

**PROJECT ALGORITMA PEMROGRAMAN I**  
**SISTEM MANAJEMEN AKUN BANK**  
**INTERAKTIF DALAM C++**



---

**SOFTWARE ARCHITECTURE  
AND QUALITY LABORATORY**

ANGGOTA	:	Zain Akbar Rizkia	202531091
	:	Nasya Healthy Thresia Sianturi	202531095
	:	Ahmaddillah	202531102
	:	Nafiratul Khasanah	202531109
	:	Kristoforus Noventa	202531112
	:	Muhammad Lutfi Ramadhan	202531115
KELAS	:	C	
DOSEN	:	Meilia Nur Indah Susanti, ST., M.Kom	
ASISTEN	:	Muhammad Khasyi Athallah	
	:	Rafi Indra Pramudhito Z.	
	:	Arif Rizki Kurniadi	
	:	Gangsar Anjasmoro	

**FAKULTAS TELEMATIKA ENERGI**  
**TEKNIK INFORMATIKA**  
**INSTITUT TEKNOLOGI PLN – JAKARTA**

**2025**

**DAFTAR ISI**

Daftar Isi .....	ii
Daftar Gambar .....	iv
1. PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Tujuan .....	1
1.3. Manfaat .....	1
1.4. Batasan Masalah .....	2
2. LANDASAN TEORI .....	3
2.1. Struktur Data .....	3
2.2. Array dan Indeksasi .....	3
2.3. Fungsi dan Parameter Referensi .....	4
2.4. Stream Input-Output .....	5
2.5. Kontrol Alur Program .....	6
2.6. Algoritma Pencarian Linear .....	7
3. PERANCANGAN .....	8
3.1. Analisis Masalah .....	8
3.1.1. Data yang Dibutuhkan .....	8
3.1.2. Output yang Diinginkan .....	8
3.1.3. Proses Data .....	8
3.2. Flowchart .....	10
3.2.1. Flowchart Pembuatan Akun .....	11
3.2.2. Flowchart Setor Tunai .....	12
3.2.3. Flowchart Tarik Tunai .....	13
3.2.4. Flowchart Tampil Akun .....	14
3.2.5. Flowchart Ubah Akun .....	15
3.2.6. Flowchart Hapus Akun .....	16
3.2.7. Flowchart Tampil Semua Akun .....	17

3.3. Algoritma .....	18
3.3.1. Pseudocode Utama Program .....	18
3.3.2. Pseudocode Fungsi-Fungsi Pendukung .....	18
3.3.2.1. Pseudocode initialize_account .....	18
3.3.2.2. Pseudocode find_available_slot .....	19
3.3.2.3. Pseudocode find_account_idx .....	19
3.3.2.4. Pseudocode deposit .....	20
3.3.2.5. Pseudocode withdraw .....	20
4. HASIL DAN PEMBAHASAN .....	21
4.1. Implementasi Struktur Data BankAccount .....	21
4.2. Implementasi Array dan Sistem Indeksasi .....	21
4.3. Implementasi Fungsi dengan Parameter Referensi .....	22
4.4. Implementasi Stream Input-Output .....	23
4.5. Implementasi Kontrol Alur dengan Switch-Case .....	24
4.6. Implementasi Algoritma Pencarian Linear .....	25
4.7. Implementasi Fungsi Validasi dan Utilitas .....	26
5. KESIMPULAN .....	28
Daftar Pustaka .....	29
Lampiran A .....	A-1

**DAFTAR GAMBAR**

Gambar 2.1 Struktur data BankAccount. ....	3
Gambar 2.2 Array BankAccount accounts[MAX_ACCOUNTS]. ....	4
Gambar 2.3 Contoh Fungsi dengan Parameter Referensi. ....	5
Gambar 2.4 Ekstraksi data dari stream menggunakan operator >> (cin). ....	5
Gambar 2.5 Insersi data ke stream menggunakan operator << (cout). ....	5
Gambar 2.6 Penggunaan cin.ignore() dan getline(). ....	6
Gambar 2.7 Struktur Kontrol Alur switch-case. ....	6
Gambar 2.8 Implementasi Algoritma Pencarian Linear. ....	7
Gambar 3.9 Diagram Alur Utama .....	10
Gambar 3.10 Diagram Alur Pembuatan Akun .....	11
Gambar 3.11 Diagram Alur Setor Tunai .....	12
Gambar 3.12 Diagram Alur Tarik Tunai .....	13
Gambar 3.13 Diagram Alur Tampil Akun .....	14
Gambar 3.14 Diagram Alur Ubah Akun .....	15
Gambar 3.15 Diagram Alur Hapus Akun .....	16
Gambar 3.16 Diagram Alur Tampil Semua Akun .....	17
Gambar 3.17 Pseudocode Utama Program. ....	18
Gambar 3.18 Pseudocode Fungsi initialize_account. ....	18
Gambar 3.19 Pseudocode Fungsi find_available_slot. ....	19
Gambar 3.20 Pseudocode Fungsi find_account_idx. ....	19
Gambar 3.21 Pseudocode Fungsi deposit. ....	20
Gambar 3.22 Pseudocode Fungsi withdraw. ....	20
Gambar 4.23 Implementasi Struktur Data BankAccount. ....	21
Gambar 4.24 Implementasi Array Statis untuk Akun Bank. ....	21
Gambar 4.25 Akses Elemen Array Menggunakan Subscript Operator. ....	22
Gambar 4.26 Iterasi Array untuk Inisialisasi Akun. ....	22
Gambar 4.27 Implementasi Fungsi deposit() dengan Parameter Referensi. ....	23

Gambar 4.28 Implementasi Fungsi <code>modify_account()</code> .....	23
Gambar 4.29 Deklarasi Header <code>&lt;iostream&gt;</code> .....	24
Gambar 4.30 Penggunaan Operator Ekstraksi <code>cin &gt;&gt;</code> .....	24
Gambar 4.31 Penggunaan <code>getline()</code> untuk Input String dengan Spasi. ....	24
Gambar 4.32 Penggunaan Operator Insersi <code>cout &lt;&lt;</code> .....	24
Gambar 4.33 Implementasi <code>switch-case</code> untuk Menu Utama. ....	25
Gambar 4.34 Implementasi Fungsi <code>find_account_idx()</code> (Pencarian Linear). ....	25
Gambar 4.35 Penggunaan Fungsi <code>find_account_idx()</code> .....	26
Gambar 4.36 Implementasi Fungsi <code>is_account_empty()</code> .....	26
Gambar 4.37 Implementasi Fungsi <code>find_available_slot()</code> .....	27
Gambar 4.38 Implementasi Fungsi <code>initialize_account()</code> .....	27

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Sistem manajemen akun bank merupakan aplikasi fundamental dalam dunia perbankan digital. Aplikasi ini dirancang untuk mengelola informasi akun pelanggan, melakukan transaksi setoran dan penarikan dana, serta menampilkan data akun yang tersimpan dalam sistem.

Pengembangan sistem manajemen akun bank menggunakan bahasa pemrograman C++ dengan paradigma pemrograman procedural memberikan fondasi yang kuat dalam memahami konsep pemrograman tingkat pemula. Melalui proyek ini, pembaca akan mempelajari penggunaan struktur data (struct), array, fungsi, sistem input-output berbasis stream, dan kontrol alur program menggunakan switch-case.

Dengan membatasi penggunaan library hanya pada `iostream` dan `cstdlib`, proyek ini menekankan pemahaman mendalam terhadap mekanisme dasar C++ tanpa bergantung pada fitur-fitur advanced atau template library. Hal ini memastikan bahwa setiap aspek program dapat dipahami secara menyeluruh dari perspektif algoritma dan logika pemrograman.

### 1.2 Tujuan

1. Memahami dan menerapkan struktur data `struct` dalam C++ untuk menyimpan informasi akun bank yang terdiri dari nomor akun, nama pemegang, alamat, jenis akun, dan saldo.
2. Menerapkan array sebagai kontainer untuk menyimpan hingga lima akun bank secara simultan.
3. Mengimplementasikan operasi dasar perbankan meliputi pembuatan akun baru, setoran, penarikan, pencarian, modifikasi, dan penghapusan akun.
4. Menerapkan fungsi dengan parameter referensi untuk memanipulasi data akun secara efisien.
5. Mengembangkan menu interaktif berbasis command-line interface (CLI) dengan sistem pilihan menggunakan `switch-case`.

### 1.3 Manfaat

1. Pemahaman konseptual tentang abstraksi data melalui struktur dan cara data diorganisir dalam memori.
2. Praktik penulisan fungsi yang reusable dan terstruktur untuk operasi spesifik.
3. Pengalaman dalam menangani input user dengan validasi dan penanganan kondisi error.
4. Kemampuan untuk merancang algoritma pencarian linear untuk menemukan akun dalam array.

5. Fondasi dasar untuk pengembangan aplikasi berbasis data yang lebih kompleks.

#### 1.4 Batasan Masalah

1. Kapasitas maksimal sistem hanya dapat menampung lima akun bank, didefinisikan oleh konstanta MAX\_ACCOUNTS.
2. Data akun hanya tersimpan dalam memori RAM selama program berjalan; tidak ada mekanisme penyimpanan data permanen ke file.
3. Sistem menggunakan array statis, bukan struktur data dinamis seperti linked list atau vector.
4. Nomor akun harus bernilai unik; sistem tidak menerapkan mekanisme otomatis untuk menghasilkan nomor akun.
5. Jenis akun dibatasi pada dua pilihan: ‘s’ untuk tabungan (savings) dan ‘c’ untuk giro (current).
6. Library yang digunakan terbatas pada `iostream` dan `cstdlib`; tidak menggunakan library tambahan seperti `string` dalam definisi struct.
7. Sistem tidak menerapkan enkripsi atau keamanan data; hanya validasi dasar pada nilai masukan.

## BAB II

### LANDASAN TEORI

#### 2.1 Struktur Data

Struktur data dalam C++ didefinisikan menggunakan keyword `struct` untuk mengelompokkan beberapa variabel dari tipe data yang berbeda menjadi satu kesatuan logis. Menurut dokumentasi (cppreference.com, 2025a) dan (GeeksforGeeks, n.d.-a), struct dalam C++ merupakan user-defined type yang mengkombinasikan multiple members dengan tipe berbeda dalam suatu composite type. Ketika struct dideklarasikan dengan body, pernyataan tersebut sekaligus merupakan definisi yang mengalokasikan template tipe dalam sistem type C++.

Dalam proyek ini, struktur `BankAccount` digunakan untuk merepresentasikan data tunggal akun:

```

26
25  struct BankAccount {
24    int accno;      // Nomor akun, tipe integer
23    string name;   // Nama pemegang akun
22    string address; // Alamat pemegang akun
21    char actype;   // Jenis akun: 's' atau 'c'
20    float amount;  // Saldo akun dalam rupiah
19
18  };

```

Gambar 2.1 Struktur data `BankAccount`.

Setiap anggota struct memiliki offset tetap dari awal struktur di memori. Dalam bahasa C++, akses terhadap member struktur dilakukan dengan operator dot (.) ketika menggunakan variabel struct langsung, dan operator arrow (->) ketika menggunakan pointer ke struct.

