FINAL PROJECT

APLIKASI PENGELOLA DATA PENGGUNAAN LISTRIK

(PE EL EN)

Disusun oleh :

Muhammad Luqman Aristio (2008561022)

Made Yosfin Saputra (2008561023)

Ira Arituddiniyah (2008561027)

I Komang Surya Adinandika (2008561040)

PROGRAM STUDI INFORMATIKA

FAKULTAS MIPA

UNIVERSITAS UDAYANA

2021

KATA PENGANTAR

Segala puji dan syukur kita panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunianya penulis dapat menyelesaikan laporan tugas final projek ini. Adapun judul dari laporan final projek ini yakni "Aplikasi Pengelola Data Penggunaan Listrik (PE EL EN)".

Adapun tujuan dari pembuatan laporan ini yakni untuk memenuhi tugas akhir mata kuliah Praktikum Algoritma dan Pemrograman. Selain itu, pembuatan laporan ini juga bertujuan untuk menambah wawasan penulis juga.

Penulis mengucapkan terima kasih sebanyak banyaknya kepada semua pihak yang telah membantu proses pembuatan laporan ini sehingga dapat diselesaikan dengan baik dan tepat waktu.

Penulis menyadari, laporan yang dibuat masih belum bisa dikatakan sempurna. Oleh karena itu kritik dan saran sangat penulis perlukan demi terwujudnya kesempurnaan laporan ini.

Denpasar, 25 Mei 2021

Penulis

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Program merupakan sebuah perangkat lunak yang memanfaatkan kemampuan komputerisasi komputer untuk melakukan sebuah perhitungan atau logika yang dibuat untuk menghasilkan output yang diinginkan. Hampir seluruh sektor di dunia ini memanfaatkan sebuah program untuk memudahkan pekerjaan mereka sehari-hari. Tentu program tersebut harus dapat berjalan sesuai dengan keinginan mereka. Sebagai contoh seperti program PE EL EN yang telah penulis buat.

PE EL EN merupakan program aplikasi yang bertujuan untuk membantu mengelola data sebuah perusahaan listrik. Selain itu tujuan pembuatan program ini yakni untuk memenuhi tugas projek akhir mata kuliah Praktikum Algoritma dan Pemrograman. Dalam program ini user dapat mengolah berbagai data yang diperlukan dalam sebuah perusahaan listrik contohnya data pengguna, data tagihan dan data pembayaran.

Seperti yang kita ketahui sebuah perusahaan listrik pasti memiliki banyak pelanggan. Jika data-data pelanggan tersebut diolah secara manual pasti akan timbul berbagai permasalahan. Oleh karena itu perlu dibuatnya sebuah aplikasi untuk membantu mengolah data-data tersebut sehingga dapat dioperasikan oleh sejumlah orang. Dengan begitu efisiensi pengolahan data perusahaan tersebut dapat maksimal dan minim kesalahan.

1.2 Tujuan

Disamping latar belakang tersebut, adapun beberapa tujuan penulis menyusun laporan ini yakni sebagai berikut :

1. Untuk memenuhi projek akhir Praktikum Algoritma dan Pemrograman
2. Untuk menjelaskan hasil program aplikasi PE EL EN
3. Mengetahui alur logika program aplikasi PE EL EN

BAB 2
LANDASAN TEORI

2.1 Variabel, Tipe Data dan Konstanta

Variabel dapat diartikan sebagai nama atau pengenal yang digunakan untuk menampung suatu data atau nilai tertentu. Dalam Bahasa C variable memiliki beberapa sifat sifat yang wajib diperhatikan diantaranya :

- Memiliki tipe data seperti integer, string, float, double dan lain-lain
- *Case sensitive* atau wajib memperhatika huruf besar dan kecilnya dalam penulisan
- Dilarang menggunakan spasi
- Tidak boleh diawalai symbol khusus contoh *?+-#@ dan lain lain
- Nilai variable dapat berubah-ubah

Berikutnya tipe data, tipe data adalah sebuah pengelompokan data untuk memberitahu compiler atau interpreter bagaimana programmer ingin mengolah data tersebut. Ada beberapa jenis tipe data pertama yakni tipe data dasar contohnya Char, Integer dan Float. Kedua tipe data turunan contohnya array, struct, pointer. Ketiga tipe data bentukan yakni enum. Keempat tipe data void yang biasanya digunakan sebagai fungsi.

Terakhir yakni konstanta. Pengertian konstanta sama dengan variable namun yang membedakan hanya pada nilainya saja. Jika pada variable nilainya dapat berubah ubah tidak pada konstanta. Nilainya tetap pada saat dideklarasikan. Sesuai dengan tujuannya konstanta digunakan jika kita ingin suatu nilai itu tetap tidak berubah selama program berjalan apapun yang terjadi. Jika kita ingin mengubahnya harus langsung dari source code itu sendiri.

2.2 Percabangan dan Perulangan

Percabangan digunakan ketika program memiliki alur yang berbeda-beda. Oleh karena itu percabangan dapat menentukan alur program tergantung dengan kondisi yang diberikannya. Ada beberapa jenis percabangan yang penulis ketahui yaitu :

1. *if*

Percabangan *if* merupakan percabangan yang hanya memiliki satu kondisi atau situasi yakni saat bernilai *true* atau benar.

2. *if else*

Percabangan *if else* merupakan percabangan yang memiliki dua kondisi atau situasi yakni saat bernilai *true* (*if*) dan saat bernilai *false* (*else*).

3. *if else if*

Percabangan *if else if* merupakan percabangan yang memiliki lebih dari dua kondisi atau situasi tergantung jumlah yang diinginkan. Percabangan if else if sangat sering digunakan saat program memiliki banyak pilihan atau kondisi didalamnya.

4. *Switch case*

Percabangan *switch case* hampir mirip dengan percabangan yang mirip dengan *if else if.* Dengan *switch case* kita bisa membuat banyak kondisi atau situasi. Selain itu penggunaannya juga cukup mudah karena kode nya yang simple dan mudah dilihat. Diawali dengan *switch(variabel)*. Variabel ini yang akan di cek nantinya saat didalam case. Berikut nya *case value, value* ini merupakan nilai yang akan dibandingkan dengan variable. Jika sesuai maka *case* akan dijalankan, setelah itu setiap *case* diakhiri dengan *break* agar program berhenti mengecek *case* lainnya saat ada *case* yang terpenuhi.

Kemudian diakhir semua *case* terdapat *default. Default* ini sama dengan fungsi *else* pada *if else* yakni nilai false jika tidak ada case yang benar.

5. Operator ternary

Percabangan ternary ini mirip dengan percabang *if else* karena terdapat 2 kondisi yaitu *true* dan *false* saja. Operator ternary menggunakan tanda tanya '?' yang digunakan untuk mengecek apakah bernilai true atau false. Contohnya yaitu seperti ini

(kondisi) ? true : false

6. Percabangan bersarang

Percabangan bersarang ini diartikan jika ada percabangan didalam sebuah percabangan. Contoh dalam kondisi *if* terdapat lagi *if else* lagi didalamnya.

Berikutnya yakni perulangan. Sesuai namanya perulangan digunakan untuk mengulang suatu alur dalam program hingga mencapai kondisi yang diinginkan atau ditentukan. Perulangan bisa digunakan untuk menampilkan data dalam jumlah yang besar atau bahkan mencari sebuah data. Beberapa jenis perulangan yakni :

1. For

Perulangan *for* merupakan jenis perulangan yang sudah ditentukan berapak kali harus mengulang atau yang sering disebut *counted loop*. Dengan perulangan for kita mengatur kondisi yang diinginkan. Untuk struktur dasar nya yaitu :

```
for(i=0;i<10;i++){

}
```

Untuk i=0 merupakan awal hitungan dimulai dari 0 kemudian berikutnya menetukan jumlah perulangannya. Dalam contoh diatas perulangan yang terjadi kurang dari 10 kali yang artinya akan berakhir saat mencapai nilai i=9. Kemudian perulangannya akan bertambah 1 karena i++. Selain pertambahan juga bisa pengurangan. Apabila ingin melakukab perulangan berkurang gunakan i--.

2. While

Perulangan while merupakan perulangan yang tergolong 2 jenis yakni bisa *counted loop* dan *uncounted loop*. Dikatakan *counted loop* apabila kita sudah menyiapkan batas atau counter nya sebelum perulangan dimulai. Jadi dalam kondisi while nya sudah di atur batas perulangannya. Contoh while (a<10).

Sedangkan perulangan while dikatakan *uncounted loop* saat perulangannya tidak bisa ditentukan kapan selesainya. Yang artinya perulangannya dipengaruhi oleh input dari user nya, Contohnya yakni :

```
while (a!=1){
printf("Masukan angka : ");
scanf("%d", &a);
}
```

Dalam kode tersebut perulangan akan terus terjadi apabila user tidak menginputkan angka 1. Sehingga dikatakan *uncounted loop*.

3. *Do/While*

Perulangan *Do/While* merupakan bentuk lebih kompleks dari *while*. Namun dari segi fungsi nya masih sama. Letak utama perbedaannya yakni ada pada pengecekan whilenya. Jika pada *while* mengecek pada bagian awal maka *Do/While* akan mengecek pada bagian akhir. Selain itu perbedaanya terletak pada penambahan *do*. Dengan *do* ini kita bisa mengatur dibagian mana program akan mengulang sehingga lebih fleksibel. Contoh penerapannya :

```
do{
printf("Masukan angka : ");
scanf("%d", &a);
}while(a!=1)
```

Dalam contoh tersebut user akan menginputkan angka terlebih dahulu baru akan di cek akan melakukan perulangan atau tidak dibagian akhir.

4. Perulangan Bersarang

Perulangan bersarang itu sama artinya melakukan perulangan didalam sebuah perulangan. Contoh dengan menggunakan for seperti ini :

```
for(i=0;i<10;i++){
    for(j=0;j<5;j++){
    }
}
```

Untuk menghitung berapa kali jumlahnya mengulang kita hanya perlu mengalikan batas setiap perulangannya saja. Contoh diatas yakni 50 kali perulangan. Untuk proses pengulangannya dimulai dari i=0. Saat i=0 perulangan j dilakukan sebanyak 5 kali yang artinya sampai nilai batas j tercapai. Saat j mencapai batasnya baru nilai berulang kembali menjadi 1 dan seterusnya.

2.3. Array

Array merupakan struktur data yang berfungsi menyimpan banyak data pada satu lokasi atau tempat yang sudah disediakan. Namun walaupun dapat menyimpan banyak data, dalam 1 array semua data wajib memiliki 1 tipe data yang sama. Untuk mengakses data dalam sebuah array bisa dilakukan dengan mencari pada indeks keberapa data tersebut berada. Indeks data pada array selalu dimulai dari 0.

Sesuai dengan pengertiannya dapat disimpulkan array dapat mempermudah kita agar tidak terlalu banyak membuat variable sehingga source code bisa menjadi lebih sedikit dan efisien. Selain itu pengaksesan datanya juga mudah dengan tinggal menentukan indeks datanya saja.

Untuk membuat array pada sebagian besar bahasa pemrograman memiliki cara yang sama yakni dimulai dengan menentukan tipe data array tersebut terlebih dahulu. Kemudian tentukan panjang dari array tersebut dan berikutnya array tersebut bisa diisi dengan data yang diinginkan sesuai dengan tipe datanya.

Terdapat 3 jenis array mulai dari yang 1 dimensi, 2 dimensi dan multidimensi. Array 1 dimensi biasanya hanya terdiri dari 1 baris data saja. Kemudian array 2 dimensi terdiri dari baris dan kolom. Sedangkan array multidimensi adalah array memiliki banyak dimensi / lebih dari 2 dimensi.

2.4. Pointer dan Fungsi Operasi String

Pointer dapat diartikan sebagai sebuah variable yang menyimpan alamat memori dari variable lain. Untuk membuat pointer dalam bahasa C kita hanya perlu menambahkan tanda bintang (*) sebelum nama variabelnya. Untuk tipe datanya sendiri harus sama dengan tipe data nilai yang akan ditunjuk nanti. Pointer biasanya digunakan ketika ingin mengakses suatu data. Data tersebut dapat berupa integer, string, array dan lain-lain.

Sehingga dengan memanfaatkan pointer prosesnya menjadi lebih mudah dan efisien dikarenakan nilai yang disimpan dalam pointer merupakan alamat memorinya. Selain itu pointer juga dapat mengubah nilai dalam suatu fungsi. Karena pointer menunjuk ke alamat memori variable itu sendiri sedangkan fungsi menduplikasi variable ke dalam parameter.

Berikutnya yakni fungsi operasi string. Ada beberapa fungsi operasi string yang sering digunakan diantaranya :

1. Fungsi strcpy()

Fungsi strcpy() digunakan untuk menyalin string dari sebuah variabel awal ke variabel tujuan. Variabel asal dan tujuan tentu harus memiliki ukuran yang sama. Bentuk dalam kode bahasa C yakni :

```
strcpy(variable_tujuan, variable_awal);
```

2. Fungsi strlen()

Fungsi strlen() digunakan untuk menghitung panjang / banyaknya karakter dalam string. Bentuk dalam kode bahasa C yakni :

```
strlen(variable_string);
```

3. Fungsi strcet()

Fungsi strcet() digunakan untuk menambahkan string pada suatu variable dari belakang . Bentuk dalam kode bahasa C yakni :

```
strcat(variable_yang_ditambahkan, variable_penambah);
```

4. Fungsi strupr()

Fungsi strupr() digunakan untuk mengubah setiap huruf kecil dalam string menjadi huruf kapital (huruf besar) . Bentuk dalam kode bahasa C yakni :

```
strupr(variabel);
```

5. Fungsi strstr()

Fungsi strstr() digunakan untuk mencari sebuah teks (string) di dalam string.. Bentuk dalam kode bahasa C yakni :

```
strstr(variable_string, "teks yang dicari");
```

6. Fungsi strchr()

Fungsi strchr() digunakan untuk mencari sebuah karakter di dalam variable string. Bentuk dalam kode bahasa C yakni :

```
strstr(variable_string,'karakter yang dicari');
```

7. Fungsi strcmp()

Fungsi strcmp() digunakan untuk membandingkan variabel string dengan variable string yang lainnya apakah sama atau tidak. Bentuk dalam kode bahasa C yakni :

```
strcmp(variable_pertama, variable_kedua);
```

2.5. Operasi File

Operasi file dalam pemrograman sesuai dengan namanya dapat diartikan sebagai proses menyimpan data input dan output dalam sebuah program dalam sebuah file, yang nanti file ini dapat diakses oleh pengguna dan digunakan untuk suatu keperluan.

Dalam sebuah program data yang kita inputkan hanya akan tersimpan sementara dalam memori dan akan terhapus ketika program dihentikan atau ditutup. Oleh karena itu dengan disimpan dalam sebuah file kita tetap bisa mengakses data tersebut walaupun program sudah ditutup. Ada 3 mode utama dalam file file yaitu read, write, dan append sedangkan dalam pengoperasian file ada membaca, menutup, menghapus, dan mengubah file. Dalam pemrograman nanti masing masing operasi akan memiliki caranya tersendiri.

Ada beberapa fungsi operasi file yang sering digunakan diantaranya :

1. fopen()

Sesuai dengan namanya fungsi fopen berfungsi untuk membuka file sesuai dengan mode yang diinginkan. Fungsi fopen() membutuhkan pointer untuk mengakses filenya sehingga sebelumnya kita harus membuat pointer dengan type data FILE yang menunjuk fungsi fopen(). Contoh implementasi :

```
FILE *ptr;

ptr=fopen("namafile.ektensi", "r");
```

2. fprintf() dan fputs()

Kedua fungsi tersebut digunakan menulis atau mencetak data kedalam file. Format untuk pendeklarasian fungsi fprintf yakni fprintf(namafile, "konstanta char", nama variabel);. Sedangkan untuk fungsi fputs() yakni fputs(nama variable, nama file);. Contoh :

```
fprintf(ptr, "%s", data);

fputs(data, ptr);
```

3. fgets() dan fscanf()

Kedua fungsi ini memilik fungsi utama yang sama yakni untuk membaca file, letak perbedaanya yakni fungsi fgets() hanya bisa membaca data bertipe string sedangkan fscanf() bisa membaca data bertipe string maupun bilangan atau angka. Untuk formatnya fgets() yakni fgets(nama variabel, jumlah string, nama file);. Sedangkan format fscanf() yakni fscanf(nama file, "konstanta char", &nama variabel);. Contoh :

```
fgets(data, 100, ptr);
```

```
fscanf(ptr, "%s", &data);
```

4. fclose()

Sesuai namanya, fungsi ini digunakan untuk menutup file. Formatnya yakni fclose(nama variabel). Contoh :

```
fclose(ptr);
```

