“Daftar kontak di handphone“

Disusun oleh :

Kelompok 7

Nama kelompok

LINDRA OKTAVIA SIANTURI (23081010027)

DINA CAHYANTI (23081010084)

Daftar isi

01. STUDI KASUS

02. FITUR PROGRAM

03. MATERI IMPLEMENTASI

04. ADT

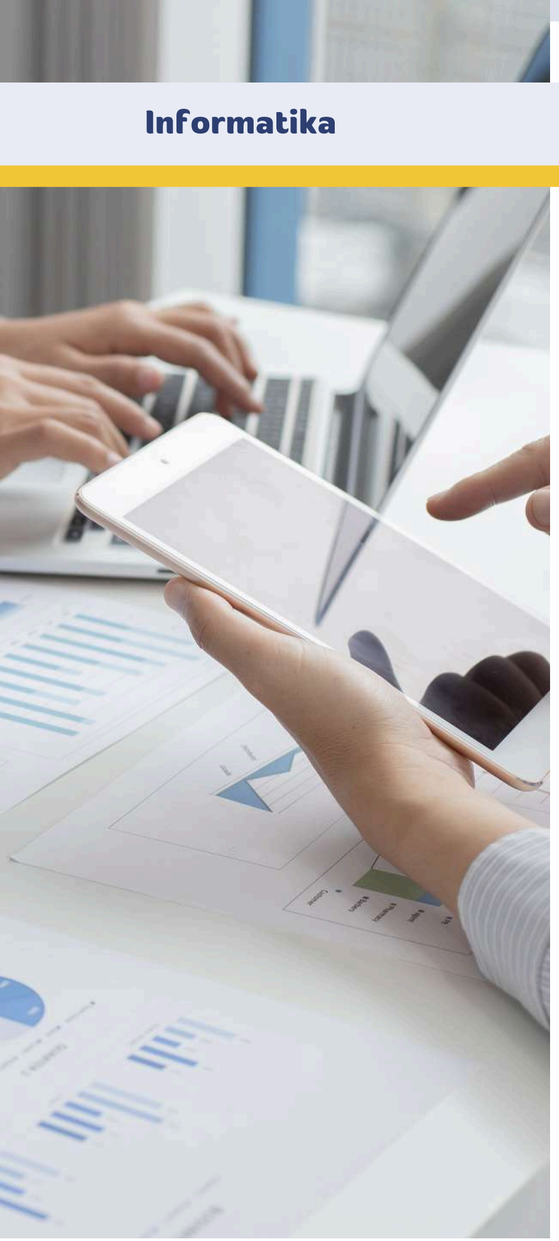
05. REKURSIF

06.QUEUE

07. INPUT PROGRAM

08. OUTPUT PROGRAM

09 KESIMPULAN



Studi Kasus

- Kami mengambil studi kasus daftar kontak di handphone karena Kontak adalah salah satu fitur paling dasar dan penting di ponsel. Pengguna perlu menyimpan informasi kontak untuk dapat berkomunikasi dengan keluarga, teman, rekan kerja, dan lainnya.

- Program ini mereplikasi fungsionalitas yang sama, memungkinkan pengguna untuk menambah, mencari, dan mengelola informasi kontak mereka.

- Program yang kami gunakan adalah bahasa C

Fitur Program

1. Menambahkan kontak
2. Menampilkan semua kontak
3. Mencari kontak
4. Hapus Kontak
5. Keluar

Materi yang di implementasikan

- **ADT (ABSTRACT DATA TYPE) - Struct**

Abstract Data Type adalah sebuah konsep dalam ilmu komputer yang mendefinisikan tipe data baru bersama dengan operasi-operasi yang dapat dilakukan pada tipe data tersebut tanpa harus memperhatikan implementasi internalnya. Dengan kata lain, ADT memisahkan antara cara data disimpan dan bagaimana data tersebut dapat diakses atau dimanipulasi.