#### 2.2 Array dan Indeksasi

Array merupakan koleksi elemen tipe data yang sama, tersimpan dalam lokasi memori yang berdampingan (contiguous) (GeeksforGeeks, n.d.-b). Dalam C++, deklarasi array menggunakan notasi `type name[size]`, di mana `size` merupakan konstanta yang menentukan jumlah maksimal elemen.

Menurut (cppreference.com, 2025b) dan (Microsoft Learn, n.d.), subscript operator [] digunakan untuk mengakses elemen array pada indeks tertentu. Operator ini didefinisikan secara identik dengan ekspresi pointer arithmetic: `E1[E2]` secara fundamental equivalent dengan `*((E1)+(E2))`. Indeksasi array dalam C++ dimulai dari 0, sehingga array dengan size `n` memiliki indeks valid dari 0 hingga `n-1`.

Dalam proyek, array BankAccount accounts[MAX\_ACCOUNTS] menyimpan hingga lima akun:

```
5  const int MAX_ACCOUNTS = 5;
6
7  struct BankAccount {
8      int accno;
9      string name;
10     string address;
11     char actype;
12     float amount;
13 };
14
15 BankAccount accounts[MAX_ACCOUNTS];
16
17 int main() {
18     // Akses elemen pertama
19     accounts[0].accno = 1001;
20     accounts[0].name = "Jane Doe";
21
22     // Iterasi seluruh array
23     for (int i = 0; i < MAX_ACCOUNTS; ++i) {
24         cout << "Account " << i
25             << " No: " << accounts[i].accno << endl;
26     }
27
28     return 0;
29 }
```

Gambar 2.2 Array BankAccount accounts[MAX\_ACCOUNTS].

Subscript operator pada array mengembalikan lvalue apabila array adalah lvalue, memungkinkan baik pembacaan maupun penulisan nilai pada posisi tersebut.

### 2.3 Fungsi dan Parameter Referensi

Fungsi dalam C++ merupakan blok kode yang dapat dipanggil berulang dengan identitas dan tipe parameter tertentu. Dokumentasi (cppreference.com, 2025c) menyatakan bahwa function declaration memperkenalkan nama fungsi dan tipe fungsi, sedangkan function definition mengasosiasikan nama/tipe dengan body fungsi.

Parameter referensi dalam C++ (reference parameter) memungkinkan fungsi memodifikasi argumen asli tanpa membuat salinan. Notasi parameter referensi menggunakan symbol ampersand (&):

```

12
13 void deposit(BankAccount &acc) {
14     float d;
15     cout << "\nMasukkan jumlah untuk disetor = ";
16     cin >> d;
17     if (d > 0) {
18         acc.amount += d; // Modifikasi saldo langsung
19         cout << "Setoran berhasil. Saldo Baru = "
20             << acc.amount << endl;
21     } else {
22         cout << "Jumlah setoran harus positif."
23             << endl;
24     }
25 }
26

```

Gambar 2.3 Contoh Fungsi dengan Parameter Referensi.

Parameter referensi berbeda dengan parameter pointer, karena referensi tidak memerlukan dereference dan tidak dapat bernilai null. Fungsi yang menerima referensi dapat secara langsung mengakses dan memodifikasi object asli melalui reference tersebut tanpa overhead pembuatan salinan (copy).

#### 2.4 Stream Input-Output

Stream dalam C++ merupakan abstraksi untuk komunikasi data antara program dan external device. Menurut dokumentasi (cppreference.com, 2025d) dan (GeeksforGeeks, n.d.-c), iostream header mendefinisikan object global `cin` dengan tipe `istream` untuk membaca input dari keyboard, dan `cout` dengan tipe `ostream` untuk menampilkan output ke layar.

Ekstraksi data dari stream menggunakan operator `>>`: (W3Schools, n.d.)

```

3
4 int main() {
5     int accno;
6     cout << "Masukkan nomor akun: ";
7     cin >> accno; // Baca integer dari input stream
8     cout << "Nomor akun yang dimasukkan: " << accno
9         << endl;
10    return 0;
11 }

```

Gambar 2.4 Ekstraksi data dari stream menggunakan operator `>>` (`cin`).

Insersi data ke stream menggunakan operator `<<`:

```

3
4 int main() {
5     float amount = 500.0;
6     cout << "Saldo: " << amount << endl;
7     return 0;
8 }

```

Gambar 2.5 Insersi data ke stream menggunakan operator << (cout).

Ketika menggunakan `cin >>` diikuti `getline()`, diperlukan `cin.ignore()` untuk membersihkan karakter newline yang tertinggal di input buffer:

```

4
5 int main() {
6     int age;
7     cout << "Enter your age: ";
8     cin >> age;
9     cin.ignore(10000, '\n'); // Bersihkan buffer
10    string name;
11    cout << "Enter your name: ";
12    getline(cin, name); // Baca string dengan spasi
13    cout << "Name: " << name << ", Age: " << age
14        << endl;
15    return 0;
16 }
```

Gambar 2.6 Penggunaan `cin.ignore()` dan `getline()`.

## 2.5 Kontrol Alur Program

Kontrol alur program menggunakan `switch-case` memungkinkan eksekusi kode berbeda berdasarkan nilai satu expression. Struktur `switch-case` dalam proyek ini merespons pilihan menu user:

```

3
4 int main() {
5     int choice;
6     cout << "Enter a choice (1 or 2): ";
7     cin >> choice;
8
9     switch (choice) {
10    case 1:
11        cout << "You chose 1." << endl;
12        break;
13    case 2:
14        cout << "You chose 2." << endl;
15        break;
16    default:
17        cout << "Pilihan tidak valid." << endl;
18    }
19    return 0;
20 }
```

Gambar 2.7 Struktur Kontrol Alur `switch-case`.

Statement `break` penting untuk menghentikan eksekusi dan keluar dari `switch-case`. Tanpa `break`, eksekusi akan berlanjut ke case berikutnya (fall-through behavior).

## 2.6 Algoritma Pencarian Linear

Pencarian linear merupakan algoritma dasar yang memeriksa setiap elemen dalam array secara berurutan sampai menemukan elemen target atau mencapai akhir array. Dalam proyek, fungsi `find_account_idx()` mengimplementasikan algoritma ini untuk mencari akun berdasarkan nomor akun:

```
9
10 int find_account_idx(int acc_num,
11                      const BankAccount accounts[],
12                      int size) {
13     for (int i = 0; i < size; ++i) {
14         if (accounts[i].accno == acc_num) {
15             return i; // Ditemukan pada indeks i
16         }
17     }
18     return -1; // Tidak ditemukan
19 }
20 }
```

Gambar 2.8 Implementasi Algoritma Pencarian Linear.

Kompleksitas waktu pencarian linear adalah  $O(n)$ , di mana  $n$  adalah jumlah elemen yang diperiksa. Algoritma ini cukup efisien untuk dataset berukuran kecil seperti lima akun, namun untuk dataset lebih besar diperlukan struktur data dan algoritma yang lebih sophisticated.

## **BAB III**

### **PERANCANGAN**

#### **3.1 Analisis Masalah**

##### **3.1.1 Data yang Dibutuhkan**

Sistem manajemen akun bank memerlukan data berikut untuk setiap akun:

1. Nomor akun (integer): identifier unik untuk setiap akun, dipilih user saat pembuatan akun.
2. Nama pemegang (string): nama lengkap pemegang akun, dapat mengandung spasi dan karakter alphabetic.
3. Alamat (string): lokasi tempat tinggal pemegang akun, dapat mengandung spasi, karakter numerik, dan punctuation.
4. Jenis akun (char): klasifikasi akun sebagai ‘s’ untuk tabungan atau ‘c’ untuk giro.
5. Saldo (float): jumlah dana yang tersimpan dalam akun, dinyatakan dalam rupiah dengan dua desimal.

Data-data tersebut dienkapsulasi dalam struktur `BankAccount` dan disimpan dalam array dengan kapasitas maksimal lima akun.

##### **3.1.2 Output yang Diinginkan**

Sistem dirancang untuk menghasilkan output berikut:

1. Menu interaktif yang menampilkan delapan pilihan operasi kepada user.
2. Konfirmasi keberhasilan operasi dengan pesan yang jelas dan informatif.
3. Pesan error ketika operasi gagal, misalnya saldo tidak mencukupi atau akun tidak ditemukan.
4. Display detail akun yang menampilkan semua data akun dalam format terstruktur.
5. Informasi saldo setelah setiap transaksi setoran atau penarikan.

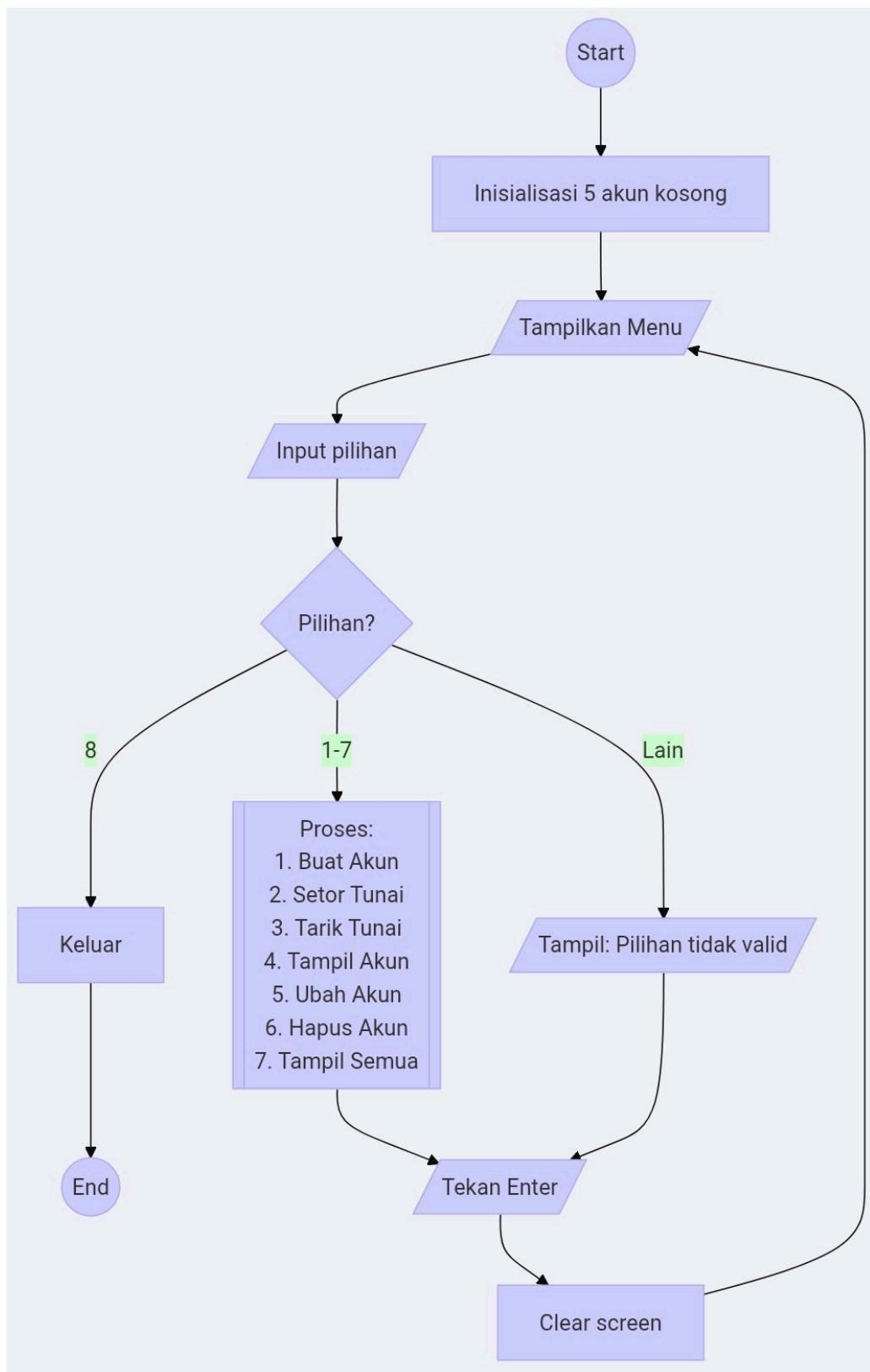
##### **3.1.3 Proses Data**

Proses dalam sistem terdiri dari:

1. Inisialisasi: Semua akun diinisialisasi dengan nilai default (nomor akun 0 menandakan slot kosong).
2. Pembuatan akun: User memasukkan data akun baru, sistem menyimpan pada slot pertama yang kosong.

