Praktikum

Struktur Data

Projek Akhir

Dosen Pembimbing:

Randi Proska Sandra, M.Sc

Disusun oleh :

Nama : Rafli Arianto

NIM : 23343051

UNIVERSITAS NEGERI PADANG

FAKULTAS TEKNIK

INFORMATIKA

2024

Nama Program

Program untuk Berlangganan Gym

Latar Belakang Pembuatan Program

Saat ini, gaya hidup sehat dan kebugaran fisik menjadi semakin penting bagi banyak orang. Gym atau pusat kebugaran telah menjadi pilihan populer untuk membantu setiap orang yang ingin mencapai tujuan kesehatan dan kebugaran. Seiring dengan meningkatnya minat terhadap kebugaran, manajemen keanggotaan gym menjadi lebih kompleks. Karenanya, sebuah sistem manajemen yang efisien dan terorganisir sangat diperlukan untuk mengelola data keanggotaan, memantau durasi berlangganan, serta memfasilitasi administrasi yang efektif.

Dengan adanya program manajemen langganan gym, informasi penting mengenai anggota dapat disimpan dan diakses dengan mudah, mengurangi risiko kesalahan dan meningkatkan efisiensi operasional. Program ini memungkinkan staf gym untuk dengan mudah menambahkan, mengedit, mencari, dan menghapus data keanggotaan. Fitur seperti pencarian berdasarkan ID atau nama, serta pengurutan data, mempermudah pengelolaan informasi anggota. Dengan akses cepat dan mudah ke data anggota, staf gym dapat memberikan layanan yang lebih baik dan responsif.

Program ini membantu memastikan bahwa data keanggotaan disimpan dengan aman dan terstruktur, mengurangi risiko kehilangan atau kerusakan data yang sering terjadi pada pencatatan manual. Data yang tersimpan secara digital memungkinkan gym untuk melakukan analisis lebih lanjut mengenai keanggotaan, seperti tren berlangganan, tingkat retensi anggota, dan efektivitas promosi. Ini membantu dalam pengambilan keputusan yang lebih baik dan strategi bisnis yang lebih terfokus. Dengan mengotomatiskan tugas-tugas administrasi, program ini membantu menghemat waktu staf, selain itu, penghematan waktu ini berguna secara tidak langsung pada penghematan biaya operasional.

Penjelasan Aplikasi

Menu:

1. Menu pertama berfungsi untuk menginputkan data konsumen baru berupa nama, nomor telepon dan durasi berlangganan yang diinginkan oleh konsumen serta menampilkan id khusus untuk konsumen yang dihasilkan berupa 3 huruf kapital secara acak.
2. Menu kedua berguna untuk menampilkan data semua konsumen yang sedang berlangganan.
3. Menu ketiga berguna untuk menampilkan urutan konsumen berdasarkan nama dari A ke Z
4. Menu keempat berguna untuk menampilkan urutan konsumen berdasarkan nama dari Z ke A
5. Menu kelima memiliki fungsi untuk mencari konsumen berdasarkan ID

6. Menu kelima memiliki fungsi untuk mencari konsumen berdasarkan nama
7. Menu ketujuh berfungsi untuk mengedit data konsumen berdasarkan ID, seperti mengubah nama, nomor telepon dan durasi berlangganan
8. Menu kedelapan berfungsi untuk membatalkan berlangganan konsumen dan menghapusnya dari data yang tersimpan berdasarkan ID
9. Menu terkakhir berfungsi untuk keluar dari program

Fitur:

1. Menghasilkan id untuk konsumen berupa 3 huruf kapital secara acak
2. Pembatalan berlangganan konsumen dan penghapusan data konsumen dari data tersimpan
3. Mengedit data konsumen
4. Mengurutkan data konsumen berdasarkan nama dari A ke Z maupun sebaliknya
5. Mencari data konsumen berdasarkan nama dan juga ID
6. Menampilkan data konsumen dengan urutan data terlama ada di paling atas

