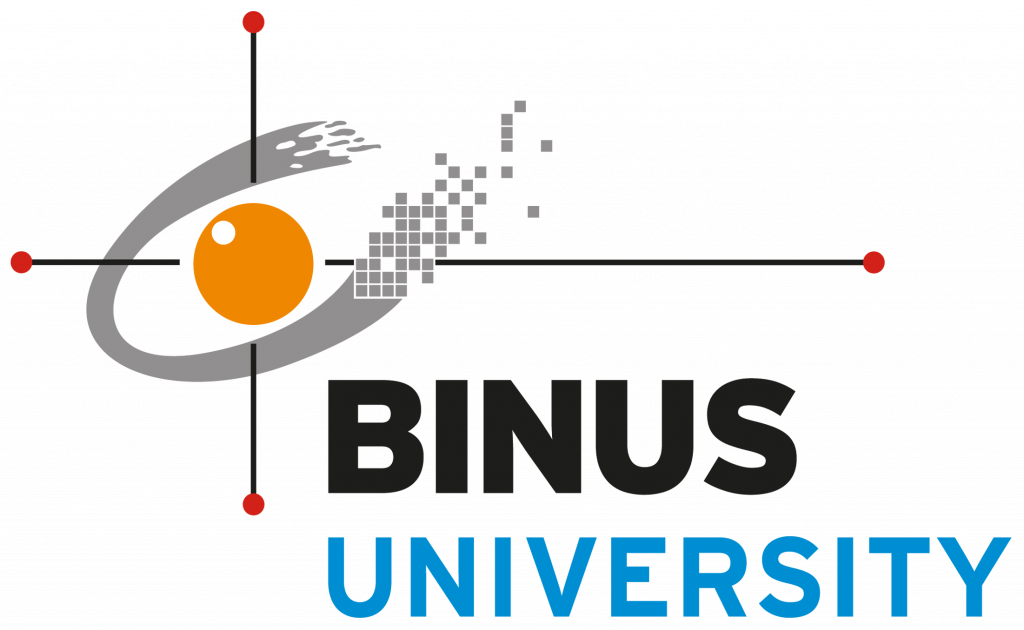
**IMPLEMENTASI HASH TABLE PADA DATA POSYANDU**

LAPORAN PROYEK AKHIR

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS BF20



Oleh :

2602181383 – APRILLIA CHANDRA DEWI

2602182865 – ALVI AULIA FATIKHA

2602185690 – NASYWA HILMI CAHYONO

Semester Genap [2022 – 2023]

MALANG

**LEMBAR PERSETUJUAN PROYEK AKHIR**

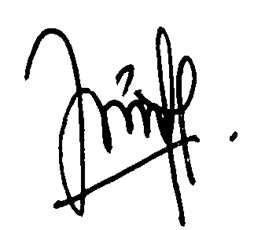
**Implementasi Hash Table pada Data Posyandu**

MATA KULIAH COMP6362004 – DATA STRUCTURES

KELAS BF20

Semester Genap [2022 – 2023]

Laporan akhir proyek ini adalah benar karya kami :

Aprillia Chandra Dewi Alvi Aulia Fatikha Nasywa Hilmi Cahyono

2602181383 2602182865 2602185690

Malang, 16 Juni 2023

Nyoman Wira Prasetya, S.Kom., M.T., M.Sc

D6494

**BAB I**

**LATAR BELAKANG**

* 1. **Latar Belakang**

Posyandu (Pos Pelayanan Terpadu) merupakan salah satu program kesehatan masyarakat yang dilaksanakan di Indonesia mulai dari pengecekan kesehatan secara berkala hingga vaksinasi. Kegiatan ini bertujuan untuk memberikan pelayanan kesehatan dasar kepada masyarakat, terutama ibu hamil dan menyusui, balita, serta lansia. Pada pelayanan posyandu, data individu seperti nama, usia, berat badan, tinggi badan, dan riwayat kesehatan lainnya akan selalu dipantau secara berkala untuk memastikan pertumbuhan dan perkembangan yang optimal.

Sesuai dengan perkembangan zaman, pemantauan data di posyandu semakin berkembang salah satunya dengan menggunakan teknologi digital. Pemrograman posyandu dengan metode hash table merupakan salah satu solusi untuk menyimpan dan mengakses data secara efisien. Metode hash table adalah struktur data yang menggunakan fungsi hash untuk mengonversi data menjadi indeks dalam array. Setiap data akan diberikan sebuah kunci (*key*) yang unik kemudian *key* tersebut akan diubah menjadi indeks dalam array. Dengan demikian, proses pencarian dan penyimpanan data menjadi lebih cepat dan efisien.

Dengan adanya sistem ini, petugas posyandu dapat dengan mudah mencari data pasien, memperbarui data, menghapus data yang tidak sesuai, atau mengambil data riwayat kesehatan dalam waktu yang singkat. Hal tersebut dapat mempermudah pemantauan pertumbuhan dan perkembangan kesehatan serta meningkatkan efisiensi dan efektivitas dalam pelayanan di posyandu. Dalam implementasinya, pemrograman posyandu dengan metode hash table memerlukan pemahaman yang baik tentang algoritma hash dan pengelolaan data. Maka dari itu, dengan adanya pemrograman posyandu balita yang menggunakan metode hash table ini, diharapkan pelayanan kesehatan di posyandu dapat lebih terorganisir, efisien, dan efektif.

**BAB II**

**TINJAUAN PUSTAKA**

* 1. **Struktur Data**

Struktur data dalam dunia komputer merupakan cara penyimpanan, pengorganisasian, serta pengaturan data dalam ruang penyimpanan komputer agar data dapat digunakan secara efisien. Struktur data biasanya diimplementasikan untuk pengolahan database, contohnya seperti data keuangan dan pengolah kata atau *word processor* di mana kolomnya (*record*) dapat berubah secara dinamis [1].

* 1. **Hash Table**

Metode *hashing* merupakan teknik untuk melakukan penambahan (*insert*) data, penghapusan (*delete*), serta pencarian (*search*) data dengan menggunakan alamat kunci (*key*) yang didapat dari sebuah fungsi. Ada beberapa macam fungsi *hashing*, seperti Mid-Square, Division, Folding, dan lain-lain. Namun, fungsi yang paling sering digunakan adalah Division karena fungsi ini paling sederhana di antara yang lainnya. Division merupakan teknik *hashing* dengan menggunakan modulus biasanya bilangan bulat akan dibagi dengan ukuran tabel *hash* dan mengambil hasil sisa bagi untuk dijadikan *key*. Fungsi *hashing* yang bagus memiliki dua kriteria, yaitu harus cepat dihitung dan tidak banyak terjadi *collision* [2].

* 1. **Linked List**

Linked list merupakan salah satu implementasi dari struktur data yang digunakan untuk menyimpan data secara urut yang memungkinkan pengguna melakukan pengurangan data, penambahan data, dan pencarian data dari elemen data yang tersimpan. Pada linked list, setiap node memiliki pointer yang digunakan sebagai penunjuk. Linked list terdiri dari empat macam, yaitu Single Linked List, Doubly Linked List, Circular Linked List, dan Circular Doubly Linked List [3].

* 1. **Tipe Data**

Tipe data merupakan pengelompokan jenis data sesuai dengan kategori data, ukuran, dan kegunaannya yang ditampung oleh sebuah variabel. Tipe data dasar dalam pemrograman komputer adalah bilangan bulat, riil, karakter, logika, dan string. Pada bahasa pemrograman C ada empat tipe data dasar, yaitu:

1. Tipe data integer

Variabel yang menggunakan tipe data ini digunakan untuk menampung data-data bernilai bilangan bulat. Tipe data ini dideklarasikan dengan tipe data berjenis int, dengan cara penulisan: **int a**.

1. Tipe data float

Variabel berjenis data float digunakan untuk menampung data-data bernilai bilangan desimal. Tipe data ini dideklarasikan dengan tipe data berjenis float, dengan cara penulisan: **float b**.

1. Tipe data char

Variabel dengan tipe data ini digunakan untuk menampung data-data berupa karakter seperti huruf abjad atau simbol. Tipe data ini dideklarasikan dengan tipe data char, dengan cara penulisan: **char x**.

