LAPORAN
PRAKTIKUM INFORMASI DAN STRUKTUR DATA

SEMESTER GANJIL 2022/2023

MODUL 4

Queue

NAMA : ALEXIS DIVASONDA SIGAT NGAING

NPM : 210711407

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

TAHUN 2022

Membuat program pembelian bahan bakar kendaraan di SPBU dengan Queue (antrian). Pertama dibuat dua buah struct. Struct pertama untuk menyimpan data pembeli dengan nama struct `Bengsin` dan struct kedua untuk menyimpan Queue dengan nama `Queue`. Jumlah Queue maksimal 5 antrian untuk setiap jenis bahan bakar. Pertama dibuat prosedur untuk inisialisasi Queue sebagai berikut `(*Q).Head = (*Q).Tail = -1;`. Karena index array dimulai dari 0 maka posisi `Head` dan `Tail` diset menjadi `-1`. Di sini juga sekaligus dibuat inisialiasi data pembeli. Selanjutnya dibuat enam menu.

Menu pertama adalah menu masuk antrian. Menu ini akan dibagi lagi menjadi dua pilihan menu sesuai jenis bahan bakar, yaitu pertalite dan pertamax. Data yang diinputkan adalah ID, nama, dan jumlah uang. Untuk ID sudah akan tergenerate otomatis, dimulai dari ID 1001. Data kemudian akan di Enqueue atau dimasukkan ke dalam antrian dengan pemanggilan prosedur `Enqueue(&pertalite, makeBengsin(B.id, B.nama, B.jenis, B.uang));` dengan kode prosedur sebagai berikut.

```
void Enqueue(Queue *Q, Bengsin B) {
    if(isEmpty(*Q)) {
        (*Q).Head = (*Q).Tail = 0;
    } else {
        if((*Q).Tail == max-1) {
            (*Q).Tail = 0;
        } else {
            (*Q).Tail++;
        }
    }
    (*Q).B[(*Q).Tail] = B;
    printf("\n\t\tBerhasil memasukkan data...");
}
```

Antrian akan dicek terlebih dahulu dengan fungsi isEmpty sebagai berikut `return (Q.Head == -1 && Q.Tail == -1);`. Fungsi akan mengembalikan kondisi antrian yang kosong. Jika sudah ada antrian, maka fungsi tidak bekerja atau akan dilewati. Jika kondisi terpenuhi, artinya Queue atau antrian berada dalam kondisi kosong. Dalam kondisi kosong, `Head` dan `Tail` diset di index 0 dan menjadi data pertama. Dalam kondisi antrian sudah ada atau kondisi normal, posisi `Tail`

akan digeser ke kanan dengan cara menambah index posisinya (`(*Q).Tail++;`). Terakhir, dalam kondisi Tail sudah berada di index akhir (belum penuh), index Tail akan diset di posisi awal (kode: `(*Q).Tail = 0;`). Antrian dibatasi sebanyak lima untuk setiap jenis bahan bakar. Antrian akan dicek menggunakan fungsi `isFull(Queue Q)`. Fungsi ini memiliki dua kondisi. Pertama kondisi `Tail` lebih besar dari `Head`, dengan kode `(Q.Head < Q.Tail && Q.Tail - Q.Head == max-1)` dan kondisi Head lebih besar dari Tail, dengan kode `(Q.Tail < Q.Head && Q.Tail + 1 == Q.Head).`

Menu kedua adalah menu pembayaran. Di sini akan diberikan dua pilihan kembali, pertama Pertalite dan kedua Pertamax. Pembayaran dilakukan sesuai dengan urutan antrian data, menggunakan prosedur `Dequeue`. Terdapat tiga kondisi, pertama kondisi 1 elemen. Sebelumnya Queue akan dicek terlebih dahulu menggunakan fungsi `isOneElmt(Queue Q)` dengan nilai balikan `return (Q.Head == Q.Tail && Q.Head != -1);`. Fungsi ini akan mengecek apakah hanya terdapat satu elemen dalam antrian atau tidak. Yang dimaksud kondisi satu elemen adalah saat index `Head` dan `Tail` sama dan Queue tidak kosong. Jika kondisi ini terpenuhi, maka akan dipanggil prosedur `createEmpty(Queue *Q)` untuk menginisialisasi semua antrian dan data.

Menu ketiga berfungsi untuk menampilkan antrian `pertalite` dan `pertamax`. Tampilan menggunakan prosedur `tampil` atau traversal. Dalam kondisi normal, traversal akan dilakukan dari index `Head` sampai sama dengan index `Tail`. Berikut kode prosedurnya.

```
if(Q.Head <= Q.Tail) {
    for(i=Q.Head; i<=Q.Tail; i++) {
        printf("%d(%s) | ", Q.B[i].id, Q.B[i].nama);
    }
```

Dalam kondisi sirkular atau Ketika `Head` lebih besar dari `Tail`, maka traversal akan dilakukan sebanyak dua kali. Pertama dari `Head` hingga batas akhir array, dan kedua dari `index 0` hingga posisi sama dengan `Tail`. Berikut kodenya.

```
} else {
    for(i=Q.Head; i<=max-1; i++) { //traversal pertama
        printf("%d(%s) | ", Q.B[i].id, Q.B[i].nama);
        temp++;
    }
    for(i=0; i<Q.Tail; i++) { //traversal kedua
        printf("%d(%s) | ", Q.B[i].id, Q.B[i].nama);
        temp++;
    }
}
```

Menu keempat adalah menu cari data. Pertama akan diinputkan ID pembeli yang ingin dicari kemudian ID akan dicari dengan dua fungsi yaitu `isFound(pertalite, B.id);` dan `isFound(pertalite, B.id);`. Fungsi masing-masinag akan mengecek antrian dalam dua jenis bahan bakar yang berbeda. Fungsi ini juga merupakan traversal. Berikut kodenya.

```
int isFound(Queue Q, int cari) {
    int i;
    if(Q.Head <= Q.Tail) {
        for(i=Q.Head; i<=Q.Tail; i++) {
            if(Q.B[i].id == cari) {
                return i;
            }
        }
    } else {
        for(i=Q.Head; i<max; i++) {
            if(Q.B[i].id == cari) {
                return i;
            }
        }
        for(i=0; i<=Q.Tail; i++) {
            if(Q.B[i].id == cari) {
                return i;
            }
        }
    }
    return -1;
}
```

Cara kerjanya sama seperti prosedur `tampil`, bedanya fungsi ini akan mengembalikan index dari data yang ditemukan.

Menu kelima merupakan menu edit data. Menu ini tidak dapat diakses apabila kedua antrian masih kosong. Pengguna akan menginputkan ID pembeli yang kemudian akan dicari dengan dua fungsi yaitu `isFound(pertalite, B.id);`

dan `isFound(pertalite, B.id);`. Jika data ditemukan maka pengguna akan menginputkan data pembeli yang baru. Data akan langsung terganti saat pengguna menginputkannya (salah satu kode: `fflush(stdin); gets(pertalite.B[index].nama);`). Index di sini merupakan fungsi `isFound` yang sudah diubah.

Menu keenam adalah menu Riwayat. Di sini akan ditampilkan jumlah pembeli dan total uang yang didapatkan.