Penjelasan Baris Kode Program:

Mendefinisikan node untuk double link list dari data konsumen yang isinya id, nama, telepon, dan durasi serta pengubung antara data sebelum dan sesudahnya dengan nama *sebelum dan *sesudah.

```
typedef struct Konsumen {
    char id[PANJANG_ID];
    char nama[PANJANG_NAMA];
    char telepon[NO_TELEPON];
    int durasi;
    struct Konsumen *sebelum;
    struct Konsumen *sesudah;
} Konsumen;
```

Bagian depan dan belakang dari list dengan nama *depan dan *belakang dengan nilai NULL.

```
Konsumen *depan = NULL, *belakang = NULL;
```

Bagian untuk membuat id otomatis dengan rand yang isinya 3 huruf acak secara kapital dari seluruh alfabet. Setiap konsumen mendapat id yang berbeda.

```
void buatID(char *id) {
    static const char alfabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for (int i = 0; i < PANJANG_ID - 1; i++) {
        id[i] = alfabet[rand() % 26];
    }
    id[PANJANG_ID - 1] = '\0';
}
```

Menambah data konsumen yang disimpan pada pointer *konsumenBaru yang ukuran datanya sama dengan Konsumen. Isinya ada id, nama, telepon, dan durasi. Data konsumen dimasukan ke dalam list yang sebelumnya ada bagian belakang dan sesudahnya ada NULL. Jika di bagian belakang maka akan ke sesudahnya, selain itu jika bagian depan maka langsung terisi konsumenBaru.

```
void tambahKonsumen() {
```

```
    Konsumen *konsumenBaru = (Konsumen *)malloc(sizeof(Konsumen));
    buatID(konsumenBaru->id);
    printf("Ketikan Nama: ");
    scanf("%s", konsumenBaru->nama);
    printf("Ketikan Nomor Telepon: ");
    scanf("%s", konsumenBaru->telepon);
    printf("Ketikan durasi berlanganan (bulan): ");
    scanf("%d", &konsumenBaru->durasi);

    konsumenBaru->sebelum = belakang;
    konsumenBaru->sesudah = NULL;
    if (belakang) {
        belakang->sesudah = konsumenBaru;
    } else {
        depan = konsumenBaru;
    }
    belakang = konsumenBaru;

    printf("ID Konsumen: %s\n", konsumenBaru->id);
}
```

Bagian untuk menampilkan data konsumen yang tersimpan, diurutkan sesuai dari data yang terlama ada di bagian yang paling atas. Pada pointer *sesuai yang pertama adalah bagian depan dilanjutkan ke sesudahnya.

```
void tampilkanKonsumen() {
    Konsumen *sesuai = depan;
    while (sesuai) {
        printf("ID: %s\n", sesuai->id);
        printf("Nama: %s\n", sesuai->nama);
        printf("Telepon: %s\n",  sesuai->telepon);
        printf("Durasi: %d bulan\n", sesuai->durasi);
        printf("--------------------\n");
        sesuai = sesuai->sesudah;
    }
}
```

Membatalkan berlangganan konsumen berdasarkan id konsumen, pointer *sesuai ada di bagian depan yang artinya data tersebut ditampilkan, saat id yang diketikkan sama dengan yang terdata menggunakan strcasecmp, jika sesuai ada di sebelum maka ada dijadikan sesudah, selain itu maka data berada di depan. Jika sesuai ada di sesudah maka akan dijadikan sebelum, selain itu maka data berada di belakang. Jika data sudah ketemu maka akan langsung dihapus menggunakan free. Jika id yang diinput tidak ada maka tidak melanjutkan proses dan aka nada output yang menyatakan bahwa id tidak ada.

```
void batalkanBerlangganan(char *id) {
    Konsumen *sesuai = depan;
    while (sesuai) {
        if (strcasecmp(sesuai->id, id) == 0) {
            if (sesuai->sebelum) {
                sesuai->sebelum->sesudah = sesuai->sesudah;
            } else {
                depan = sesuai->sesudah;
            }
            if (sesuai->sesudah) {
                sesuai->sesudah->sebelum = sesuai->sebelum;
            } else {
```

```
                belakang = sesuai->sebelum;
            }
            free(sesuai);
            printf("Berlangganan dengan ID %s batal.\n", id);
            return;
        }
        sesuai = sesuai->sesudah;
    }
    printf("Konsumen dengan ID %s tidak ada.\n", id);
}
```