- **LINKED LIST - Array**

Kami menggunakan Array karena kami ingin membatasi input

- **REKURSIF**

Rekursif adalah sebuah konsep dalam ilmu komputer di mana suatu fungsi atau prosedur memanggil dirinya sendiri selama pemrosesan. Dalam konteks pemrograman, rekursi memungkinkan penyelesaian masalah kompleks dengan memecahnya menjadi submasalah yang lebih kecil dan lebih mudah diatasi.

- **Queue**

Queue adalah struktur data yang memungkinkan penambahan elemen hanya di satu ujung (rear) dan penghapusan elemen hanya di ujung yang lain (front). Prinsip ini dikenal dengan konsep FIFO (First-In-First-Out), artinya elemen pertama yang masuk adalah elemen pertama yang keluar.

ADT

```c
#define MAX_KONTAK 10 // Jumlah maksimum kontak yang dapat disimpan

// Struktur untuk menyimpan kontak
typedef struct Kontak {
    char nama[10]; // Nama kontak
    char nomor[20]; // Nomor telepon kontak
} Kontak;
```

Kami membatasi jumlah maxsimum kontak yang disimpan menggunakan array

ADT

```c
// ADT untuk daftar kontak
typedef struct {
    Kontak kontak[MAX_KONTAK]; // Array untuk menyimpan kontak
    int jumlah; // Jumlah kontak yang tersimpan
} ListKontak;
```

struct ListKontak: pengguna hanya perlu memikirkan operasi yang bisa dilakukan pada daftar kontak ini, seperti menambah, menghapus, mencari, dan menampilkan kontak. Mereka tidak perlu memikirkan bagaimana kontak disimpan di dalam array.

ADT

```
// Prototipe fungsi
void initListKontak(ListKontak *list);
void tambahKontak(ListKontak *list, const char *nama, const char *nomor);
void menampilkanKontak(const ListKontak *list);
void cariKontak(const ListKontak *list, const char *nama);
int cariKontakRekursif(const ListKontak *list, const char *nama, int index);
void dequeue(ListKontak *list);
void hapusKontak(ListKontak *list);
```

mereka menyediakan interface yang jelas dan menyembunyikan detail implementasi struktur data dari pengguna.

Rekursif

```c
// Fungsi rekursif untuk mencari kontak
int cariKontakRekursif(const ListKontak *list, const char *nama, int index) {
    if (index >= list->jumlah) {
        return -1; // Kontak tidak ditemukan
    }
    if (strcmp(list->kontak[index].nama, nama) == 0) {
        printf("Kontak Ditemukan: Nama: %s, Nomor Telepon: %s\n", list->kontak[index].nama, list->kontak[index].nomor
        return index; // Kontak ditemukan
    }
    return cariKontakRekursif(list, nama, index + 1); // Pemanggilan diri untuk kontak berikutnya
}
```

Fungsi ini memanggil dirinya sendiri dengan parameter index + 1, sehingga secara rekursif melanjutkan pencarian kontak berikutnya hingga akhir daftar.

Enqueue

```c
// Menambahkan kontak baru
void tambahKontak(ListKontak *list, const char *nama, const char *nomor) {
    if (list->jumlah >= MAX_KONTAK) {
        printf("Daftar kontak sudah penuh.\n");
        return;
    }

    // Enqueue: Menambahkan kontak baru ke belakang queue
    strcpy(list->kontak[list->jumlah].nama, nama);
    strcpy(list->kontak[list->jumlah].nomor, nomor);
    list->jumlah++; // Increment jumlah kontak

    printf("Kontak berhasil ditambahkan.\n");
```

Fungsi tambahKontak() bisa dianggap sebagai fungsi enqueue dalam konteks queue karena cara kerjanya sejalan dengan prinsip dasar antrian, di mana elemen baru ditambahkan di belakang elemen yang sudah ada.

ketika daftar kontak sudah mencapai batas maxsimum array maka akan muncul pesan “Daftar kontak sudah penuh”

Contoh Penggunaan

Misalnya, jika kamu menambahkan kontak berikut secara berurutan:

- Kontak A
- Kontak B
- Kontak C

Maka dalam daftar kontak, urutannya adalah:

- Indeks 0: Kontak A (masuk pertama)
- Indeks 1: Kontak B (masuk kedua)
- Indeks 2: Kontak C (masuk ketiga)

Saat kamu memanggil dequeue/hapus kontak, kontak A akan dihapus pertama kali, diikuti kontak B, dan akhirnya kontak C. Ini menggambarkan karakteristik dari queue.