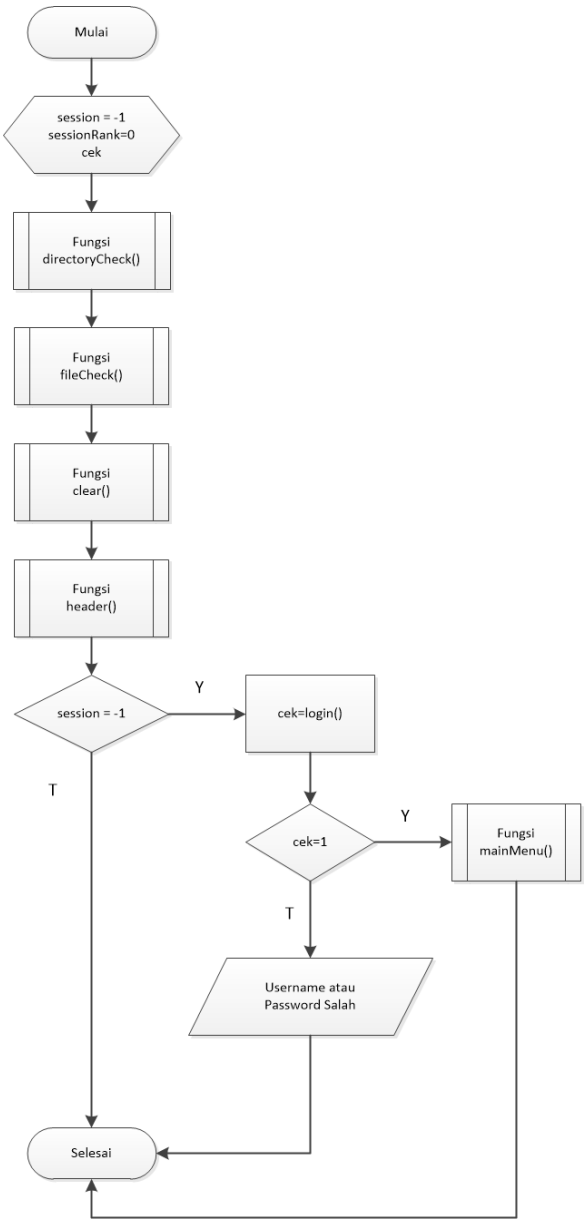
BAB 3

DESAIN DAN METODE

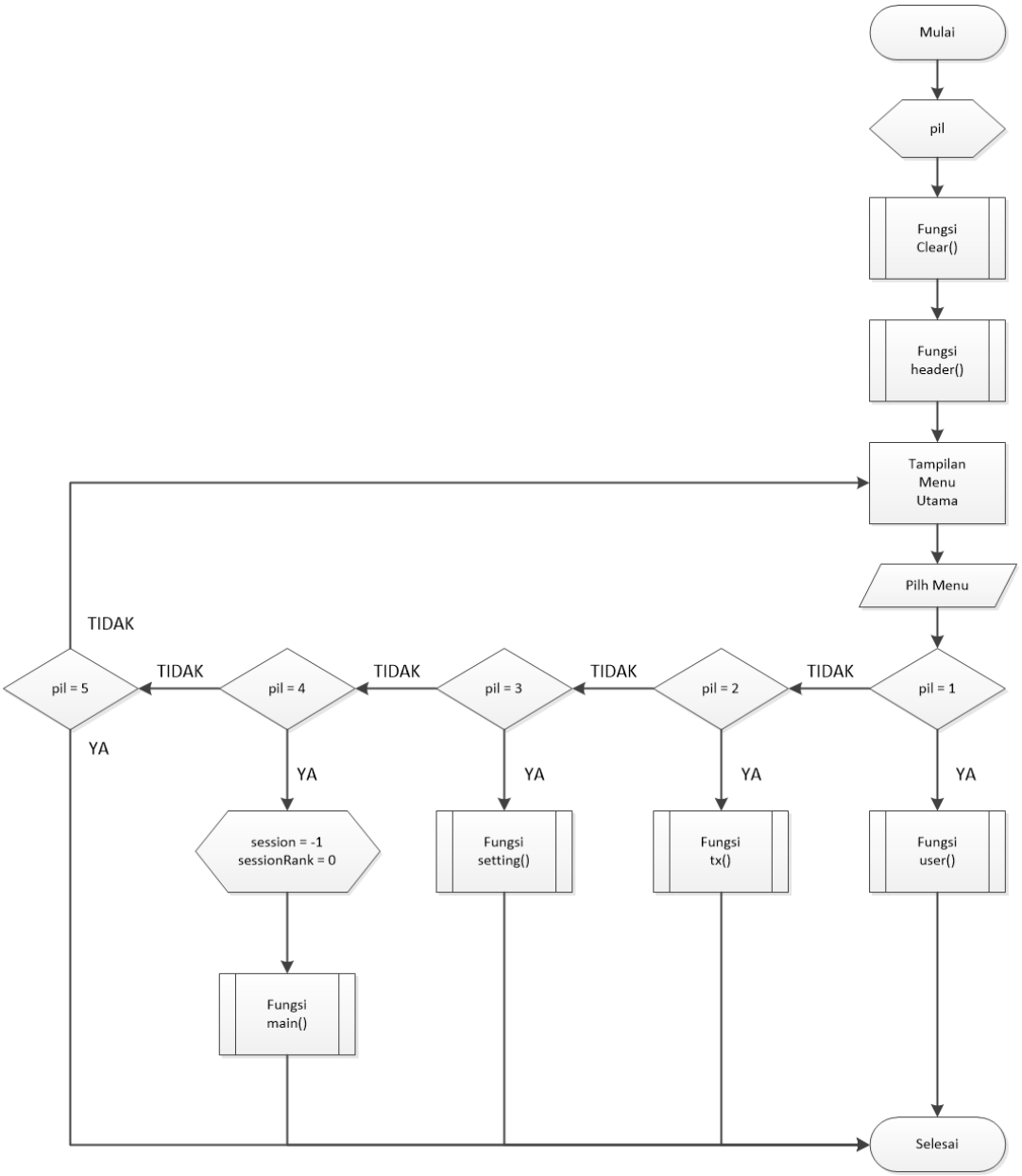
3.1 Flowchart Program

Dalam program aplikasi PE EL EN tersebut terdapat banyak fungsi yang disimpan dalam file yang berbeda-beda oleh karena itu flowchart dibuat terpisah sesuai fungsi masing-masing sebagai berikut :