Membuat fungsi untuk sort data dengan Merge, void pertama isinya ada Konsumen yang menggunakan pointer ke **depanDepan dan meningkat untuk mengurutkan data ke atas. Void kedua isinya ada Konsumen pointer ke *asal, Konsumen pointer ke **bagianDepan dan Konsumen pointer ke **bagianBelakang. Dan ada return ke Konsumen dari urutandenganMerge yang isinya konsumen pointer ke *a, konsumen pointer ke *b dan meningkat.

```
void urutanKonsumendenganMerge(Konsumen **depanDepan, int meningkat);
void bagiDaftar(Konsumen *asal, Konsumen **bagianDepan, Konsumen
**bagianBelakang);
Konsumen* urutandenganMerge(Konsumen *a, Konsumen *b, int meningkat);
```

Mengurutkan menggunakan nama dengan memanggil fungsi mengurutkan dengan merge dari bagian depan dan meningkat.

```
void urutanKonsumendengannama(int meningkat) {
    urutanKonsumendenganMerge(&depan, meningkat);
}
```

Fungsi urutan dengan merge yang isinya ada konsumen *depan sama dengan *depanDepan, konsumen *a dan konsumen *b. jika depan adalah NULL atau depan sesudah adalah NULL maka akan mendapatkan hasil. Dilanjutkan dengan membagi daftar menjadi dua a dan b. a dibuat meningkat dan b dibuat meningkat juga yang nantinya akan mengahilkan urutan yang sesuai pada *depanDepan setelah selesai diurutkan. Dilanjutkan dengan return konsumen urutandenganMerge yang isinya konsumen *a, konsumen *b dan meningkat. Konsumen *hasil adalah NULL. Jika a adalah NULL maka return b selain itu jika b adalah NULL maka return a. Jika kondisi benar maka ekspresi pertama yang akan dieksekusi yang hasilnya adalah untuk mengurutkan nama dari bawah ke atas. Jika kondisi salah maka yang akan dieksekusi adalah ekspresi kedua untuk mengurutkan data dari atas ke bawah. Jadi dari fungsi tersebut dapat digunakan untuk dua jenis pengurutan yaitu dari A ke Z dan juga Z ke A.

```
void urutanKonsumendenganMerge(Konsumen **depanDepan, int meningkat) {
    Konsumen *depan = *depanDepan;
    Konsumen *a;
    Konsumen *b;

    if ((depan == NULL) || (depan->sesudah == NULL)) {
        return;
    }

    bagiDaftar(depan, &a, &b);

    urutanKonsumendenganMerge(&a, meningkat);
    urutanKonsumendenganMerge(&b, meningkat);

    *depanDepan = urutandenganMerge(a, b, meningkat);
```

```
}

Konsumen* urutandenganMerge(Konsumen *a, Konsumen *b, int meningkat) {
    Konsumen *hasil = NULL;

    if (a == NULL)
        return b;
    else if (b == NULL)
        return a;

    if (meningkat ? (strcmp(a->nama, b->nama) <= 0) : (strcmp(a->nama, b-
>nama) >= 0)) {
        hasil = a;
        hasil->sesudah = urutandenganMerge(a->sesudah, b, meningkat);
        hasil->sesudah->sebelum = hasil;
        hasil->sebelum = NULL;
    } else {
        hasil = b;
        hasil->sesudah = urutandenganMerge(a, b->sesudah, meningkat);
        hasil->sesudah->sebelum = hasil;
        hasil->sebelum = NULL;
    }
    return hasil;
}
```