1. Tipe data Boolean

Variabel tipe data ini digunakan apabila suatu fungsi bertipe logika. Tipe data ini mewakili nilai benar dan salah. Tipe data ini menggunakan tambahan file header **stdbool.h**, dengan cara penulisan:

**bool y,z;**

**y = true;**

**z = false;** [4].

* 1. **Struct**

Struct digunakan untuk mempresentasikan kumpulan komponen (element) yang memiliki satu jenis data saja atau bisa juga terdiri dari beberapa jenis tipe data dalam satu struct. Setiap element disebut juga sebagai property atau attribute. Contoh implementasi struct, yaitu:

**struct Node {**

**int id;**

**char nama[25];**

**int usia;**

**char gender;**

**float tinggi;**

**struct Node\* next;**

**};** [5]

**BAB III**

**GAMBARAN UMUM PROGRAM**

* 1. **Pseudocode**

CONSTANT TABLE\_SIZE = 1000

CONSTANT FILE\_NAME = "data.txt"

STRUCT Node

id: integer

nama: character array of size 25

usia: integer

gender: character

tinggi: float

berat: float

next: pointer to Node structure

END STRUCT

FUNCTION key(id)

RETURN id MOD TABLE\_SIZE

END FUNCTION

FUNCTION checkID(id)

IF id < 1000 OR id > 9999 THEN

RETURN 0

END IF

RETURN 1

END FUNCTION

FUNCTION checkNama(nama)

length = length of nama

IF length < 3 OR length > 25 THEN

RETURN 0

END IF

FOR i = 0 TO length-1

IF NOT (nama[i] is a letter) AND (nama[i] is not '\n') AND (nama[i] is not ' ') THEN

RETURN 0

END IF

END FOR

RETURN 1

END FUNCTION

FUNCTION checkGender(gender)

IF gender is not 'M' AND gender is not 'F' THEN

RETURN 0

END IF

RETURN 1

END FUNCTION

FUNCTION checkUsia(usia)

IF usia < 1 OR usia > 5 THEN

RETURN 0

END IF

RETURN 1

END FUNCTION

FUNCTION checkTinggi(tinggi)

IF tinggi < 50 OR tinggi > 99 THEN

RETURN 0

END IF

RETURN 1

END FUNCTION

FUNCTION checkBerat(berat)

IF berat < 5 OR berat > 25 THEN

RETURN 0

END IF

RETURN 1

END FUNCTION

FUNCTION insert(hashTable, id, nama, usia, gender, tinggi, berat)

i = key(id)

newNode = create new Node

newNode.id = id

copy nama to newNode.nama

newNode.usia = usia

newNode.gender = gender

newNode.tinggi = tinggi

newNode.berat = berat

newNode.next = NULL

IF hashTable[i] is NULL THEN

hashTable[i] = newNode

ELSE

current = hashTable[i]

WHILE current.next is not NULL DO

current = current.next

END WHILE

current.next = newNode

END IF

END FUNCTION

FUNCTION searchID(hashTable, id)

i = key(id)

current = hashTable[i]

WHILE current is not NULL DO

IF current.id is equal to id THEN

RETURN current

END IF

current = current.next

END WHILE

RETURN NULL

END FUNCTION

FUNCTION viewData(hashTable)

PRINT "-------------------------------------------------------------------------------------"

PRINT "| ID Balita | Nama Balita | Usia | Gender | Tinggi Badan | Berat Badan |"

PRINT "-------------------------------------------------------------------------------------"

FOR i = 0 TO TABLE\_SIZE-1

current = hashTable[i]

WHILE current is not NULL DO

PRINT "| %-9d | %-24s | %d | %c | %.2f cm | %.2f kg |" (current.id, current.nama, current.usia, current.gender, current.tinggi, current.berat)

current = current.next

END WHILE

END FOR

PRINT "-------------------------------------------------------------------------------------"

END FUNCTION

FUNCTION update(hashTable, id)

current = searchID(hashTable, id)

IF current is not NULL THEN

PRINT "Data ditemukan!"

PRINT "ID : " + current.id

PRINT "Nama : " + current.nama

PRINT "Usia : " + current.usia + " tahun"

PRINT "Gender : " + current.gender

PRINT "Tinggi Badan: " + current.tinggi + " cm"

PRINT "Berat Badan : " + current.berat + " kg"

PRINT "Masukkan data yang ingin diupdate:"

usia, gender, tinggi, berat = 0, '', 0.0, 0.0

DO

PRINT "Usia (1-5 tahun): "

READ usia

WHILE checkUsia(usia) is FALSE

current.usia = usia

DO

PRINT "Gender (M/F): "

READ gender

WHILE checkGender(gender) is FALSE

current.gender = gender

DO

PRINT "Tinggi Badan (50-99 cm): "

READ tinggi

WHILE checkTinggi(tinggi) is FALSE

current.tinggi = tinggi

DO

PRINT "Berat Badan (5-25 kg): "

READ berat

WHILE checkBerat(berat) is FALSE

current.berat = berat

PRINT "Data berhasil diupdate!"

ELSE

PRINT "Data tidak ditemukan."

END IF

END FUNCTION

FUNCTION delete(hashTable, id)

i = key(id)

current = hashTable[i]

previous = NULL

WHILE current is not NULL DO

IF current.id is equal to id THEN

IF previous is NULL THEN

hashTable[i] = current.next

ELSE

previous.next = current.next

END IF

free current

PRINT "Data berhasil dihapus!"

RETURN

END IF

previous = current

current = current.next

END WHILE

PRINT "Data tidak ditemukan."

END FUNCTION

FUNCTION meanData(hashTable)

j = 0

totalBerat = 0

totalTinggi = 0

FOR i = 0 TO TABLE\_SIZE-1

current = hashTable[i]

WHILE current is not NULL DO

totalBerat = totalBerat + current.berat

totalTinggi = totalTinggi + current.tinggi

j = j + 1

current = current.next

END WHILE

END FOR

IF j > 0 THEN

meanBerat = totalBerat / j

meanTinggi = totalTinggi / j

PRINT "Rata-rata Tinggi Badan : " + meanTinggi + " cm"

PRINT "Rata-rata Berat Badan : " + meanBerat + " kg"

PRINT "Dari " + j + " data yang tersedia."

ELSE

PRINT "Tidak ada data yang tersedia."

END IF

END FUNCTION

FUNCTION header()

EXECUTE system command "cls"

PRINT "==========================================================================="

PRINT "=x=x=x=x=x=x x=x=x=x=x=x="

PRINT "=x=x=x=x=x=x <3 Data Balita <3 x=x=x=x=x=x="

PRINT "=x=x=x=x=x=x x=x=x=x=x=x="

PRINT "==========================================================================="

END FUNCTION

FUNCTION loading()

PRINT "Loading... "

FOR i = 0 TO 4

PRINT i \* 100 / 4 + "% "

FLUSH standard output

EXECUTE sleep command for 1 second

PRINT "\b\b\b\b"

END FOR

PRINT "Loading complete!"

END FUNCTION

FUNCTION welcomeMessage()

PRINT "\n\n\n\n\n"

PRINT "=================================================="

PRINT "= ="

PRINT "= WELCOME TO ="

PRINT "= POSYANDU ="

PRINT "= ="

PRINT "=================================================="

CALL loading()

END FUNCTION

FUNCTION login()

EXECUTE system command "cls"

CALL header()

SET defaultUsername = "admin"

SET defaultPassword = "admin"

DECLARE username[20]

DECLARE password[20]

SET tryL = 0

DO

PRINT "Username: "

READ username

PRINT "Password: "

READ password

IF username = defaultUsername AND password = defaultPassword THEN

RETURN

ELSE

PRINT "Maaf, informasi yang Anda masukkan salah"

tryL = tryL + 1

END IF

WHILE tryL <= 3

IF tryL > 3 THEN

EXECUTE system command "cls"

CALL header()

PRINT "Login Gagal"

PRINT "Maaf, Anda belum terdaftar."

