

RESPONSI
PRAKTIKUM STRUKTUR DATA DAN ALGORITMA
2024

IDENTITAS

Nama : Havizhan Rhaiya Ardhana

NIM : L0123063

Kelas : B

Judul Program : Game Story Novel : Sensei : Beyond Journey's End

Deskripsi Program : Program Game Visual Novel dengan beberapa fitur seperti Buat Akun, Hapus akun, Cek Akun, menampilkan Peta, dan memulai story dalam Bahasa C++

DOKUMENTASI PROGRAM

Analisis Kode

Kode Lengkap :

```

1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <map>
5  #include <list>
6  #include <queue>
7  #include <set>
8  #include <ctime>
9  #include <vector>
10 #include <stack>
11
12 using namespace std;
13
14 // Struktur data untuk menyimpan informasi akun
15 > struct Akun...
25
26 // Struktur data untuk node dalam peta
27 > struct Node...
35
36 // Fungsi untuk membersihkan layar terminal
37 > void clearScreen()...
45
46 // Fungsi untuk menunggu hingga pengguna menekan tombol apapun sebelum melanjutkan
47 > void pressAnyKeyToContinue()...
54
55 // Fungsi untuk membuat akun baru
56 > void buatAkun(map<string, Akun> &daftarAkun, list<Akun> &listAkun, set<string> &daftarUsername)...
92
93 // Fungsi untuk memeriksa keberadaan akun berdasarkan username
94 > bool cekAkun(map<string, Akun> &daftarAkun, string username)...
101
102 // Fungsi untuk menghapus akun
103 > void hapusAkun(map<string, Akun> &daftarAkun, list<Akun> &listAkun, set<string> &daftarUsername, string
    username)...
141
142 // Fungsi untuk menampilkan daftar akun
143 > void lihatAkun(list<Akun> &listAkun)...
161
162 // Fungsi untuk menampilkan menu utama
163 > void tampilkanMenu()...
175
176 // Fungsi untuk menampilkan peta
177 > void tampilkanPeta(Node *locationTree)...
195
196 // Fungsi untuk membuat travel history
197 > void tampilkanTravelHistory(stack<string> historyStack, map<string, string> endings)...
232
233 // Fungsi untuk menampilkan teks cerita di lokasi tertentu
234 > string teks(string lokasi)...
488
489 // Fungsi untuk memulai cerita
490 > void mulaiStory(map<string, Akun> &daftarAkun, Node *locationTree)...
551
552 > int main()...
657

```

Berikut kode program yang membuat Game Visual Novel. Kode di atas dibagi menjadi beberapa bagian fungsi untuk memudahkan dalam menjalankan setiap programnya. Berikut adalah penjelasan setiap fungsi pada kode di atas :

1. Header



Header berfungsi untuk menjalankan program yang akan dibuat. Tanpa adanya header, program tidak akan jalan.

a. `#include <iostream>`

Pustaka standar pada C++ untuk menyediakan fungsi input/output seperti Cout dan Cin.

b. `#include <fstream>`

Untuk melakukan operasi Input/Output pada file teks dan memungkinkan untuk membaca dan menulis file.

c. `#include <string>`

Memungkinkan penggunaan fungsi-fungsi untuk memanipulasi string.

d. `#include <map>`

Sebuah struktur data koleksi asosiatif dari pasangan kunci-nilai Dimana setiap kunci dipetakan untuk ke nilai tertentu.

e. `#include <list>`

Sebuah struktur data koleksi linier dari elemen-elemen yang tersusun dalam urutan tertentu namun hanya dapat memungkinkan akses secara sekuensial.

f. `#include <queue>`

Struktur data yang memiliki konsep First In First Out (FIFO) Dimana elemen pertama yang masuk adalah elemen pertama yang keluar.

g. `#include <set>`

Struktur data yang dapat digunakan untuk mencari, menambah, dan menghapus data dan memastikan tidak ada data yang terduplikat.

h. `#include <ctime>`

Fungsi yang dapat mengambil waktu dan tanggal sesuai sistem.

i. `#include <vector>`

Struktur data yang berisi Kumpulan elemen yang disusun secara berurutan dan digunakan untuk mengakses elemen dan melakukan penambahan/ penyisipan.