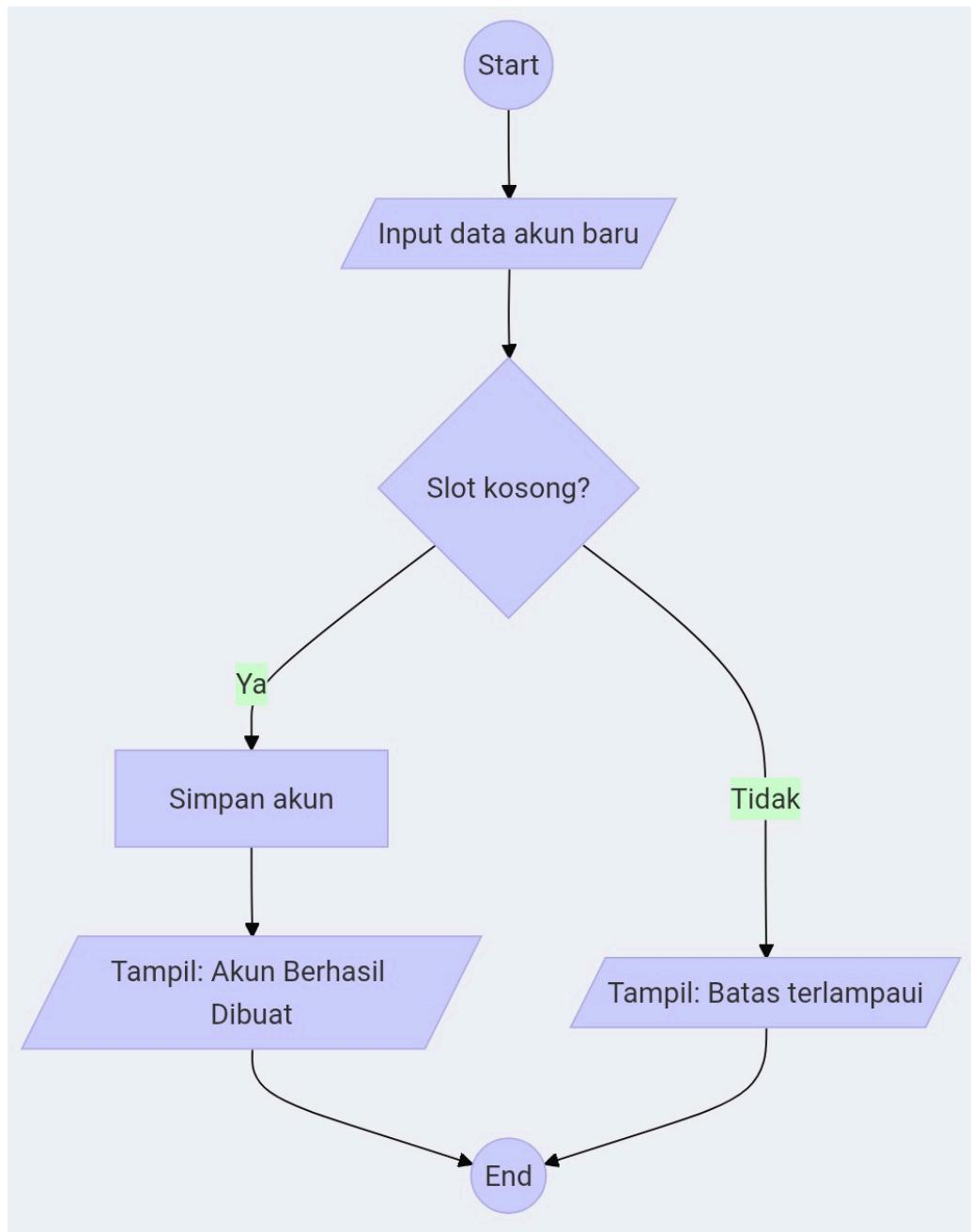
3. Pencarian akun: Sistem melakukan pencarian linear untuk menemukan akun berdasarkan nomor akun.
4. Transaksi setoran: Saldo ditambahkan dengan jumlah yang disetorkan, dengan validasi jumlah positif.
5. Transaksi penarikan: Saldo dikurangi dengan jumlah yang ditarik, dengan validasi saldo cukup dan jumlah positif.
6. Modifikasi akun: User dapat mengubah nama, alamat, dan jenis akun (tidak termasuk nomor akun dan saldo).
7. Penghapusan akun: Slot akun dikosongkan dengan reinitialisasi nilai-nilai-nya.
8. Tampilan semua akun: Sistem mengiterasi seluruh array dan menampilkan akun yang valid (nomor akun bukan 0).