1. Fungsi main()

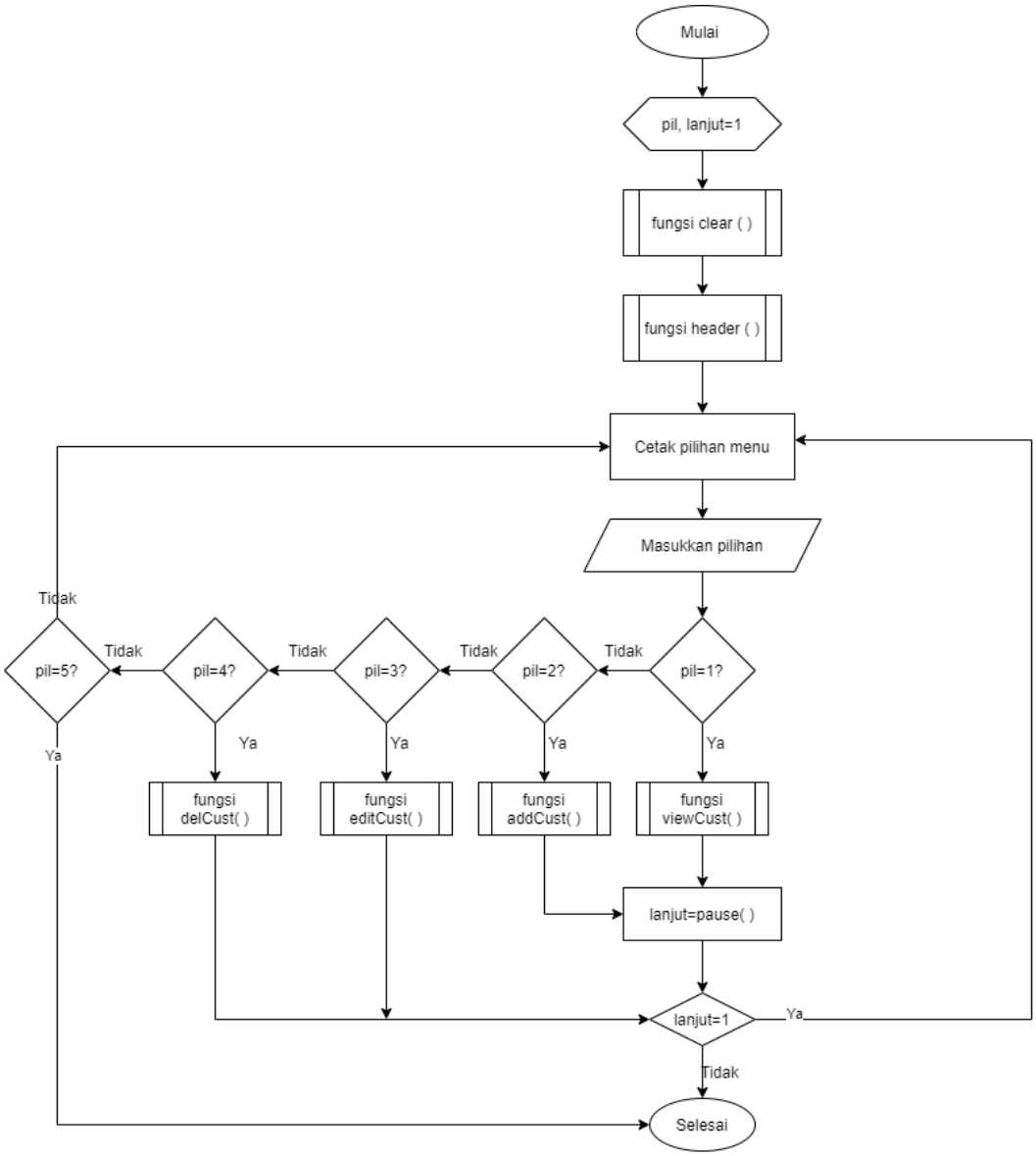


2. Fungsi mainMenu()

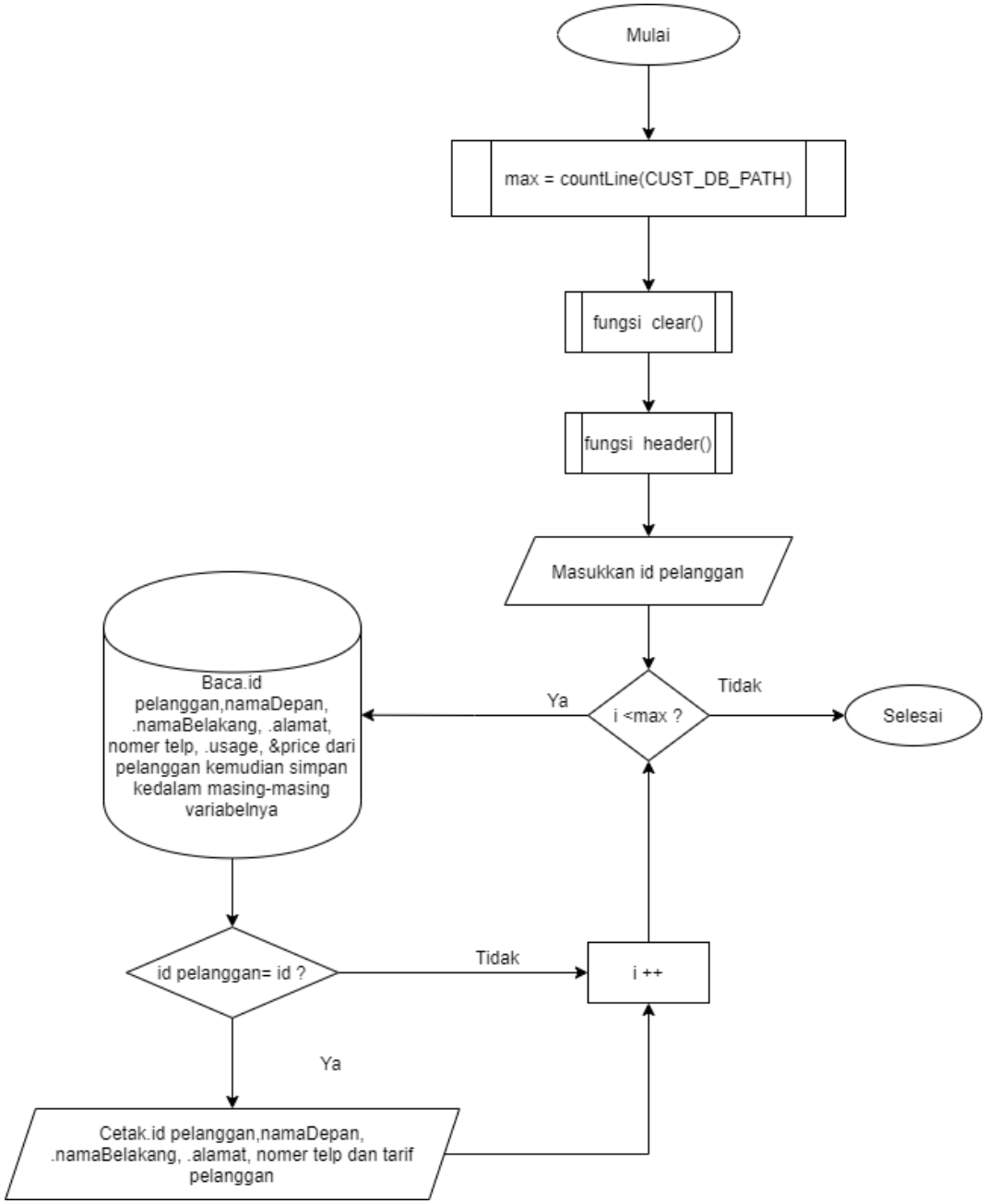


3. Fungsi user()

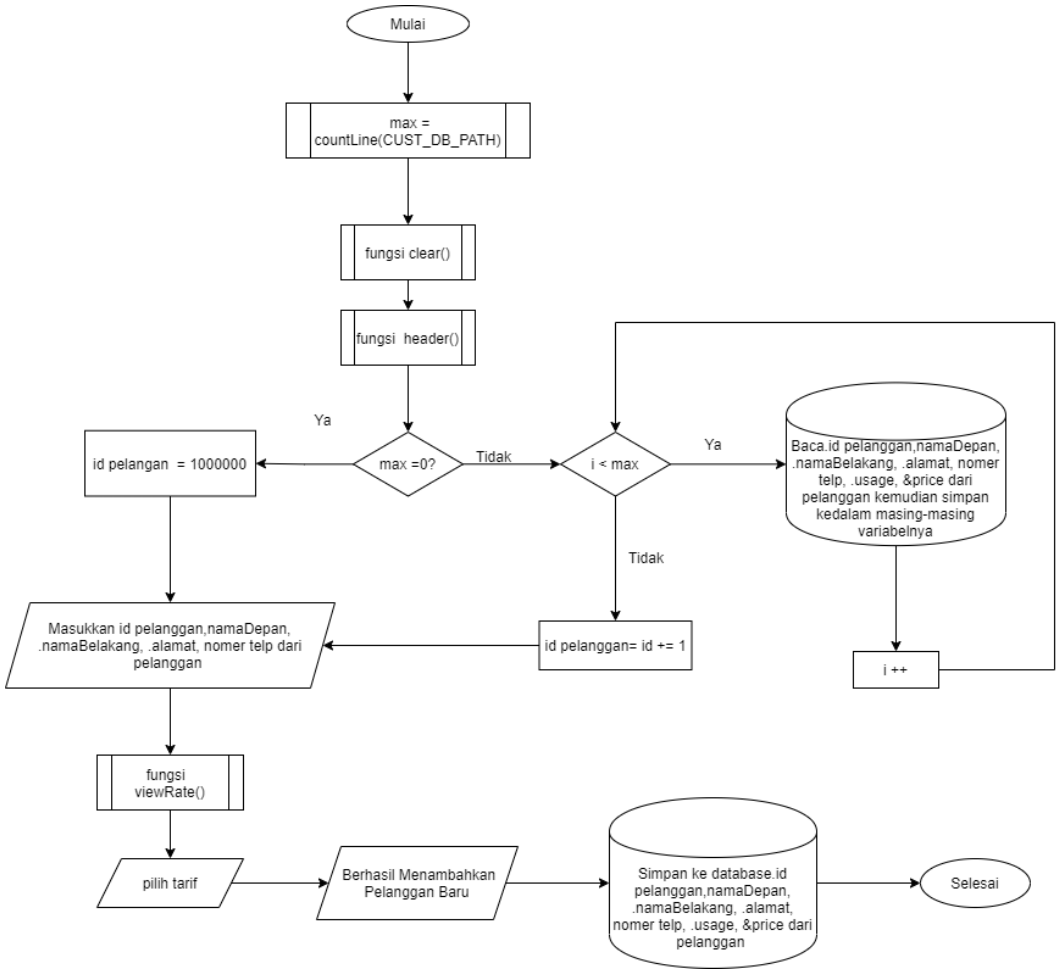
❖ Menu Pelanggan



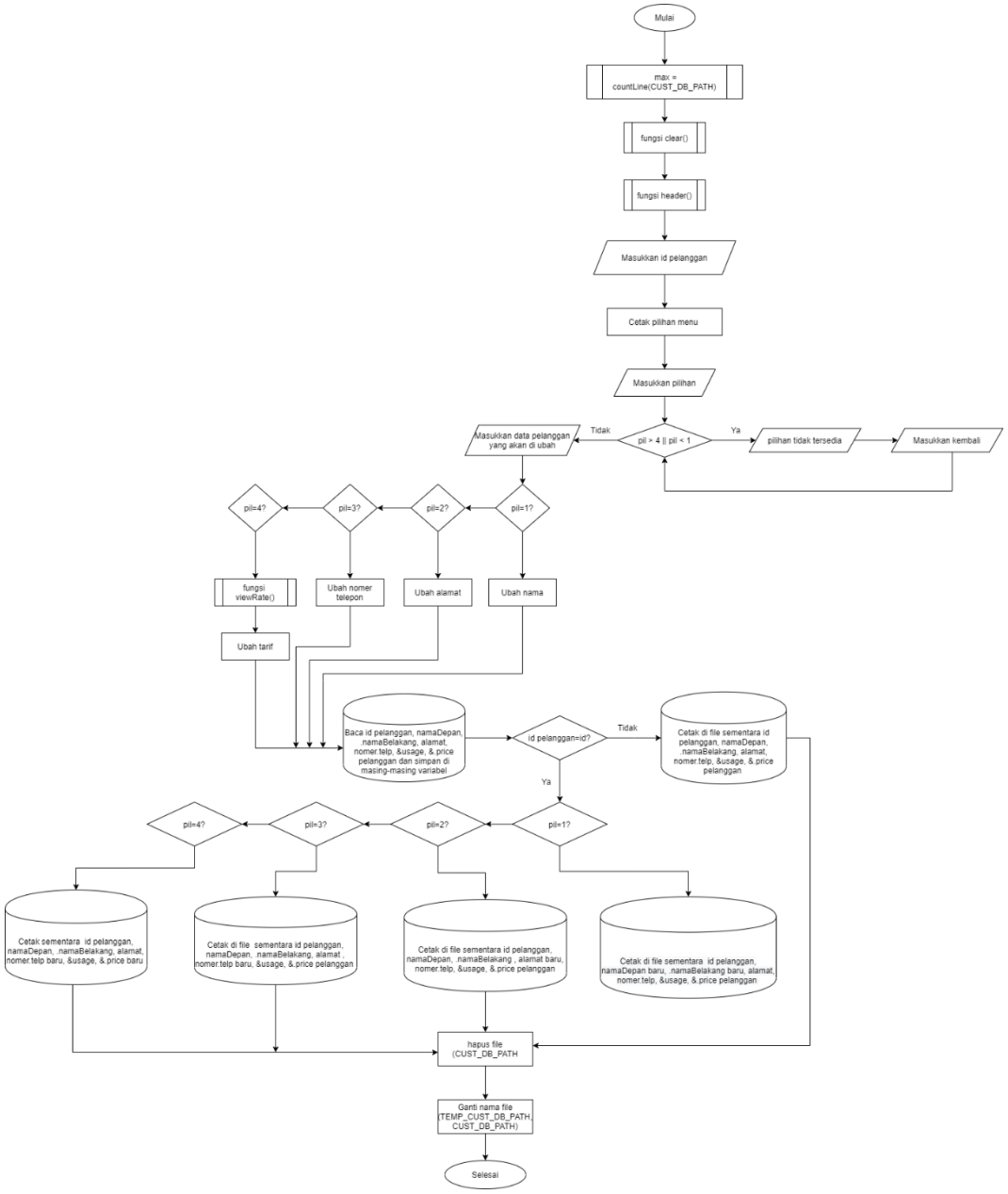
❖ **Tampilkan pelanggan**



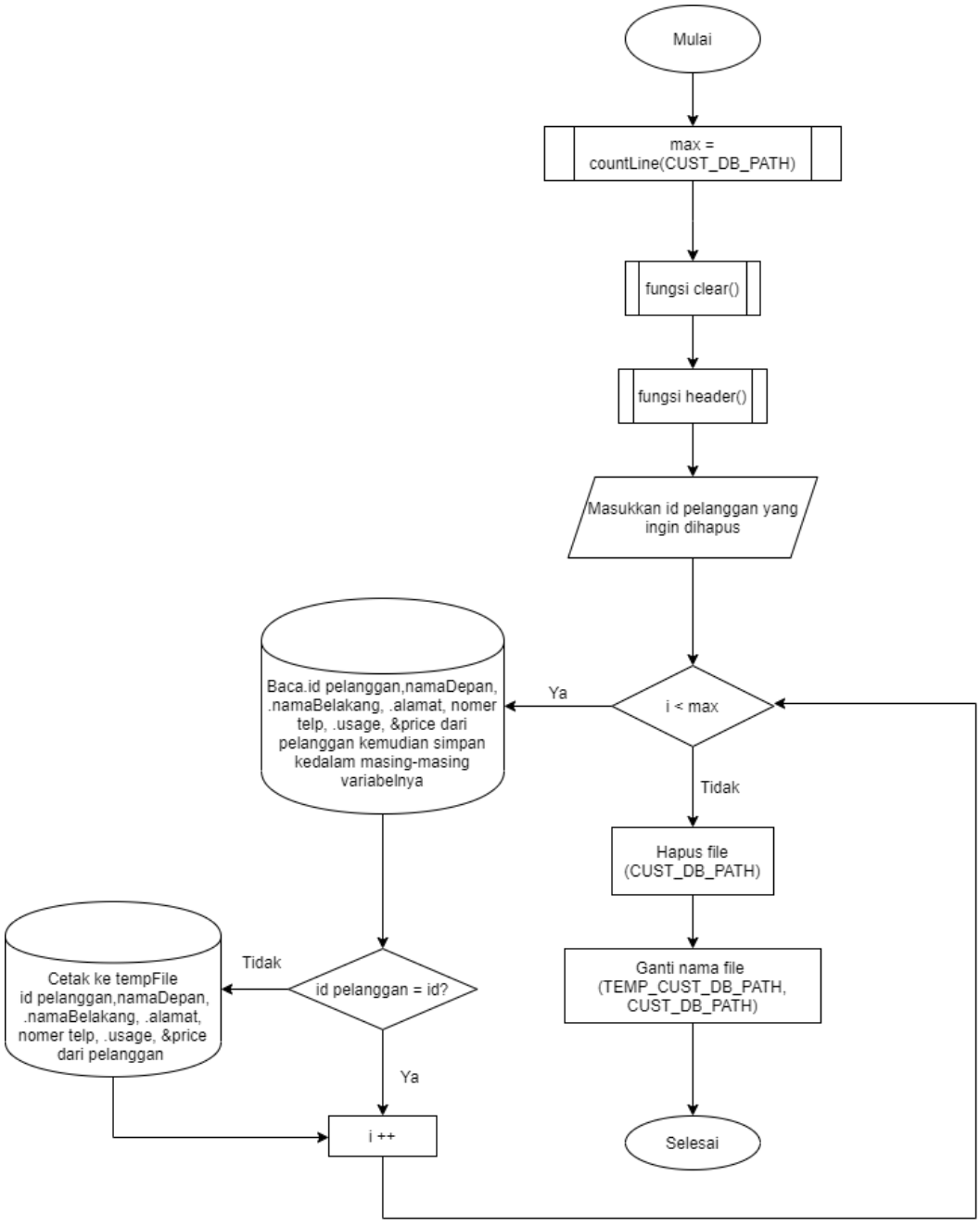
❖ Tambah Pelanggan



❖ Edit Pelanggan