Bagian untuk membagi daftar menjadi dua untuk pengurutan merge dengan isi konsumen *asal, konsumen **bagianDepan dan konsumen **bagianBelakang. Pada **bagianBelakang berisi konsumen *cepat dan konsumen *lama. Dengan kondisi jika asal adalah NULL atau asal sesudah adalah NULL maka *bagianDepan menjadi asal dan *bagianBelakang menjadi NULL yang artinya tidak ada data atau data hanya ada satu. Kondisi lain yaitu data lebih dari satu pada bagian else maka lama menjadi asal dan cepat menjadi asal sesudah. Dilanjutkan dengan kondisi selama cepat tidak NULL dengan tujuan cepat menjadi NULL pada akhirnya. Dengan perulangan menerus untuk mencapai akhir dari list. *bagianDepan diatur ke asal yang merupakan bagian awal dari list dan *bagianBelakang diatur ke lama sesudah yang merupakan awal dari bagian kedua list. Terakhir mengatur lama sesudah menjadi NULL untuk memutus list pada pemisahan.

```
void bagiDaftar(Konsumen *asal, Konsumen **bagianDepan, Konsumen
**bagianBelakang) {
    Konsumen *cepat;
    Konsumen *lama;
    if (asal == NULL || asal->sesudah == NULL) {
        *bagianDepan = asal;
        *bagianBelakang = NULL;
    } else {
        lama = asal;
        cepat = asal->sesudah;

        while (cepat != NULL) {
            cepat = cepat->sesudah;
            if (cepat != NULL) {
                lama = lama->sesudah;
                cepat = cepat->sesudah;
            }
        }

        *bagianDepan = asal;
```

```
        *bagianBelakang = lama->sesudah;
        lama->sesudah = NULL;
    }
}
```

Bagian untuk searching dengan linear, return konsumen* cariKonsumendenganID yang isinya *id, saat konsumen *sesuai ada dibagian depan maka diproses id yang sesuai dengan strcasecmp dan mengembalikan hasil sesuai, jika id tidak ditemukan maka akan terus mengeser pada list yaitu sesuai ke sesuai sesudah. Jika setelah dicari id tidak ada maka return adalah NULL.

```
Konsumen* cariKonsumendenganID(char *id) {
    Konsumen *sesuai = depan;
    while (sesuai) {
        if (strcasecmp(sesuai->id, id) == 0) {
            return sesuai;
        }
        sesuai = sesuai->sesudah;
    }
    return NULL;
}
```

Bagian untuk searching dengan linear, return konsumen* cariKonsumendengannama yang isinya *nama, saat konsumen *sesuai ada dibagian depan maka diproses nama yang sesuai dengan strcasecmp dan mengembalikan hasil sesuai, jika nama tidak ditemukan maka akan terus mengeser pada list yaitu sesuai ke sesuai sesudah. Jika setelah dicari nama tidak ada maka return adalah NULL.

```
Konsumen* cariKonsumendengannama(char *nama) {
    Konsumen *sesuai = depan;
    while (sesuai) {
        if (strcasecmp(sesuai->nama, nama) == 0) {
            return sesuai;
        }
        sesuai = sesuai->sesudah;
    }
    return NULL;
}
```

Mengedit data konsumen dengan mencari menggunakan id, Konsumen pointer ke *konsumen untuk memanggil funsi cariKonsumendenganID, jika tidak ada konsumen yang dicari maka outputnya adalah konsumen yang dimaksud tidak ada. Jika konsumen yang dicari ketemu maka dapat mulai mengedit data konsumen.

```
void editdataKonsumen(char *id) {
    Konsumen *konsumen = cariKonsumendenganID(id);
    if (!konsumen) {
        printf("Konsumen dengan ID %s tidak ada.\n", id);
        return;
    }

    printf("Edit data konsumen dengan ID %s\n", id);
    printf("Ketikan nama baru: ");
    scanf("%s", konsumen->nama);
    printf("Ketikan nomor telepon baru: ");
    scanf("%s", konsumen->telepon);
    printf("Ketikan durasi berlangganan baru (bulan): ");
```

```
    scanf("%d", &konsumen->durasi);
}
```