### 3.2 Flowchart



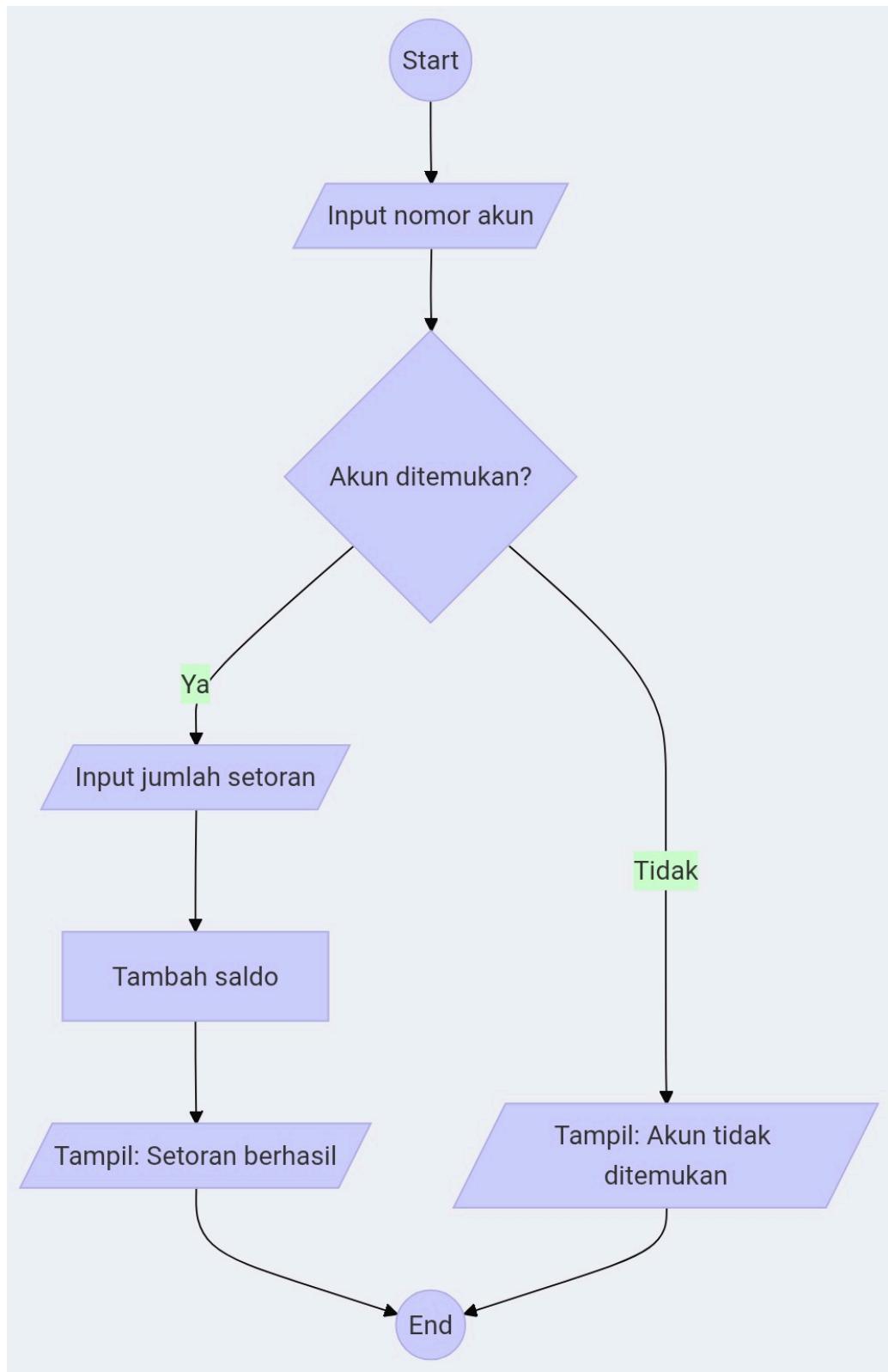
Gambar 3.9 Diagram Alur Utama

### 3.2.1 Flowchart Pembuatan Akun



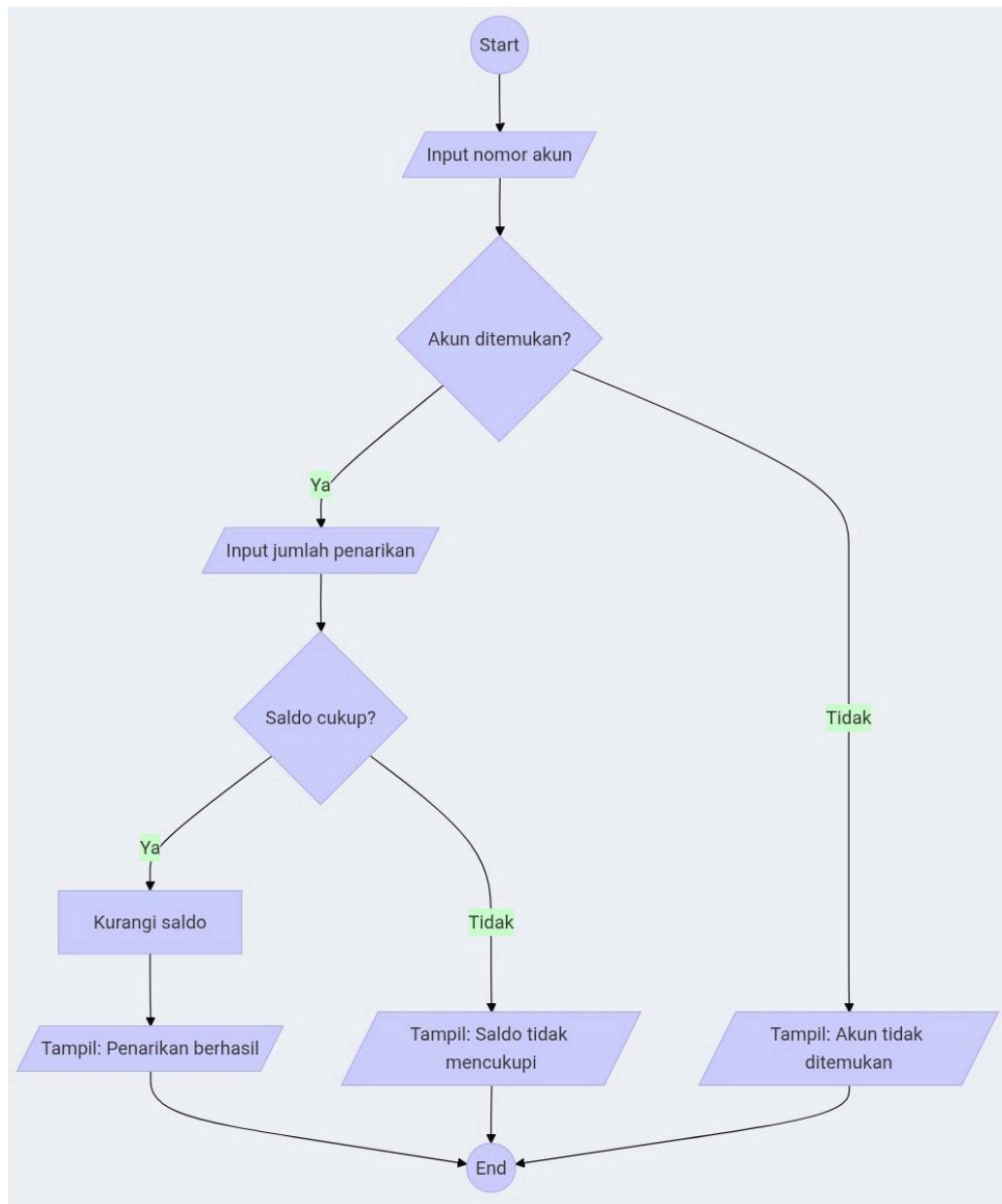
Gambar 3.10 Diagram Alur Pembuatan Akun

### 3.2.2 Flowchart Setor Tunai



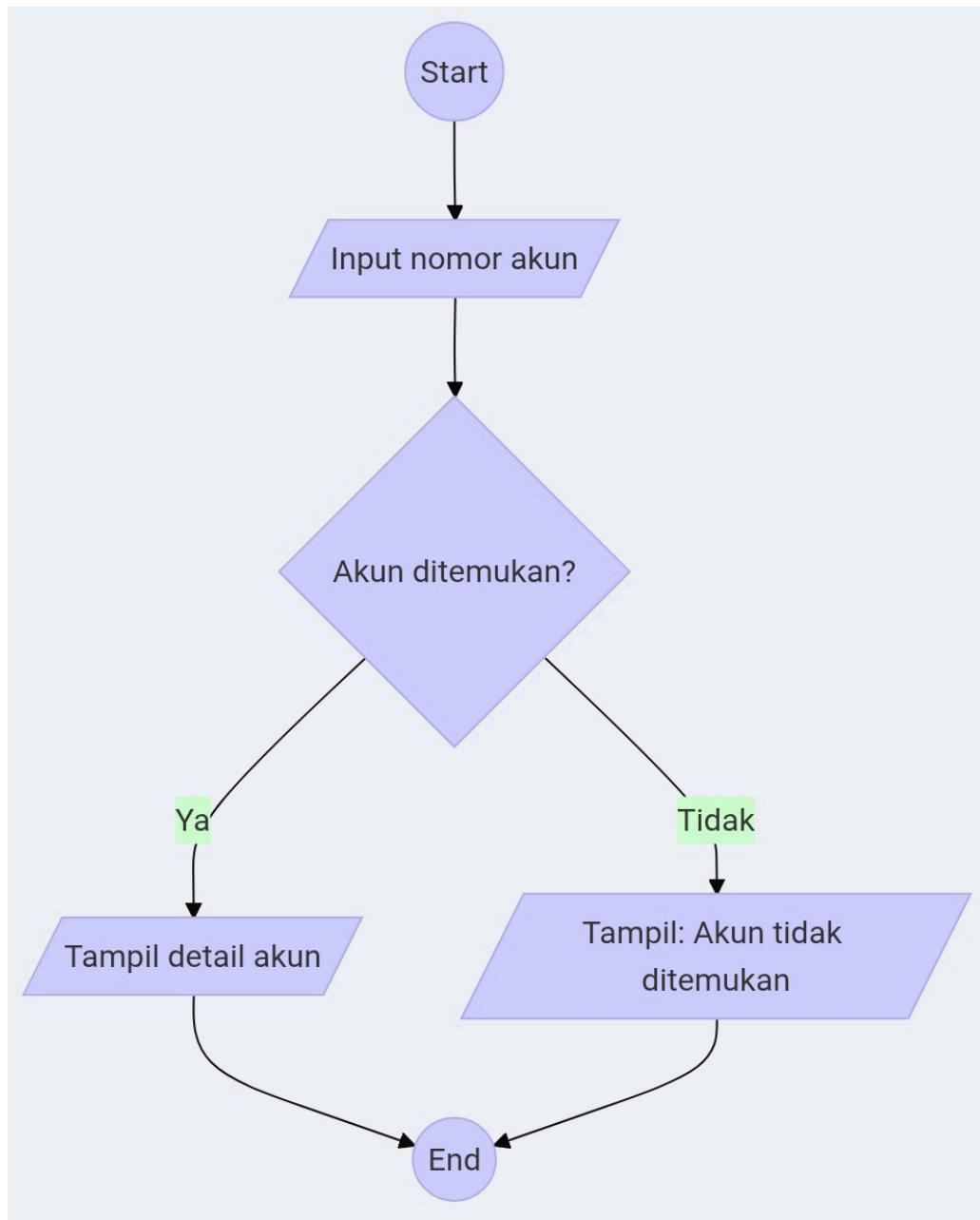
Gambar 3.11 Diagram Alur Setor Tunai

### 3.2.3 Flowchart Tarik Tunai



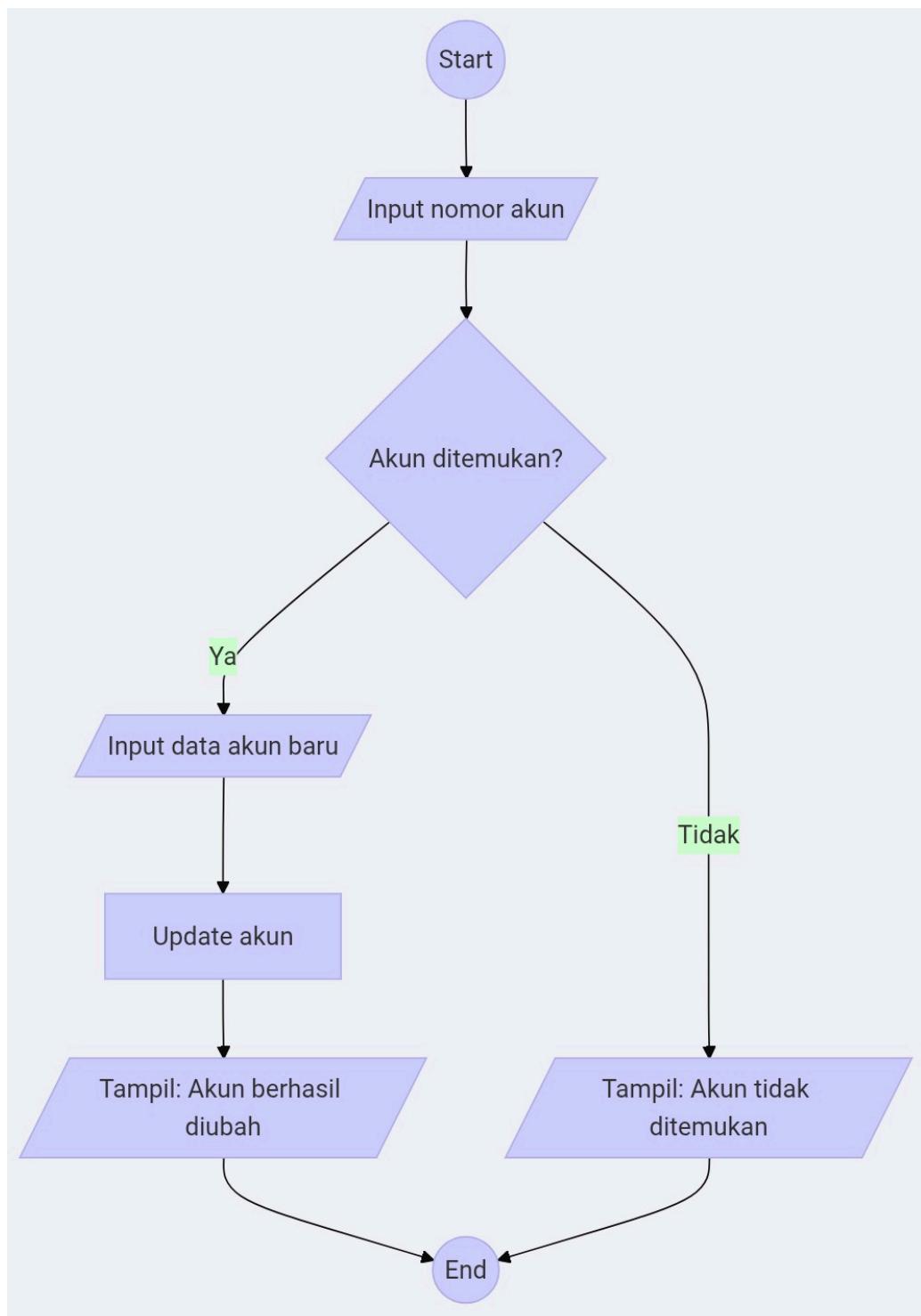
Gambar 3.12 Diagram Alur Tarik Tunai

### 3.2.4 Flowchart Tampil Akun



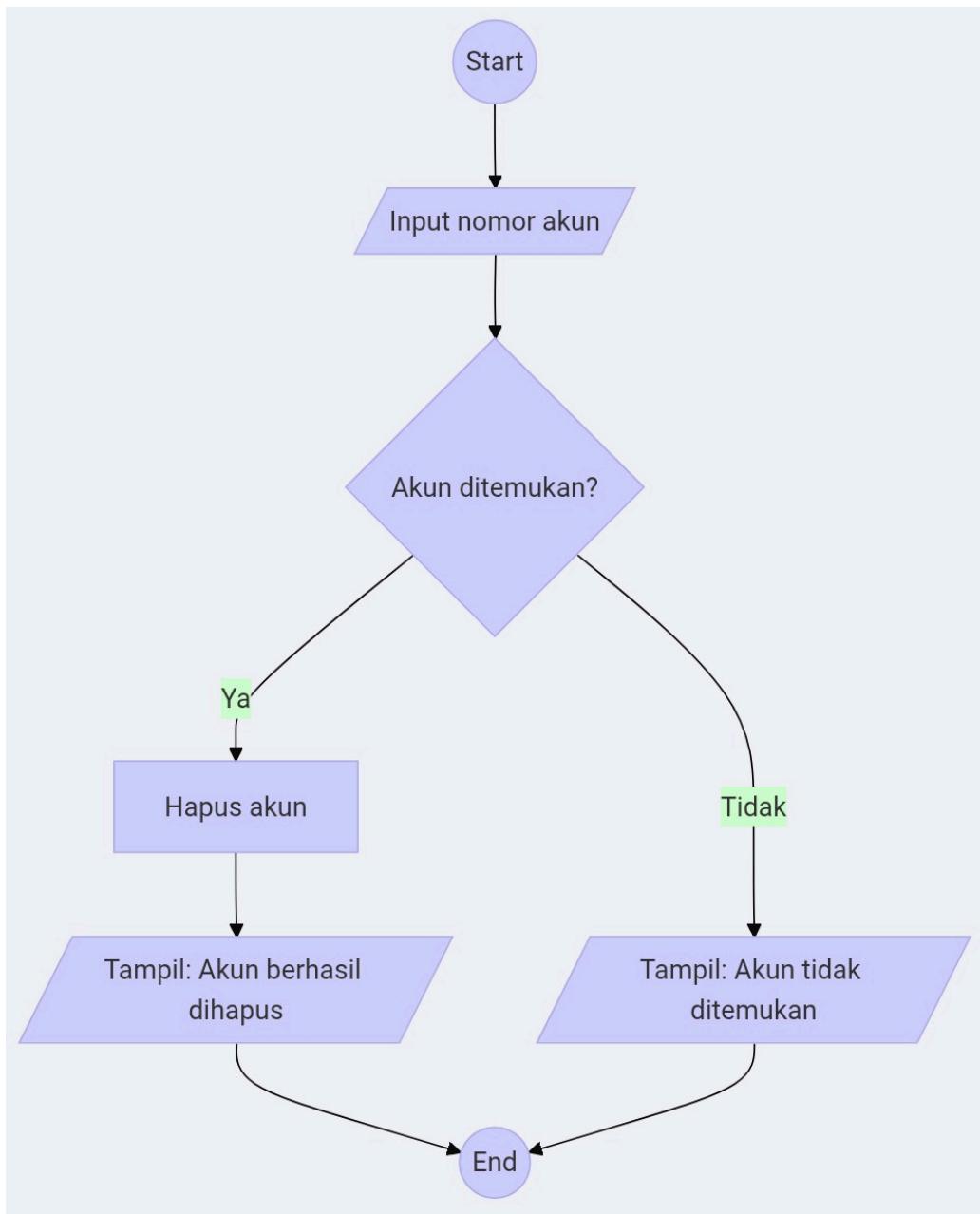
Gambar 3.13 Diagram Alur Tampil Akun

### 3.2.5 Flowchart Ubah Akun



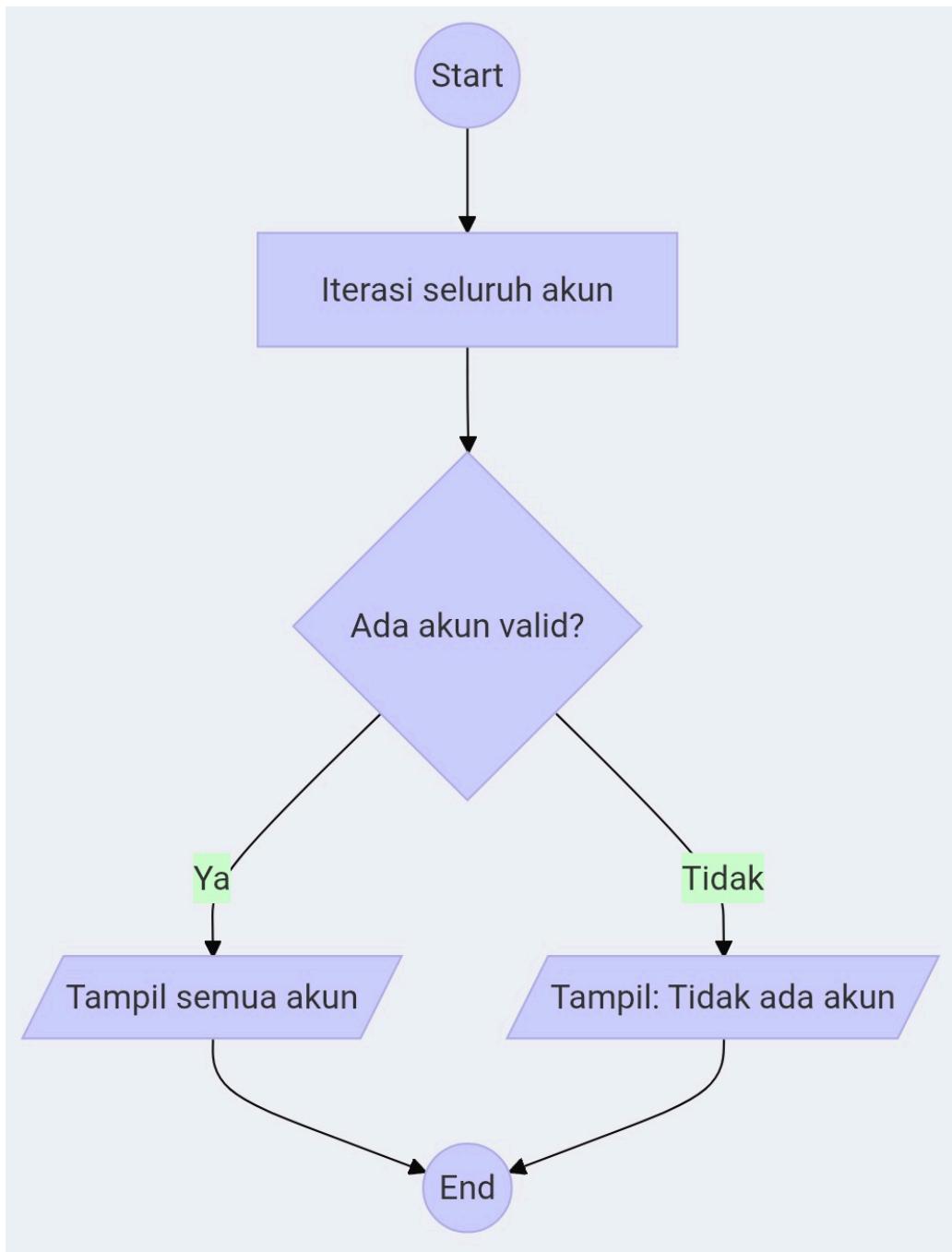
Gambar 3.14 Diagram Alur Ubah Akun

### 3.2.6 Flowchart Hapus Akun



Gambar 3.15 Diagram Alur Hapus Akun

### 3.2.7 Flowchart Tampil Semua Akun



Gambar 3.16 Diagram Alur Tampil Semua Akun

### 3.3 Algoritma

#### 3.3.1 Pseudocode Utama Program

```

9:04 PM >_
 0 0 2 KELOMPOK 6 AP 21:04:35 | 1 2 3
1 ALGORITHM Main
2 INPUT: User menu selection
3 OUTPUT: Menu-driven Interface for bank
4 account management
5
6 BEGIN
7   DECLARE accounts AS ARRAY[5] OF BankAccount
8   DECLARE choice AS INTEGER
9   DECLARE acc_num AS INTEGER
10  DECLARE acc_idx AS INTEGER
11
12 FOR i = 0 TO 4 DO
13   CALL initialize_account(accounts[i])
14 ENFOR
15
16 WHILE true DO
17   CALL system("clear")
18   CALL display_menu()
19   READ choice FROM user
20
21 SWITCH choice
22 CASE 1:
23   acc_idx = CALL find_available_slot(
24     accounts, 5)
25   IF acc_idx != -1 THEN
26     CALL create_new_account(
27       acc_num, accounts[acc_idx])
28     PRINT "Akun Berhasil Dibuat..."
29   ELSE
30     PRINT "Batas akun terlampaui"
31   END IF
32
33 CASE 2:
34   PRINT "Masukkan no. akun: "
35   READ acc_num FROM user
36   acc_idx = CALL find_account_idx(
37     acc_num, accounts, 5)
38   IF acc_idx != -1 THEN
39     CALL deposit(accounts[acc_idx])
40   ELSE
41     PRINT "Akun tidak ditemukan"
42   END IF
43
44 CASE 3:
45   PRINT "Masukkan no. akun: "
46   READ acc_num FROM user
47   acc_idx = CALL find_account_idx(
48     acc_num, accounts, 5)
49   IF acc_idx != -1 THEN
50     CALL withdraw(accounts[acc_idx])
51   ELSE
52     PRINT "Akun tidak ditemukan"
53   END IF
54
55 CASE 4:
56   PRINT "Masukkan no. akun: "
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99

```

The pseudocode describes a menu-driven banking application. It initializes accounts, handles user input for account selection, and provides options for creating new accounts, depositing, withdrawing, and displaying account details. Error handling is included for invalid account numbers and full account limits.

Gambar 3.17 Pseudocode Utama Program.

#### 3.3.2 Pseudocode Fungsi-Fungsi Pendukung

##### 3.3.2.1 Pseudocode initialize\_account

```

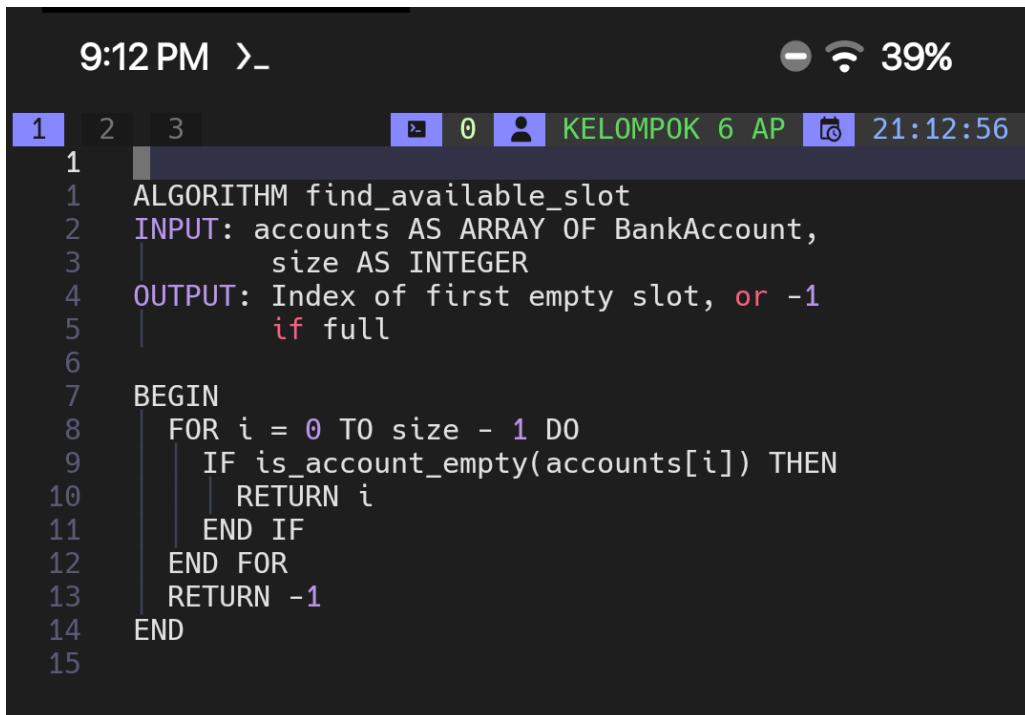
9:08 PM >_
 0 0 2 KELOMPOK 6 AP 21:08:47 | 1 2 3
1 ALGORITHM initialize_account
2 INPUT: acc AS REFERENCE TO BankAccount
3 OUTPUT: Initialized account with default
4 values
5
6 BEGIN
7   acc.accno = 0
8   acc.name = ""
9   acc.address = ""
10  acc.actype = '\0'
11  acc.amount = 0.0
12 END
13

```

This pseudocode defines a function named initialize\_account that takes a reference to a BankAccount object. It sets the account number to 0, initializes the name and address fields as empty strings, and sets the account type to '\0'. The initial amount is set to 0.0.