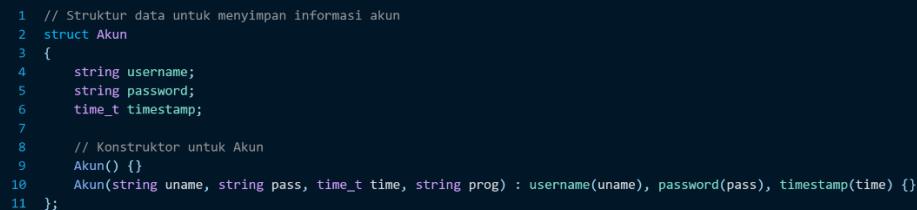
j. `#include <stack>`

Struktur data yang memiliki konsep Last In First Out (LIFO) Dimana elemen terakhir yang masuk adalah elemen pertama yang keluar.

2. Using namespace std;

Pernyataan yang digunakan untuk menghindari penulisan seperti `std::` sebelum tipe data atau fungsi-fungsi pada program.

3. Struct Akun



```
1 // Struktur data untuk menyimpan informasi akun
2 struct Akun
3 {
4     string username;
5     string password;
6     time_t timestamp;
7
8     // Konstruktor untuk Akun
9     Akun() {}
10    Akun(string uname, string pass, time_t time, string prog) : username(uname), password(pass), timestamp(time) {}
11 };
```

Struct digunakan untuk menyimpan informasi mengenai akun setiap user.

a. `string username` : Atribut untuk menyimpan nama user

b. `string password` : Atribut untuk menyimpan password

c. `time_t timestamp` : Atribut untuk menyimpan waktu pembuatan akun

Konstruktor untuk akun digunakan untuk menerima informasi dari struktur data yang di deklarasikan sebelumnya.

4. Struct Node

```
1 // Struktur data untuk node dalam reta
2 struct Node
3 {
4     string location;
5     Node *parent;
6     vector<Node *> children;
7
8     Node(string loc, Node *par) : location(loc), parent(par) {}
9 };
```

Struktur Node digunakan untuk merepresentasikan node dalam sebuah Tree.

- string location : untuk menyimpan informasi lokasi dari Node
- Node *Parent : pointer yang menunjuk ke node parent dari lokasi node saat ini.
- vector<Node *> children : Vektor yang menunjuk node-node anak dari node saat ini.
- String loc : konstruktor yang menyimpan lokasi di dalam node
- Node *par : Konstruktor yang menunjuk ke parent dari node yang akan di buat.

5. ClearScreen

```
1 // Fungsi untuk membersihkan layar terminal
2 void clearScreen()
3 {
4     #ifdef _WIN32
5         system("cls"); // For Windows
6     #else
7         system("clear"); // For Unix-based systems (Linux, macOS)
8     #endif
9 }
```

Sebuah fungsi untuk membersihkan layar terminal secara otomatis dengan cara memanggil fungsi tersebut. Fungsi tersebut dibagi menjadi 2 yaitu windows dan Linux/macOS.

6. PressAnyKey

```

1 // Fungsi untuk menunggu hingga pengguna menekan tombol apapun sebelum melanjutkan
2 void pressAnyKeyToContinue()
3 {
4     cout << "\n";
5     cout << "Tekan tombol apapun untuk melanjutkan...";
6     cin.ignore(); // Membersihkan input buffer
7     cin.get();    // Menunggu pengguna menekan tombol apapun
8 }

```

Fungsi yang digunakan untuk mengkonfirmasi user sebelum melanjutkan program. terdapat 2 fungsi yaitu `cin.ignore()` dan `cin.get()`.

- `cin.ignore()` : Digunakan untuk membersihkan input buffer.
- `cin.get()` : Untuk mengkonfirmasi user apakah sudah menekan tombol untuk melanjutkan.