CALL getchar() // Waits for a key press

EXIT program with error code 1

END IF

END FUNCTION

FUNCTION main()

DECLARE hashTable[TABLE\_SIZE]

DECLARE hash[1000]

SET i = 0

FOR i = 0 TO TABLE\_SIZE - 1

SET hashTable[i] = NULL

END FOR

DECLARE fp = OPEN FILE\_NAME for reading

IF fp == NULL THEN

PRINT "Data tidak tersedia"

EXIT program with error code 1

END IF

WHILE not end of file(fp)

READ hash[j] from fp

SET i = key(hash[j].id)

CREATE new node, newNode

SET newNode values from hash[j]

SET newNode.next = NULL

IF hashTable[i] is empty THEN

SET hashTable[i] = newNode

ELSE

SET current = hashTable[i]

WHILE current.next is not NULL

SET current = current.next

END WHILE

SET current.next = newNode

END IF

INCREMENT j

END WHILE

CLOSE fp

CALL welcomeMessage()

CALL login()

DECLARE choice

DO

EXECUTE system command "cls"

CALL header()

PRINT "1. Insert"

PRINT "2. Update"

PRINT "3. Search"

PRINT "4. Display"

PRINT "5. Delete"

PRINT "6. Rata-Rata"

PRINT "7. Exit"

PRINT "Enter your choice: "

READ choice

SWITCH choice

CASE 1:

CALL insertData(hashTable)

BREAK

CASE 2:

CALL updateData(hashTable)

BREAK

CASE 3:

CALL searchData(hashTable)

BREAK

CASE 4:

CALL displayData(hashTable)

BREAK

CASE 5:

CALL deleteData(hashTable)

BREAK

CASE 6:

CALL meanData(hashTable)

BREAK

CASE 7:

PRINT "Thank you!"

BREAK

DEFAULT:

PRINT "Invalid choice!"