Void judul untuk nantinya dipanggil pada intmain() di bagian awal.

```
void judul(){
      printf("Program Manajemen Langganan Gym\n\n");
}
```

Int main() program dengan inisialisai untuk pilihan dengan tipe data int, id dan nama dengan tipe data char. Memanggil void judul di awal, dan menggunakan do while untuk menjalankan pilihan dari user yang memanggil fungsi-fungsi sesuai pilihan. Pada case 5 dan 6 menampilkan output yang tidak dibuat fungsinya diluar int main(). Case 0 untuk keluar program, dan default jika pilihan user tidak sesuai dengan yang tertera.

```
int main() {
    int pilihan;
    char id[PANJANG_ID];
    char nama[PANJANG_NAMA];

    judul();
    do {
        printf("1. Tambah konsumen\n");
        printf("2. Tampilkan semua data konsumen\n");
        printf("3. Urutan konsumen berdasarkan nama (A-Z)\n");
        printf("4. Urutan konsumen berdasarkan nama (Z-A)\n");
        printf("5. Cari data konsumen dengan ID\n");
        printf("6. Cari data konsumen dengan nama\n");
        printf("7. Edit data konsumen berdasarkan ID\n");
        printf("8. Batalkan berlangganan berdasarkan ID\n");
        printf("0. Keluar\n");
        printf("Pilih menu: ");
        scanf("%d", &pilihan);

        switch (pilihan) {
            case 1:
                tambahKonsumen();
                break;
            case 2:
                tampilkanKonsumen();
                break;
            case 3:
                urutanKonsumendengannama(1);
                tampilkanKonsumen();
                break;
            case 4:
                urutanKonsumendengannama(0);
                tampilkanKonsumen();
                break;
            case 5: {
                printf("Ketikan ID konsumen: ");
                scanf("%s", id);
                Konsumen *konsumenberdasarID = cariKonsumendenganID(id);
                if (konsumenberdasarID) {
                    printf("ID: %s\n", konsumenberdasarID->id);
                    printf("Nama: %s\n", konsumenberdasarID->nama);
                    printf("Telepon: %s\n", konsumenberdasarID->telepon);
```

```
                        printf("Durasi: %d bulan\n", konsumenberdasarID-
>durasi);
                    printf("--------------------\n");
                } else {
                    printf("Konsumen dengan ID %s tidak ada.\n", id);
                }
                break;
            }
            case 6: {
                printf("Ketikan nama konsumen: ");
                scanf("%s", nama);
                Konsumen *konsumenberdasarnama =
cariKonsumendengannama(nama);
                if (konsumenberdasarnama) {
                    printf("ID: %s\n", konsumenberdasarnama->id);
                    printf("Nama: %s\n", konsumenberdasarnama->nama);
                    printf("Telepon: %s\n", konsumenberdasarnama->telepon);
                        printf("Durasi: %d bulan\n", konsumenberdasarnama-
>durasi);
                    printf("--------------------\n");
                } else {
                    printf("Konsumen dengan nama %s tidak ada.\n", nama);
                }
                break;
            }
            case 7:
                printf("Ketikan ID konsumen: ");
                scanf("%s", id);
                editdataKonsumen(id);
                break;
            case 8:
                printf("Ketikan ID konsumen: ");
                scanf("%s", id);
                batalkanBerlangganan(id);
                break;
            case 0:
                printf("Keluar dari program.\n");
                break;
            default:
                printf("Pilih menu yang lain.\n");
        }
    } while (pilihan != 0);

    return 0;
}
```