7. Buat Akun

```

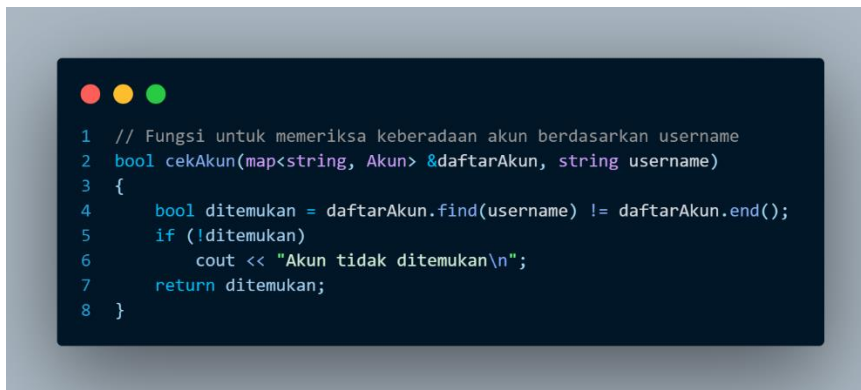
1 // Fungsi untuk membuat akun baru
2 void buatAkun(map<string, Akun> &daftarAkun, list<Akun> &listAkun, set<string> &daftarUsername)
3 {
4     clearScreen(); // membersihkan layar sebelum membuat akun
5
6     Akun akunBaru;
7     cout << "Masukkan username: ";
8     cin >> akunBaru.username;
9
10    // Memeriksa apakah username sudah digunakan
11    if (daftarUsername.find(akunBaru.username) != daftarUsername.end())
12    {
13        cout << "Username sudah digunakan\n";
14        return;
15    }
16
17    daftarUsername.insert(akunBaru.username);
18
19    cout << "Masukkan password: ";
20    cin >> akunBaru.password;
21    akunBaru.timestamp = time(0); // Mengambil timestamp saat ini
22    daftarAkun[akunBaru.username] = akunBaru;
23    listAkun.push_back(akunBaru);
24
25    // Simpan akun ke dalam file
26    ofstream file("akun.txt", ios::app);
27    if (file.is_open())
28    {
29        file << akunBaru.username << " " << akunBaru.password << " " << akunBaru.timestamp << endl;
30        file.close();
31        cout << "Akun berhasil dibuat\n";
32    }
33    else
34    {
35        cout << "Gagal membuat akun\n";
36    }
37 }

```

Sebuah fungsi yang digunakan oleh user untuk membuat akun. Fungsi ini akan menampilkan tampilan untuk memasukkan username dan password. Lalu program akan menyimpan waktu pembuatan akun dan menambahkan akun baru ke dalam file akun.txt.

- `map<string, Akun> &daftarAkun` : Struktur data map yang digunakan untuk menyimpan informasi akun, dimana string untuk username dan Akun untuk struktur data yang berisi informasi akun
- `list<Akun> &listAkun` : struktur data list yang digunakan untuk menyimpan daftar akun.
- `set<string> &daftarUsername` : digunakan untuk menyimpan daftar username yang telah dibuat.
- `if(daftarUsername.find...)` : untuk memeriksa apakah ada duplikat terhadap akun yang dibuat.
- `daftarAkun`: Menambah akun baru ke dalam map `daftarAkun`, dengan username sebagai kunci.
- `listAkun.push_back(akunBaru);`: Menambah akun baru ke dalam list `listAkun`.
- `ofstream file("akun...");`: Membuka file "akun.txt" untuk penulisan dengan mode append.
- `if(file.is_open())...` : Memeriksa apakah file "akun.txt" berhasil dibuka.

8. **bool cekAkun**



```
1 // Fungsi untuk memeriksa keberadaan akun berdasarkan username
2 bool cekAkun(map<string, Akun> &daftarAkun, string username)
3 {
4     bool ditemukan = daftarAkun.find(username) != daftarAkun.end();
5     if (!ditemukan)
6         cout << "Akun tidak ditemukan\n";
7     return ditemukan;
8 }
```

Fungsi ini untuk mengecek akun berdasarkan usernamenya dengan menggunakan bool untuk memeriksa apakah akun terdaftar atau belum.

- `CekAkun` : Fungsi yang memiliki 2 parameter yaitu `map<string, Akun> &daftarAkun` dan `string username` yang digunakan untuk memetakan string ke Akun dan string yang

mewakili usernamenya.