BREAK

END SWITCH

WHILE choice != 7

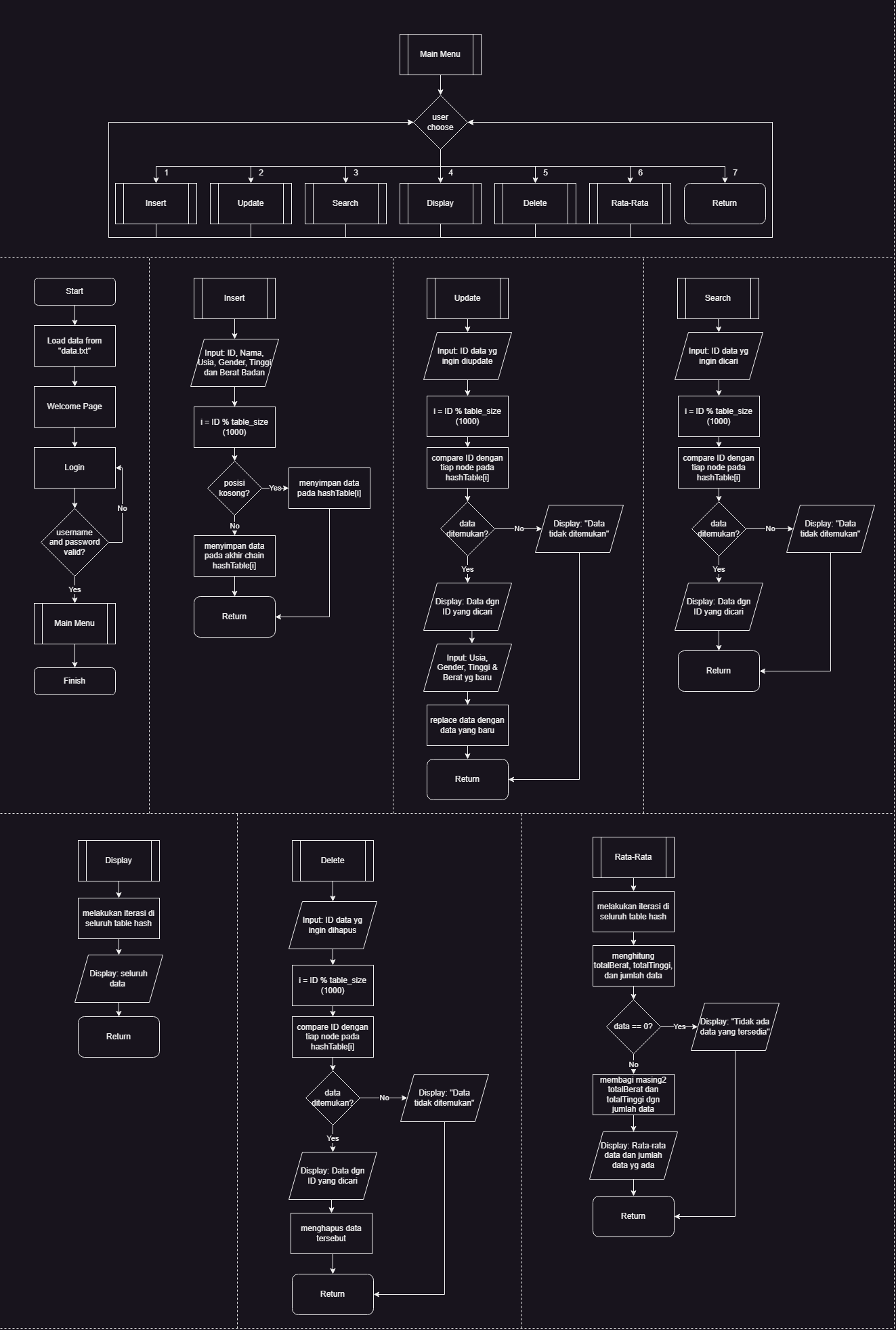
RETURN 0

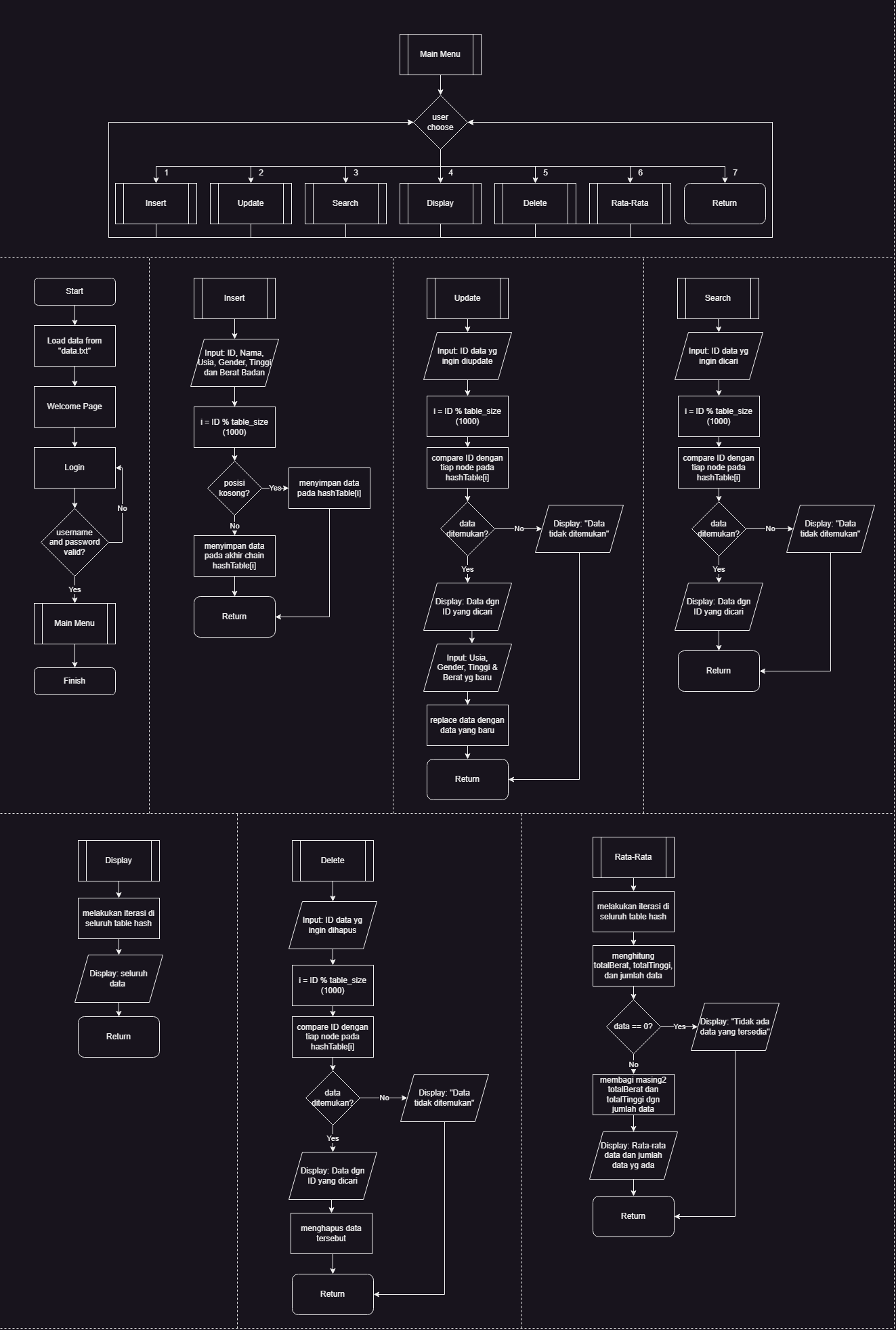
END FUNCTION

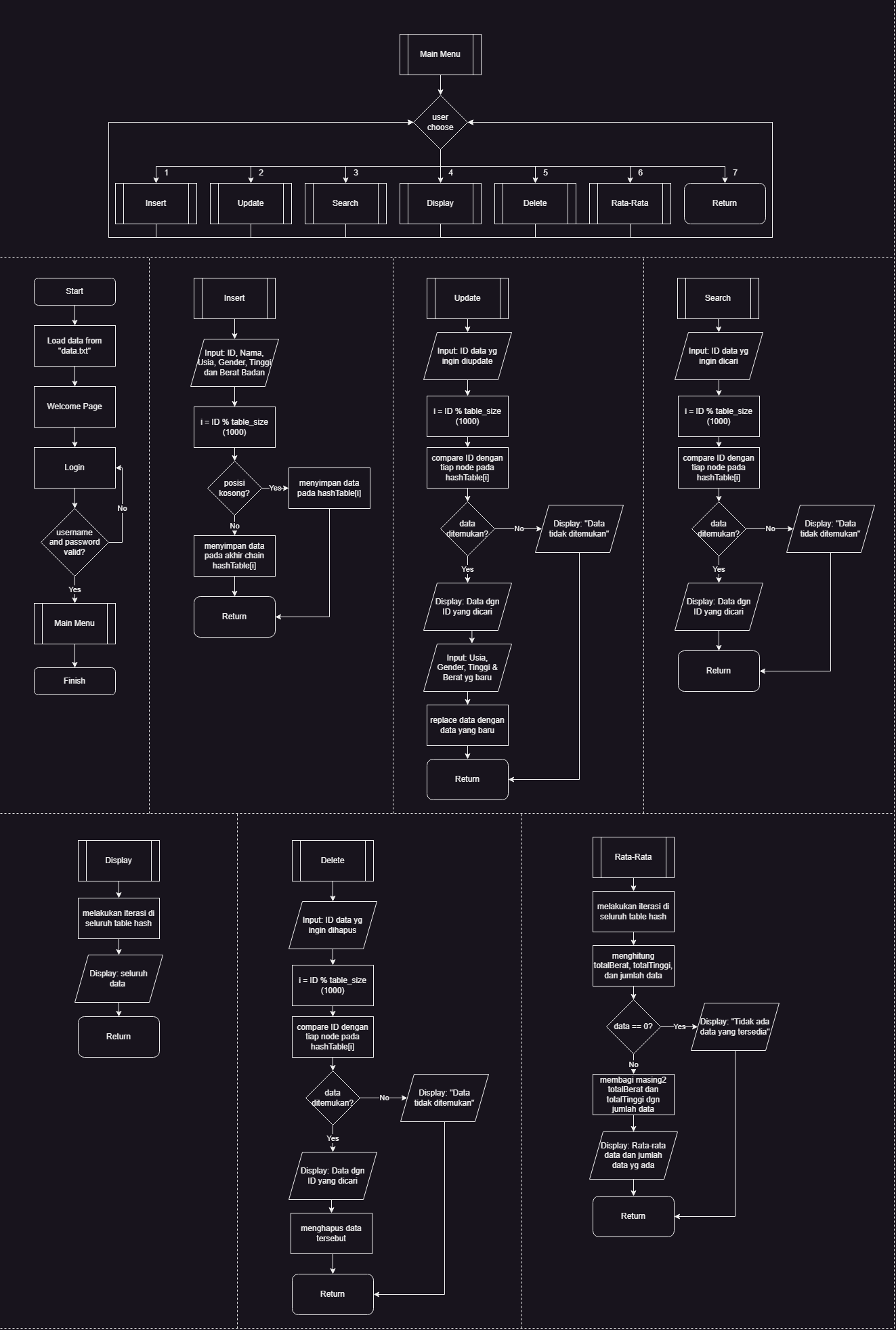
* 1. **Flowchart**

Berikut adalah lampiran dari flowchart kami:

<https://drive.google.com/file/d/1OfPkMnT2G6_vQPm5jhzC1BZjL9HZ9aqL/view?usp=sharing>







* 1. **Deskripsi Program**

Program “Posyandu” berfungsi sebagai sistem manajemen data untuk informasi tentang balita. Program ini dapat digunakan oleh berbagai pengguna, seperti tenaga medis profesional, administrator klinik kesehatan anak, atau individu yang bertanggung jawab untuk memantau perkembangan anak. Tenaga medis yang bekerja di layanan kesehatan anak dapat memanfaatkan program ini untuk menyimpan catatan kesehatan dan pertumbuhan anak. Mereka dapat dengan mudah memasukkan data baru, memperbarui catatan yang ada, mencari balita tertentu berdasarkan ID mereka, dan menghitung nilai rata-rata tinggi dan berat badan balita. Program ini membantu para profesional memantau perkembangan anak secara individu dan mengidentifikasi suatu *outlier* atau pola. Secara keseluruhan, program ini menyediakan *interface* yang ramah pengguna dan berbagai fungsi untuk mengelola dan menganalisis data kesehatan anak. Program ini membantu proses pengorganisasian informasi, sehingga memudahkan profesional kesehatan, administrator klinik, dan orang tua/wali untuk memantau dan mengevaluasi perkembangan dan pertumbuhan anak.

* 1. **Fitur Program**

1. Loading Screen

Program diawali dengan loading screen yang mencetak pesan memuat awal program. Kemudian, program memasuki loop yang diulang sebanyak lima kali. Dalam setiap iterasi, program mencetak persentase progres saat ini, lalu menghapus output untuk menampilkan teks "Loading" dengan langsung, menunggu selama 1 detik untuk perintah sleep, kemudian memindahkan kursor ke belakang sejauh empat posisi untuk menimpa persentase progres dengan karakter backspace. Terakhir, program akan mencetak pesan "Loading complete!" setelah perulangan selesai.

1. Login

Fitur login membersihkan layar konsol setelah loading screen, menampilkan header, dan menginisialisasi nilai username dan password default. Kemudian program meminta pengguna untuk memasukkan username dan password. Jika nilai yang diberikan cocok dengan username dan password default, maka program akan berlanjut ke proses berikutnya. Jika tidak, program akan menampilkan pesan kesalahan dan menambahkan variabel tryL. Looping akan berlanjut hingga pengguna memasukkan username dan password yang benar atau melebihi tiga kali percobaan gagal. Jika jumlah percobaan maksimum tercapai, program menampilkan pesan gagal masuk, menunggu input tombol, dan keluar dengan kode errror 1.