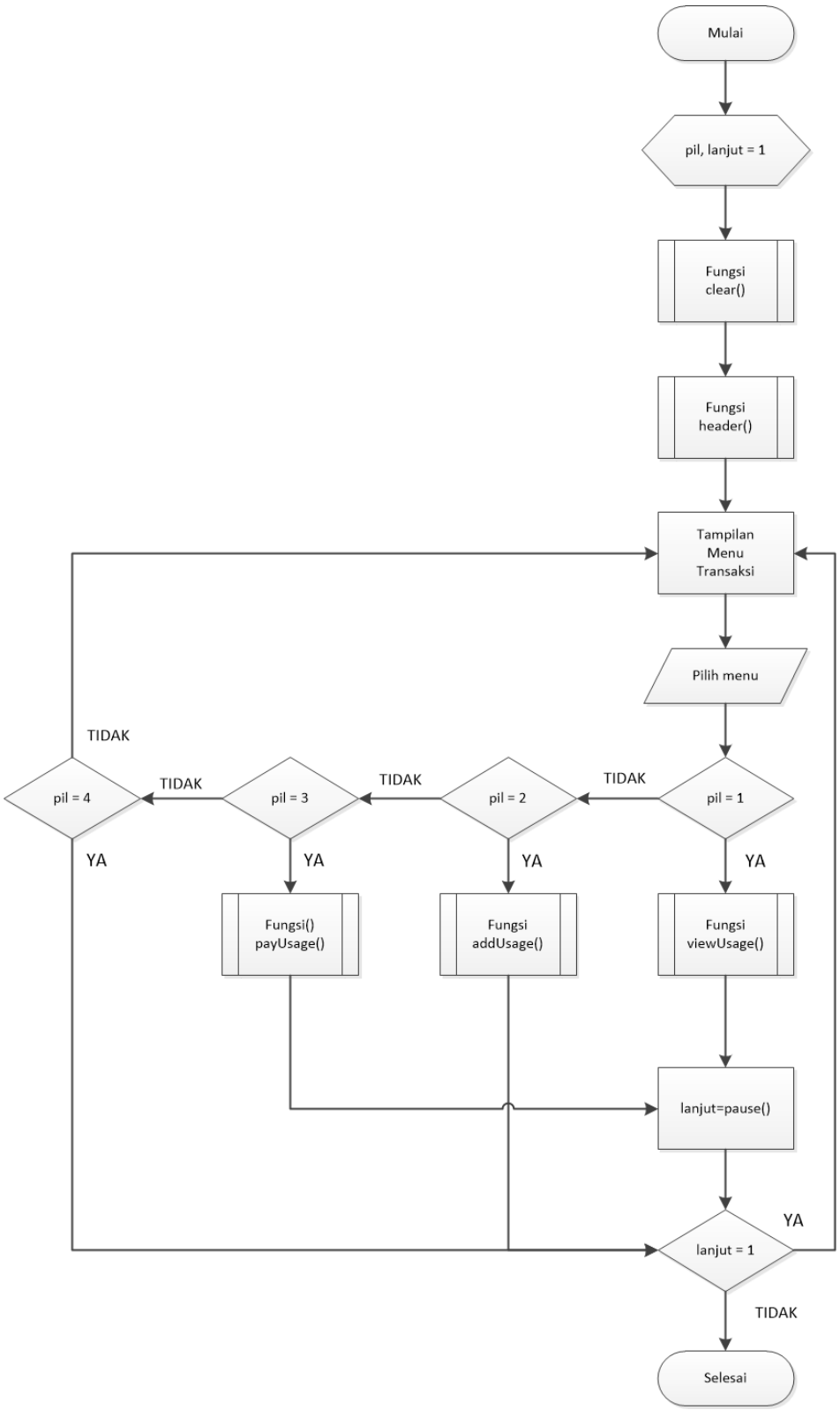


❖ **Hapus pelanggan**

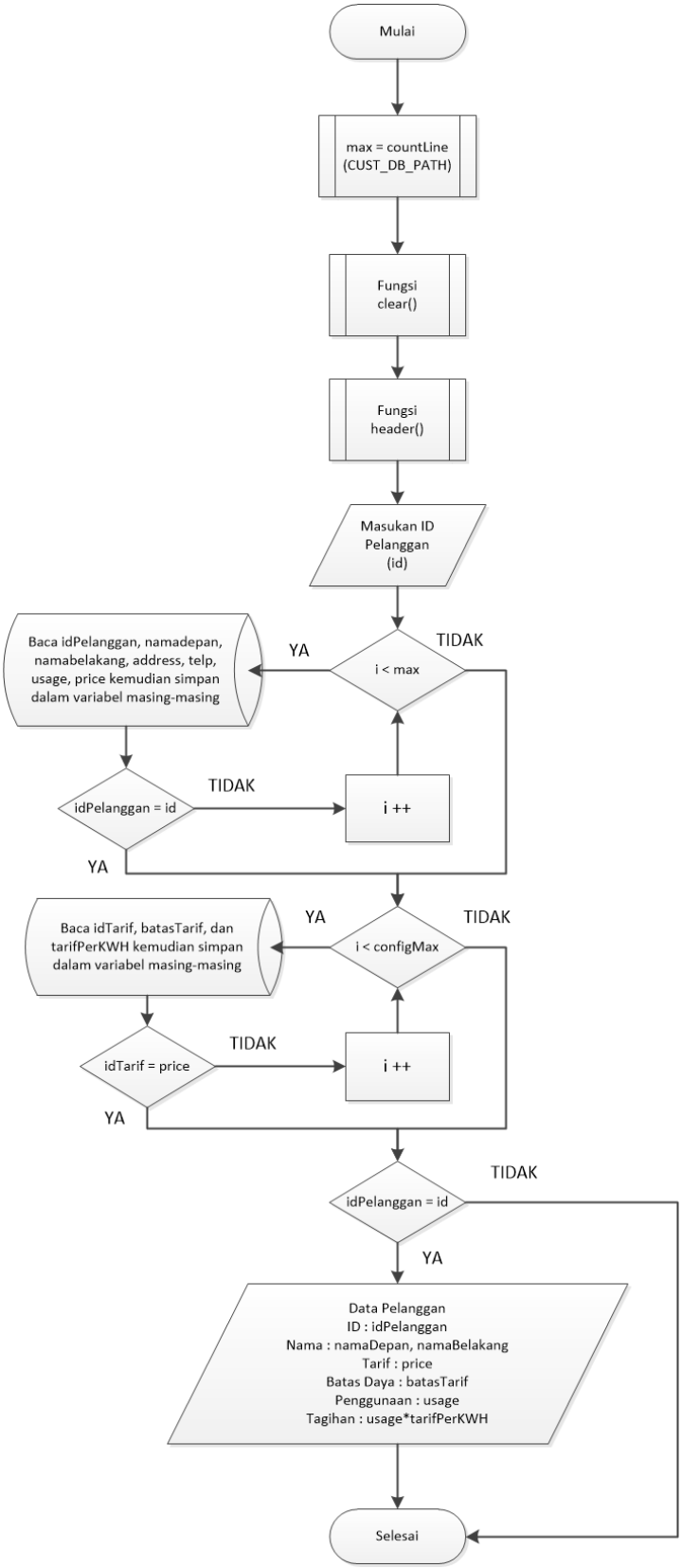


4. Fungsi transaksi

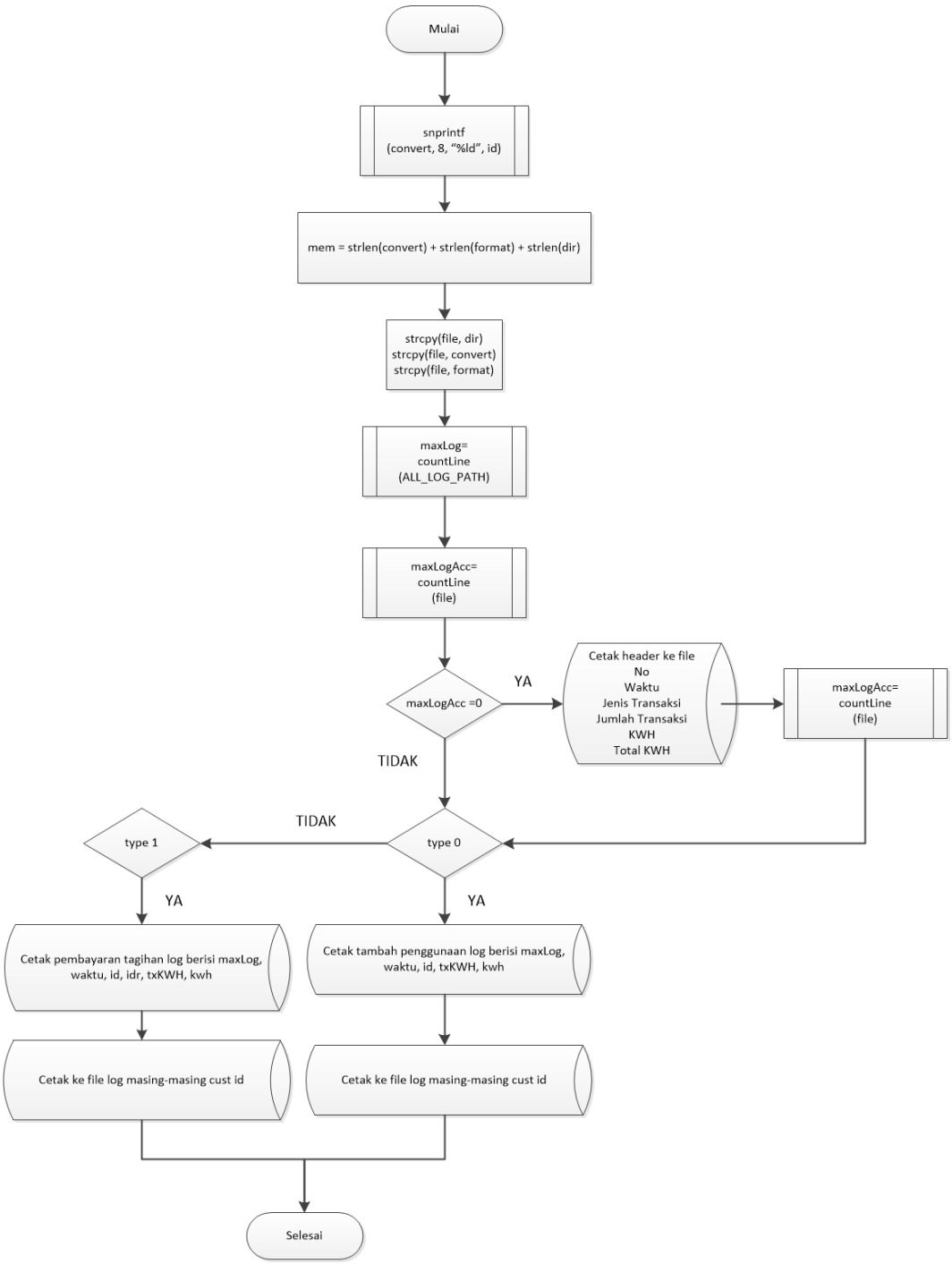
- **Menu TX()**



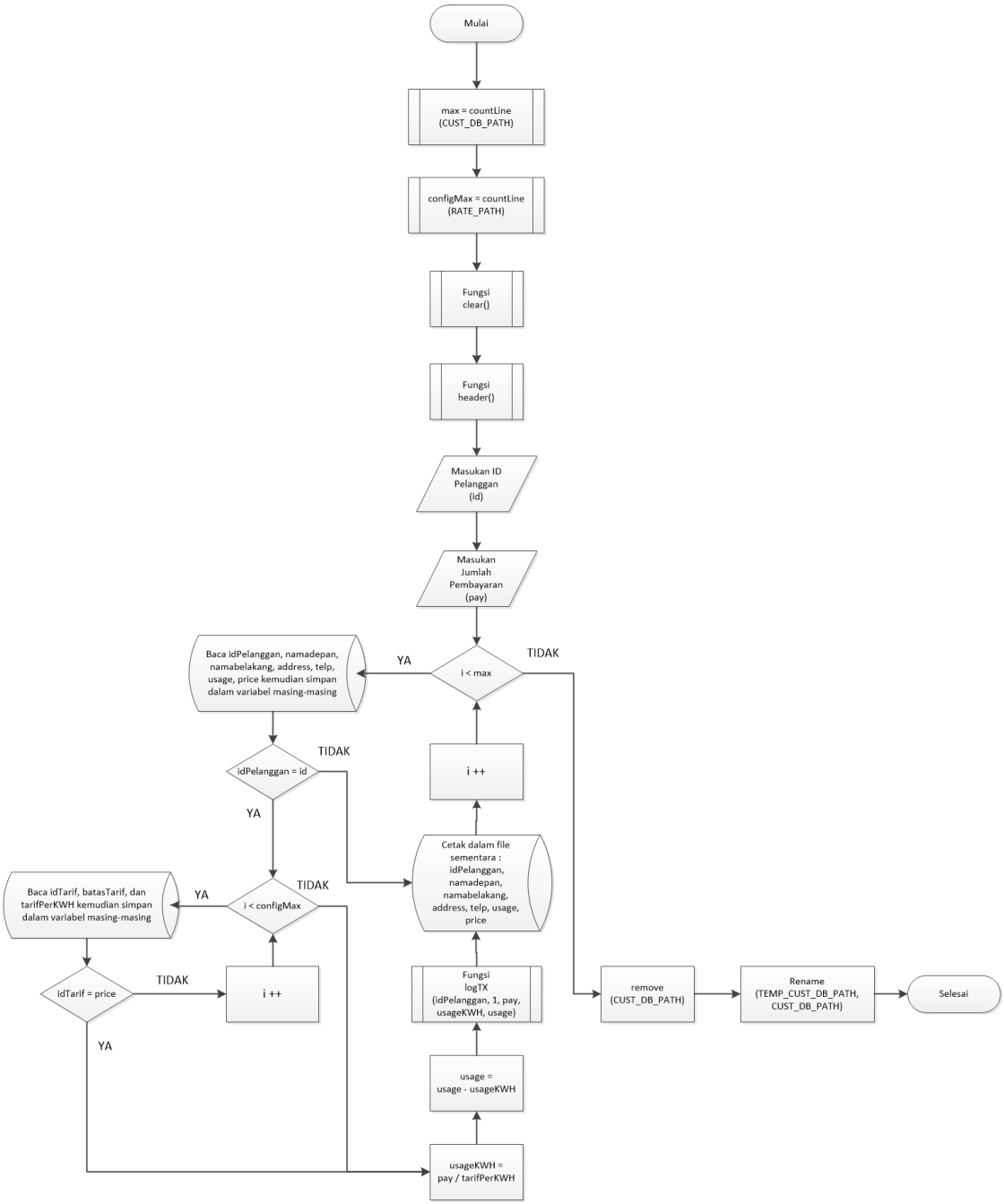
❖ **Fungsi viewUsage()**



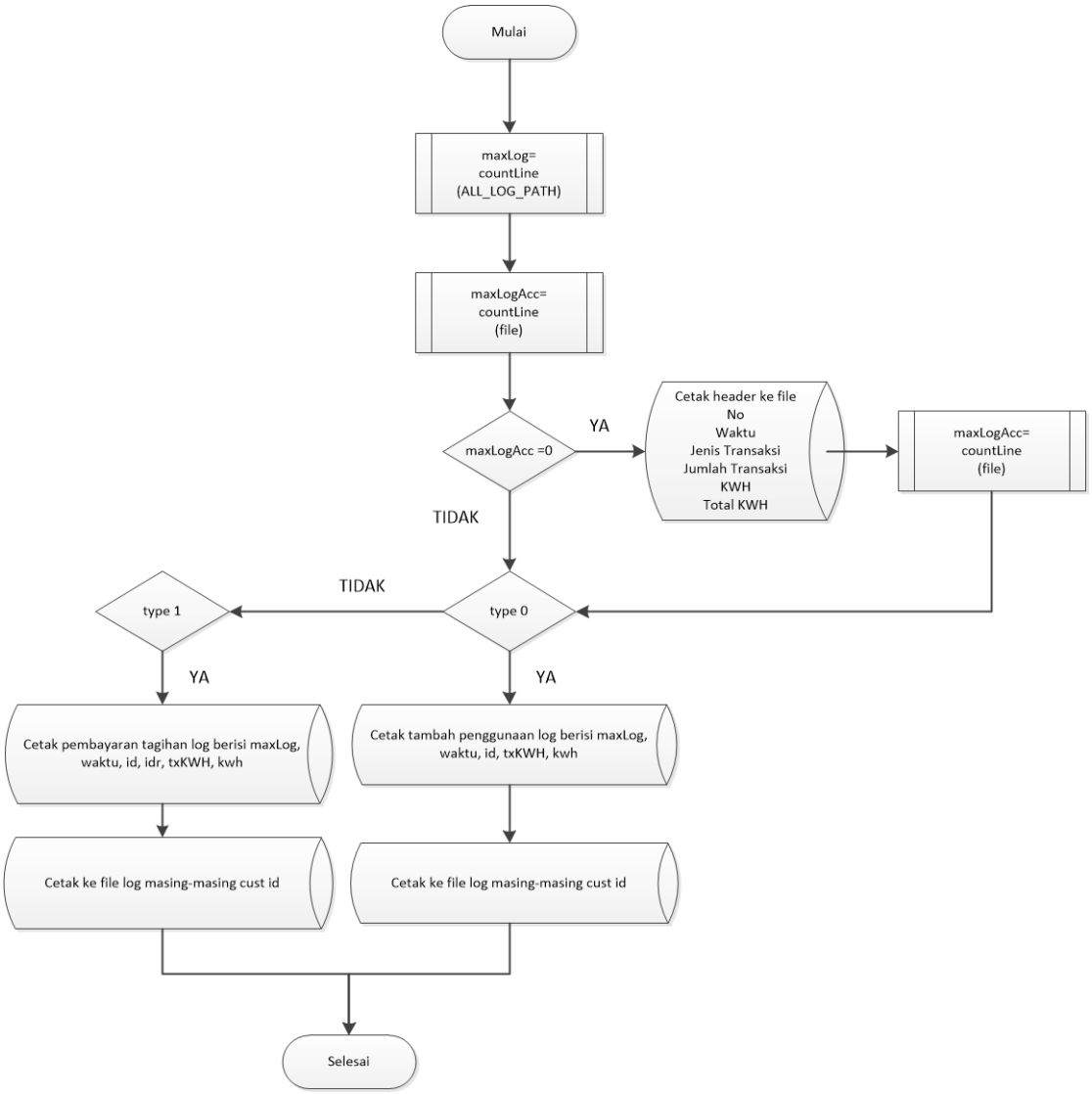
❖ **Fungsi addUsage()**



❖ Fungsi payUsage()