- `return` : Digunakan untuk mengembalikan dan mencari apakah username terdaftar pada map `daftarAkun`.
- `find(username)` : Digunakan untuk mencari elemen dalam map dengan kata kunci `username`.
- Metode `end`: Digunakan untuk mendapatkan iterator ke elemen setelah elemen terakhir dalam map.
- Operator `!=` : Digunakan untuk memeriksa apakah hasil pencarian username tidak sama dengan posisi setelah elemen terakhir dalam map.
- `pressAnyKey...` : pemanggilan fungsi untuk menunggu masukkan dari user sebelum melanjutkan ke program berikutnya.

9. Hapus Akun


```

1 // Fungsi untuk menghapus akun
2 void hapusAkun(map<string, Akun> &daftarAkun, list<Akun> &listAkun, set<string> &daftarUsername, string username)
3 {
4     auto it = daftarUsername.find(username);
5     if (it != daftarUsername.end())
6     {
7         daftarUsername.erase(it);
8         daftarAkun.erase(username);
9         for (auto itList = listAkun.begin(); itList != listAkun.end(); ++itList)
10        {
11            if (itList->username == username)
12            {
13                listAkun.erase(itList);
14                break;
15            }
16        }
17
18        cout << "Akun berhasil dihapus\n";
19
20        // Menyimpan kembali akun-akun ke dalam file
21        ofstream file("akun.txt");
22        if (file.is_open())
23        {
24            for (auto &entry : daftarAkun)
25            {
26                file << entry.second.username << " " << entry.second.password << " " << entry.second.timestamp << endl;
27            }
28            file.close();
29        }
30        else
31        {
32            cout << "Gagal menyimpan data akun\n";
33        }
34    }
35    else
36    {
37        cout << "Akun tidak ditemukan\n";
38    }
39 }

```

Fungsi untuk menghapus sebuah akun dengan menerima input berupa username dari user dan memeriksa apakah input yang diberikan terdapat pada list akun.

- daftarAkun: Sebuah parameter yang mewakili daftar akun yang memiliki tipe data map<string, Akun> dimana sebuah map yang menggunakan string sebagai kunci dan Akun sebagai nilai.
- listAkun : Sebuah parameter yang mewakili daftar akun dalam bentuk list.
- daftarUsername : Sebuah parameter yang mewakili daftar username dalam bentuk set dimana tipe datanya adalah set<string>.
- username : Sebuah parameter yang mewakili username dari akun yang ingin dihapus dimana tipe datanya adalah string.
- daftarUsername.find(username); : Digunakan untuk mencari apakah username yang ingin dihapus ada dalam set daftarUsername.

- `daftarUsername.end()` : Digunakan untuk memeriksa setelah pencarian username ditemukan.
- `daftarAkun.erase(username);` : Digunakan untuk menghapus akun sesuai dengan username yang ditemukan.
- `(ofstream file("akun.txt"));` : Digunakan untuk menulis ulang akun yang tersisa dalam file `akun.txt`.

10. Lihat Akun



```

1 // Fungsi untuk menampilkan daftar akun
2 void lihatAkun(list<Akun> &listAkun)
3 {
4     clearScreen(); // Membersihkan layar sebelum memeriksa akun
5
6     cout << "Daftar Akun:\n";
7     cout << "-----\n";
8     cout << "| No. | Username | Password | Waktu |\n";
9     cout << "-----\n";
10    int nomor = 1;
11    for (auto &akun : listAkun)
12    {
13        // Konversi timestamp ke string
14        char timestampStr[20];
15        strftime(timestampStr, 20, "%Y-%m-%d %H:%M:%S", localtime(&akun.timestamp));
16        printf("| %-4d | %-12s | %-12s | %-20s |\n", nomor++, akun.username.c_str(), akun.password.c_str(), timestampStr);
17    }
18    cout << "-----\n";
19 }

```

Fungsi untuk melihat daftar list akun di dalam terminal dengan parameter `list<Akun> &listAkun` untuk menampilkan daftar akun.