1. Insert

Fitur insert berfungsi untuk memasukkan data baru ke dalam Hash Table dengan parameter ID, nama, usia, gender, tinggi, dan berat badan. Parameter ID digunakan untuk membuat key yang nantinya digunakan sebagai index. Key ini didapat dari modulus ID dibagi ukuran Hash Table yaitu 1000. Fitur ini membuat stucture node baru, memberi value ke setiap anggota struct tersebut, kemudian memasukkan node baru itu ke dalam hash table berdasarkan index ‘i’. Jika hash table di index ‘i’ kosong (NULL), maka node baru akan dimasukkan ke dalam index tersebut. Jika index tidak kosong, maka node tersebut akan melewati linked list pada index yang diberikan dan membuat node baru di akhir. Untuk persyaratan value masing-masing data agar dapat diterima dengan program yaitu:

1. ID: tidak kurang dari 1000 dan tidak lebih dari 9999
2. Nama: terdiri dari 3-25 karakter
3. Usia: berusia 1-5 tahun
4. Gender: Male atau Female
5. Tinggi badan: 50 - 99 cm
6. Berat badan: 5-25 kg

Jika pengguna tidak menginput value sesuai persyaratan di atas, maka program akan meminta kembali input dari user hingga sesuai dengan persyaratan.

1. Update

Fitur update memungkinkan pengguna untuk mengupdate data sebuah node di dalam hash table. Pertama, program akan meminta input user berupa ID balita. Jika ada node yang sesuai dengan ID yang diinput user, maka program akan menampilkan data node tersebut kemudian meminta pengguna untuk memasukkan value yang baru lalu data node tersebut akan terupdate sesuai dengan masukkan pengguna. Value baru yang akan dimasukkan oleh user juga harus memenuhi persyaratan seperti yang ada pada fitur insert. Jika tidak ada node yang ditemukan dari ID, program menampilkan pesan bahwa tidak tidak ada data yang ditemukan.

1. Search

Fitur search berfungsi untuk mencari data balita berupa node dalam hash table. Fitur ini diawali dengan program meminta input dari user berupa ID balita. Kemudian, jika ada node yang memiliki ID yang sama dengan input pengguna maka program akan menampilkan data-data balita tersebut. Jika program tidak menemukan ID yang sama dengan input pengguna, maka program akan menampilkan pesan bahwa data tidak ditemukan.

1. Display

Fitur display menampilkan data yang disimpan dalam hash table dalam bentuk tabel yang sudah diformat. Program akan mengiterasi hash table dan mencetak ID, nama, usia, gender, tinggi, dan berat badan balita. Data-data tersebut ditampilkan dalam sebuah border.

1. Delete

Fitur ini menghapus sebuah node dengan ID yang sesuai di dalam hash table. Pertama, program akan meminta pengguna untuk memasukkan sebuah ID yang ingin dihapus. Kemudian program akan mencari node dalam hash table yang sesuai dengan ID tersebut kemudian menghapus node tersebut dari linked list. Jika node berhasil ditemukan dan dihapus, program akan menampilkan pesan data berhasil dihapus. Jika node tidak ditemukan, program akan menampilkan pesan bahwa data tidak ditemukan.

1. Rata-rata

Fitur ini berguna untuk menghitung rata-rata dari nilai tinggi dan berat badan balita dari data yang disimpan di dalam hash table. Program akan mengiterasi hash table, lalu mengakumulasi total nilai tinggi dan berat badan, dan menghitung jumlah node dalam hash table tersebut. Jika ada node dalam hash table, program akan menghitung rata-rata tinggi dan berat badan (total nilai tinggi/berat dibagi dengan jumlah node) dan menampilkan hasilnya serta mencetak jumlah data yang ada. Jika tidak ada node, maka program mencetak pesan bahwa tidak ada data yang tersedia.

1. File Processing

Program ini menggunakan file processing untuk membaca data dari file eksternal dan mengisi hash table dengan informasi yang diterima. File ini diberi nama “data.txt” yang berisi data balita termasuk ID, nama, usia, gender, tinggi, dan berat badan. Program diawali dengan membuka file dengan perintah ‘fopen’ dan mengecek apakah file tersebut ada atau tidak. Jika file ditemukan, program akan melanjutkan membaca setiap baris file menggunakan ‘fscanf’ dan mengekstrak data ke dalam variabel yang sesuai. Kemudian, program akan membuat node baru untuk setiap data balita dan menetapkan value yang diekstrak ke atribut node. Node kemudian dimasukkan ke dalam tabel hash berdasarkan indeks hash yang dihitung. Proses ini berlanjut hingga akhir file tercapai. Terakhir, program menutup file menggunakan ‘fclose’. Dengan memanfaatkan file processing, program dapat menginisialisasi tabel hash dengan data yang sudah ada sebelumnya, membuatnya mudah untuk mengelola dan memperbarui catatan balita.

* 1. **Layout Design**

1. Header

Setiap fitur pada program menampilkan header pada bagian atas halaman yang berisi teks “Data Balita” dan elemen-elemen dekoratif. Teks “Data Balita” diberi kombinasi warna merah dan biru yang membuatnya menarik secara visual.

1. Pilihan Menu

Program menampilkan menu bernomor dari opsi yang tersedia. Setiap opsi dicantumkan secara jelas dan ringkas, sehingga memudahkan pengguna untuk memahami dan memilih fungsionalitas yang diinginkan. Opsi menu disejajarkan secara vertikal yang memudahkan keterbacaan.

1. User Input

Program meminta pengguna untuk memasukkan data tertentu bila diperlukan, seperti saat memasukkan atau memperbarui data balita. Petunjuknya disajikan dalam format yang terstruktur dengan baik, dengan instruksi yang jelas dan jarak yang sesuai. Hal ini memastikan bahwa pengguna dapat memasukkan data yang diperlukan secara akurat.

1. Display Data

Saat menampilkan data balita, program mengatur informasi dalam format tabel. Setiap data, seperti ID, nama, usia, gender, tinggi, dan berat badan disajikan dalam kolom yang memungkinkan perbandingan dan pemahaman yang mudah. Data juga disejajarkan dengan rapi agar meningkatkan keterbacaan.

1. Notifikasi Pesan

Program menggunakan warna teks yang berbeda, seperti hijau untuk pesan sukses dan merah untuk pesan error untuk menarik perhatian ke informasi penting. Perbedaan warna ini membantu pengguna dengan cepat mengidentifikasi status atau hasil operasi sehingga memberikan mekanisme umpan balik yang jelas.

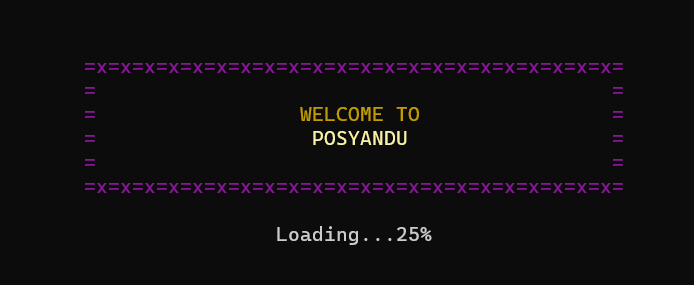
**BAB IV**

**HASIL**

* 1. **Screenshot Program**

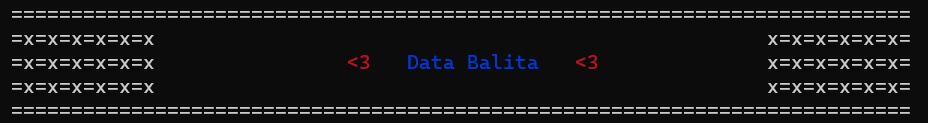
1. Welcome Message

Saat program pertama kali dijalankan, program akan menampilkan Welcome Message dan loading untuk menuju ke Main Menu.



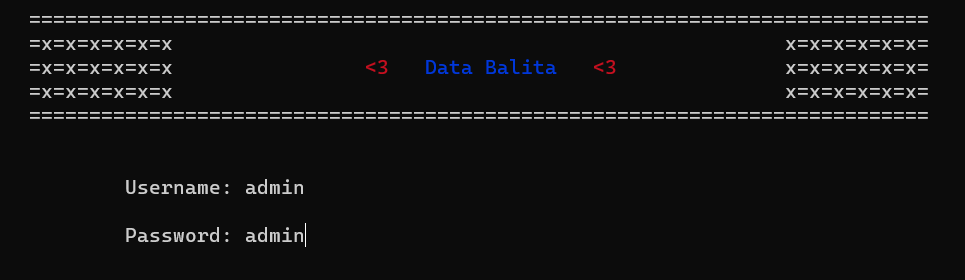
1. Header

Header yang bertuliskan “Data Balita” ini akan ada pada setiap fitur menu yang ada pada program ini.