Gambar 3.18 Pseudocode Fungsi initialize\_account.

### 3.3.2.2 Pseudocode find\_available\_slot



The screenshot shows a mobile phone screen with a dark theme. At the top, the status bar displays the time (9:12 PM), signal strength, battery level (39%), and a timestamp (21:12:56). The main area contains pseudocode for an algorithm named 'find\_available\_slot'. The code uses a combination of numbers and letters for line numbers. It defines inputs for an array of bank accounts and a size, and outputs the index of the first empty slot or -1 if full. The logic involves a FOR loop from i=0 to size-1, checking if each account is empty. If found empty, it returns the index i. If no empty slot is found, it returns -1.

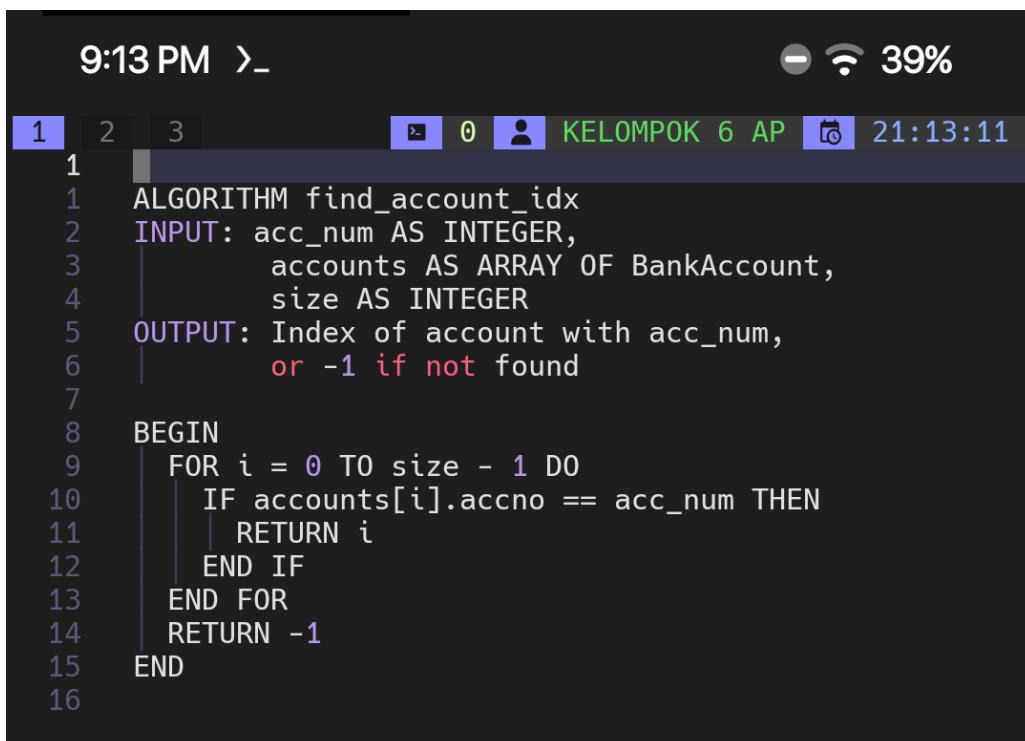
```

1 ALGORITHM find_available_slot
2 INPUT: accounts AS ARRAY OF BankAccount,
3         size AS INTEGER
4 OUTPUT: Index of first empty slot, or -1
5         if full
6
7 BEGIN
8     FOR i = 0 TO size - 1 DO
9         IF is_account_empty(accounts[i]) THEN
10            RETURN i
11        END IF
12    END FOR
13    RETURN -1
14 END
15

```

Gambar 3.19 Pseudocode Fungsi find\_available\_slot.

### 3.3.2.3 Pseudocode find\_account\_idx



The screenshot shows a mobile phone screen with a dark theme. At the top, the status bar displays the time (9:13 PM), signal strength, battery level (39%), and a timestamp (21:13:11). The main area contains pseudocode for an algorithm named 'find\_account\_idx'. The code uses a combination of numbers and letters for line numbers. It defines inputs for an account number, an array of bank accounts, and a size, and outputs the index of the account with the specified account number, or -1 if not found. The logic involves a FOR loop from i=0 to size-1, checking if the account number in accounts[i] matches the input account number. If found matching, it returns the index i. If no matching account is found, it returns -1.

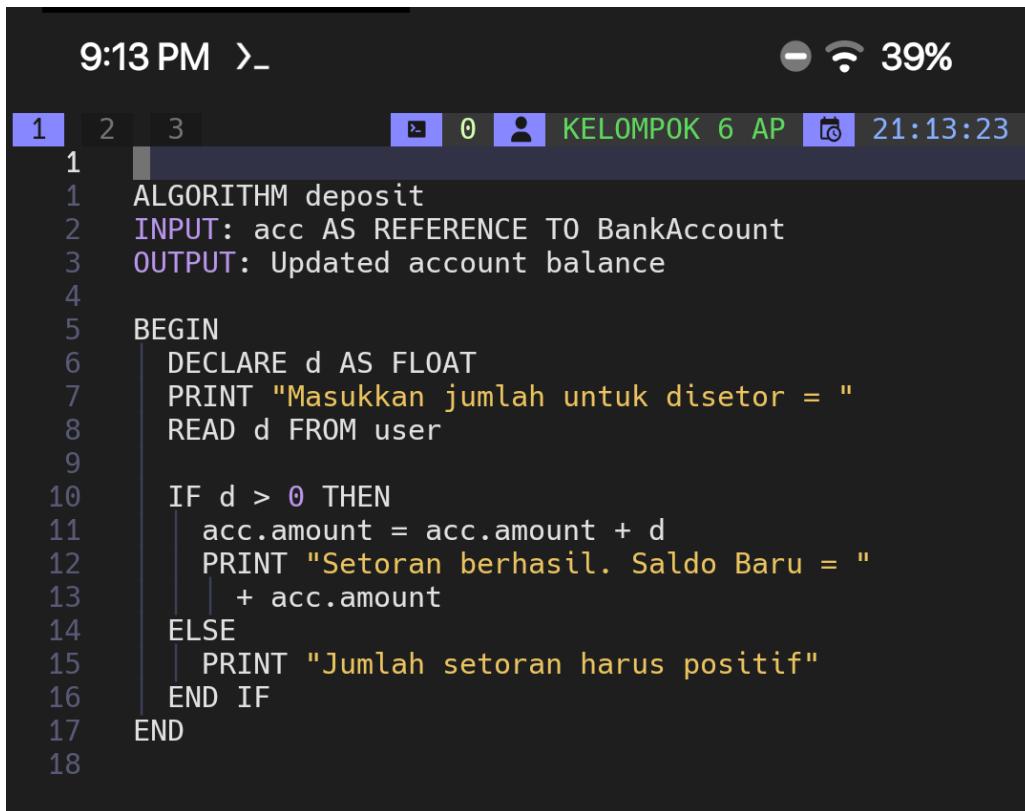
```

1 ALGORITHM find_account_idx
2 INPUT: acc_num AS INTEGER,
3         accounts AS ARRAY OF BankAccount,
4         size AS INTEGER
5 OUTPUT: Index of account with acc_num,
6         or -1 if not found
7
8 BEGIN
9     FOR i = 0 TO size - 1 DO
10        IF accounts[i].accno == acc_num THEN
11            RETURN i
12        END IF
13    END FOR
14    RETURN -1
15 END
16

```

Gambar 3.20 Pseudocode Fungsi find\_account\_idx.

### 3.3.2.4 Pseudocode deposit



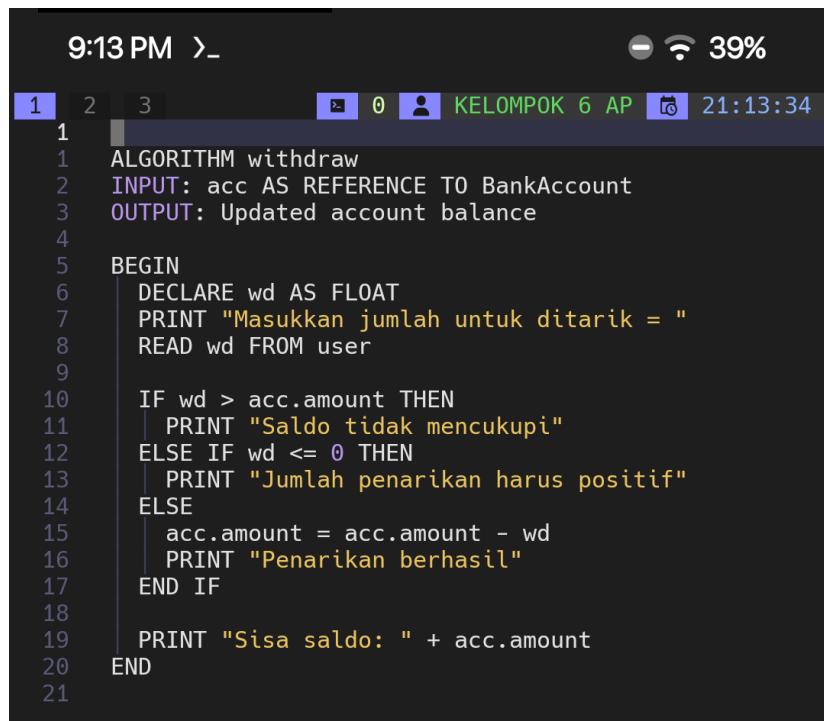
```

9:13 PM >_
2 3 0 KELOMPOK 6 AP 21:13:23
1
1 ALGORITHM deposit
2 INPUT: acc AS REFERENCE TO BankAccount
3 OUTPUT: Updated account balance
4
5 BEGIN
6   DECLARE d AS FLOAT
7   PRINT "Masukkan jumlah untuk disetor = "
8   READ d FROM user
9
10 IF d > 0 THEN
11   acc.amount = acc.amount + d
12   PRINT "Setoran berhasil. Saldo Baru = "
13   + acc.amount
14 ELSE
15   PRINT "Jumlah setoran harus positif"
16 END IF
17
18 END
19
20
21

```

Gambar 3.21 Pseudocode Fungsi deposit.