- `Cout` : Digunakan untuk mencetak kata dan bentuk tabel untuk memudahkan pembacaan user.
- `for (auto &Akun)` : Digunakan untuk menampilkan informasi setiap akun yang memanggil beberapa informasi berupa nomor, username, password, dan waktu pembuatan.

11. Tampilkan Menu

```

1 // Fungsi untuk menampilkan menu utama
2 void tampilkanMenu()
3 {
4     cout << "Selamat datang di game\n";
5     cout << "Sensei : Beyond Journey's End\n";
6     cout << "Silahkan pilih menu di bawah\n";
7     cout << "1. Buat Akun\n";
8     cout << "2. Hapus Akun\n";
9     cout << "3. Lihat Akun\n";
10    cout << "4. Lihat Peta\n";
11    cout << "5. Mulai Story\n";
12    cout << "6. Keluar\n";
13 }

```

Sebuah fungsi untuk menampilkan tampilan menu awal ketika user memulai menjalankan program. Fungsi ini terdiri dari beberapa Cout untuk mencetak tampilan menu.

12. TampilkanPeta

```

1 // Fungsi untuk menampilkan peta
2 void tampilkanPeta(Node *locationTree)
3 {
4     clearScreen(); // Membersihkan layar sebelum menampilkan peta
5
6     cout <<
7     R"(
8         /      \      Kota
9         /        \
10        /          \
11       /            \
12      /              \
13     /                \
14    /                  \
15   /                    \
16  /                      \
17 /                        \
18 )" << endl;
19 }

```

Sebuah fungsi untuk menampilkan peta yang dibuat secara manual menggunakan Cout untuk mencetak tampilan peta yang berbentuk struktur data Tree.

13. Travel History

```

1
2 // Fungsi untuk membuat travel history
3 void tampilkanTravelHistory(stack<string> historyStack, map<string, string> endings)
4 {
5     if (historyStack.empty())
6     {
7         cout << "Travel history kosong\n";
8         return;
9     }
10
11     stack<string> tempStack;
12     while (!historyStack.empty())
13     {
14         tempStack.push(historyStack.top());
15         historyStack.pop();
16     }
17
18     int i = 1;
19     while (!tempStack.empty())
20     {
21         cout << i << ". " << tempStack.top();
22         if (tempStack.size() > 1)
23         {
24             cout << " < ";
25         }
26         else
27         {
28             auto it = endings.find(tempStack.top());
29             if (it != endings.end())
30             {
31                 cout << " < " << it->second;
32             }
33         }
34         tempStack.pop();
35         i++;
36     }
37 }
38

```

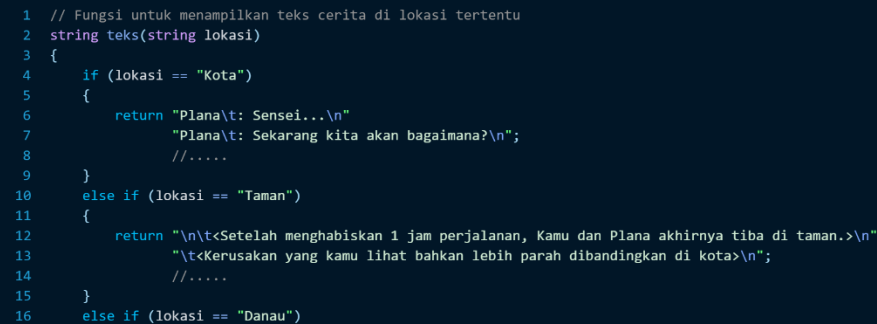
Sebuah fungsi yang digunakan untuk memperlihatkan kepada user tentang jalan yang telah dilalui selama dalam story. Fungsi ini akan terus mencetak perjalanan setiap user berpindah tempat ke lokasi berikutnya.