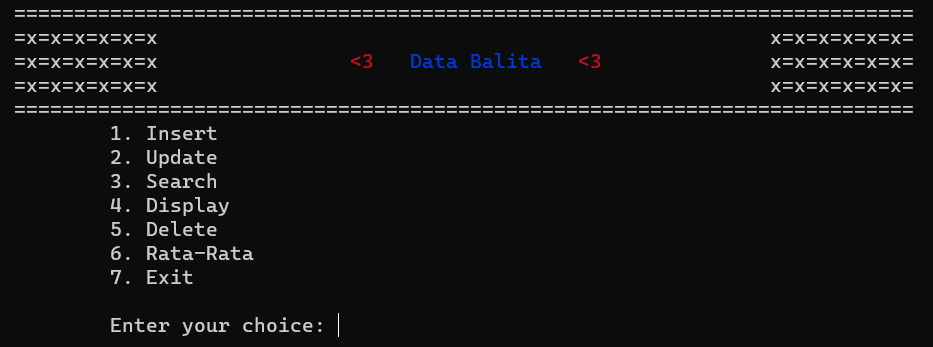
1. Fitur Login

Setelah *user* menunggu *loading*  pada Welcome Message di awal, *user* akan diminta menginput username dan password untuk memasuki program.



1. Main Menu

Jika *user* berhasil login, maka *user* akan dialihkan ke Main Menu yang berisi fitur-fitur yang ada di dalam program ini, yaitu meliputi Insert Data, Update Data, Search Data, Display Data. Delete Data, Rata-rata tinggi badan dan berat badan, dan Exit Program.



1. Fitur Insert

Pada fitur Insert Data, *user* diminta untuk menginput data balita, yaitu meliputi ID, Nama, Usia, Gender, Tinggi Badan, dan Berat Badan. Lalu jika data berhasil ditambahkan akan muncul pesan berwarna hijau “Data berhasil ditambahkan!”.

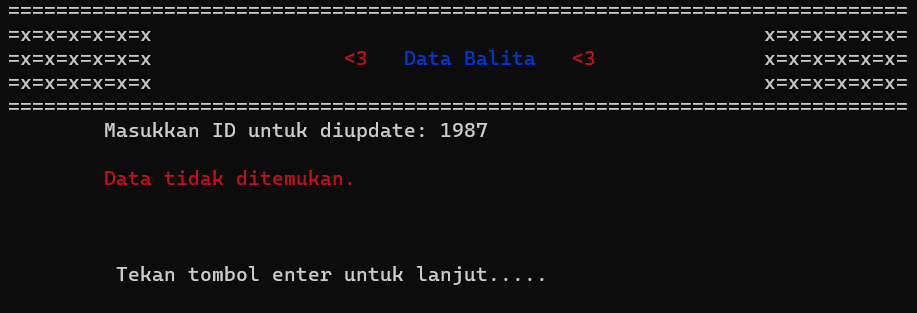


1. Fitur Update

Pada awal fitur ini, *user* akan diminta menginput ID balita yang ingin diupdate. Jika data berhasil ditemukan program akan menampilkan pesan berwarna hijau “Data ditemukan!” lalu dilanjutkan dengan menampilkan data sebelum dilakukan update. Setelah itu *user* akan diminta menginput data balita yang baru, yaitu meliputi Usia, Gender, Tinggi Badan, dan Berat Badan. Jika data berhasil diupdate, program akan menampilkan pesan berwarna hijau “Data berhasil diupdate!”



Jika *user* menginputkan ID yang salah atau tidak tersimpan, maka program akan menampilkan pesan berwarna merah “Data tidak ditemukan.”

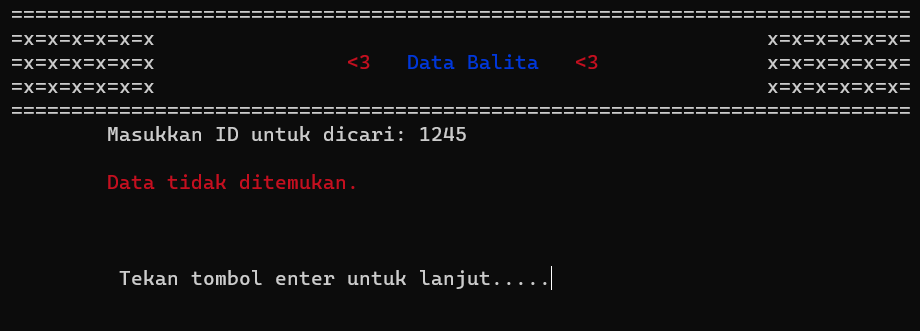


1. Fitur Search

*User* akan diminta untuk menginput ID yang ingin dilakukan pencarian data. Jika data ditemukan, program akan menampilkan pesan berwarna hijau “Data ditemukan!” dilanjut dengan menampilkan data balita meliputi ID, Nama, Usia, Gender, Tinggi Badan, dan Berat Badan.

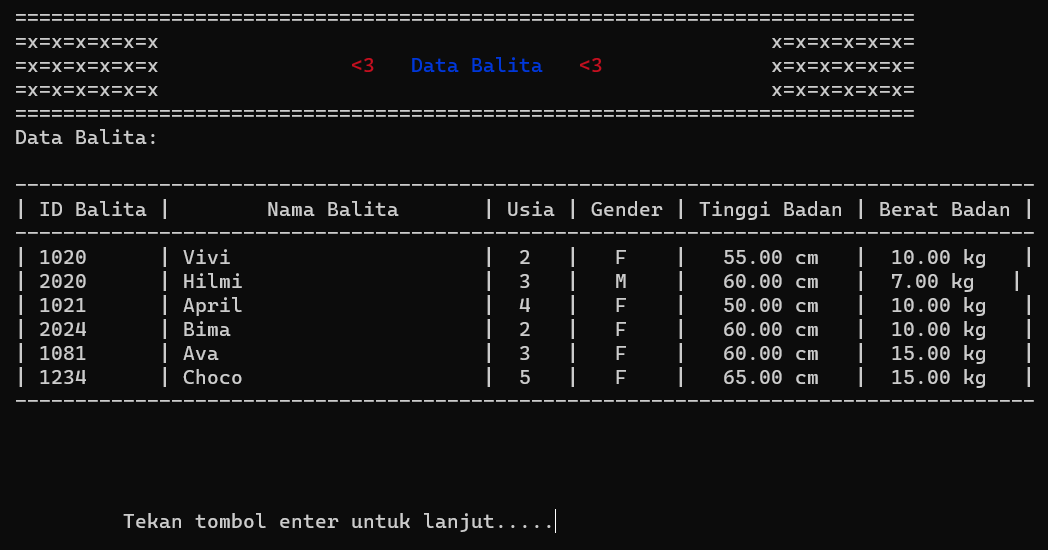


Jika data tidak ditemukan maka program akan menampilkan pesan berwarna merah “Data tidak ditemukan.”



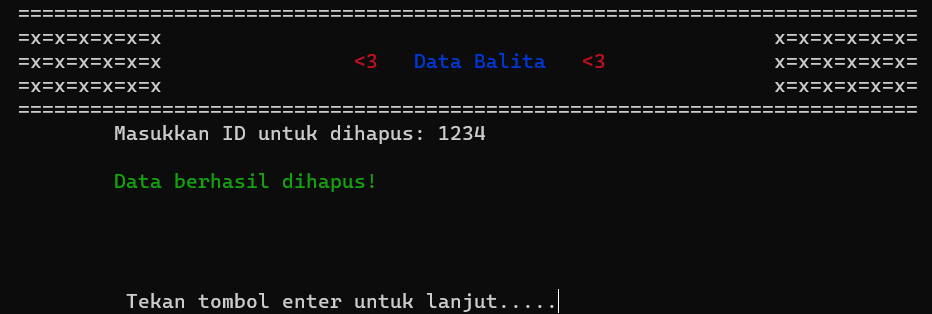
1. Fitur Display

Pada fitur ini program akan menampilkan semua data balita yang tersedia dalam bentuk tabel.