Dequeue

```c
// Menghapus kontak dari queue
void dequeue(ListKontak *list) {
    if (list->jumlah == 0) {
        printf("Daftar kontak kosong. Tidak ada yang bisa dihapus.\n");
        return;
    }

    // Menghapus kontak pertama
    printf("Kontak dengan nama '%s' berhasil dihapus.\n", list->kontak[0].nama);

    // Geser kontak ke kiri untuk menghapus yang pertama
    for (int i = 1; i < list->jumlah; i++) {
        list->kontak[i - 1] = list->kontak[i];
    }

    list->jumlah--; // Decrement jumlah kontak
}
```

ia menghapus elemen pertama dari antrian dan menggeser elemen-elemen yang tersisa untuk menjaga urutan.

Jika di list kontak tidak ada kontak yang akan di hapus maka akan muncul pesan “Daftar kontak kosong, Tidak ada yang bisa dihapus”

Contoh Penggunaan

Misalnya, kamu memiliki daftar kontak sebagai berikut:

- Kontak A
- Kontak B
- Kontak C

Ketika kamu memanggil dequeue():

Sebelum:

- Indeks 0: Kontak A
- Indeks 1: Kontak B
- Indeks 2: Kontak C

Sesudah:

- Kontak A dihapus, jadi:
- Indeks 0: Kontak B
- Indeks 1: Kontak C
- Jumlah kontak menjadi 2.

Input Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_KONTAK 10 // Jumlah maksimum kontak yang dapat disimpan

// Struktur untuk menyimpan kontak
typedef struct Kontak {
    char nama[10]; // Nama kontak
    char nomor[20]; // Nomor telepon kontak
} Kontak;

// ADT untuk daftar kontak
typedef struct {
    Kontak kontak[MAX_KONTAK]; // Array untuk menyimpan kontak
    int jumlah; // Jumlah kontak yang tersimpan
} ListKontak;

// Prototipe fungsi
void initListKontak(ListKontak *list);
void tambahKontak(ListKontak *list, const char *nama, const char *nomor);
void menampilkanKontak(const ListKontak *list);
void cariKontak(const ListKontak *list, const char *nama);
int cariKontakRekursif(const ListKontak *list, const char *nama, int index);
void dequeue(ListKontak *list);
void hapusKontak(ListKontak *list);
```

Input Program

```c
// Inisialisasi daftar kontak
void initListKontak(ListKontak *list) {
    list->jumlah = 0; // Inisialisasi jumlah kontak
}

// Menambahkan kontak baru
void tambahKontak(ListKontak *list, const char *nama, const char *nomor) {
    if (list->jumlah >= MAX_KONTAK) {
        printf("Daftar kontak sudah penuh.\n");
        return;
    }

    // Enqueue: Menambahkan kontak baru ke belakang queue
    strcpy(list->kontak[list->jumlah].nama, nama);
    strcpy(list->kontak[list->jumlah].nomor, nomor);
    list->jumlah++; // Increment jumlah kontak

    printf("Kontak berhasil ditambahkan.\n");
}

// Menampilkan semua kontak
void menampilkanKontak(const ListKontak *list) {
    if (list->jumlah == 0) {
        printf("Daftar kontak kosong.\n");
        return;
    }
```

```c
// Menampilkan semua kontak
void menampilkanKontak(const ListKontak *list) {
    if (list->jumlah == 0) {
        printf("Daftar kontak kosong.\n");
        return;
    }

    printf("Daftar Kontak:\n");
    for (int i = 0; i < list->jumlah; i++) {
        printf("Nama: %s, Nomor Telepon: %s\n", list->kontak[i].nama, list->kontak[i].nomor);
    }
}

// Mencari kontak berdasarkan nama
void cariKontak(const ListKontak *list, const char *nama) {
    if (cariKontakRekursif(list, nama, 0) == -1) {
        printf("Kontak dengan nama '%s' tidak ditemukan.\n", nama);
    }
}
```

Input Program