### 3.3.2.5 Pseudocode withdraw



```

9:13 PM >_
2 3 0 KELOMPOK 6 AP 21:13:34
1
1 ALGORITHM withdraw
2 INPUT: acc AS REFERENCE TO BankAccount
3 OUTPUT: Updated account balance
4
5 BEGIN
6   DECLARE wd AS FLOAT
7   PRINT "Masukkan jumlah untuk ditarik = "
8   READ wd FROM user
9
10 IF wd > acc.amount THEN
11   PRINT "Saldo tidak mencukupi"
12 ELSE IF wd <= 0 THEN
13   PRINT "Jumlah penarikan harus positif"
14 ELSE
15   acc.amount = acc.amount - wd
16   PRINT "Penarikan berhasil"
17 END IF
18
19 PRINT "Sisa saldo: " + acc.amount
20
21 END
22

```

Gambar 3.22 Pseudocode Fungsi withdraw.

## BAB IV

# HASIL DAN PEMBAHASAN

### 4.1 Implementasi Struktur Data BankAccount

Struktur BankAccount merupakan inti dari sistem penyimpanan data. Definisi struktur ini mengelompokkan lima anggota yang mewakili atribut-atribut lengkap satu akun bank:

```

4
5  struct BankAccount {
6    int accno; // Tipe int untuk nomor akun numerik
7    string name; // Tipe string untuk nama pemegang
8    string
9      address; // Tipe string untuk alamat pemegang
10   char actype; // Tipe char untuk jenis akun (s
11     // atau c)
12   float amount; // Tipe float untuk saldo dalam
13     // rupiah
14 };
15

```

Gambar 4.23 Implementasi Struktur Data BankAccount.

Penggunaan `int` untuk nomor akun memastikan identifikasi numerik yang efisien. Tipe `float` untuk saldo memungkinkan representasi nilai dengan presisi desimal, meskipun untuk aplikasi finansial production-grade biasanya menggunakan tipe integer dengan unit cent untuk menghindari floating-point precision issues. Tipe `char` untuk jenis akun menghemat memori dibandingkan menggunakan `string`.

Ketika `struct` dideklarasikan, compiler mengalokasikan template tipe ini, namun tidak mengalokasikan memori untuk data aktual sampai variabel `struct` dideklarasikan. Deklarasi array `BankAccount accounts[MAX_ACCOUNTS]` mengalokasikan memori untuk lima instance struktur secara kontinu dalam memori.

### 4.2 Implementasi Array dan Sistem Indeksasi

Array statis digunakan untuk menyimpan akun-akun dengan kapasitas tetap:

```

4
5  const int MAX_ACCOUNTS = 5;
6
7  struct BankAccount {
8    int accno;
9    string name;
10   string address;
11   char actype;
12   float amount;
13 };
14
15  BankAccount accounts[MAX_ACCOUNTS];
16

```

Gambar 4.24 Implementasi Array Statis untuk Akun Bank.

Konstanta MAX\_ACCOUNTS didefinisikan dengan nilai 5, membentuk batas atas jumlah akun yang dapat disimpan. Array ini dideklarasikan dengan scope global dalam fungsi main(), sehingga seluruh fungsi dapat mengaksesnya.

Akses elemen array menggunakan subscript operator []:

```

10   string address;
11   char acctype;
12   float amount;
13 };
14
15 BankAccount accounts[MAX_ACCOUNTS];
16
17 int main() {
18     accounts[0].accno = 1001; // Akses elemen pertama
19     accounts[4].amount =
20         500000.0; // Akses elemen terakhir
21

```

Gambar 4.25 Akses Elemen Array Menggunakan Subscript Operator.

Subscript operator [] dalam C++ secara definitif equivalent dengan pointer arithmetic. Expresi accounts[i] sebenarnya diterjemahkan menjadi \*(accounts + i), di mana accounts + i menghitung address dengan offset sebesar i \* sizeof(BankAccount) dari base address array.

Inisialisasi array dilakukan melalui looping:

```

24
25 int main() {
26     for (int i = 0; i < MAX_ACCOUNTS; ++i) {
27         initialize_account(accounts[i]);
28     }
29     cout << "All accounts initialized." << endl;
30     return 0;
31 }

```

Gambar 4.26 Iterasi Array untuk Inisialisasi Akun.

Setiap iterasi memanggil fungsi initialize\_account() dengan parameter referensi, mengakses setiap elemen array secara berurutan. Indeks dimulai dari 0 dan berakhir pada 4, mencakup seluruh kapasitas array.

### 4.3 Implementasi Fungsi dengan Parameter Referensi

Fungsi-fungsi dalam sistem menggunakan parameter referensi untuk memodifikasi data akun secara langsung tanpa membuat salinan:

```

12
13 void deposit(BankAccount &acc) {
14     float d;
15     cout << "\nMasukkan jumlah untuk disetor = ";
16     cin >> d;
17     if (d > 0) {
18         acc.amount += d;
19         cout << "Setoran berhasil. Saldo Baru = "
20             << acc.amount << endl;
21     } else {
22         cout << "Jumlah setoran harus positif."
23             << endl;
24     }
25 }
26

```

Gambar 4.27 Implementasi Fungsi `deposit()` dengan Parameter Referensi.

Parameter `BankAccount& acc` mendeklarasikan referensi ke struktur `BankAccount`. Ketika fungsi memodifikasi `acc.amount`, perubahan tersebut mempengaruhi object asli dalam array, bukan salinan. Ini berbeda dengan parameter pass-by-value yang akan membuat salinan dan perubahan tidak akan terlihat di caller.

Referensi dalam C++ bersifat immutable setelah inisialisasi; setelah referensi `acc` terikat pada object tertentu, tidak dapat diubah untuk mereferensikan object lain. Referensi juga tidak dapat bernilai null, berbeda dengan pointer.

Fungsi `modify_account()` memanfaatkan referensi untuk mengubah multiple members:

```

12
13 void modify_account(BankAccount &acc) {
14     cout << "\nMengubah No. Akun: " << acc.accno
15     |     << endl;
16     cout << "Masukkan nama baru: ";
17     cin.ignore();
18     getline(cin, acc.name);
19     cout << "Masukkan alamat baru: ";
20     getline(cin, acc.address);
21     cout << "Masukkan tipe akun baru (s/c): ";
22     cin >> acc.acctype;
23     cout << "\nAkun berhasil diubah." << endl;
24 }
25

```

Gambar 4.28 Implementasi Fungsi `modify_account()`.

Dalam fungsi ini, `cin.ignore()` membersihkan karakter newline dari buffer input setelah `cin >> acctype` pada operasi sebelumnya, memastikan `getline()` bekerja dengan benar. Referensi memungkinkan modifikasi langsung terhadap anggota-anggota struktur di dalam array.

#### 4.4 Implementasi Stream Input-Output

Stream input-output menggunakan object global `cin` dan `cout` dari header `<iostream>`:

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "iostream included and namespace std "
6         "used."
7         << endl;
8     return 0;
9 }
```

Gambar 4.29 Deklarasi Header &lt;iostream&gt;.

Operator ekstraksi `>>` membaca data dari input stream:

```

3
4 int main() {
5     int choice;
6     cout << "Masukkan nomor pilihan: ";
7     cin >> choice;
8     cout << "Pilihan Anda: " << choice << endl;
9     return 0;
10 }
```

Gambar 4.30 Penggunaan Operator Ekstraksi `cin >>`.

Untuk membaca string dengan spasi, digunakan `getline()`:

```

3
4 int main() {
5     string name;
6     cout << "Masukkan nama lengkap: ";
7     cin.ignore();
8     getline(cin, name);
9     cout << "Nama Anda: " << name << endl;
10    return 0;
11 }
```

Gambar 4.31 Penggunaan `getline()` untuk Input String dengan Spasi.

Operator insersi `<<` menampilkan data ke output stream:

```

7
8 int main() {
9     BankAccount acc = {5000.0f};
10    cout << "Saldo Baru = " << acc.amount << endl;
11    return 0;
12 }
```

Gambar 4.32 Penggunaan Operator Insersi `cout <<`.

#### 4.5 Implementasi Kontrol Alur dengan Switch-Case

Menu utama program menggunakan `switch-case` untuk mengarahkan eksekusi berdasarkan pilihan user:

```

26
27     switch (choice) {
28     case 1:
29         acc_idx = find_available_slot(accounts,
30                                         MAX_ACCOUNTS);
31         if (acc_idx != -1) {
32             create_new_account(accounts[acc_idx]);
33             cout << "Akun Berhasil Dibuat..." << endl;
34         } else {
35             cout << "Batas akun terlampaui. Tidak dapat "
36                 "membuat akun baru."
37                 << endl;
38         }
39         break;
40     case 2:
41         // Logika setoran
42         break;
43     case 8:
44         exit(0);
45     default:
46         cout << "Pilihan tidak valid. Silakan coba "
47                 "lagi."
48                 << endl;
49     }
50

```

Gambar 4.33 Implementasi switch-case untuk Menu Utama.

Statement `break` penting untuk menghentikan eksekusi dan keluar dari switch block. Tanpa `break`, eksekusi akan terus ke case berikutnya (fall-through), yang umumnya tidak diinginkan.

Case 8 memanggil `exit(0)` untuk menghentikan program dengan status 0 (normal termination). `cstdlib` header menyediakan fungsi `exit()`.

Case `default` menangani pilihan yang tidak valid. Struktur switch-case ini membuat menu-driven program yang mudah dipahami dan diperluas.

#### 4.6 Implementasi Algoritma Pencarian Linear

Fungsi `find_account_idx()` mengimplementasikan pencarian linear untuk menemukan akun berdasarkan nomor akun:

```

8
9     int find_account_idx(int acc_num,
10                           const BankAccount accounts[],
11                           int size) {
12         for (int i = 0; i < size; ++i) {
13             if (accounts[i].accno == acc_num) {
14                 return i;
15             }
16         }
17         return -1;
18     }
19

```

Gambar 4.34 Implementasi Fungsi `find_account_idx()` (Pencarian Linear).

Fungsi menerima tiga parameter: nomor akun yang dicari (`acc_num`), array akun (`accounts` dengan qualifier `const` karena fungsi hanya membaca), dan ukuran array (`size`).

Loop `for` iterasi dari indeks 0 sampai `size - 1`. Setiap iterasi membandingkan `accounts[i].accno` dengan `acc_num`. Ketika match ditemukan, fungsi segera mengembalikan indeks `i`. Jika seluruh array diperiksa tanpa menemukan match, fungsi mengembalikan `-1` sebagai sentinel value yang menunjukkan akun tidak ditemukan.

Caller menggunakan return value ini untuk pemeriksaan:

```

29
30     acc_idx = find_account_idx(acc_num, accounts,
31                               MAX_ACCOUNTS);
32     if (acc_idx != -1) {
33         deposit(accounts[acc_idx]);
34     } else {
35         cout << "Akun tidak ditemukan." << endl;
36     }
37     return 0;
38 }
```

Gambar 4.35 Penggunaan Fungsi `find_account_idx()`.

Pemeriksaan `acc_idx != -1` menentukan apakah pencarian berhasil. Jika berhasil, akun pada indeks `acc_idx` dapat diakses untuk operasi lebih lanjut.

#### 4.7 Implementasi Fungsi Validasi dan Utilitas

Fungsi `is_account_empty()` digunakan untuk memeriksa apakah slot akun kosong:

```

7
8     bool is_account_empty(const BankAccount &acc) {
9         return acc.accno == 0;
10    }
11
12    int main() {
13        BankAccount acc1 = {0};
14        BankAccount acc2 = {101};
15        cout << "Account 1 is empty: " << boolalpha
16        << is_account_empty(acc1) << endl;
```

Gambar 4.36 Implementasi Fungsi `is_account_empty()`.

Fungsi ini menggunakan konvensi bahwa nomor akun 0 menunjukkan slot kosong (tidak valid). Return type `bool` memberikan hasil boolean yang dapat langsung digunakan dalam kondisi `if`.

Fungsi `find_available_slot()` mencari slot kosong pertama:

```
11
12     int find_available_slot(
13         const BankAccount accounts[], int size) {
14     for (int i = 0; i < size; ++i) {
15         if (is_account_empty(accounts[i])) {
16             return i;
17         }
18     }
19     return -1;
20 }
21
```

Gambar 4.37 Implementasi Fungsi `find_available_slot()`.

Fungsi ini memanfaatkan `is_account_empty()` untuk memeriksa setiap slot. Ketika slot kosong ditemukan, indeks tersebut dikembalikan. Jika semua slot penuh, `-1` dikembalikan.

Fungsi `initialize_account()` mereset akun ke state kosong:

```
11
12     void initialize_account(BankAccount &acc) {
13         acc.accno = 0;
14         acc.name = "";
15         acc.address = "";
16         acc.actype = '\0';
17         acc.amount = 0.0;
18     }
19
```

Gambar 4.38 Implementasi Fungsi `initialize_account()`.

Penggunaan parameter referensi memungkinkan modifikasi object asli. Semua anggota diatur ke nilai default: nomor akun 0, string kosong, karakter null, dan saldo 0.0.

## BAB V

### KESIMPULAN

Sistem manajemen akun bank yang dikembangkan dalam proyek ini mendemonstrasikan penerapan konsep-konsep fundamental pemrograman C++ secara komprehensif. Melalui implementasi, pembaca telah mempelajari dan memahami:

1. Penggunaan struktur data `struct` untuk abstraksi dan enkapsulasi atribut-atribut logis ke dalam composite type yang bermakna. Struct `BankAccount` mengelompokkan lima anggota dengan tipe berbeda menjadi kesatuan yang merepresentasikan entitas akun bank.
2. Penerapan array statis sebagai kontainer untuk menyimpan multiple instances dari struktur dengan kapasitas tetap. Array ini memungkinkan akses  $O(1)$  ke setiap akun melalui subscript operator, meskipun dengan trade-off kapasitas terbatas.
3. Implementasi fungsi dengan parameter referensi yang memungkinkan fungsi memodifikasi data asli tanpa overhead pembuatan salinan. Referensi memberikan semantik yang lebih intuitif dibandingkan pointer untuk kasus-kasus seperti ini.
4. Penggunaan stream input-output berbasis object `cin` dan `cout` untuk komunikasi interaktif dengan user. Operator ekstraksi dan insersi memudahkan pembacaan dan penulisan data dengan tipe otomatis conversion.
5. Desain menu interaktif berbasis `switch-case` yang memungkinkan user memilih dari operasi yang berbeda-beda. Struktur ini scalable dan mudah untuk menambahkan operasi baru.
6. Implementasi algoritma pencarian linear untuk menemukan akun dalam array. Meskipun kompleksitas  $O(n)$ , algoritma ini cukup efisien untuk dataset berukuran kecil dan mudah dipahami.
7. Pemahaman tentang convention dan best practices dalam desain program, seperti menggunakan sentinel value (-1) untuk menunjukkan kondisi khusus, dan `const` qualifier untuk parameter yang tidak dimodifikasi.

Proyek ini memberikan fondasi yang solid untuk memahami konsep pemrograman tingkat pemula dan untuk melanjutkan ke topik-topik lebih advanced seperti linked list, dynamic memory allocation, file I/O, dan object-oriented programming. Sistem yang dikembangkan dapat diperluas dengan fitur-fitur tambahan seperti penyimpanan data ke file, enkripsi password, atau analisis transaksi historis.

## **DAFTAR PUSTAKA**

- cppreference.com. (2025c). *Function declaration*. <https://en.cppreference.com/w/cpp/language/function>
- cppreference.com. (2025d). *Input/output library*. <https://en.cppreference.com/w/cpp/header/iostream>
- cppreference.com. (2025a). *Struct Declaration*. <https://en.cppreference.com/w/c/language/struct>
- cppreference.com. (2025b). *Subscript Operator*. [https://en.cppreference.com/w/cpp/language/operator\\_member\\_access](https://en.cppreference.com/w/cpp/language/operator_member_access)
- GeeksforGeeks. (n.d.-b). *Arrays in C++*. <https://www.geeksforgeeks.org/cpp/cpp-arrays/>
- GeeksforGeeks. (n.d.-c). *Basic Input / Output in C++*. <https://www.geeksforgeeks.org/cpp/basic-input-output-c/>
- GeeksforGeeks. (n.d.-a). *Structures in C++*. <https://www.geeksforgeeks.org/cpp/structures-in-cpp/>
- Microsoft Learn. (n.d.). *Subscript Operator*. <https://learn.microsoft.com/en-us/cpp/cpp/subscript-operator>
- W3Schools. (n.d.). *C++ cin object*. [https://www.w3schools.com/cpp/ref\\_iostream\\_cin.asp](https://www.w3schools.com/cpp/ref_iostream_cin.asp)

LAMPIRAN A

## Kode Sumber (1)

## Kode Sumber (2)

```

11:58 AM ↓ > 2.14 87% 11:58:49 11:59 AM ↓ > 87% 11:59:12
1 2 3 KELOMPOK 6 AP 11:58:49 1 2 3 KELOMPOK 6 AP 11:59:12
=====  

| Menu Bank |  

=====  

1. Buat akun baru  

2. Setoran  

3. Penarikan  

4. Cek akun  

5. Ubah akun  

6. Hapus akun  

7. Tampilkan semua akun  

8. Keluar  

=====  

Masukkan nomor pilihan: 1  

Masukkan no. akun baru: 67  

Masukkan nama lengkap: zain  

Masukkan alamat: jakarta  

Masukkan jenis akun (s untuk tabungan, c untuk giro): s  

Masukkan jumlah setoran awal: Rp. 61  

Akun Berhasil Dibuat...  

Tekan Enter untuk melanjutkan...

```

### Buat Akun Baru (1), Setoran (2)

```

11:59 AM ↓ > 87% 11:59:40 12:00 PM ↓ > 87% 12:00:08
1 2 3 KELOMPOK 6 AP 11:59:40 1 2 3 KELOMPOK 6 AP 12:00:08
=====  

| Menu Bank |  

=====  

1. Buat akun baru  

2. Setoran  

3. Penarikan  

4. Cek akun  

5. Ubah akun  

6. Hapus akun  

7. Tampilkan semua akun  

8. Keluar  

=====  

Masukkan nomor pilihan: 3  

Masukkan no. akun: 67  

Masukkan jumlah untuk ditarik = Rp. 200  

Saldo tidak mencukupi.  

Sisa saldo: Rp. 130  

Tekan Enter untuk melanjutkan...

```

```

1. Buat akun baru  

2. Setoran  

3. Penarikan  

4. Cek akun  

5. Ubah akun  

6. Hapus akun  

7. Tampilkan semua akun  

8. Keluar  

=====  

Masukkan nomor pilihan: 3  

Masukkan no. akun: 67  

Masukkan jumlah untuk ditarik = Rp. 100  

Penarikan berhasil.  

Sisa saldo: Rp. 30  

Tekan Enter untuk melanjutkan...

```

### Penarikan Gagal dan Berhasil (3)

```

12:00 PM ↓ > 87% 12:00:22 12:01PM ↓ > 87% 12:01:33
1 2 3 KELOMPOK 6 AP 12:00:22 1 2 3 KELOMPOK 6 AP 12:01:33
=====  

| Menu Bank |  

=====  

1. Buat akun baru  

2. Setoran  

3. Penarikan  

4. Cek akun  

5. Ubah akun  

6. Hapus akun  

7. Tampilkan semua akun  

8. Keluar  

=====  

Masukkan nomor pilihan: 4  

Masukkan no. akun: 67  

No. Akun: 67  

Nama: zain  

Alamat: jakarta  

Tipe Akun: s  

Saldo: Rp. 30  

Tekan Enter untuk melanjutkan...

```

```

1. Buat akun baru  

2. Setoran  

3. Penarikan  

4. Cek akun  

5. Ubah akun  

6. Hapus akun  

7. Tampilkan semua akun  

8. Keluar  

=====  

Masukkan nomor pilihan: 5  

Masukkan no. akun: 67  

Mengubah No. Akun: 67  

Masukkan nama baru: ikan  

Masukkan alamat baru: tptln lt 3  

Masukkan tipe akun baru (s/c): c  

Akun berhasil diubah.  

Tekan Enter untuk melanjutkan...

```

### Cek Akun (4), Ubah Akun (5)

```

12:01 PM 1 2 3 KELOMPOK 6 AP 12:01:46 12:02 PM 1 2 3 KELOMPOK 6 AP 12:02:36
==== Menu Bank ====
1. Buat akun baru
2. Setoran
3. Penarikan
4. Cek akun
5. Ubah akun
6. Hapus akun
7. Tampilkan semua akun
8. Keluar
=====
Masukkan nomor pilihan: 7
--- Semua Akun ---
No. Akun: 67
Nama: ikan
Alamat: itpln lt 3
Tipe Akun: c
Saldo: Rp. 30
-----
Tekan Enter untuk melanjutkan...

```

```

1. Buat akun baru
2. Setoran
3. Penarikan
4. Cek akun
5. Ubah akun
6. Hapus akun
7. Tampilkan semua akun
8. Keluar
=====
Masukkan nomor pilihan: 6
Masukkan no. akun: 999
Akun tidak ditemukan.
Tekan Enter untuk melanjutkan...

```

Tampilkan Semua Akun (7), Hapus Akun (6)

```

12:02 PM 1 2 3 KELOMPOK 6 AP 12:02:51 12:03 PM 1 2 3 KELOMPOK 6 AP 12:03:01
==== Menu Bank ====
1. Buat akun baru
2. Setoran
3. Penarikan
4. Cek akun
5. Ubah akun
6. Hapus akun
7. Tampilkan semua akun
8. Keluar
=====
Masukkan nomor pilihan: 6
Masukkan no. akun: 67
Akun berhasil dihapus.
Tekan Enter untuk melanjutkan...

```

```

1. Buat akun baru
2. Setoran
3. Penarikan
4. Cek akun
5. Ubah akun
6. Hapus akun
7. Tampilkan semua akun
8. Keluar
=====
Masukkan nomor pilihan: 8
~/algo003

```

Feedback Hapus Akun (6), Keluar (8)

Rizkiacry/bank-interactive-cli-saq



1 Contributor    0 Issues    1 Star    0 Forks



<https://github.com/Rizkiacry/bank-interactive-cli-saq>