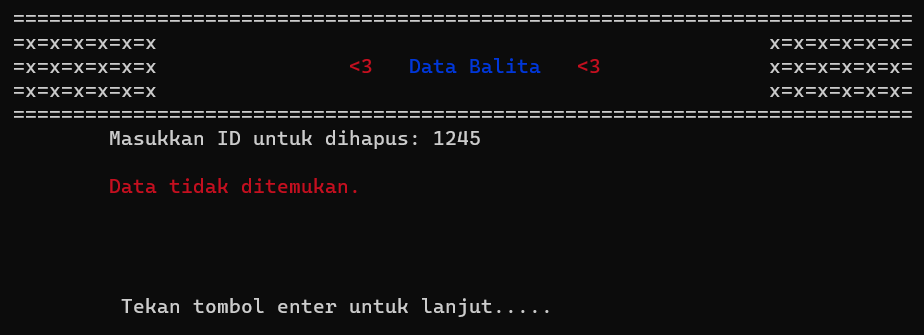


1. Fitur Delete

Untuk melakukan fitur ini, *user* diminta untuk menginput ID yang ingin dihapus. Lalu program akan melakukan pencarian data dan jika data berhasil dihapus maka program akan menampilkan pesan berwarna hijau “Data berhasil dihapus!”



Namun, jika data tidak ditemukan maka program akan menampilkan pesan berwarna merah “Data tidak ditemukan.”



1. Fitur Rata-rata

Pada fitur ini program akan menampilkan hasil rata-rata dari seluruh data Tinggi Badan dan Berat Badan yang ada disertai keterangan dari berapa jumlah data yang ada.



1. Exit

jika *user* memilih untuk keluar maka program akan menampilkan pesan “Thank you!”



* 1. **Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <conio.h>

#include <unistd.h>

#define TABLE\_SIZE 1000

#define FILE\_NAME "data.txt"

struct Node {

int id;

char nama[25];

int usia;

char gender;

float tinggi, berat;

struct Node\* next;

};

// modulus untuk menghasilkan key berdasarkan ID

int key(int id){

return id % TABLE\_SIZE;

}

// validasi input usia

int checkID(int id){

if(id < 1000 || id > 9999){

return 0;

}

return 1;

}

// validasi input nama

int checkNama(char nama[]){

int length = strlen(nama);

if(length < 3 || length > 25){

return 0;

}

for(int i = 0; i < length; i++){

if (!(isalpha(nama[i])) && (nama[i] != '\n') && (nama[i] != ' ')) {

return 0;

}

}

return 1;

}

// validasi input gender

int checkGender(char gender){

if(gender != 'M' && gender != 'F'){

return 0;

}

return 1;

}

// validasi input usia

int checkUsia(int usia){

if(usia < 1 || usia > 5){

return 0;

}

return 1;

}

// validasi input tinggi badan

int checkTinggi(float tinggi){

if(tinggi < 50 || tinggi > 99){

return 0;

}

return 1;

}

int checkBerat(float berat){

if(berat < 5 || berat > 25){

return 0;

}

return 1;

}

// menambahkan data ke tabel hash

void insert(struct Node\* hashTable[], int id, char nama[], int usia, char gender, float tinggi, float berat){

int i = key(id);

// membuat node baru

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->id = id;

strcpy(newNode->nama, nama);

newNode->usia = usia;

newNode->gender = gender;

newNode->tinggi = tinggi;

newNode->berat = berat;

newNode->next = NULL;

// memeriksa apakah posisi kosong

if (hashTable[i] == NULL){

hashTable[i] = newNode; // node ditambah pada posisi kosong

} else{

// menambahkan node baru ke akhir jika terjadi collision

struct Node\* current = hashTable[i];

while (current->next != NULL) { // mencari posisi kosong pada chain

current = current->next;

}

current->next = newNode; // menambah node pada akhir chain

}

}

// mencari data berdasarkan ID balita

struct Node\* searchID(struct Node\* hashTable[], int id){

int i = key(id);

// mencari node dengan ID yang cocok

struct Node\* current = hashTable[i];

while(current != NULL){

if(current->id == id){

return current;

}

current = current->next;

}

// node tidak ditemukan

return NULL;

}

// menampilkan seluruh data

void viewData(struct Node\* hashTable[]){

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

printf("\t\t\t| ID Balita | Nama Balita | Usia | Gender | Tinggi Badan | Berat Badan |\n");

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

// melakukan iterasi di seluruh tabel hash

for(int i = 0; i < TABLE\_SIZE; i++){

struct Node\* current = hashTable[i];

while(current != NULL){

printf("\t\t\t| %-9d | %-24s | %d | %c | %.2f cm | %.2f kg |\n", current->id, current->nama, current->usia, current->gender, current->tinggi, current->berat);

current = current->next;

}

}

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

}

// mengupdate data berdasarkan ID balita

void update(struct Node\* hashTable[], int id){

struct Node\* current = searchID(hashTable, id);

if(current != NULL){

printf("\033[0;32m");

printf("\n\t\t\t\tData ditemukan!\n");

printf("\033[0m");

printf("\n\t\t\t\tID\t\t: %d\n", current->id);

printf("\t\t\t\tNama\t\t: %s\n", current->nama);

printf("\t\t\t\tUsia\t\t: %d tahun\n", current->usia);

printf("\t\t\t\tGender\t\t: %c\n", current->gender);

printf("\t\t\t\tTinggi Badan\t: %.2f cm\n", current->tinggi);

printf("\t\t\t\tBerat Badan\t: %.2f kg\n", current->berat);

printf("\n\t\t\t\tMasukkan data yang ingin diupdate:\n");

int usia;

char gender;

float tinggi, berat;

do{

printf("\n\t\t\t\tUsia (1-5 tahun)\t: ");

scanf("%d", &usia);

}while(!checkUsia(usia));

current->usia = usia;

do{

printf("\t\t\t\tGender (M/F)\t\t: ");

scanf(" %c", &gender);

}while(!checkGender(gender));

current->gender = gender;

do{

printf("\t\t\t\tTinggi Badan (50-99 cm)\t: ");

scanf("%f", &tinggi);

}while(!checkTinggi(tinggi));

current->tinggi = tinggi;

do{

printf("\t\t\t\tBerat Badan (5-25 kg)\t: ");

scanf("%f", &berat);

}while(!checkBerat(berat));

current->berat = berat;

printf("\033[0;32m");

printf("\n\t\t\t\tData berhasil diupdate!\n");

printf("\033[0m");

} else {

printf("\033[0;31m");

printf("\n\t\t\t\tData tidak ditemukan.\n");

printf("\033[0m");

}

}

// menghapus data berdasarkan ID balita

void delete(struct Node\* hashTable[], int id){

int i = key(id);

// mencari node dengan ID yang cocok

struct Node\* current = hashTable[i];

struct Node\* previous = NULL;

while(current != NULL){

if(current->id == id){

if(previous == NULL){

// jika node pertama pada chain

hashTable[i] = current->next;

} else {

// jika node berada di tengah atau akhir chain

previous->next = current->next;

}

free(current);

printf("\033[0;32m");

printf("\n\t\t\t\tData berhasil dihapus!\n\n");

printf("\033[0m");

return;

}

previous = current;

current = current->next;

}

printf("\033[0;31m");

printf("\n\t\t\t\tData tidak ditemukan.\n\n");

printf("\033[0m");

}

// menghitung hasil rata-rata data

void meanData(struct Node\* hashTable[]){

int j = 0;

float totalBerat = 0, totalTinggi = 0;

for (int i = 0; i < TABLE\_SIZE; i++){

struct Node\* current = hashTable[i];

while(current != NULL){

totalBerat += current->berat;

totalTinggi += current->tinggi;

j++;

current = current->next;

}

}

if(j > 0){

float meanBerat, meanTinggi;

meanBerat = totalBerat / j;

meanTinggi = totalTinggi / j;

printf("\n\t\t\t\tRata-rata Tinggi Badan\t: %.2f cm\n", meanTinggi);

printf("\t\t\t\tRata-rata Berat Badan\t: %.2f kg\n\n", meanBerat);

printf("\t\t\t\tDari %d data yang tersedia.", j);

} else{

printf("\033[0;31m");

printf("\n\t\t\t\tTidak ada data yang tersedia.\n");

printf("\033[0m");

}

}

//header untuk tiap menu

void header(){

system("cls");

printf("\t\t\t===========================================================================");

