

LAPORAN TUGAS BESAR
PRAKTIKUM ALGORITMA & STRUKTUR DATA

**Program Penerimaan Pegawai Menggunakan Bahasa Pemrograman C++ Dengan
Mengimplementasikan Algoritma & Struktur Data Stack dan Queue**

“Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Praktikum Algoritma & Struktur Data”

Dosen Pengampu: Indri Tri Julianto, S.Kom., M.Kom.

Instruktur Praktikum: : Mita Hidayani Putri



Disusun Oleh:

Ahmad Nur Sahid (2206042)

Yoga Agustiansyah (2206050)

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI GARUT

2023

KATA PENGANTAR

Seiring dengan kemajuan teknologi dan persaingan di dunia kerja yang semakin ketat, proses penerimaan pegawai telah menjadi salah satu aspek yang penting dalam menjaga kualitas dan efisiensi suatu perusahaan. Dalam mata kuliah **Praktikum Algoritma & Struktur Data**, kami memiliki kesempatan untuk menerapkan pengetahuan yang telah kami pelajari dalam bentuk sebuah tugas besar praktikum.

Tugas besar praktikum ini kami kerjakan sebagai tim yang terdiri dari dua orang. Kami merasa terinspirasi oleh tantangan dalam proses penerimaan pegawai dan memutuskan untuk membuat sebuah program yang dapat membantu dalam menyederhanakan dan mempercepat proses tersebut. Program yang kami buat bertujuan untuk mempermudah pengelolaan data pendaftar, dan seleksi berkas calon pegawai.

Dalam pembuatan program ini, kami menerapkan berbagai algoritma dan struktur data yang telah kami pelajari selama perkuliahan. Kami juga memanfaatkan pengetahuan kami dalam pemrograman dan analisis algoritma untuk mengoptimalkan performa program agar dapat mengelola data dengan efisien dan memberikan hasil yang akurat.

Dalam laporan ini, kami akan menjelaskan tentang rancangan program, algoritma dan struktur data yang kami implementasikan, serta hasil dan evaluasi dari pengujian yang kami lakukan. Kami berharap bahwa laporan ini dapat memberikan gambaran yang komprehensif tentang proses pengembangan program penerimaan pegawai yang kami buat.

Kami ingin mengucapkan terima kasih kepada dosen pengampu dan instruktur praktikum atas bimbingan dan dukungan mereka selama proses pembuatan tugas besar ini.

Akhir kata, kami berharap laporan ini dapat memberikan gambaran yang jelas tentang program penerimaan pegawai yang kami buat. Semoga laporan ini bermanfaat dan dapat dijadikan referensi bagi pembaca yang tertarik dalam mengimplementasikan algoritma dan struktur data dalam pengembangan program.

Hormat kami,

Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
DAFTAR GAMBAR	iii
DAFTAR KODE	iv
BAB I PENDAHULUAN.....	1
BAB II PEMBAHASAN	2
2.1. Pembahasan Program	2
2.2. Direktori Root	5
2.3. Direktori src.....	8
2.4. Sub Direktori Modules	10
2.4.1. Modul stack.cpp.....	10
2.4.2. Modul queue.cpp.....	12
2.4.3. Modul userman.cpp.....	14
2.5. Sub Direktori Components.....	17
2.5.1. Modul programStack.cpp.....	17
2.5.2. Modul programQueue.cpp	20
2.5.3. Modul programUserManager.cpp.....	23
2.5.4. Modul programInfo.cpp.....	26
2.6. Sub Direktori DB	27
2.7. Sub Direktori Utils	28
2.7.1. Header dataType.h	28
2.7.2. Header utilityFunctions.h.....	29
BAB III REVIEW PENGUJI.....	31
3.1. Screenshot Program.....	31
3.2. Pertanyaan pertanyaan.....	32
BAB IV KESIMPULAN	34

DAFTAR GAMBAR

Gambar 1 - Direktori Program.....	2
Gambar 2 - Struktur Direktori Program.....	4
Gambar 3 - Direktori src	8
Gambar 4 - Tampilan Database	27
Gambar 5 - Tampilan Menu Utama Dari Program	31
Gambar 6 - Tampilan Program Info.....	31
Gambar 7 - Tampilan Login	31
Gambar 8 - Tampilan Program User Manager	31
Gambar 9 - Tampilan Menu Update User.....	32
Gambar 10 - Tampilan Program Stack	32
Gambar 11 - Tampilan Program Queue.....	32
Gambar 12 - Implementasi Proses Aritmatika.....	33

DAFTAR KODE

.\main.cpp.....	5
.\src\modules\stack.cpp	10
.\src\modules\queue.cpp.....	12
.\src\modules\userman.cpp.....	14
.\src\components\programStack.cpp	17
.\src\components\programQueue.cpp	20
.\src\components\programUserman.cpp	23
.\src\components\programInfo.cpp	26
.\src\DB\users.csv	27
.\src\utils\dataType.h.....	28
.\src\utils\utilityFunctions.h	29