- tampilkanTravelHistory : deklarasi fungsi untuk menampilkan travel history dan memiliki 2 parameter yaitu stack<string> historyStack dan map<string, string> endings.
- historyStack : Parameter stack yang berisi setiap lokasi yang telah dikunjungi.
- endings : Parameter map yang digunakan untuk memetakan lokasi ke lokasi tujuan. Ketika lokasi berubah, lokasi tersebut akan ditambahkan ke dalam history perjalanan.
- (historyStack.empty()); : Sebuah fungsi untuk mengecek apakah user telah melakukan perjalanan atau belum. Jika belum, maka akan mencetak Travel history kosong.
- while (!tempStack.empty()) : Sebuah kondisi dimana akan mencetak setiap riwayat

perjalanan dengan prinsip Last in First Out (LIFO) dengan membuat stack sementara dari (tempStack) dan historyStack dan dipindahkan ke dalam tempStack dengan urutan terbalik.

Setelah menampilkan history stack yang dibalik, program akan memberikan nomor setiap urutan lokasi yang di datangi dan mencetak lokasi dengan memberi tanda "<" untuk memisahkan antara 2 lokasi.

14. Teks (string lokasi)



```
1 // Fungsi untuk menampilkan teks cerita di lokasi tertentu
2 string teks(string lokasi)
3 {
4     if (lokasi == "Kota")
5     {
6         return "Plana\t: Sensei...\n"
7             "Plana\t: Sekarang kita akan bagaimana?\n";
8         //.....
9     }
10    else if (lokasi == "Taman")
11    {
12        return "\n\t<Setelah menghabiskan 1 jam perjalanan, Kamu dan Plana akhirnya tiba di taman.>\n"
13            "\t<Kerusakan yang kamu lihat bahkan lebih parah dibandingkan di kota>\n";
14        //.....
15    }
16    else if (lokasi == "Danau")
```

Sebuah fungsi yang digunakan untuk menampilkan teks cerita ketika user berada di lokasi tertentu menggunakan if else untuk mengecek lokasi sekarang. (Kode di atas hanyalah contoh karena kode asli memiliki teks yang sangat panjang).

15. MulaiStory

```

1 // Fungsi untuk memulai cerita
2 void mulaiStory(map<string, Akun> &daftarAkun, Node *locationTree)
3 {
4     clearScreen(); // Membersihkan layar sebelum memulai cerita
5
6     string username;
7     cout << "Masukkan username: ";
8     cin >> username;
9
10    if (cekAkun(daftarAkun, username))
11    {
12        cout << "Akun tidak ditemukan\n";
13        return;
14    }
15
16    queue<Node *> travelHistory;
17    stack<string> tempTravelHistory;
18
19    Node *currentLocation = locationTree;
20    bool selesai = false;
21
22    while (!selesai)
23    {
24        clearScreen(); // Membersihkan layar sebelum menampilkan lokasi saat ini
25        tampilkanPeta(locationTree);
26
27        cout << "Current Location: " << currentLocation->location << endl;
28        cout << "Travel History: ";
29        tampilkanTravelHistory(tempTravelHistory, {"Selokan", "Ending 1: Kota < Taman < Danau < Selokan"});
30        cout << endl;
31
32        cout << "Anda berada di " << currentLocation->location << ".\n";
33        cout << teks(currentLocation->location);
34
35        cout << "Pilihan Lokasi:\n";
36        cout << "0. Menu\n";
37        for (int i = 0; i < currentLocation->children.size(); ++i)
38        {
39            cout << i + 1 << ". " << currentLocation->children[i]->location << endl;
40        }
41
42        int pilihan;
43        cout << "Pilihan Anda: ";
44        cin >> pilihan;
45
46        if (pilihan == 0)
47        {
48            break;
49        }
50        else if (pilihan >= 1 && pilihan <= currentLocation->children.size())
51        {
52            tempTravelHistory.push(currentLocation->location);
53            travelHistory.push(currentLocation);
54
55            currentLocation = currentLocation->children[pilihan - 1];
56        }
57        else
58        {
59            cout << "Pilihan tidak valid\n";
60        }
61    }
62 }

```

Fungsi untuk menyimpan program untuk menjalankan story utama dalam program ini. Fungsi ini menggunakan 2 parameter yaitu daftarAkun dengan struktur data map yang berisi daftar akun dan parameter locationTree dengan menggunakan pointer untuk menghubungkan root ke lokasi Tree.

- string Username : tipe data string yang digunakan untuk menyimpan username pengguna dan mengeceknya apakah username yang dimasukkan sesuai dengan akun

yang telah di buat.