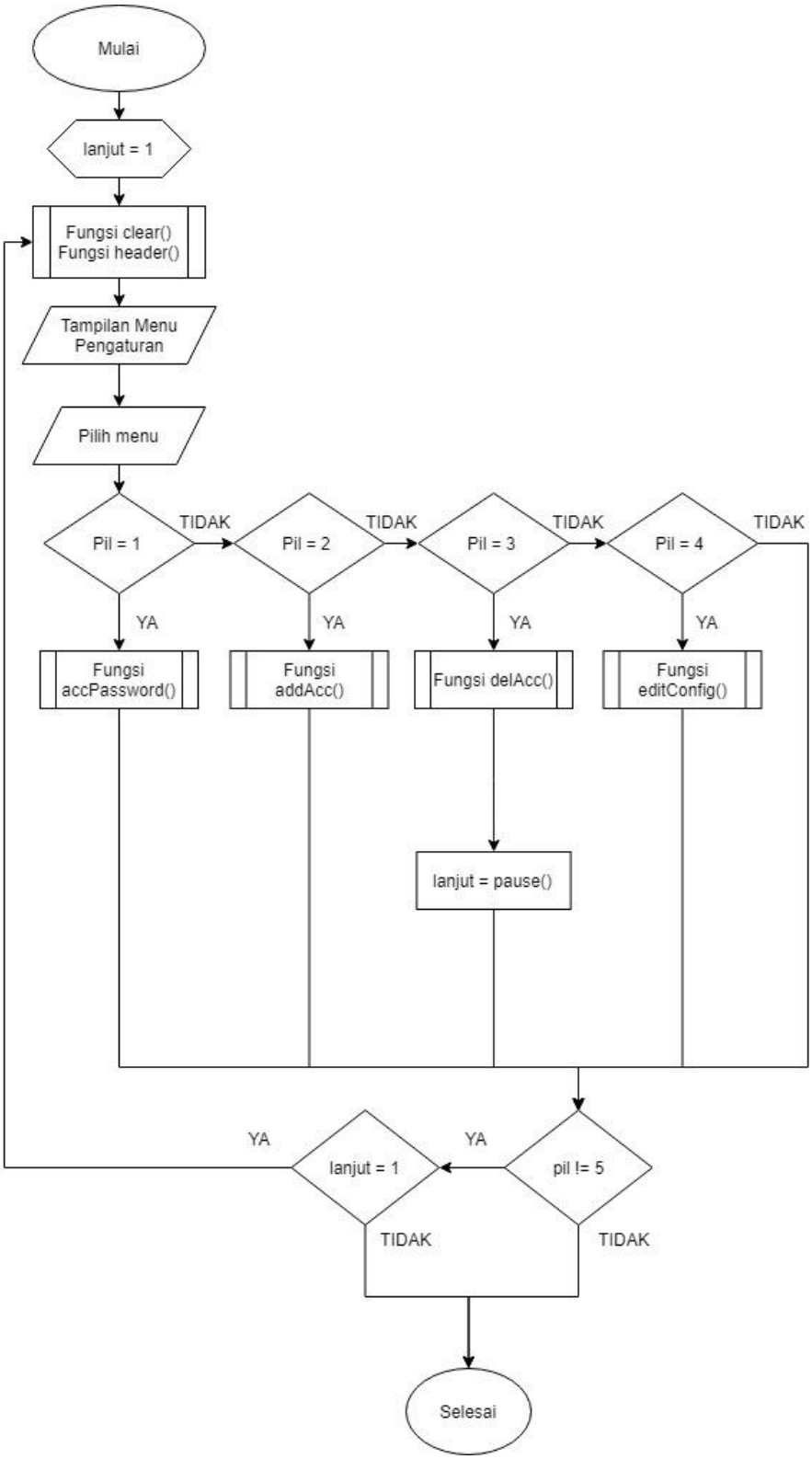


❖ **Fungsi logTX()**

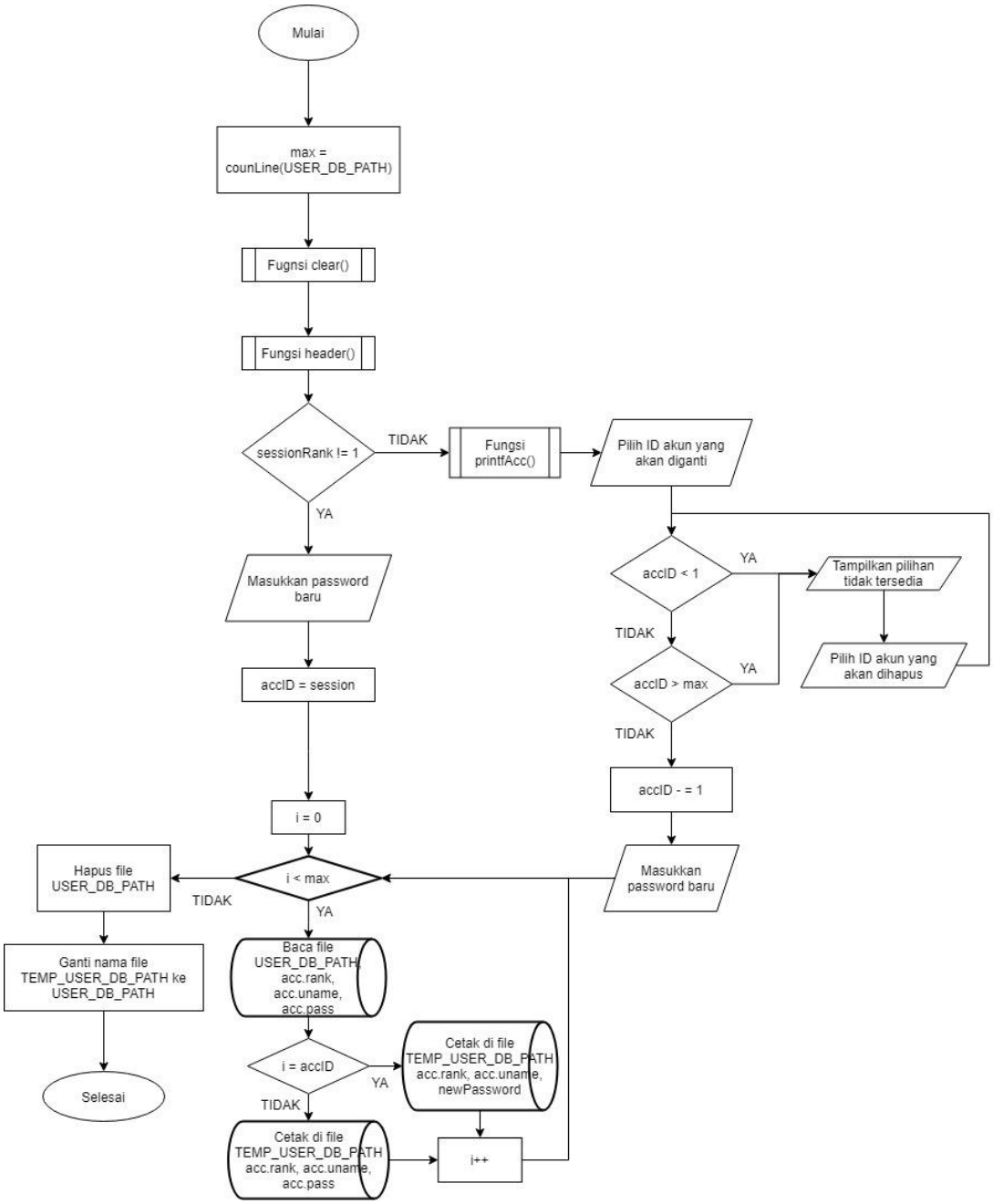


5. Fungsi setting()

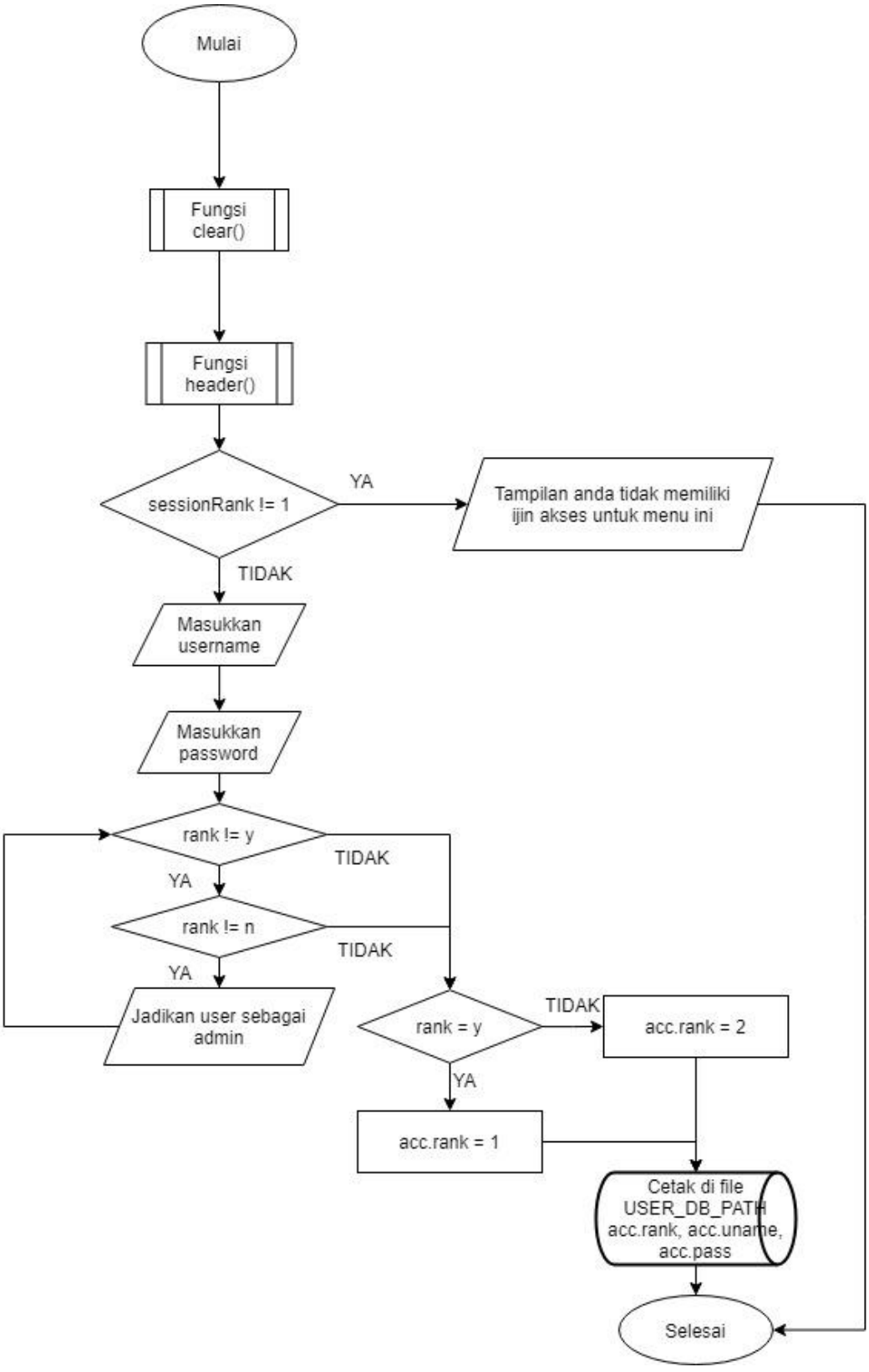
❖ Menu Pengaturan



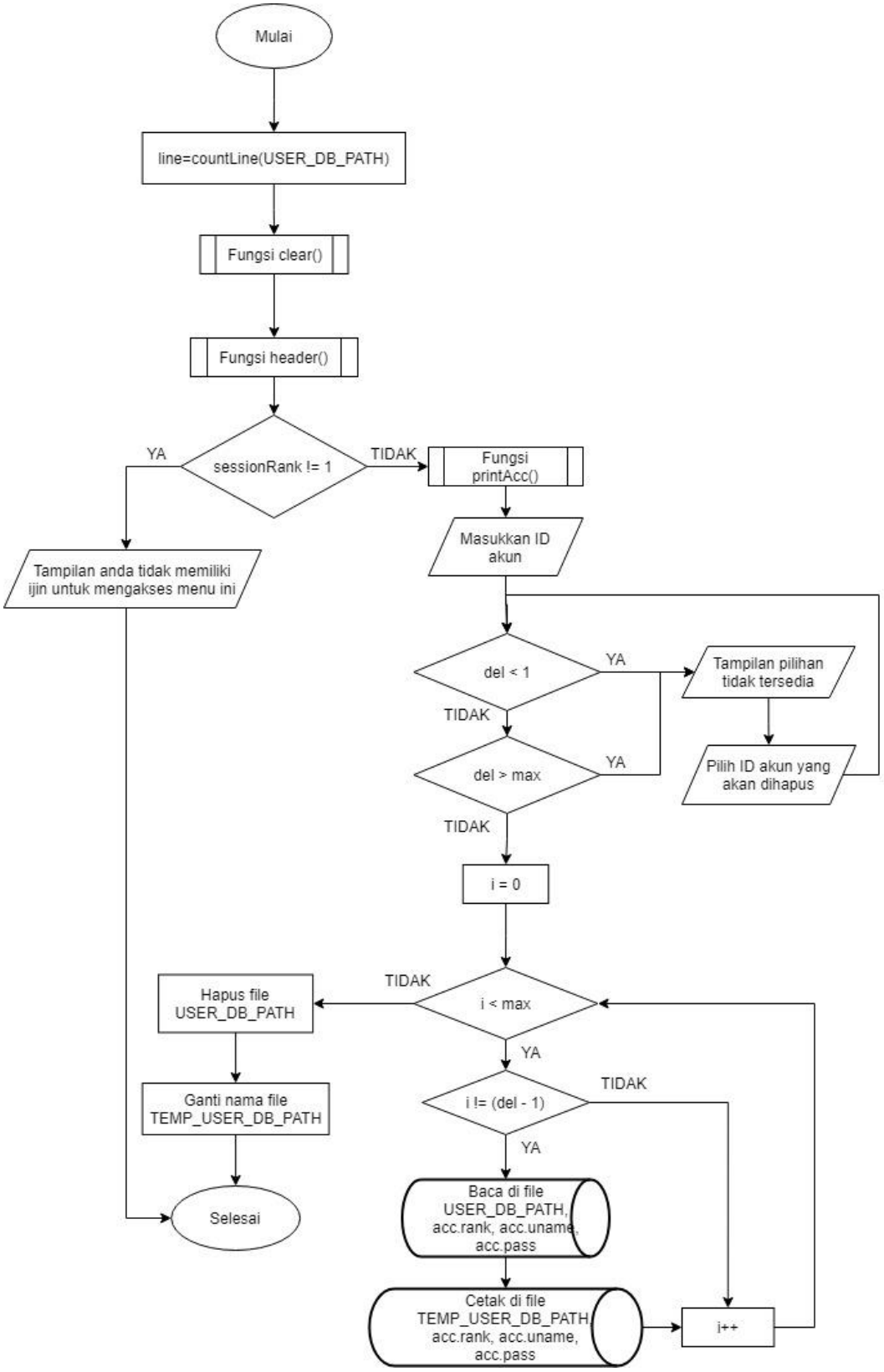
❖ **Fungsi accPassword()**



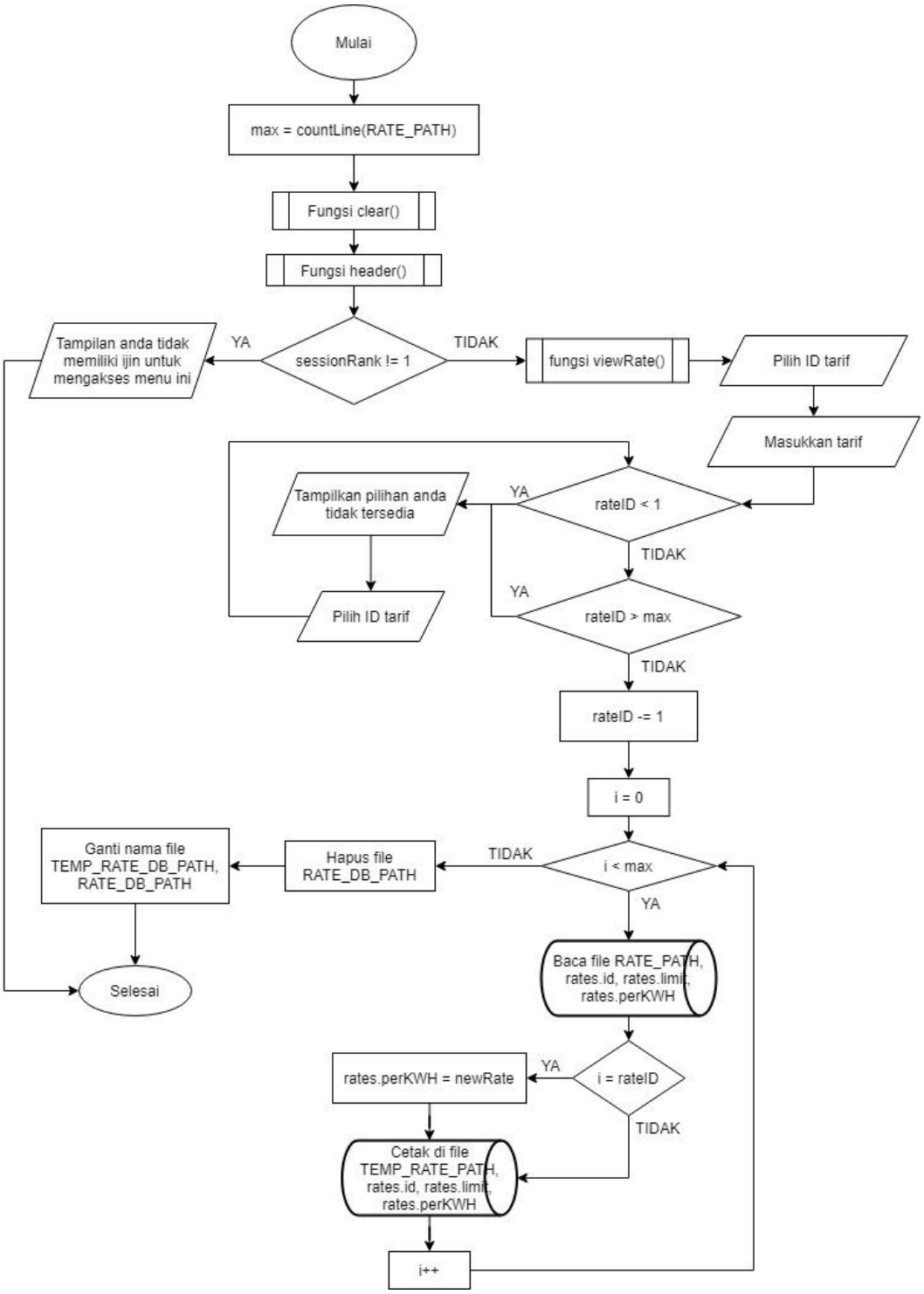
❖ **Fungsi addAcc()**



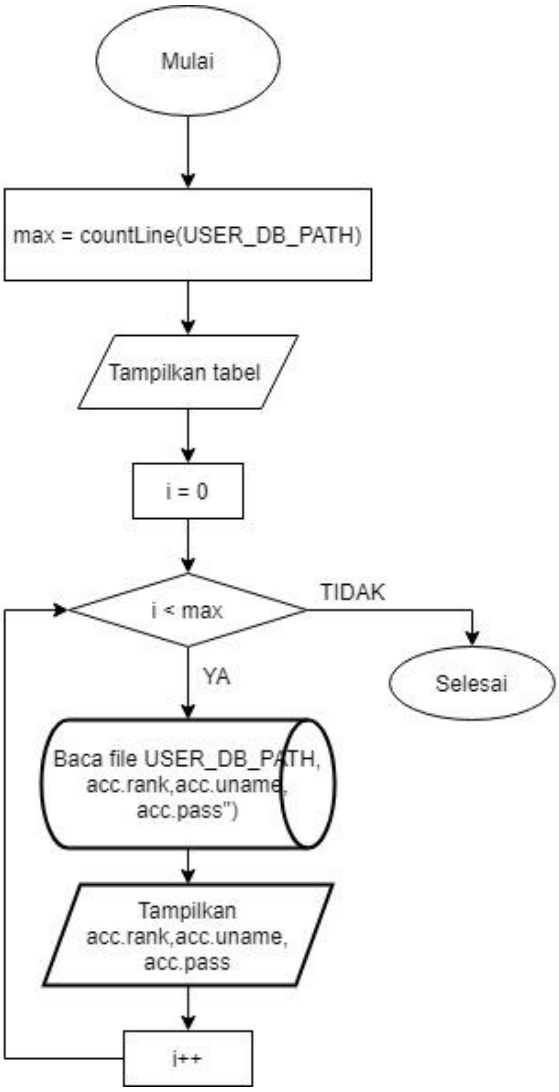
❖ **Fungsi delAcc()**



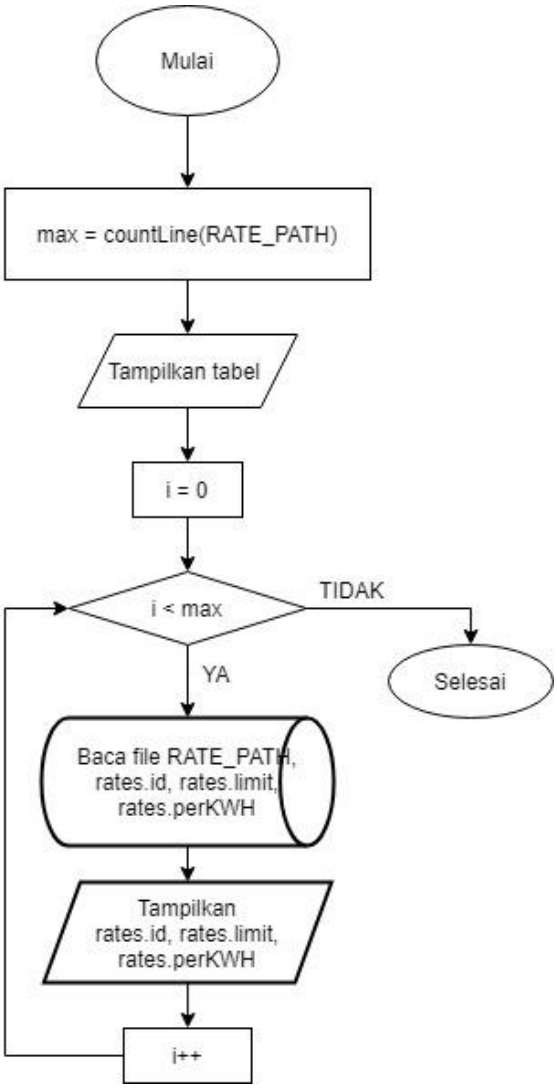
❖ **Fungsi editConfig()**



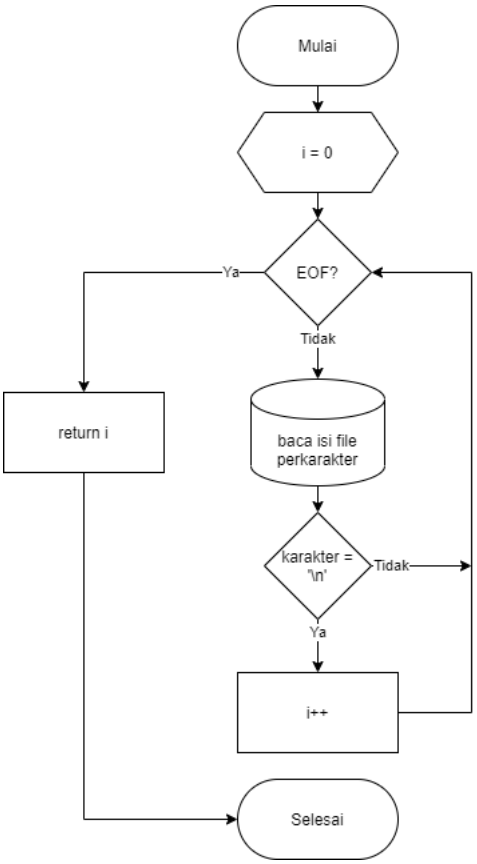
❖ **Fungsi printAcc()**



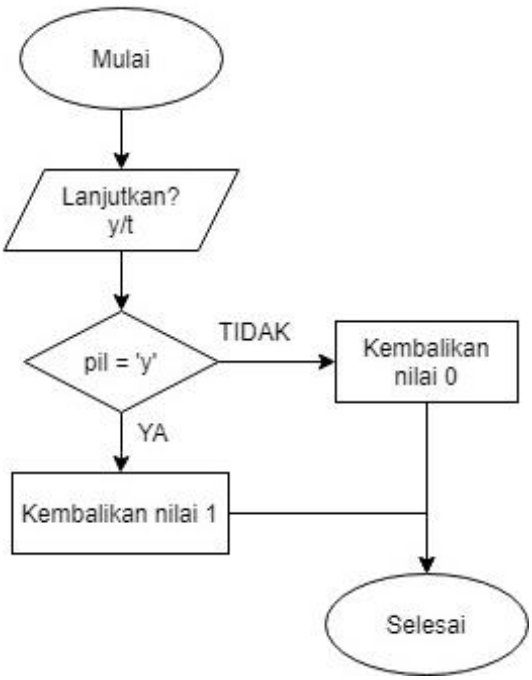
❖ **Fungsi viewRate()**



6. Fungsi countLine()



7. Fungsi pause()



3.2 Implementasi Metode Landasan Teori dalam Program

Dalam program tersebut sudah mengimplementasikan semua landasan teori yang terdapat dalam bab 2. Untuk penjelasan lebih detail sebagai berikut :

1. Variabel, Tipe Data dan Konstanta

Dalam program tersebut sudah menggunakan banyak tipe data dari integer, float, char, array, struct, pointer, dan void. Tipe data integer digunakan saat program akan menyimpan data berupa bilangan bulat sebagai contoh saat memilih menu program. Berikutnya float digunakan saat proses menghitung proses pembayaran dalam tagihan. Setelah itu struct saat ingin menyimpan data pelanggan, akun, dan tarif

Dan konstanta digunakan saat ingin membuat direktori penyimpanan data yang akan ditambahkan. Sehingga saat operasi file nanti hanya perlu memanggil konstanta tersebut saja.

2. Percabangan dan Perulangan

Percabangan sangat banyak digunakan dalam program tersebut terutama switch case dan if else. Contohnya dalam pemilihan menu-menu yang tersedia semua menggunakan switch case sebagai percabangannya. Kemudian di beberapa file lain juga digunakan if else sebagai percabangan.

Berikutnya yakni perulangan. Perulangan yang sering digunakan dalam program tersebut yakni do while dan for. Perulangan do while digunakan ketika ingin mengulang program tersebut kembali ke menu sebelumnya. Sebagai contoh saat selesai memilih salah satu menu maka otomatis kembali ke menu sebelumnya. Sedangkan perulangan for banyak digunakan ketika ingin menampilkan, mengedit, menghapus dan menambah data seperti contoh data pelanggan dan transaksi.

3. Array

Untuk array sendiri banyak digunakan ketika ingin menyimpan suatu data yang memiliki ukuran yang cukup panjang. Seperti contohnya sebuah nama, alamat, username atau password. Array yang digunakan yakni array 1 dimensi.

4. Pointer dan Fungsi Operasi String

Untuk pointer banyak digunakan ketika ingin menunjuk sebuah database penyimpanan data dalam suatu file. Jadi pointer tersebut akan menyimpan alamat dari file tersebut, sehingga nanti saat melakukan operasi file hanya perlu memanggil pointer tersebut saja.

Berikutnya fungs operasi string. Dalam program tersebut terdapat beberapa fungsi operasi string seperti strcmp(), strcpy(), strlen() dan strcat(). Sebagai contoh strcmp() digunakan dalam fungsi login yakni untuk membandingkan username dan password yang diinputkan dengan username dan password yang ada di database.

5. Operasi File

Dalam program tersebut sudah banyak fungsi operasi file yang dilakukan. Contohnya yakni dalam menu pelanggan. Di menu tersebut terdapat fitur CRUD data pelanggan yang semuanya menggunakan fungsi operasi file untuk menyimpan data data yang telah diinputkan atau akan diolah nantinya. Selain dalam menu pelanggan, menu transaksi dan setting juga menggunakan CRUD dalam programnya. Beberapa fungsi yang sering digunakan yakni fopen(), fprintf(), fscanf() dan fclose().

