas tertinggi / terendah**.

Implementasi dg Array

```
#include <stdio.h>
 #define PJG_MAX 20

typedef int DataType;
struct Elemen{
    DataType data;
    int priority;
};
typedef struct Elemen ElemenType;
int Jum_El, Head;


ElemenType Q[PJG_MAX];
```

Implementasi dg Array(2)

```
void create()
{
    int Jum_El=0;
    int Head = 0;
}
int empty()
{
   if(Jum_El == 0) return(1);
   else return(0);
}
int full()
{
    if(Jum_El == PJG_MAX - 1) return(1);
    else return(0);
}
```

Implementasi dg Array(3)

```
void move_queue_down(int awal, int akhir)
{   int i;
    for(i=akhir; i >= awal;i--) Q[i] = Q[i-1];
}
void enqueue(ElemenType e)
{     int Pos;
      Pos = Head;
      if(!full()){
           while ((e.priority < Q[Pos].priority) && (Jum_El >
             Pos))
                Pos=Pos+1;
           if(Jum_El > Pos)
           move_queue_down(Pos+1,Jum_El);
           Q[Pos] = e;
           Jum_El= Jum_El + 1;
      }
}
```

Implementasi dg Array(4)

```
void move_queue_up(int awal, int akhir)
{   int i;
    for(i=awal; i <= akhir; i++) Q[i]=Q[i+1];
}
void dequeue(ElemenType *e)
{   if(!empty()) {
        *e=Q[Head];
        Jum_El = Jum_El - 1;
        if(!empty())move_queue_up(Head,Jum_El-1);
    }
}
```

Implementasi dg Array(4)

```
void main()
{   int i;
    ElemenType j,k;
    create();
    k.data = 100; k.priority = 0; enqueue(k);
    k.data = 200; k.priority = 6; enqueue(k);
    k.data = 50; k.priority = 3; enqueue(k);
    k.data = 140; k.priority = 10; enqueue(k);
    k.data = 900; k.priority = 7; enqueue(k);
    k.data = 180; k.priority = 0; enqueue(k);

    for(i=20; i <= 40; i+=5) {
       k.data = i; k.priority = 5;
```

Implementasi dg Array(4)

```
    printf("Lihat isi Antrian Prioritas\n\n);
    printf("================================\n");
    printf("          Data     Prioritas\n");
    printf("================================\n");
     While(!empty()) {
        dequeue(&j);
        printf("       %5d         %5d  \n", j.data, j.priority);
      }
}
```

Implementasi dg Linked List

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>

typedef int priority_type;
typedef char data_type;

struct pQueueNode{
    priority_type key;
    data_type data;
    pQueueNode *next; };

struct pQueuePtr{
    pQueueNode *head, *current, *tail;
} PQ;

typedef pQueuePtr *PQueuePtr;
```

Implementasi dg Linked List(2)

```
void create(PQueuePtr *PQ)
{
    (*PQ)->head=(*PQ)->current=(*PQ)->tail=NULL;
}

int empty(PQueuePtr *PQ)
{
    if ((*PQ)->head==NULL)
        return(1);
    else return(0);
}

void FindFirst(PQueuePtr *PQ)
{   (*PQ)->current = (*PQ)->head; }

void FindNext(PQueuePtr *PQ)
{
    if(((*PQ)->current)->next!=NULL)
        (*PQ)->current = ((*PQ)->current)->next;
}
```

Implementasi dg Linked List(3)

```
void enqueue(PQueuePtr *PQ, data_type N, priority_type K)
{     pQueueNode *p;
    p=(pQueueNode *)malloc(sizeof(struct pQueueNode));
    p->data=N;
    p->key=K;
    p->next=NULL;
    if (empty(PQ))            { (*PQ)->head = (*PQ)->tail = p;     }
    else
    {       FindFirst(PQ);
            while((*PQ)->current->key<(p->key)&&
                    (*PQ)->current->next->key<(p->key)&&(*PQ)->current-
    >next!=NULL)
                        FindNext(PQ);
            if ((*PQ)->current->key>p->key)    // Insert di Head
            {           p->next = (*PQ)->current;
                        (*PQ)->head = p; }
            else {      p->next = (*PQ)->current->next; // Insert selain di Head
                        (*PQ)->current->next   = p;  };
    };
    (*PQ)->current = p;
}
```

Implementasi dg Linked List(4)

```
void dequeue(PQueuePtr *PQ, data_type N)
{   pQueueNode *p;
    if (!empty(PQ))
    {
          p=(*PQ)->head;
          N=p->data;
          (*PQ)->head = (*PQ)->head->next;
          free(p);
    }
    (*PQ)->current = (*PQ)->head;
}
```

Implementasi dg Linked List(4)

```
void main()
{   int i;
    char x;
    Create();
     enqueue(PQ, “C”, 10);
     enqueue(PQ, “E”, 3);
     enqueue(PQ, “D”, 8);
     enqueue(PQ, “W”, 5);

     for(i=20; i <= 40; i+=5) {
        enqueue(PQ, “Z”, I);
     }
```

Implementasi dg Linked List(4)

```
printf("Lihat isi Antrian Prioritas\n\n);
    printf("============================\n");
    printf("          Data     Prioritas\n");
    printf("============================\n");
     While(!empty()) {
        dequeue(PQ, x);
        printf("       %5s         %5d  \n", x, PQ.priority);
      }
}
```

Selesai