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

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

printf("\033[0;31m");

printf("<3");

printf("\033[0;34m");

printf(" Data Balita ");

printf("\033[0;31m");

printf("<3");

printf("\033[0m");

printf(" x=x=x=x=x=x=");

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

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

}

void loading() {

printf("\n\n\t\t\t\t\t\tLoading... ");

for (int i = 0; i <= 4; i++) {

printf("%d%% ", (i \* 100) / 4);

fflush(stdout);

sleep(1); // Sleep for 1 second

printf("\b\b\b\b");

}

printf("\n\n\t\t\t\t\tLoading complete!\n");

}

//display welcome message

void welcomeMessage(){

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

printf("\033[0;35m");

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

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

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

printf("\033[0;33m");

printf("WELCOME TO");

printf("\033[0;35m");

printf(" =");

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

printf("\033[1;33m");

printf("POSYANDU");

printf("\033[0;35m");

printf(" =");

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

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

printf("\033[0m");

loading();

}

void login(){

//default login

system("cls");

header();

const char defaultUsername[] = "admin";

const char defaultPassword[] = "admin";

char username[20];

char password[20];

int tryL = 0;

do{

printf("\n\n\n\t\t\t\tUsername: ");

scanf(" %[^\n]", username);

printf("\n\t\t\t\tPassword: ");

scanf(" %[^\n]", password);

if((!strcmp(username, defaultUsername)) && (!strcmp(password, defaultPassword))){

return;

}

else{

printf("\033[0;31m");

printf("\t\t\t\tMaaf, informasi yang Anda masukkan salah");

printf("\033[0m");

tryL++;

}

}

while(tryL<=3);

if(tryL>3){ //jika percobaan gagal lebih dari 3 kali

system("cls");

header();

printf("\033[0;31m");

printf("\n\t\t\t\tLogin Gagal");

printf("\n\t\t\t\tMaaf, Anda belum terdaftar.");

printf("\033[0m");

getchar();

exit(1);

}

}

int main(){

struct Node\* hashTable[TABLE\_SIZE];

struct Node hash[1000];

// Inisialisasi tabel hash dengan NULL

for(int i = 0; i < TABLE\_SIZE; i++){

hashTable[i] = NULL;

}

FILE\* fp = fopen(FILE\_NAME, "r");

if (fp == NULL) {

printf("\n\t\t\t\tData tidak tersedia\n");

exit(1);

}

int j = 0;

while(!feof(fp)){

fscanf(fp, "%d\t%s\t%d\t%c\t%f\t%f\n", &hash[j].id, hash[j].nama, &hash[j].usia, &hash[j].gender, &hash[j].tinggi, &hash[j].berat);

int i = key(hash[j].id);

// membuat node baru

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

newNode->id = hash[j].id;

strcpy(newNode->nama, hash[j].nama);

newNode->usia = hash[j].usia;

newNode->gender = hash[j].gender;

newNode->tinggi = hash[j].tinggi;

newNode->berat = hash[j].berat;

newNode->next = NULL;

// memeriksa apakah posisi kosong

if (hashTable[i] == NULL){

hashTable[i] = newNode; // node ditambah pada posisi kosong

} else{

// menambahkan node baru ke akhir jika terjadi collision

struct Node\* current = hashTable[i];

while (current->next != NULL) { // mencari posisi kosong pada chain

current = current->next;

}

current->next = newNode; // menambah node pada akhir chain

}

j++;

}

fclose(fp);

welcomeMessage();

login();

int choice;

do{

system("cls");

header();

printf("\n\t\t\t\t1. Insert\n");

printf("\t\t\t\t2. Update\n");

printf("\t\t\t\t3. Search\n");

printf("\t\t\t\t4. Display\n");

printf("\t\t\t\t5. Delete\n");

printf("\t\t\t\t6. Rata-Rata\n");

printf("\t\t\t\t7. Exit\n");

printf("\n\t\t\t\tEnter your choice: ");

scanf("%d", &choice);

switch(choice){

case 1:{

system("cls");

header();

int id;

char nama[50];

int usia;

char gender;

float tinggi, berat;

printf("\n\t\t\t\tMasukkan data balita:\n");

do{

printf("\n\t\t\t\tID (4-digit)\t\t: ");

scanf("%d", &id);

}while(!checkID(id));

do{

printf("\t\t\t\tNama (3-25 karakter)\t: ");

scanf(" %[^\n]", nama);

}while(!checkNama(nama));

do{

printf("\t\t\t\tUsia (1-5 tahun)\t: ");

scanf("%d", &usia);

}while(!checkUsia(usia));

do{

printf("\t\t\t\tGender (M/F)\t\t: ");

scanf(" %c", &gender);

}while(!checkGender(gender));

do{

printf("\t\t\t\tTinggi Badan (50-99 cm)\t: ");

scanf("%f", &tinggi);

}while(!checkTinggi(tinggi));

do{

printf("\t\t\t\tBerat Badan (5-25 kg)\t: ");

scanf("%f", &berat);

}while(!checkBerat(berat));

insert(hashTable, id, nama, usia, gender, tinggi, berat);

printf("\033[0;32m");

printf("\n\t\t\t\tData berhasil ditambahkan!\n\n");

printf("\033[0m");

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 2:{

system("cls");

header();

int id;

printf("\n\t\t\t\tMasukkan ID untuk diupdate: ");

scanf("%d", &id);

update(hashTable, id);

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 3:{

system("cls");

header();

int id;

printf("\n\t\t\t\tMasukkan ID untuk dicari: ");

scanf("%d", &id);

struct Node\* result = searchID(hashTable, id);

if(result != NULL){

printf("\033[0;32m");

printf("\n\t\t\t\tData ditemukan!\n");

printf("\033[0m");

printf("\n\t\t\t\tID\t\t: %d\n", result->id);

printf("\t\t\t\tNama\t\t: %s\n", result->nama);

printf("\t\t\t\tUsia\t\t: %d tahun\n", result->usia);

printf("\t\t\t\tGender\t\t: %c\n", result->gender);

printf("\t\t\t\tTinggi Badan\t: %.2f cm\n", result->tinggi);

printf("\t\t\t\tBerat Badan\t: %.2f kg\n\n", result->berat);

} else{

printf("\033[0;31m");

printf("\n\t\t\t\tData tidak ditemukan.\n");

printf("\033[0m");

}

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 4:{

system("cls");

header();

printf("\n\t\t\tData Balita:\n");

viewData(hashTable);

printf("\n");

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 5:{

system("cls");

header();

int id;

printf("\n\t\t\t\tMasukkan ID untuk dihapus: ");

scanf("%d", &id);

delete(hashTable, id);

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 6:{

system("cls");

header();

printf("\n\t\t\t\tHasil Rata-Rata Data:\n");

meanData(hashTable);

printf("\n");

printf("\n\n\n\t\t\t\t Tekan tombol enter untuk lanjut.....");

getch();

break;

}

case 7:{

system("cls");

header();

printf("\n\n\n\t\t\t\tThank you!\n\n\n");

break;

}

default:{

printf("\n\t\t\t\tInvalid choice!\n\n");

break;

}

}

}while(choice != 7);

return 0;

}

**DAFTAR PUSTAKA**

1. Jodi, M. R. D. (2020). Algoritma dan struktur data. *Jurnal Fakultas Komputer*, 4—5.
2. Syahrir, M., & Fatimatuzzahra. (2020). Integrasi pendekatan metode custom hashing dan data partitioning untuk mempercepat proses pencarian frekuensi item-set pada algoritma apriori. *Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *20*(1), 149—150. <https://doi.org/10.30812/matrik.v20i1.833>
3. Dharma, A., & Syahputra, H. H. (2017). Aplikasi pembelajaran linked list berbasis mobile learning. *Riau Journal Of Computer Science*, *4*(1), 2—4.
4. Jamaludin. (2022). *Konsep algoritma dan pemrograman: Mengenai konsep dasar dan praktis dalam bahasa pascal dan C*. Indie Press. Page 17—18.
5. Watrianthos, R. (2014). Struktur bahasa pemrograman pascal atau bahasa C. *Jurnal Ilmiah AMIK Labuhan BatuI*, *2*(1), 16.