BAB 4

HASIL DAN IMPLEMENTASI

4.1 Isi Kode Program

Program ini terpisah menjadi beberapa file. Untuk semua file program selain PeeLen.c dimasukan ke dalam folder lib agar berjalan dengan baik. Berikut isi kode file :

1. PeeLen.c

```
#include "lib/first.h"
#include "lib/mainLogic.h"
#include "lib/acc_func.h"
#include "lib/cust_func.h"
#include "lib/tx_func.h"

void mainMenu();
void user();
void tx();
void setting();

int main() {
        int cek;

        directoryCheck();
        fileCheck();

        clear();
        header();
```

```
	if (session == -1)
		cek = login();

	if (cek)
		mainMenu();
	else
		printf("Username atau Password Salah!\n");

	return 0;
}

void mainMenu() {
	int pil;

	do {
		clear();
		header();

	printf("=================================================================
==========\n");
		printf("|                  APLIKASI PENGELOLA DATA PENGGUNA LISTRIK
|\n");

	printf("=================================================================
==========\n");
		printf("|                                                    MENU  UTAMA
|\n");

	printf("=================================================================
==========\n");
		printf(" --> 1. Pelanggan\n");
		printf(" --> 2. Transaksi\n");
		printf(" --> 3. Pengaturan\n");
		printf(" --> 4. Log Out\n");
```

```
                printf(" --> 5. Keluar\n");

        printf("==========================================================================
===========\n");
                printf(" --> Pilih menu : ");
                scanf("%d", &pil);

        printf("==========================================================================
===========\n");

                switch (pil) {
                case 1:
                        user();
                        break;
                case 2:
                        tx();
                        break;
                case 3:
                        setting();
                        break;
                case 4:
                        session = -1;
                        sessionRank = 0;
                        main();
                        return;
                case 5:
                        break;
                default:
                        break;
                }
        } while (pil != 5);
}
```

```
void user() {
	int pil, lanjut = 1;
	do {
		clear();
		header();
		printf(" --> PELANGGAN\n");

	printf("=========================================================================
==========\n");
		printf(" --> 1. Tampilkan Pelanggan\n");
		printf(" --> 2. Tambah Pelanggan\n");
		printf(" --> 3. Ubah Data Pelanggan\n");
		printf(" --> 4. Hapus Pelanggan\n");
		printf(" --> 5. Kembali\n");

	printf("=========================================================================
==========\n");
		printf(" --> Pilih menu : ");
		scanf("%d", &pil);

	printf("=========================================================================
==========\n");

		switch (pil) {
		case 1:
			viewCust();
			lanjut = pause();
			break;
		case 2:
			addCust();
			lanjut = pause();
			break;
		case 3:
```

```
                editCust();
                break;
        case 4:
                delCust();
                break;
        case 5:
                break;
        default:
                break;
        }
    } while (pil != 5 && lanjut == 1);
}


void tx() {
    int pil, lanjut = 1;
    do {
        clear();
        header();
        printf(" --> DATA TRANSAKSI\n");

    printf("=========================================================================
==========\n");
        printf(" --> 1. Tampilkan Data Penggunaan\n");
        printf(" --> 2. Tambah Penggunaan\n");
        printf(" --> 3. Pembayaran Tagihan\n");
        printf(" --> 4. Kembali\n");

    printf("=========================================================================
==========\n");
        printf(" --> Pilih menu : ");
        scanf("%d", &pil);
```

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


                switch (pil) {
                case 1:
                        viewUsage();
                        lanjut = pause();
                        break;
                case 2:
                        addUsage();
                        break;
                case 3:
                        payUsage();
                        lanjut = pause();
                        break;
                case 4:
                        break;
                default:
                        break;
                }
        } while (pil != 4 && lanjut == 1);
}


void setting() {
        int pil, lanjut;
        do {
                clear();
                header();
                printf(" --> PENGATURAN\n");


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

```
                printf(" --> 1. Ubah Password\n");
                printf(" --> 2. Buat Akun Baru\n");
                printf(" --> 3. Hapus Akun\n");
                printf(" --> 4. Ubah Tarif\n");
                printf(" --> 5. Kembali\n");

        printf("=========================================================================
===========\n");
                printf(" --> Pilih menu : ");
                scanf("%d", &pil);

        printf("=========================================================================
===========\n");

                switch (pil) {
                case 1:
                        accPassword();
                        break;
                case 2:
                        addAcc();
                        break;
                case 3:
                        delAcc();
                        lanjut = pause();
                        break;
                case 4:
                        editConfig();
                        break;
                case 5:
                        break;
                default:
                        break;
                }
```

```
        } while (pil != 5);
}
```

2. acc_func.c

```
#include "acc_func.h"


int i;
int login() {
    FILE* database;
    account acc;


    char user[20];
    char pass[20];
    int i, status = 0;


    int line = countLine(USER_DB_PATH);
    database = fopen(USER_DB_PATH, "r");


    printf("\nMasukan Username dan password");
    printf("\nUsername : ");
    scanf("\n");
    scanf("%[^\n]%*c", user);
    printf("Password : ");
    scanf("%[^\n]%*c", pass);



    for (i = 0; i < line && status == 0; i++) {
        fscanf(database, "%d,%[^,],%[^\n]", &acc.rank, acc.uname, acc.pass);
        int cekU = strcmp(acc.uname, user);
        int cekP = strcmp(acc.pass, pass);


        if (cekU == 0 && cekP == 0) {
```

```
            status = 1;
            session = i;
            sessionRank = acc.rank;
        }
    }

    fclose(database);
    return status;
}

void addAcc() {
    FILE* database;
    account acc;
    char rank;

    database = fopen(USER_DB_PATH, "a+");

    clear();
    header();

    if (sessionRank != 1) {
        printf("Anda tidak memiliki ijin untuk mengakses menu ini");
        return;
    }

    printf(" --> BUAT AKUN BARU \n");

printf("================================================================================\n");
    printf("                    Masukan Data untuk Akun Baru                    \n");

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

```
    printf(" Username : ");
    fflush(stdin);
    scanf("%[^\n]%*c", acc.uname);
    printf(" Password : ");
    fflush(stdin);
    scanf("%[^\n]%*c", acc.pass);

    while (rank != 'y' && rank != 'n') {
        printf("\nJadikan User Sebagai admin? (y/n) : ");
        fflush(stdin);
        scanf("%c", &rank);
    }

    if (rank == 'y')
        acc.rank = 1;
    else
        acc.rank = 2;

    fprintf(database, "%d,%s,%s\n", acc.rank, acc.uname, acc.pass);

    fclose(database);
}

void delAcc() {
    FILE* database, * tempFile;
    account acc;
    char rank;

    int max = countLine(USER_DB_PATH);
    int del;
```

```
    database = fopen(USER_DB_PATH, "a+");
    tempFile = fopen(TEMP_USER_DB_PATH, "w");

    clear();
    header();

    if (sessionRank != 1) {
        printf("Anda tidak memiliki ijin untuk mengakses menu ini");
        return;
    }

    printf(" --> HAPUS AKUN \n");

printf("================================================================================
====\n");
    printAcc();
    printf("\n Pilih ID Akun yang akan Dihapus : ");
    scanf("%d", &del);

    while (del < 1 && del > max) {
        printf("\n Pilihan Tidak Tersedia");
        printf("\n Pilih ID Akun yang akan Dihapus : ");
        scanf("%d", &del);
    }

    for (i = 0; i < max; i++) {
        if (i != (del - 1)) {
            fscanf(database, "%d,%[^,],%[^\n]", &acc.rank, acc.uname, acc.pass);
            fprintf(tempFile, "%d,%s,%s\n", acc.rank, acc.uname, acc.pass);
        }
    }
```

```
    fclose(database);
    fclose(tempFile);

    remove(USER_DB_PATH);
    rename(TEMP_USER_DB_PATH, USER_DB_PATH);
}

void accPassword() {
    FILE* database, * tempFile;
    account acc;
    char newUser[20];
    char newPassword[20];

    int max = countLine(USER_DB_PATH);
    int accID;

    database = fopen(USER_DB_PATH, "a+");
    tempFile = fopen(TEMP_USER_DB_PATH, "w");

    clear();
    header();

    printf(" --> UBAH PASSWORD \n");

printf("================================================================================
====\n");
    if (sessionRank != 1) {
        printf("\n Masukan Password Baru");
        printf("\n Password : ");
        scanf("\n");
        scanf("%[^\n]%*c", newPassword);
        accID = session;
```

```
} else {
    printAcc();
    printf("\n Pilih ID Akun yang akan Diganti Password : ");
    scanf("%d", &accID);

    while (accID < 1 && accID > max) {
        printf("\n Pilihan Tidak Tersedia");
        printf("\n Pilih ID Akun yang akan Dihapus : ");
        scanf("%d", &accID);
    }
    accID -= 1;

    printf("\n Masukan Password Baru");
    printf("\n Password : ");
    scanf("\n");
    scanf("%[^\n]%*c", newPassword);

}



for (i = 0; i < max; i++) {
    fscanf(database, "%d,%[^,],%[^\n]", &acc.rank, acc.uname, acc.pass);
    if (i == accID) {
        fprintf(tempFile, "%d,%s,%s\n", acc.rank, acc.uname, newPassword);
    } else {
        fprintf(tempFile, "%d,%s,%s\n", acc.rank, acc.uname, acc.pass);
    }
}

fclose(database);
fclose(tempFile);
```

```
    remove(USER_DB_PATH);
    rename(TEMP_USER_DB_PATH, USER_DB_PATH);
}

void editConfig() {
    FILE* config, * tempFile;
    rate rates;

    int max = countLine(RATE_PATH);
    int rateID, newRate;

    config = fopen(RATE_PATH, "r");
    tempFile = fopen(TEMP_RATE_PATH, "w");

    clear();
    header();

    if (sessionRank != 1) {
        printf("Anda tidak memiliki ijin untuk mengakses menu ini");
        return;
    }

    printf(" --> UBAH TARIF \n");

printf("================================================================================
====\n");
    viewRate();
    printf("\n Pilih ID Tarif yang akan Diubah : ");
    fflush(stdin);
    scanf("%d", &rateID);
    printf("\n Masukan Tarif                   : ");
```

```
    fflush(stdin);
    scanf("%d", &newRate);

printf("================================================================================
====\n");

    while (rateID < 1 && rateID > max+1) {
        printf("\n Pilihan Tidak Tersedia");
        printf(" Pilih ID Tarif yang akan Diubah : ");
        scanf("%d", &rateID);
    }
    rateID-=1;

    for (i = 0; i < max; i++) {
        fscanf(config, "%d,%d,%d", &rates.id, &rates.limit, &rates.perKWH);
        if (i == rateID) {
            rates.perKWH = newRate;
        }
        fprintf(tempFile, "%d,%d,%d\n", rates.id, rates.limit, rates.perKWH);
    }

    fclose(config);
    fclose(tempFile);

    remove(RATE_PATH);
    rename(TEMP_RATE_PATH, RATE_PATH);
}

void printAcc() {
    FILE* database;
    account acc;
```

```
    int max = countLine(USER_DB_PATH);
    database = fopen(USER_DB_PATH, "r");

    printf(" +----+--------------------+\n");
    printf(" | ID |      Username      |\n");
    printf(" +----+--------------------+\n");
    for (i = 0; i < max; i++) {
        fscanf(database, "%d,%[^,],%[^\n]", &acc.rank, acc.uname, acc.pass);
        printf(" | %-2d | %-18s |\n", i + 1, acc.uname);
    }
    printf(" +----+--------------------+\n");
    fclose(database);
}
```

3. acc_func.h

```
#ifndef ACC_FUNC_H_
#define ACC_FUNC_H_

#include "mainLogic.h"

int login();
void addAcc();
void delAcc();
void accPassword();
void printAcc();
void editConfig();

#endif
```

4. cust_func.c

```
#include "cust_func.h"

int i;

void viewCust() {

    FILE* database;

    customer cust;


    long id;

    int max = countLine(CUST_DB_PATH);


    database = fopen(CUST_DB_PATH, "r");


    clear();

    header();

    printf(" --> TAMPILKAN PELANGGAN \n");


printf("================================================================================
====\n");

    printf("                          Masukan ID Pelanggan yang Ingin Ditampilkan
\n");


printf("================================================================================
====\n");

    printf(" ID Pelanggan    : ");

    fflush(stdin);

    scanf("%ld", &id);


printf("================================================================================
====\n");

    for (i = 0; i < max; i++) {

        fscanf(database,    "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",    &cust.id,
cust.namaDepan,    cust.namaBelakang,    cust.address,    cust.telp,    &cust.usage,
&cust.price);

        if (cust.id == id) {
```

```
            printf(" ID            : %ld\n", cust.id);
            printf(" Nama Depan    : %s\n", cust.namaDepan);
            printf(" Nama Belakang : %s\n", cust.namaBelakang);
            printf(" Alamat        : %s\n", cust.address);
            printf(" No Telepon    : %s\n", cust.telp);
            printf(" Tarif         : %d\n", cust.price);
        }
    }
printf("================================================================================\n");
    fclose(database);
}

void addCust() {
    FILE* database;
    customer cust;

    int max = countLine(CUST_DB_PATH);
    database = fopen(CUST_DB_PATH, "a+");

    clear();
    header();

    if (max == 0) {
        cust.id = 1000000;
    } else {
        for (i = 0; i < max; i++)
            fscanf(database, "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n", &cust.id, cust.namaDepan, cust.namaBelakang, cust.address, cust.telp, &cust.usage, &cust.price);
        cust.id += 1;
    }
```

```
    printf(" --> TAMBAH PELANGGAN \n");

printf("================================================================================\n");
    printf("                                  Masukan Data untuk Pelanggan Baru \n");

printf("================================================================================\n");
    printf(" Nama Depan    : ");
    fflush(stdin);
    scanf("%[^\n]%*c", cust.namaDepan);
    printf(" Nama Belakang : ");
    fflush(stdin);
    scanf("%[^\n]%*c", cust.namaBelakang);
    printf(" Alamat        : ");
    fflush(stdin);
    scanf("%[^\n]%*c", cust.address);
    printf(" No Telepon    : ");
    fflush(stdin);
    scanf("%[^\n]%*c", cust.telp);

printf("================================================================================\n");
    viewRate();
    printf(" Pilih Tarif         : ");
    fflush(stdin);
    scanf("%d", &cust.price);

printf("================================================================================\n");
    printf(" Berhasil Menambahkan Pelanggan Baru dengan ID : %ld \n", cust.id);
```

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


    fprintf(database,   "%ld,%s,%s,\"%s\",%s,0.0,%d\n",   cust.id,   cust.namaDepan,
cust.namaBelakang, cust.address, cust.telp, cust.price);

    fclose(database);

}


void editCust() {

    FILE* database, * tempFile;

    customer cust, newData;


    long id;

    int pil;

    int max = countLine(CUST_DB_PATH);


    database = fopen(CUST_DB_PATH, "r");

    tempFile = fopen(TEMP_CUST_DB_PATH, "w");


    clear();

    header();

    printf(" --> UBAH DATA PELANGGAN \n");


printf("================================================================================
====\n");

    printf("                              Masukan  ID  Pelanggan  yang  Ingin  Diubah
\n");


printf("================================================================================
====\n");

    printf(" ID Pelanggan    : ");

    fflush(stdin);

    scanf("%ld", &id);
```

```
printf("================================================================================
====\n");
    printf("                    Pilih Data yang Ingin Diubah                    \n");

printf("================================================================================
====\n");
    printf(" Data :  \n");
    printf(" --> 1. Nama \n");
    printf(" --> 2. Alamat \n");
    printf(" --> 3. Nomor Telepon \n");
    printf(" --> 4. Tarif \n");
    printf(" Pilihan : ");
    fflush(stdin);
    scanf("%d", &pil);

    while (pil > 4 && pil < 1) {
        printf(" Pilihan Tidak Tersedia\n");
        printf(" Masukan Kembali : ");
        fflush(stdin);
        scanf("%d", &pil);
    }

printf("================================================================================
====\n");
    printf("                            Masukan Dat Pelanggan yang Ingin Diubah
\n");

printf("================================================================================
====\n");

    switch (pil) {
    case 1:
        printf(" Nama Depan    : ");
```

```
        fflush(stdin);
        scanf("%[^\n]%*c", newData.namaDepan);
        printf(" Nama Belakang : ");
        fflush(stdin);
        scanf("%[^\n]%*c", newData.namaBelakang);
        break;
    case 2:
        printf(" Alamat        : ");
        fflush(stdin);
        scanf("%[^\n]%*c", newData.address);
        break;
    case 3:
        printf(" No Telepon    : ");
        fflush(stdin);
        scanf("%[^\n]%*c", newData.telp);
        break;
    case 4:
        viewRate();
        printf(" Tarif         : ");
        fflush(stdin);
        scanf("%d", &newData.price);
        break;
    }

printf("================================================================================\n");

    for (i = 0; i < max; i++) {
        fscanf(database,   "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",   &cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   cust.telp,   &cust.usage,
&cust.price);
```

```
        if (cust.id == id) {
            switch (pil) {
            case 1:
                fprintf(tempFile,      "%ld,%s,%s,\"%s\",%s,%f,%d\n",      cust.id,
newData.namaDepan,  newData.namaBelakang,  cust.address,  cust.telp,  cust.usage,
cust.price);
                break;
            case 2:
                fprintf(tempFile,      "%ld,%s,%s,\"%s\",%s,%f,%d\n",      cust.id,
cust.namaDepan,   cust.namaBelakang,   newData.address,   cust.telp,   cust.usage,
cust.price);
                break;
            case 3:
                fprintf(tempFile,      "%ld,%s,%s,\"%s\",%s,%f,%d\n",      cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   newData.telp,   cust.usage,
cust.price);
                break;
            case 4:
                fprintf(tempFile,      "%ld,%s,%s,\"%s\",%s,%f,%d\n",      cust.id,
cust.namaDepan,    cust.namaBelakang,    cust.address,    cust.telp,    cust.usage,
newData.price);
                break;
            }
        } else {
            fprintf(tempFile,        "%ld,%s,%s,\"%s\",%s,%f,%d\n",        cust.id,
cust.namaDepan,    cust.namaBelakang,    cust.address,    cust.telp,    cust.usage,
cust.price);
        }
    }

    fclose(database);
    fclose(tempFile);

    remove(CUST_DB_PATH);
    rename(TEMP_CUST_DB_PATH, CUST_DB_PATH);
```

```
}

void delCust() {
    FILE* database, * tempFile;
    customer cust;

    long id;
    int max = countLine(CUST_DB_PATH);

    database = fopen(CUST_DB_PATH, "r");
    tempFile = fopen(TEMP_CUST_DB_PATH, "w");

    clear();
    header();
    printf(" --> HAPUS PELANGGAN \n");

printf("================================================================================
====\n");
    printf("                              Masukan ID Pelanggan yang Ingin Dihapus
\n");

printf("================================================================================
====\n");
    printf(" ID Pelanggan    : ");
    fflush(stdin);
    scanf("%ld", &id);

printf("================================================================================
====\n");

    for (i = 0; i < max; i++) {
        fscanf(database,   "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",   &cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   cust.telp,   &cust.usage,
&cust.price);
        if (cust.id != id)
```

```
            fprintf(tempFile,        "%ld,%s,%s,\"%s\",%s,%f,%d\n",        cust.id,
cust.namaDepan,    cust.namaBelakang,    cust.address,    cust.telp,    cust.usage,
cust.price);

    }


    fclose(database);

    fclose(tempFile);


    remove(CUST_DB_PATH);

    rename(TEMP_CUST_DB_PATH, CUST_DB_PATH);

}
```

5. cust_func.h

```
#ifndef CUST_FUNC_H_

#define CUST_FUNC_H_


#include "mainLogic.h"


void viewCust();

//void printCust();

void addCust();

void editCust();

void delCust();

void viewRate();


#endif
```

6. first.c

```
#include "first.h"


void directoryCheck() {
```

```
    struct stat st = { 0 };

    if (stat("db_cust", &st) == -1) {
        mkdir("db_cust");
    }

    if (stat("db_user", &st) == -1) {
        mkdir("db_user");
    }

    if (stat("config", &st) == -1) {
        mkdir("config");
    }

    if (stat("tx_log", &st) == -1) {
        mkdir("tx_log");
    }
}

void fileCheck() {
    FILE* userDB, *custDB, *rate, *log;

    userDB = fopen(USER_DB_PATH, "a+");
    custDB = fopen(CUST_DB_PATH, "a+");
    rate = fopen(RATE_PATH, "a+");
    log = fopen(ALL_LOG_PATH, "a+");

    int baris = countLine(USER_DB_PATH);
    if (baris == 0)
        fprintf(userDB, "1,admin,admin\n");
```

```
    baris = countLine(RATE_PATH);
    if (baris == 0){
        fprintf(rate, "1,450,200\n");
        fprintf(rate, "2,900,1400\n");
        fprintf(rate, "3,1300,1600\n");
        fprintf(rate, "4,2200,1900\n");
        fprintf(rate, "5,3100,2300\n");
    }

    baris = countLine(ALL_LOG_PATH);
    if (baris == 0) {
        fprintf(log,   "No,Waktu,ID,Jenis   Transaksi,Jumlah   Transaksi,KWH,Total
KWH\n");
    }

    fclose(userDB);
    fclose(custDB);
    fclose(rate);
    fclose(log);
}
```

