PRAKTIKUM ALGORITMA & STRUKTUR DATA

"QUEUE"

DOSEN :

Umi Sa'adah S.Kom, M.Kom (197404162000032003)

PENYUSUN :

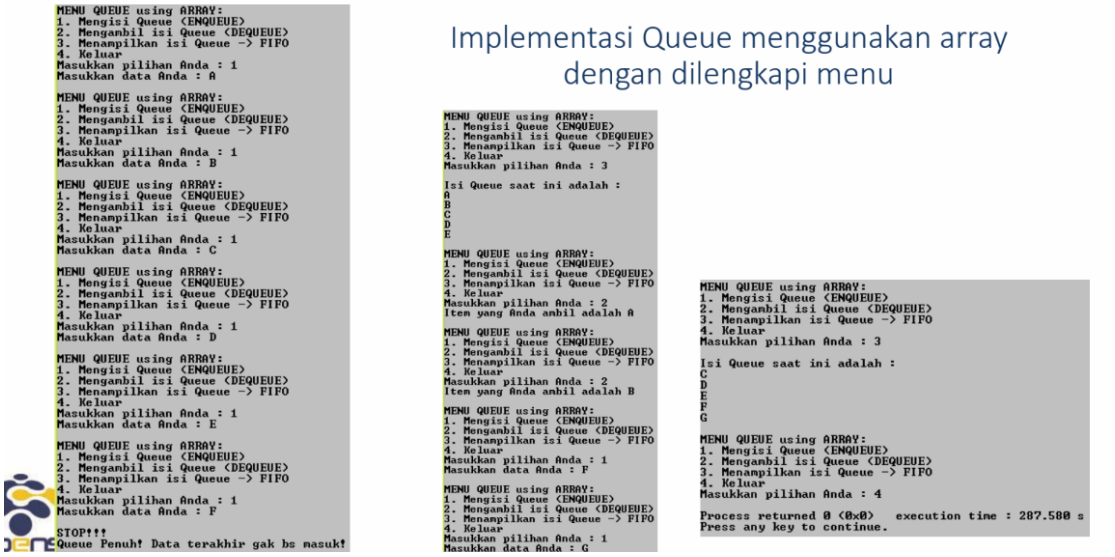
Mohammad Ihsan Septiano Firdaus (3125600041)

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

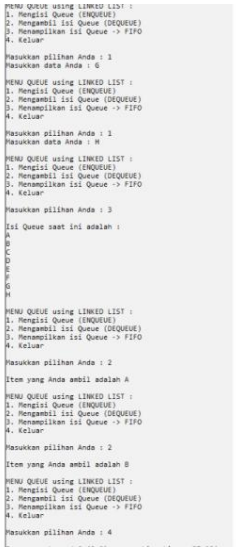
Program




```c
#include <stdio.h>
#define MAX 3
typedef char itemType;
typedef struct {
    itemType data[MAX];
    int f, r, count;
} queue;
//prottoype
void init(queue *);
int isFull(queue);
int isEmpty(queue);
void enqueue(queue *);
void dequeue(queue *);
void tampil(queue);

//main function
int main () {
    int ch; queue antrian;
    init(&antrian);
    do {
        printf("Menu QUEUE\n");
        printf("1. Mengisi Queue\n");
        printf("2. Mengambil Queue\n");
        printf("3. Menampilkan Queue\n");
        printf("4. Keluar\n");
        printf("Operasi : ");
        scanf("%d", &ch);
        switch(ch) {
            case 1:
                enqueue(&antrian); break;
            case 2:
```

```
                dequeue(&antrian); break;
            case 3:
                tampil(antrian); break;
            case 4:
                break;
            default:
                printf("Masukkan Operasi Yang Benar\n"); break;
        }} while (ch != 4); }

//inisialisasi
void init (queue *q) {
    q->f = 0;
    q->r = 0;
    q->count = 0; }

//enque
int isFull(queue q) {
    return(q.count == MAX); }

void enqueue(queue *q) {
    if(isFull(*q)) {
        printf("PENUH\n");
        return; }
    itemType x;
    printf("Masukkan Data = ");
    scanf(" %c", &x);
    q->data[(q->r)] = x;
    q->r = (q->r+1) % MAX;
    q->count ++; }

//deque
int isEmpty(queue q) {
    return(q.count == 0); }

void dequeue(queue *q) {
    if(isEmpty(*q)) {
        printf("KOSONG\n");
        return; }
    printf("Data yang diambil =  %c\n", q->data[q->f]);
    q->f = (q->f+1) % MAX;
    q->count--; }

void tampil(queue q) {
    for (int i = 0; i != q.count; i++) {
        int j = (q.f + i) %MAX ;
        printf("%c\n", q.data[j]); }}
```