- travelHistory : Inisiasi variabel yang digunakan untuk menyimpan history penjelajahan dalam bentuk queue.
- tempTravelHistory : Digunakan untuk menyimpan history dalam bentuk stack.
- While : Perulangan utama yang digunakan untuk menampilkan output setiap user berpindah tempat dengan kondisi-kondisi pada kode yang di tuliskan.
- TampilkanPeta : pemanggilan fungsi untuk menampilkan peta Tree yang sudah dibuat.
- tampilkanTravelHistory : Fungsi yang digunakan untuk menampilkan history setiap user berpindah lokasi dengan contoh output yang berada di kanan (tampilkanTravelHistory).
- for int : perulangan yang digunakan untuk mencetak nomor dan menu setiap user berpindah tempat.
- Jika pilihan user adalah 0, maka program akan kembali ke menu utama. Dan jika pilihan tidak terdapat pada terminal, akan menampilkan pesan pilihan tidak valid.

16. Int Main



```
1
2 int main()
3 {
4     map<string, Akun> daftarAkun;
5     list<Akun> listAkun;
6     set<string> daftarUsername;
7
8     // Membaca akun dari file akun.txt
9     ifstream file("akun.txt");
10    if (file.is_open())
11    {
12        string username, password;
13        time_t timestamp;
14        while (file >> username >> password >> timestamp)
15        {
16            daftarUsername.insert(username);
17            Akun akun(username, password, timestamp, "");
18            daftarAkun[username] = akun;
19            listAkun.push_back(akun);
20        }
21        file.close();
22    }
```

Fungsi di atas digunakan untuk melakukan pembacaan akun dari file Akun.txt dengan informasi berupa username, password, dan waktu pembuatan.

- map<string, Akun> daftarAkun : Sebuah map yang berisi daftar akun dengan string

sebagai key yang merepresentasikan username.

- list<Akun> listAkun : sebuah linked list yang berisi informasi mengenai daftar akun.
- set<string> daftarUsername : Sebuah set yang berisi username dan digunakan untuk memastikan username yang dimasukkan unik.
- ifstream file("akun.txt") : Melakukan pembacaan akun dari file akun.txt
- if (file.is.open) : Program untuk membaca data pada akun.txt yang berisi username password dan waktu pembuatan. Setiap akun yang dibuat dimasukkan ke dalam daftarAkun dan ke dalam listAkun.

```
1 // Node awal adalah "Kota"
2 Node *locationTree;
3 locationTree = new Node("Kota", nullptr);
4
5 Node *taman = new Node("Taman", locationTree);
6 Node *danau = new Node("Danau", taman);
7 Node *selokan = new Node("Selokan", danau); // Memperbaiki letak node "Selokan"
8 Node *sekolah = new Node("Sekolah", danau);
9 Node *tamanBermain = new Node("TamanBermain", sekolah);
10 Node *rumahSakit = new Node("RumahSakit", sekolah);
11
12 taman->children.push_back(danau);
13
14 danau->children.push_back(sekolah);
15 danau->children.push_back(selokan);
16
17 sekolah->children.push_back(tamanBermain);
18 sekolah->children.push_back(rumahSakit);
19
20 Node *pasar = new Node("Pasar", locationTree);
21 Node *padangPasar = new Node("PadangPasar", pasar);
22 Node *stasiun = new Node("Stasiun", pasar);
23 Node *jembatan = new Node("Jembatan", stasiun);
24 Node *pusatPermainan = new Node("PusatPermainan", stasiun);
25
26 pasar->children.push_back(padangPasar);
27 pasar->children.push_back(stasiun);
28
29 stasiun->children.push_back(jembatan);
30 stasiun->children.push_back(pusatPermainan);
31
32 locationTree->children.push_back(taman);
33 locationTree->children.push_back(pasar);
```

Sebuah sturktur data Tree yang digunakan untuk merepresentasikan lokasi.

- Node *locationTree
Location Tree = ... : Pointer ke root dari pohon lokasi. Node digunakan untuk merepresentasikan setiap simpul node dalam Tree dan Kota sebagai root.
- Node *(lokasi) = new : sebuah node baru yang merupakan anak dari node Kota. setiap node terhubung ke node induknya.