7. first.h

```
#ifndef FIRST_H_
#define FIRST_H_

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include "mainLogic.h"

void directoryCheck();
```

```
void fileCheck();


#endif
```

8. mainLogic.c

```
#include "mainLogic.h"


int i;

int session = -1;

int sessionRank = 0;


void header() {

	time_t mentah;

	struct tm* waktu;

	time(&mentah);

	waktu = localtime(&mentah);


	printf("\n=========================================================================
=============\n");

	printf("|                                                                    |\n");

	printf("|   /$$$$$$$  /$$$$$$$$        /$$$$$$$$ /$$              /$$$$$$$$
/$$   /$$  |\n");

	printf("|  | $$__  $$| $$_____/       | $$_____/| $$             | $$_____/|
$$$ | $$  |\n");

	printf("|  | $$    $$| $$             | $$       | $$             | $$       |
$$$$| $$  |\n");

	printf("|  | $$$$$$$/| $$$$$          | $$$$$    | $$             | $$$$$    |
$$ $$ $$  |\n");

	printf("|  | $$____/ | $$__/          | $$__/    | $$             | $$__/    |
$$  $$$$  |\n");

	printf("|  | $$      | $$             | $$       | $$             | $$       |
$$   $$$  |\n");

	printf("|  | $$      | $$$$$$$$       | $$$$$$$$| $$$$$$$$       | $$$$$$$$|
$$    $$  |\n");
```

```
        printf("|  |__/      |________/      |________/|________/      |________/|__/
__/  |\n");

        printf("|                                                                    |\n");

        printf("|                                        %02d/%02d/%d %02d:%02d:%02d
|\n", waktu->tm_mday, waktu->tm_mon + 1, waktu->tm_year + 1900, waktu->tm_hour,
waktu->tm_min, waktu->tm_sec);

        printf("=====================================================================
==========\n");

}


void viewRate() {

    FILE* database;

    rate rates;


    database = fopen(RATE_PATH, "r");

    int max = countLine(RATE_PATH);


    printf(" +----+-----------+---------------+\n");

    printf(" | ID | Batas VA  | Tarif Per-KWH |\n");

    printf(" +----+-----------+---------------+\n");

    for (i = 0; i < max; i++) {

        fscanf(database, "%d,%d,%d", &rates.id, &rates.limit, &rates.perKWH);

        printf("  |  %-2d  |  %-8d    |    Rp.%-8d    |\n", rates.id, rates.limit,
rates.perKWH);

    }

    printf(" +----+-----------+---------------+\n");

    fclose(database);

}


int countLine(char file[]){

        FILE* cek;

    int i = 0;

    int cs;
```

```
    cek = fopen(file, "r");
    while (!feof(cek)) {               //Loop hingga EOF  1
        cs = fgetc(cek);               //simpan stream char ke c
        if (cs == '\n') i++;          //Jika dideteksi \n , tambah i
    }
    fclose(cek);
    return i;
}

int pause() {
    char pil;
    printf("\nLanjutkan? (y/t) : ");
    fflush(stdin);
    scanf("%c", &pil);

    if (pil == 'y') return 1;
    else return 0;
}

void clear() {
#ifdef _WIN32
    std : system("cls");
#else
std: system("clear");
#endif
}
```

9. mainLogic.h

```
#ifndef MAINLOGIC_H_

#define MAINLOGIC_H_


#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <time.h>


#define USER_DB_PATH "./db_user/name_pass.csv"

#define TEMP_USER_DB_PATH "./db_user/temp.csv"

#define CUST_DB_PATH "./db_cust/data_cust.csv"

#define TEMP_CUST_DB_PATH "./db_cust/temp.csv"

#define RATE_PATH "./config/config.csv"

#define TEMP_RATE_PATH "./config/temp.csv"

#define ALL_LOG_PATH "./tx_log/tx_log.csv"


struct tm* waktu;


typedef struct customer {

    long id;

    char namaDepan[20];

    char namaBelakang[20];

    char address[70];

    char telp[15];

    float usage;

    int price;

} customer;


typedef struct account {

    int rank;
```

```
    char uname[20];
    char pass[20];
}account;

typedef struct rate {
    int id;
    int limit;
    int perKWH;
} rate;

void header();
void viewRate();
int pause();
int countLine(char file[]);
void clear();

extern int session;
extern int sessionRank;

#endif
```

10. tx_func.c

```
#include "tx_func.h"
int i;
void viewUsage() {
    FILE* database, * config;
    customer cust;
    rate rates;

    long id;
    int max = countLine(CUST_DB_PATH);
```

```
    int configMax = countLine(RATE_PATH);


    config = fopen(RATE_PATH, "r");
    database = fopen(CUST_DB_PATH, "r");


    clear();
    header();
    printf(" --> TAMPILKAN DATA PENGGUNAAN \n");

printf("================================================================================
====\n");
    printf("                        Masukan ID Pelanggan                          \n");

printf("================================================================================
====\n");
    printf(" ID Pelanggan    : ");
    fflush(stdin);
    scanf("%ld", &id);

printf("================================================================================
====\n");


    for (i = 0; i < max; i++) {
        fscanf(database,   "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",   &cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   cust.telp,   &cust.usage,
&cust.price);
        if (cust.id == id)
            break;
    }


    for (i = 0; i < configMax; i++) {
        fscanf(config, "%d,%d,%d", &rates.id, &rates.limit, &rates.perKWH);
        if (rates.id == cust.price)
            break;
```

```
    }

    if (cust.id == id) {
        printf(" ID          : %ld\n", cust.id);
        printf(" Nama        : %s %s\n", cust.namaDepan, cust.namaBelakang);
        printf(" Tarif       : %d\n", cust.price);
        printf(" Batas Daya  : %d VA\n", rates.limit);
        printf(" Penggunaan  : %.4f KWH\n", cust.usage);
        printf(" Tagihan     : Rp.%.2f\n", cust.usage * rates.perKWH);
    }

printf("================================================================================\n");

    fclose(config);
    fclose(database);
}

void addUsage() {
    FILE* database, * tempFile;
    customer cust;

    long id;
    float add;
    int max = countLine(CUST_DB_PATH);

    database = fopen(CUST_DB_PATH, "r");
    tempFile = fopen(TEMP_CUST_DB_PATH, "a");

    clear();
    header();
    printf(" --> TAMBAH PENGGUNAAN \n");
```

```
printf("================================================================================
====\n");
    printf("                        Masukan ID Pelanggan                        \n");

printf("================================================================================
====\n");
    printf(" ID Pelanggan    : ");
    fflush(stdin);
    scanf("%ld", &id);

printf("================================================================================
====\n");
    printf(" Penggunaan      : ");
    fflush(stdin);
    scanf("%f", &add);

printf("================================================================================
====\n");

    for (i = 0; i < max; i++) {
        fscanf(database,   "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",   &cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   cust.telp,   &cust.usage,
&cust.price);
        if (cust.id == id) {
            cust.usage += add;
            logTX(cust.id, 0, -1, add, cust.usage);
        }
        fprintf(tempFile,  "%ld,%s,%s,\"%s\",%s,%f,%d\n",  cust.id,  cust.namaDepan,
cust.namaBelakang, cust.address, cust.telp, cust.usage, cust.price);
    }

    fclose(database);
    fclose(tempFile);
```

```
    remove(CUST_DB_PATH);
    rename(TEMP_CUST_DB_PATH, CUST_DB_PATH);
}

void payUsage() {
    FILE* database, * tempFile, * config;
    customer cust;
    rate rates;

    long id;
    int pay;
    int max = countLine(CUST_DB_PATH);
    int configMax = countLine(RATE_PATH);

    config = fopen(RATE_PATH, "r");
    database = fopen(CUST_DB_PATH, "r");
    tempFile = fopen(TEMP_CUST_DB_PATH, "a");

    clear();
    header();
    printf(" --> BAYAR TAGIHAN \n");

printf("================================================================================\n");
    printf("                        Masukan ID Pelanggan                        \n");

printf("================================================================================\n");
    printf(" ID Pelanggan      : ");
    fflush(stdin);
    scanf("%ld", &id);
```

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

    printf(" Jumlah Pembayaran : ");

    fflush(stdin);

    scanf("%d", &pay);


printf("================================================================================
====\n");


    for (i = 0; i < max; i++) {

        fscanf(database,   "%ld,%[^,],%[^,],\"%[^\"]\",%[^,],%f,%d\n",   &cust.id,
cust.namaDepan,   cust.namaBelakang,   cust.address,   cust.telp,   &cust.usage,
&cust.price);

        if (cust.id == id) {

            for (i = 0; i < configMax; i++) {

                fscanf(config, "%d,%d,%d", &rates.id, &rates.limit, &rates.perKWH);

                if (rates.id == cust.price)

                    break;

            }

            float usageKWH = ((float)pay / (float)rates.perKWH);

            cust.usage -= usageKWH;

            logTX(cust.id, 1, pay, usageKWH, cust.usage);

        }

        fprintf(tempFile,  "%ld,%s,%s,\"%s\",%s,%f,%d\n",  cust.id,  cust.namaDepan,
cust.namaBelakang, cust.address, cust.telp, cust.usage, cust.price);

    }


    fclose(config);

    fclose(database);

    fclose(tempFile);


    remove(CUST_DB_PATH);

    rename(TEMP_CUST_DB_PATH, CUST_DB_PATH);
```

```
}

void logTX(long id, int type, int idr,float txKWH, float kwh) {
    FILE* logAcc, * log;

    time_t mentah;
    struct tm* waktu;
    time(&mentah);
    waktu = localtime(&mentah);

    char convert[8];
    char* format = ".csv";
    char* dir = "./tx_log/";
    snprintf(convert, 8, "%ld", id);

    int mem = strlen(convert) + strlen(format) + strlen(dir);
    char file[mem];

    strcpy(file, dir);
    strcat(file, convert);
    strcat(file, format);

    log = fopen(ALL_LOG_PATH, "a");
    logAcc = fopen(file, "a");

    int maxLog = countLine(ALL_LOG_PATH);
    int maxLogAcc = countLine(file);

    if (maxLogAcc == 0) {
        fprintf(logAcc,   "No,Waktu,Jenis   Transaksi,Jumlah   Transaksi,KWH,Total
KWH\n");
        maxLogAcc = countLine(file);
```

```
    }


    switch (type) {
    case 0:
        fprintf(log,    "%d,%02d:%02d:%02d-%02d/%02d/%d,%ld,TAMBAH    PENGGUNAAN,-
,%.4f,%.4f\n",  maxLog,  waktu->tm_hour,  waktu->tm_min,  waktu->tm_sec,  waktu-
>tm_mday, waktu->tm_mon + 1, waktu->tm_year + 1900, id, txKWH, kwh);
        fprintf(logAcc,     "%d,%02d:%02d:%02d-%02d/%02d/%d,TAMBAH     PENGGUNAAN,-
,%.4f,%.4f\n",  maxLog,  waktu->tm_hour,  waktu->tm_min,  waktu->tm_sec,  waktu-
>tm_mday, waktu->tm_mon + 1, waktu->tm_year + 1900, txKWH, kwh);
        break;
    case 1:
        fprintf(log,                "%d,%02d:%02d:%02d-%02d/%02d/%d,%ld,PEMBAYARAN
TAGIHAN,%d,%.4f,%.4f\n",  maxLog,  waktu->tm_hour,  waktu->tm_min,  waktu->tm_sec,
waktu->tm_mday, waktu->tm_mon + 1, waktu->tm_year + 1900, id, idr, txKWH, kwh);
        fprintf(logAcc,                 "%d,%02d:%02d:%02d-%02d/%02d/%d,PEMBAYARAN
TAGIHAN,%d,%.4f,%.4f\n",  maxLog,  waktu->tm_hour,  waktu->tm_min,  waktu->tm_sec,
waktu->tm_mday, waktu->tm_mon + 1, waktu->tm_year + 1900, idr, txKWH, kwh);
        break;
    }


    fclose(log);
    fclose(logAcc);
}
```

11. tx_func.h

```
#ifndef TX_FUNC_H_
#define TX_FUNC_H_


#include "mainLogic.h"


void viewUsage();
void addUsage();
void payUsage();
```

```
void logTX(long id, int type, int idr, float kwh, float txKWH);