```c
// Fungsi rekursif untuk mencari kontak
int cariKontakRekursif(const ListKontak *list, const char *nama, int index) {
    if (index >= list->jumlah) {
        return -1; // Kontak tidak ditemukan
    }
    if (strcmp(list->kontak[index].nama, nama) == 0) {
        printf("Kontak Ditemukan: Nama: %s, Nomor Telepon: %s\n", list->kontak[index].nama, list->kontak[index].nomor);
        return index; // Kontak ditemukan
    }
    return cariKontakRekursif(list, nama, index + 1); // Pemanggilan diri untuk kontak berikutnya
}
```

Input Program

```c
// Menghapus kontak dari queue
void dequeue(ListKontak *list) {
    if (list->jumlah == 0) {
        printf("Daftar kontak kosong. Tidak ada yang bisa dihapus.\n");
        return;
    }

    // Menghapus kontak pertama
    printf("Kontak dengan nama '%s' berhasil dihapus.\n", list->kontak[0].nama);

    // Geser kontak ke kiri untuk menghapus yang pertama
    for (int i = 1; i < list->jumlah; i++) {
        list->kontak[i - 1] = list->kontak[i];
    }

    list->jumlah--; // Decrement jumlah kontak
}

// Membersihkan seluruh daftar kontak
void hapusKontak(ListKontak *list) {
    list->jumlah = 0; // Reset jumlah kontak
}
```

Input Program

```c
int main() {
    ListKontak listkontak;
    initListKontak(&listkontak);

    int choice;
    char nama[10];
    char nomor[20];

    while (1) {
        printf("\nKelompok 7\n");
        printf("Nama : Lindra Oktavia Sianturi (23081010027)\n");
        printf("Nama : Dina Cahyanti (23081010084)\n");
        printf("\nMenu:\n");
        printf("1. Tambah Kontak\n");
        printf("2. Tampilkan Semua Kontak\n");
        printf("3. Cari Kontak\n");
        printf("4. Hapus Kontak\n");
        printf("5. Keluar\n");
        printf("Pilihan Anda: ");
        scanf("%d", &choice);
        getchar();  // Menghapus karakter newline dari buffer
```

Input Program

```c
switch (choice) {
    case 1:
        printf("Masukkan Nama: ");
        fgets(nama, sizeof(nama), stdin);
        nama[strcspn(nama, "\n")] = '\0'; // Menghapus newline dari input

        printf("Masukkan Nomor Telepon: ");
        fgets(nomor, sizeof(nomor), stdin);
        nomor[strcspn(nomor, "\n")] = '\0'; // Menghapus newline dari input

        tambahKontak(&listkontak, nama, nomor);
        break;
    case 2:
        menampilkanKontak(&listkontak);
        break;
    case 3:
        printf("Masukkan Nama untuk Dicari: ");
        fgets(nama, sizeof(nama), stdin);
        nama[strcspn(nama, "\n")] = '\0'; // Menghapus newline dari input

        cariKontak(&listkontak, nama);
        break;
```

```c
            case 4:
                dequeue(&listkontak);
                break;
            case 5:
                hapusKontak(&listkontak);  // Membersihkan data
                exit(0);
            default:
                printf("Pilihan tidak valid. Silakan coba lagi.\n");
        }
    }

    return 0;
}
```

Output Program

1. Tambah kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 1
Masukkan Nama: dina
Masukkan Nomor Telepon: 081283014
Kontak berhasil ditambahkan.
```

Output Program

1. Tambah kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 1
Masukkan Nama: nina
Masukkan Nomor Telepon: 0174862183
Daftar kontak sudah penuh.

Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)
```

Karena kami menggunakan array untuk membatasi input menambahkan 10 kontak, maka jika menambahkan kontak lebih dari 10 akan muncul pesan “Daftar kontak sudah penuh.”

Output Program

2. Tampilkan semua kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 2
Daftar Kontak:
Nama: dina, Nomor Telepon: 081389471
Nama: lindra, Nomor Telepon: 0813971032
Nama: sophia, Nomor Telepon: 03893921
Nama: amel, Nomor Telepon: 0812839714
```

Jika memilih menu 2 yaitu tampilkan semua kontak, maka akan muncul output menampilkan semua kontak yang telah ditambahkan sebelumnya

Output Program

3. Cari Kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 3
Masukkan Nama untuk Dicari: lindra
Kontak Ditemukan: Nama: lindra, Nomor Telepon: 0813971032
```

Jika memilih menu 2 yaitu tampilkan semua kontak, maka akan muncul output menampilkan semua kontak yang telah ditambahkan sebelumnya

Output Program

3. Cari Kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 3
Masukkan Nama untuk Dicari: rere
Kontak dengan nama 'rere' tidak ditemukan.
```

Jika memasukan nama untuk dicari dan nama tersebut tidak pernah ditambahkan sebelumnya, maka akan muncul pesan “Kontak dengan nama ‘rere’ tidak ditemukan”

Output Program

4. Hapus Kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 4
Kontak dengan nama 'dina' berhasil dihapus.
```

Karena kami menggunakan queue jadi saat memilih menu 4 yaitu hapus kontak, maka secara otomatis nama yang pertama kali ditambahkan yang akan terhapus

Output Program

4. Hapus Kontak

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 4
Daftar kontak kosong. Tidak ada yang bisa dihapus.
```

Jika sebelumnya sudah menghapus semua kontak atau tidak ada kontak yang ditambahkan maka akan muncul pesan “Daftar kontak kosong. Tidak ada yang bisa dihapus.” saat memilih menus hapus kontak

Output Program

5. Keluar

```
Kelompok 7
Nama : Lindra Oktavia Sianturi (23081010027)
Nama : Dina Cahyanti (23081010084)

Menu:
1. Tambah Kontak
2. Tampilkan Semua Kontak
3. Cari Kontak
4. Hapus Kontak
5. Keluar
Pilihan Anda: 5
PS C:\Users\USER\Documents\Struktur Data\output>
```

Jika memilih menu keluar, maka secara otomatis akan keluar dari terminal output.

Kesimpulan

1. **Struktur Data** : Program menggunakan struktur data array untuk menyimpan maksimal 10 kontak. Setiap kontak terdiri dari nama dan nomor telepon, disimpan dalam struktur `Kontak`, dan kumpulan kontak tersebut dikelola dalam struktur `ListKontak`.

2. **Fungsi-fungsi Utama**:

- I**nisialisasi** (`initListKontak`) untuk memulai daftar kontak kosong.
- T**ambah Kontak** (`tambahKontak`): Menambahkan kontak baru ke dalam daftar selama belum mencapai batas maksimum.
- T**ampilkan Kontak** (`menampilkanKontak`): Menampilkan semua kontak yang telah disimpan.
- C**ari Kontak** (`cariKontak`): Mencari kontak berdasarkan nama, menggunakan fungsi rekursif untuk pencarian.
- **Hapus Kontak** (`dequeue`): Menghapus kontak pertama dari daftar (sesuai dengan prinsip queue di mana elemen pertama keluar pertama).
- **Hapus Semua Kontak** (`hapusKontak`): Menghapus semua kontak sekaligus.

Kesimpulan

3. **Menu Interaktif**: Program menyediakan antarmuka berbasis teks untuk pengguna dengan pilihan menu seperti tambah kontak, tampilkan kontak, cari kontak, hapus kontak, dan keluar dari program.

4. **Antrean (Queue)** : Program ini menggunakan konsep queue dalam penanganan kontak, di mana kontak pertama yang ditambahkan akan menjadi yang pertama dihapus.

5. **Fungsi Rekursif** : Pencarian kontak menggunakan fungsi rekursif, yang memeriksa setiap kontak satu per satu berdasarkan nama yang dimasukkan.

Program ini cocok untuk simulasi sederhana dari pengelolaan daftar kontak dengan fitur dasar seperti menambah, menampilkan, mencari, dan menghapus kontak.

Link Canva

Terima Kasih

Kelompok 7

“Daftar kontak di handphone“