- (lokasi)->children..... : Untuk menjelaskan setiap Node memiliki daftar anak (children).

```

1  int pilihan;
2  while (true)
3  {
4      clearScreen(); // Membersihkan layar sebelum menampilkan menu
5
6      tampilkanMenu();
7
8      cout << "\nPilihan: ";
9      cin >> pilihan;
10
11     switch (pilihan)
12     {
13     case 1:
14         buatAkun(daftarAkun, listAkun, daftarUsername);
15         pressAnyKeyToContinue();
16         break;
17     case 2:
18         clearScreen();
19         cout << "Masukkan username yang akan dihapus: ";
20         {
21             string username;
22             cin >> username;
23             hapusAkun(daftarAkun, listAkun, daftarUsername, username);
24         }
25         pressAnyKeyToContinue();
26         break;
27     case 3:
28         lihatAkun(listAkun);
29         pressAnyKeyToContinue();
30         break;
31     case 4:
32         tampilkanPeta(locationTree);
33         pressAnyKeyToContinue();
34         break;
35     case 5:
36         mulaiStory(daftarAkun, locationTree);
37         break;
38     case 6:
39         clearScreen();
40         cout << "Terima kasih telah bermain!\n";
41         return 0;
42     default:
43         cout << "Pilihan tidak valid\n";
44         break;
45     }
46 }
47
48 return 0;
49 }

```

- While : Loop utama yang digunakan untuk menampilkan menu utama dan mengambil input dari user.
- tampilkanMenu : Fungsi untuk memanggil tampilan menu.
- switch : Untuk memproses pilihan user yang dimasukkan ke dalam cin << pilihan. Jika pilihan tidak ada, maka akan mencetak output pilihan tidak valid dan akan secara otomatis mengulangi program untuk memilih opsi menu.

Tabel penggunaan struktur data:

NO	STRUKTUR DATA	POIN	PERAN
1	List	5	<ul style="list-style-type: none"> - Struktur data list digunakan untuk menyimpan serangkaian elemen data yang terletak di lokasi memori yang berurutan. - dalam kode saya, list di implementasikan untuk menyimpan daftar akun. Berikut adalah implementasi dari list : - list<Akun> listAkun;
2	Stack	5	<ul style="list-style-type: none"> - Stack digunakan untuk mengimplementasikan struktur data berdasarkan prinsip LIFO dimana elemen terakhir yang di masukkan adalah elemen pertama yang di ambil. - Pada kode saya, stack diimplementasikan untuk menyimpan history penjelajahan user dengan kode sebagai berikut : - (Stack<string> historyStack,....)
3	Queue	5	<ul style="list-style-type: none"> - Queue digunakan untuk mengimplementasikan struktur data berdasarkan prinsip FIFO dimana elemen pertama yang dimasukkan adalah elemen pertama yang keluar. - Pada kode saya, Queue digunakan untuk mengimplementasikan travel history dengan kode sebagai berikut : - Queue<Node *> travelHistory;

4	Set	10	<ul style="list-style-type: none"> - Set digunakan untuk menyimpan elemen-elemen unik dengan urutan tertentu. - Saya mengimplementasikan set untuk menyimpan username setiap pengguna dan membuat agar username yang di buat tidak sama dengan username yang sudah ada. - <code>set<string> username;</code>
5	Map	10	<ul style="list-style-type: none"> - Map digunakan untuk memetakan kunci ke nilai. - Program saya menggunakan map untuk menghubungkan antara nama pengguna dengan akun dengan pengimplementasian sebagai berikut : - <code>map<string, Akun> daftarAkun;</code>
6	Tree	15	<ul style="list-style-type: none"> - Tree merupakan struktur data yang terdiri dari beberapa node yang saling terhubung ke Parent. - Saya menggunakan Tree untuk menghubungkan nama setiap tempat dan memberikan informasi suatu node terhubung dengan children yang mana. - Misal: <code>Node *taman = new Node("Taman", locationTree);</code> Berarti taman terhubung dengan locationTree yang dimana locationTree sebagai parent
TOTAL POIN: 50			