#endif
```

4.2 Penjelasan Kode Program

Pertama saat program di run maka akan menjalankan fungsi main() kemudian memanggil fungsi directoryCheck() dan fileCheck(). Fungsi directoryCheck() digunakan untuk mengecek apakah direktori untuk menyimpan file yang akan diinput nanti sudah ada atau tidak. Jika tidak atau sama dengan -1, maka buat direktori atau folder tersebut. Untuk membuatnya digunakan perintah mkdir(). Kemudian fileCheck() digunakan untuk mengecek apakah data dalam file yang diperlukan untuk proses nanti seperti login dan tarif sudah ada atau belum. Untuk mengeceknya disini dengan menghitung baris yang ada didalam file. Jika baris = 0 maka artinya file itu kosong. Jika kosong maka isi dengan data default dari kode. Untuk menghitung barisnya digunakan fungsi countLine(). Fungsi countLine() memanfaatkan perulangan while(!feof) dengan increment.

Setelah itu masuk ke login. User menginputkan username dan password kemudian sistem mengeceknya dengan memanggil fungsi login. Di fungsi login ini username dan password inputan dibandingkan dengan username dan password dari database dengan fungs string strcmp(). Jika sama maka dibawa ke fungsi mainMenu(). Jika salah maka ada tampilan username dan password salah.

Berikutnya di fungsi mainMenu() ini terdapat 5 pilihan menu yakni Pelanggan, Transaksi, Pengaturan, Log Out dan Keluar. Kemudian user menginputkan menu yang diinginkan disimpan dalam variable pil. Kemudian dengan switch case menu dipilih sesuai dengan inputan user. Masing-masing menu nantinya akan terdapat menu lagi didalamnya kecuali keluar dan log out.

1. Pelanggan

Pada menu pelanggan, terdapat beberapa menu CRUD data pelanggan. Masing-masing menu memiliki fungsinya tersendiri. Yang pertama menampilkan, saat ingin menampilkan user menginputkan ID pelanggan yang diinginkan

kemudian jika ada maka akan ditampilkan seluruh data pelanggan tersebut. Untuk menampilkannya digunakan operasi file mode read kemudian dengan perulangan for dengan batas jumlah baris dalam file data tersebut dibaca. Jika sama dengan ID inputan user maka data ditampilkan.

Berikutnya menambah data, untuk menambah pertama dengan mengatur ID pelanggan. Jika data tersebut merupakan data yang diinputkan pertama kali maka ID diatur 1000000. Dan jika bukan data pertama maka ID diatur 1000000 + urutan ID berikutnya. Setelah mengatur ID maka tinggal menginputkan data kemudian disimpan didalam file dengan mode append.

Berikutnya yakni mengedit data, untuk mengedit data pertama user menginputkan ID pelanggannya kemudian pilih data apa yang ingin diubah. Kemudian dengan perulangan for cek apakah didalam file ada yang ID nya sama, jika ada maka pindahkan semua data dengan data baru ke file sementara. Kemudian hapus file utama dan rename file sementara dengan nama file utama. Untuk mengedit digunakan mode read dan write.

Terakhir yakni menghapus data. Untuk menghapus data user menginputkan ID Pelanggan yang ingin dihapus kemudian sama dengan mengedit yakni semua data yang ID nya berbeda dipindahkan ke dalam file sementara. Kemudian file utama dihapus dan file sementara di rename dengan nama file utama.

2. Transaksi

Dalam menu transaksi juga memiliki beberapa menu lagi didalamya. Menu-menu tersebut masing-masing terdapat fungsi CRUD. Setelah user menginputkan pilihannya terdapat percabangan switch case didalamnya. Menu-menu yang terdapat didalam transaksi yakni menampilkan data penggunaan, menambah penggunaan dan pembayaran tagihan.

Pertama yakni menampilkan data penggunaan. Data penggunaan disini merupakan data pelanggan yang sudah terdapat jumlah penggunaannya beserta total tagihan pembayarannya. Untuk menampilkannya pertama buka file penyimpanan dengan operasi file fopen() mode read. Kemudian input ID Pelanggan yang ingin

dilihat tagihannya. Berikutnya dengan perulangan for cek apakah ID Pelanggan yang diinput ada didalam file penyimpanan. Jika ada maka break dan berikutnya dengan perulangan for lagi cek apakah ID tarif per KWH milik pelanggan ada didalam data penyimpanan data tarif, jika ada maka break. Kemudian untuk menghitung total tagihan tinggal mengalikan penggunaan dengan tarif per KWH nya.

Berikutnya yaitu menambah data tagihan. Untuk membuat data tagihan user perlu menginputkan ID dan jumlah penggunaan milik pelanggan. Setelah itu cek dengan perulangan for apakah ID pelanggan yang diinputkan ada didalam file data pelanggan. Jika ada maka ubah usage milik pengguna dengan menambahkannya dengan jumlah penggunaannya. Kemudian panggil fungsi logTX() untuk mencetak record penambahan tersebut kedalam file log. Berikutnya yakni mencetak semua data ke dalam file sementara agar data baru berubah. Setelah itu hapus file utamanya kemudian rename file sementara dengan nama file utama.

Terakhir yaitu menu pembayaran tagihan. Untuk membayar tagihan user perlu menginputkan ID pelanggan dan juga jumlah uang pembayarannya. Berikutnya dengan perulangan for dicari apakah ID ditemukan atau tidak didalam file. Jika ada maka lakukan perulangan for lagi untuk mecari apakah ID tarif milik pelanggan ada didalam data tarif. Jika ada maka break, setelah itu hitung jumlah KWH yang dibayar dengan nominal pembayaran. Untuk menghitung KWH yang dibayar bisa dengan membagi jumlah pembayaran dengan tarif per kwhnya. Sehingga didapat total KWH yang dibayar. Hasil tersebut disimpan dan total penggunaan KWH pelanggan dikurangi dengan total KWH yang sudah dibayar. Kemudian panggil fungsi logTX() untuk mencetak pembayaran tersebut didalam file history.

Dalam fungsi logTX() bertujuan untuk mencetak proses yang terjadi didalam transaksi kedalam file. File dicetak dipisah sesuai dengan ID Pelanggan agar memudahkan pengecekan. Namun ada 1 file yang menyimpan seluruh datanya juga. Didalam file tersebut disimpan nomor, waktu pembayaran, jenis transaksi, jumlah transaksi, KWH dan Total KWH. Pertama untuk mengambil waktu deklarasikan time_t dengan variable mentah untuk menyimpan data waktu saat iTU

lalu simpan didalam struct waktu. Untuk penamaan file sesuai dengan ID pelanggan pertama atur tipe file dan direktorinya kemudian simpan dalam variable masing-masing. Setelah itu ubah namanya menjadi ID Pelanggan kemudian hitung panjang variable tipe file dan direktori ditambah dengan ID dengan fungsi string strlen(). Hasilnya disimpan sebagai panjang char file.

Kemudian variable file itu diberi nama sesuai dengan ID nya. Untuk menamainya dimanfaatkan fungsi strcpy() dan strcat() untuk menduplikat dan menambah nama file tersebut. Kemudian jika sudah selesai penamaan tinggal menambah data yang sudah diberikan sebelumnya sesuai dengan jenisnya. Jika type yang diterima 0 maka merupakan proses penambahan penggunaan atau tagihan dan jika 1 maka proses pembayaran.

Diakhir menu tagihan terdapat fungsi pause() untuk menentukan apakah ingin lanjut ke menu sebelumnya atau selesai. Didalam fungsi pause() user ditanya ingin lanjut atau tidak. Jika iya ketik y dan tidak n. Maka fungsi akan mereturn nilai 1 jika iya dan 0 jika tidak.

3. Pengaturan

Dalam menu pengaturan ini berfungsi mengatur data yang berkaitan dengan penggunaan program seperti admin dan data tarif. Beberapa menunya yakni edit password akun, buat akun baru, hapus akun dan ubah harga tarif. Digunakan percabangan switc case dalam pemilihan menu.

Pertama yakni menu edit password. Dalam menu ini pemilik akun dapat mengubah password akun miliknya. Jika akun milik admin maka dapat mengubah semua password akun lainnya dan jika akun milik karyawan hanya bisa mengubah password miliknya saja. Untuk membedakan akun admin dan karyawan hanya perlu melihat rank akun tersebut. Jika 1 maka admin dan jika bukan 1 maka karyawan. Jika admin maka proses pertama yakni memilih ID akun yang akan diubah passwordnya. Setelah itu masukan password barunya kemudian dengan perulangan for cek jika ID akun yang diinput sama dengan ID dalam file maka ubah passwordnya dan jika tidak maka biarkan. Kemudian cetak semua data termasuk

data akun berpassword baru tersebut ke file sementara. Hapus file utama dan rename file sementara dengan nama file utama.

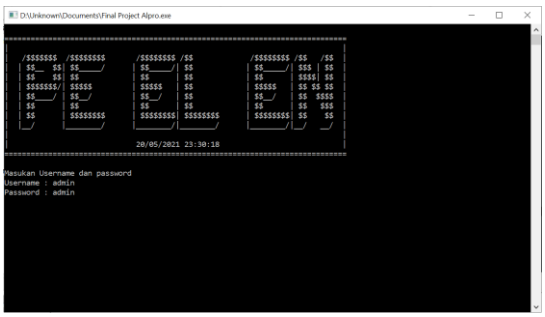
Kedua yakni menambah akun. Menu tambah akun hanya dapat diakses oleh admin dan jika karyawan mencoba akses maka akan langsung kembalikan. Untuk menambah akun baru admin harus menginputkan username dan passwordnya terlebih dahulu kemudian putuskan apakah akun tersebut akan dijadikan akun admin atau akun karyawan saja. Jika admin ketik y dan jika karyawan ketik n. Kemudian tinggal mencetak data baru tersebut kedalam file penyimpanan.

Ketiga yakni hapus akun. Untuk menghapus hanya bisa dilakukan oleh admin. Pertama input ID Akun yang akan dihapus. Kemudian dengan perulangan for cetak semua data yang ID nya tidak sama dengan yang diinputkan di file utama akun ke file sementara. Kemudian hapus file utama dan renam file sementara dengan nama file utama.

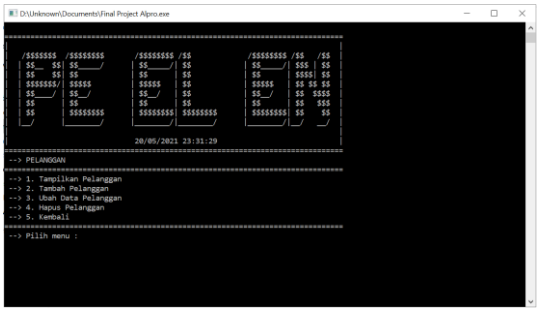
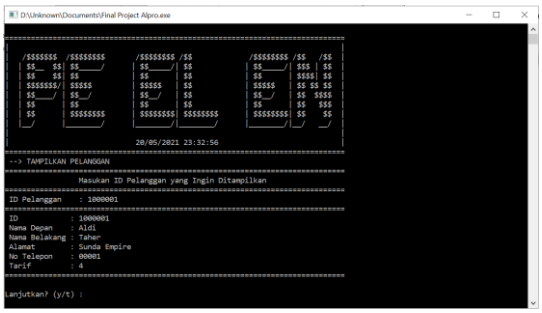
Keempat yakni ubah harga tarif. Untuk mengubah harga tarif hanya bisa dilakukan oleh admin saja. Pertama input ID tarif yang akan diubah dan masukan harga tarifnya yang baru. Kemudian dengan perulangan for cek apakah ID tarif yang diinputkan sama dengan ID tarif dalam file penyimpanan. Jika sama ubah harganya. Kemudian cetak semua data termasuk data baru kedalam file sementara setelah itu hapus file utama dan rename file sementara dengan nama file utama.

4.3 Screenshot Hasil Run Program

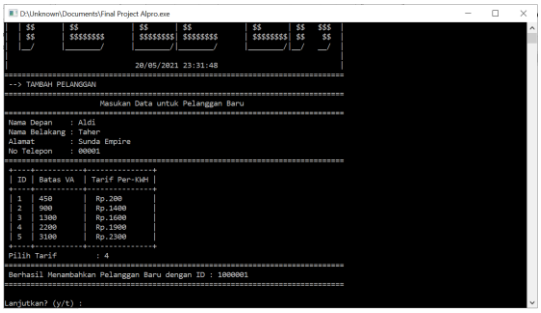
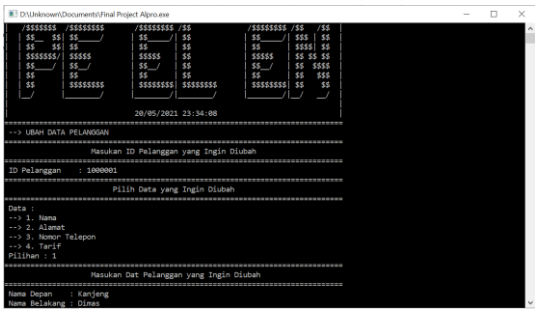
- **Login dan Menu Utama**



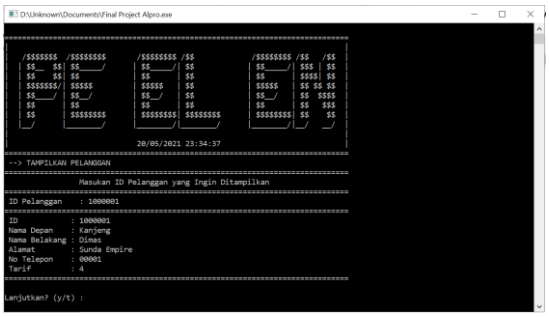
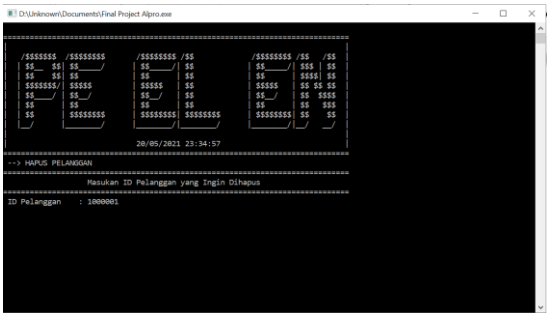
- **Menu Pelanggan dan Tampilan Data Pelanggan**

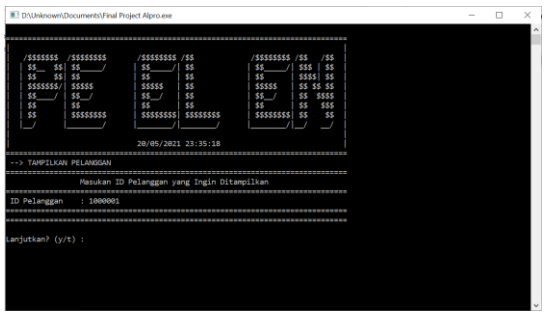
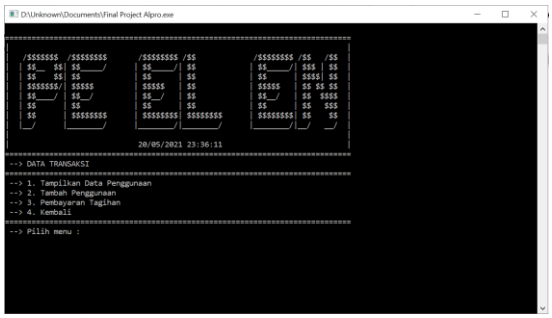
- **Tambah Pelanggan dan Edit Data Pelanggan**

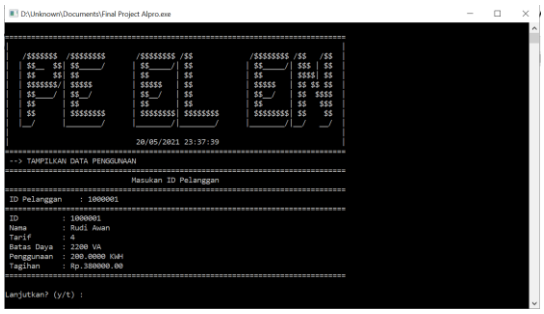
- **Tampilan Sesudah Edit dan Hapus Pelanggan**

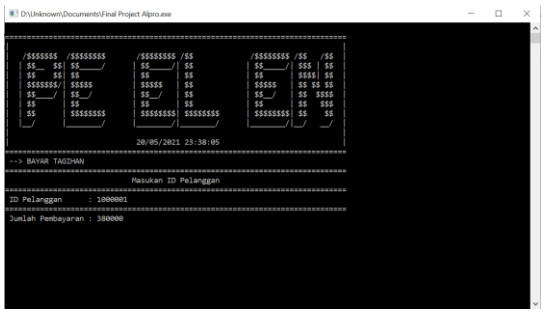
- **Tampilan Setelah Hapus dan Menu Transaksi**

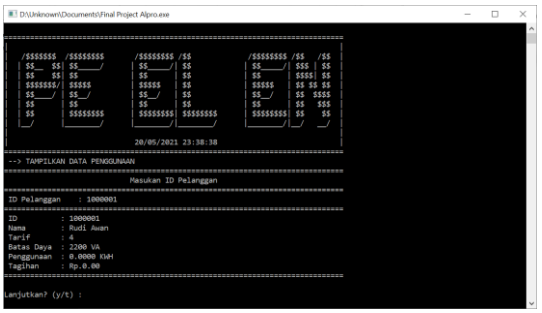



- **Tambah Tagihan dan Tampilan Tagihan**

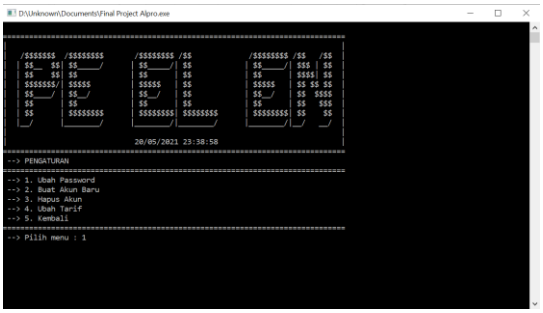


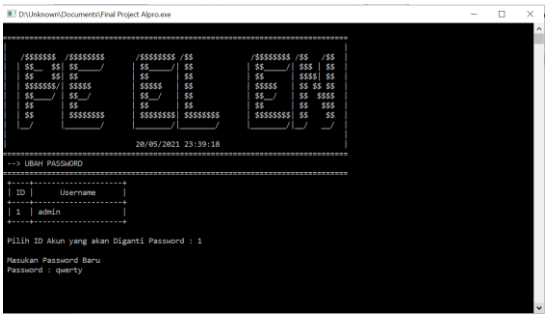
- **Pembayaran Tagihan dan Tampilan Setelah Bayar**



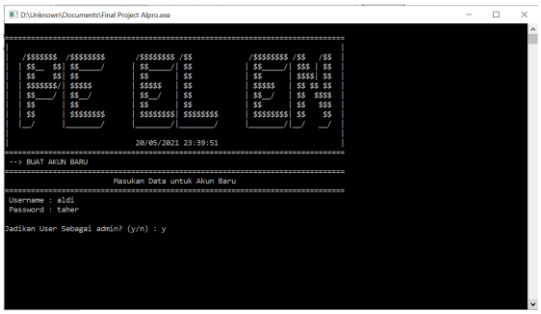


- **Menu Pengaturan dan Ubah Password**

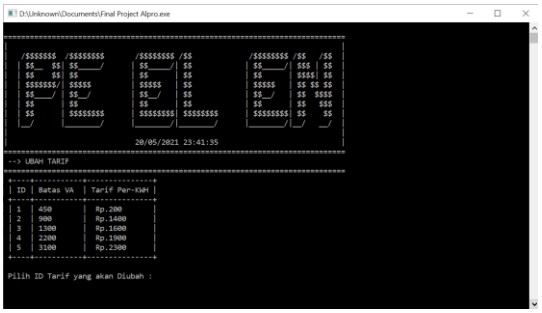


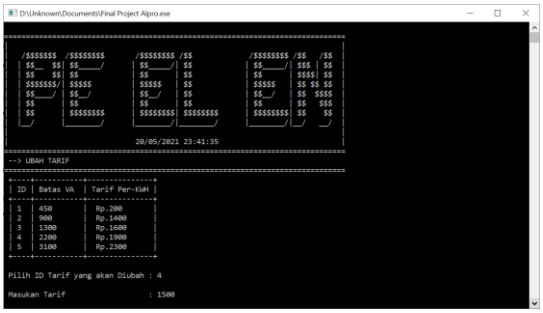


- **Buat Akun dan Hapus Akun**



- **Ubah Tarif**


D:\Unknown\Documents\Final Project Alpro.exe
20/05/2021 23:41:35
--> UBAH TARIF
ID	Batas VA	Tarif Per-KWH
1	450	Rp.200
2	900	Rp.1400
3	1300	Rp.1600
4	2200	Rp.1900
5	3100	Rp.2300
Pilih ID Tarif yang akan Diubah :


D:\Unknown\Documents\Final Project Alpro.exe
20/05/2021 23:41:35
--> UBAH TARIF
ID	Batas VA	Tarif Per-KWH
1	450	Rp.200
2	900	Rp.1400
3	1300	Rp.1600
4	2200	Rp.1900
5	3100	Rp.2300
Pilih ID Tarif yang akan Diubah : 4
Masukan Tarif : 1500

BAB 5

PENUTUP

5.1 Kesimpulan

Dalam suatu perusahaan yang memiliki banyak data yang perlu diolah pasti memerlukan sebuah aplikasi atau program yang dapat membantu mereka mengolah data tersebut. Program tersebut harus dapat mengolah data dalam jumlah besar dan juga memiliki algoritma yang sesuai dengan sistem perusahaan tersebut.

Sebagai contoh program PE EL EN ini. Program PE EL EN merupakan program sederhana dalam bahasa C yang dapat membantu sebuah perusahaan listrik menyimpan data, mengolah data dan menampilkannya kepada user. Dengan program tersebut perusahaan dapat melihat proses-proses transaksi yang telah dilakukan beserta data pelanggannya juga. Dengan demikian perusahaan mampu terus berjalan dengan baik walaupun memiliki banyak pelanggan dan transaksi yang terus berlangsung setiap harinya.

5.2 Saran

Tak ada gading yang tak retak, begitu pula dengan penulis. Penulis menyadari masih banyak kekurangan dalam laporan atau program yang perlu dibenahi. Oleh karena kritik dan saran sangat penulis perlukan agar kedepannya laporan atau program dapat dibuat lebih baik lagi lebih maksimal.

DAFTAR PUSTAKA

Muhardian, Ahmad. (2019, Mei 18). Belajar Pemrograman C #5: Mengenal Variabel, Tipe Data, Konstanta. Diakses pada 17 Mei 2021 melalui https://www.petanikode.com/c-variabel/

Muhardian, Ahmad. (2019, Mei 18). Belajar Pemrograman C #7: Mengenal 6 Macam Bentuk Blok Percabangan. Diakses pada 20 Mei 2021 melalui https://www.petanikode.com/c-percabangan/

Muhardian, Ahmad. (2019, Mei 18). Belajar Pemrograman C #8: Memahami Blok Perulangan pada C. Diakses pada 20 Mei 2021 melalui https://www.petanikode.com/c-perulangan/

Andre. (2018, Februari 20). Tutorial Belajar C Part 18: Pengertian dan Contoh Kode Program Tipe Data Array. Diakses pada 23 Mei 2021 melalui https://www.duniailkom.com/tutorial-belajar-c-pengertian-dan-contoh-kode-program-tipe-data-array/

Muhardian, Ahmad. (2019, November 26). Belajar Pemrograman C #14: Mengenal Tipe Data String pada C. Diakses pada 23 Mei 2021 melalui https://www.petanikode.com/c-string/

Muhardian, Ahmad. (2019, Desember 17). Belajar Pemrograman C #15: Cara Membaca dan Menulis File di C. Diakses pada 23 Mei 2021 melalui https://www.petanikode.com/c-file/