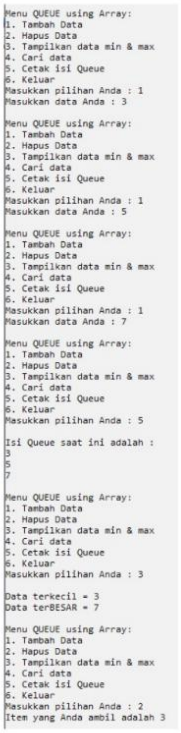
Implementasi Queue
dengan Single Linked List




```c
#include <stdio.h>
#include <stdlib.h>
typedef char itemType;
typedef struct node {
    struct node *next;
    itemType data;
} node;

node *p, *front, *rear, *del;

void alokasi();
void enqueue();
void dequeue();
void tampil();
void bebaskan(node **);

int main () {
    int ch;
    do {
        printf("Menu QUEUE (SLL)\n");
        printf("1. Mengisi Queue\n");
        printf("2. Mengambil Queue\n");
        printf("3. Menampilkan Queue\n");
        printf("4. Keluar\n");
        printf("Operasi : ");
        scanf("%d", &ch);
        switch(ch) {
            case 1:
                enqueue(); break;
            case 2:
                dequeue(); break;
            case 3:
```

```
                tampil(); break;
            case 4:
                break;
            default:
                printf("Masukkan Operasi Yang Benar\n"); break;
        }} while (ch != 4); }

//enqu
void alokasi() {
    p = (node *) malloc (sizeof(node));
    if(p) {
        p->next = NULL;
        printf("Data yang ingin dimasukkan = ");
        scanf(" %c", &p->data); }
    else printf("Gagal ALokasi"); }

void enqueue() {
    alokasi();
    if(front== NULL) front = p;
    else rear->next = p;
    rear = p; }

//dequ
void dequeue() {
    if(front == NULL ) {
        printf("DATA KOSONG");
        return; }
    printf("data yang diambil adalah %c\n", front->data);
    del = front;
    front = front->next;
    if(front == NULL) rear = NULL;
    bebaskan(&del); }

void bebaskan(node **a) {
    if (a == NULL) return;
    free(*a);
    *a = NULL; }

//menampilkan
void tampil() {
    node *tampil = front;
    if (tampil == NULL) {
        printf("DATA KOSONG\n");
        return; }
    printf("DATA = ");
    while (tampil != NULL) {
        printf("%c ", tampil->data);
        tampil= tampil->next;
```

```
    }
    printf("\n"); }
```



Menu Queue using Array

```c
#include <stdio.h>
#define MAX 3
typedef int itemType;
typedef struct {
    itemType data[MAX];
    int f, r, count;
} queue;

//prottoype
void init(queue *);
int isFull(queue);
int isEmpty(queue);
void enqueue(queue *);
void dequeue(queue *);
void tampil(queue);
void minmax(queue);
void search(queue);

//main function
int main () {
    int ch; queue antrian;
    init(&antrian);
    do {
        printf("Menu QUEUE\n");
        printf("1. Mengisi Queue\n");
        printf("2. Mengambil Queue\n");
        printf("3. Menampilkan nilai max & min\n");
```

```
        printf("4. Mencari Data\n");
        printf("5. Menampilkan Queue\n");
        printf("6. Keluar\n");
        printf("Operasi : ");
        scanf("%d", &ch);
        switch(ch) {
            case 1:
                enqueue(&antrian); break;
            case 2:
                dequeue(&antrian); break;
            case 3:
                minmax(antrian); break;
            case 4:
                search(antrian); break;
            case 5:
                tampil(antrian); break;
            case 6:
                break;
            default:
                printf("Masukkan Operasi Yang Benar\n"); break;
        }} while (ch != 6); }

//inisialisasi
void init (queue *q) {
    q->f = 0;
    q->r = 0;
    q->count = 0; }

//enque
int isFull(queue q) {
    return(q.count == MAX); }

void enqueue(queue *q) {
    if(isFull(*q)) {
        printf("PENUH\n");
        return; }
    itemType x;
    printf("Masukkan Data = ");
    scanf("%d", &x);
    q->data[(q->r)] = x;
    q->r = (q->r+1) % MAX;
    q->count ++; }

//deque
int isEmpty(queue q) {
    return(q.count == 0); }

void dequeue(queue *q) {
```

```
    if(isEmpty(*q)) {
        printf("KOSONG\n");
        return; }
    printf("Data yang diambil = %d\n", q->data[q->f]);
    q->f = (q->f+1) % MAX;
    q->count--; }

//tampil
void tampil(queue q) {
    for (int i = 0; i != q.count; i++) {
        int j = (q.f + i) %MAX ;
        printf("%d\n", q.data[j]); }}

//cari maxmin
void minmax(queue q) {
    if(q.count == 0) {printf("Data Masih Kosong\n"); return; }
    ;;;
    int max = q.data[q.f], min = q.data[q.f], queuemax = 1, queuemin =
1;
    for (int i = 1; i < q.count; i++) {
        if(max < q.data[(q.f+i)%MAX]) {
            max = q.data[(q.f+i)%MAX];
            queuemax = i+1; }
        if(min > q.data[(q.f+i)%MAX]) {
            min = q.data[(q.f+i)%MAX];
            queuemin = i+1; } }
     printf("Data max adalah %d dan berada di urutan %d\n", max,
queuemax);
     printf("Data min adalah %d dan berada di urutan %d\n", min,
queuemin); }

//mencari data
void search (queue q) {
    int found = 0, data, key, queue, multi;
    printf("Data yang dicari = ");
    scanf("%d", &key);
    for (int i = 0; i < q.count; i++) {
        if(key == q.data[(q.f+i)%MAX]) {
            queue = i+1; found = 1;
            printf("Data %d ditemukan dan berada di urutan %d\n", key,
queue);}}

    if(!found) printf("Data %d tidak ditemukan!\n", key);
    }
```


```
MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : A
Nilai prioritasnya : 3

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : B
Nilai prioritasnya : 2

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : C
Nilai prioritasnya : 2

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : D
Nilai prioritasnya : 1

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 3

Isi Queue saat ini adalah :
Data    Prioritas
D       1
B       2
C       2
A       3

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 2

Item yang Anda ambil adalah D
```

```
MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : E
Nilai prioritasnya : 2

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 1
Masukkan data Anda : F
Nilai prioritasnya : 1

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 3

Isi Queue saat ini adalah :
Data    Prioritas
F       1
B       2
C       2
E       2
A       3

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 2

Item yang Anda ambil adalah F

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 2

Item yang Anda ambil adalah B

MENU PRIORITY QUEUE using LINKED LIST :
1. Mengisi Queue
2. Mengambil isi Queue
3. Menampilkan isi Queue
4. Keluar

Masukkan pilihan Anda : 4

Process returned 0 (0x0)   execution time : 47.524 s
```

```c
#include <stdio.h>
#include <stdlib.h>
typedef char itemType;
typedef struct node {
    struct node *next;
    itemType data;
    int priority;
} node;

node *p, *front, *del;

void alokasi();
void enqueue();
void dequeue();
void tampil();
void bebaskan(node **);

int main () {
    int ch;
    do {
        printf("Menu QUEUE (SLL)\n");
        printf("1. Mengisi Queue\n");
        printf("2. Mengambil Queue\n");
        printf("3. Menampilkan Queue\n");
        printf("4. Keluar\n");
        printf("Operasi : ");
        scanf("%d", &ch);
```

```
        switch(ch) {
            case 1:
                enqueue(); break;
            case 2:
                dequeue(); break;
            case 3:
                tampil(); break;
            case 4:
                break;
            default:
                printf("Masukkan Operasi Yang Benar\n"); break;
        }} while (ch != 4); }

//enqu
void alokasi() {
    p = (node *) malloc (sizeof(node));
    if(p) {
        p->next = NULL;
        printf("Data yang ingin dimasukkan = ");
        scanf(" %c", &p->data);
        printf("Prioritas (lebih kecil lebih dahulu) = ");
        scanf("%d", &p->priority); }
    else printf("Gagal ALokasi"); }

void enqueue() {
    alokasi();
    node *search = front, *psearch;
    if(front== NULL) {front = p; return;}
    if(search->priority > p->priority) {
        p->next = front;
        front = p; return; }
    while (p->priority >= search->priority) {
        if (search->next == NULL) {
            search->next = p;
            return; }
        psearch = search;
        search = search->next;
    }
    p->next = search;
    psearch->next = p;
}



//dequ
void dequeue() {
    if(front == NULL ) {
        printf("DATA KOSONG");
```

```
        return; }
    printf("data yang diambil adalah %c\n", front->data);
    del = front;
    front = front->next;
    bebaskan(&del); }

void bebaskan(node **a) {
    if (a == NULL) return;
    free(*a);
    *a = NULL; }

//menampilkan
void tampil() {
    node *tampil = front;
    if (tampil == NULL) {
        printf("DATA KOSONG\n");
        return; }
    printf("DATA = ");
    while (tampil != NULL) {
        printf("%c ", tampil->data);
        tampil= tampil->next;
    }
    printf("\n"); }
```