BAB I

PENDAHULUAN

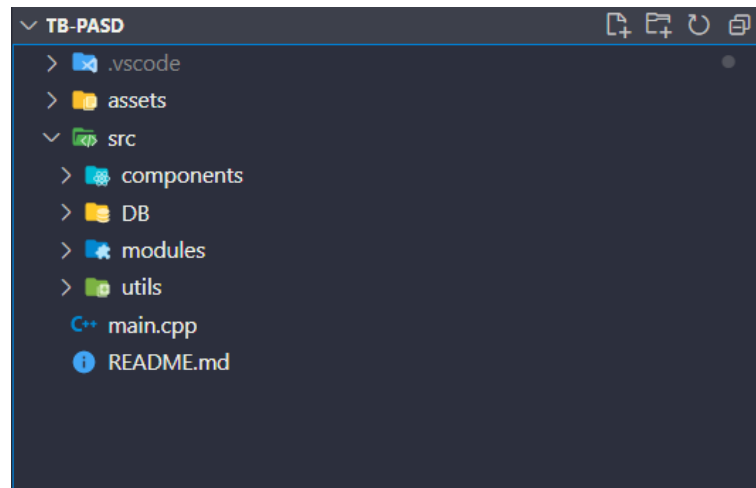
Laporan ini membahas tentang program penerimaan pegawai yang telah kami buat dalam mata kuliah "Praktikum Algoritma & Struktur Data". Program ini dirancang untuk memberikan antrian pelamar dan mengolah berkas lamaran pelamar dengan menggunakan algoritma dan struktur data stack dan queue. Dalam program ini, surat lamaran yang diberikan oleh pelamar akan langsung diantarkan ke bagian pengolah surat lamaran melalui algoritma yang kami kembangkan.

Program penerimaan pegawai kami terbagi menjadi tiga bagian yang dipisahkan oleh sistem login. Setiap pengguna memiliki akses terbatas berdasarkan peran atau role yang mereka miliki. Secara default, program ini telah memiliki tiga pengguna dengan peran yang berbeda. Pengguna dengan peran stack dapat mengakses program pengelolaan berkas lamaran (HRD). Pengguna dengan peran queue dapat mengakses program pengaturan antrian pelamar. Sementara itu, pengguna dengan peran admin memiliki hak akses sebagai manajer pengguna dan dapat melakukan operasi Create, Read, Update, dan Delete (CRUD) terhadap database pengguna.

BAB II PEMBAHASAN

2.1. Pembahasan Program

Program ini dikembangkan menggunakan bahasa pemrograman C++ dengan menerapkan metode modular programming. Modul utama program dijalankan dalam file `main.cpp` yang berada di direktori utama (root). Selain itu, terdapat sub-direktori `src` yang berisi modul-modul program yang dipanggil oleh `main.cpp`. Di dalam sub-direktori `src`, terdapat beberapa sub-sub-direktori seperti `modules`, `components`, `utils`, dan `DB`. Untuk menjalankan program ini, cukup jalankan file `main.cpp` yang berada di direktori root.



Gambar 1 - Direktori Program

Sub-direktori `modules` berisi modul-modul dasar untuk mengoperasikan program. Modul `stack.cpp` berisi algoritma dasar untuk struktur data stack seperti operasi pop dan push. Modul `queue.cpp` berisi algoritma dasar untuk struktur data queue seperti operasi enqueue dan dequeue. Selain itu, terdapat modul `userManager.cpp` yang mengatur pengelolaan pengguna dengan menerapkan skema CRUD. Modul `userManager` dapat membaca, membuat, mengedit, dan menghapus pengguna dari database lokal. Modul ini juga memiliki skema login untuk mengatur proses login pengguna.

Selain sub-sub-direktori `modules`, dalam sub-direktori `src` terdapat sub-sub-direktori `components`. Di dalamnya terdapat program-program telah menerapkan

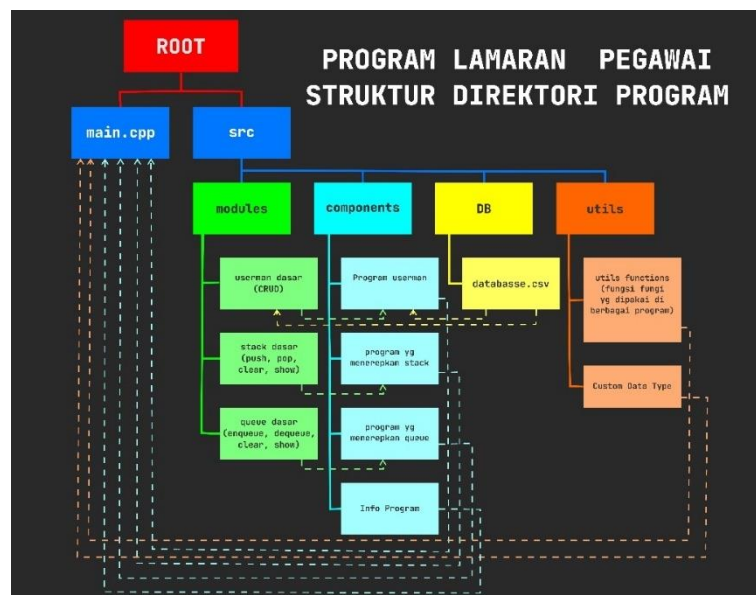
algoritma dan struktur data stack dan queue. Program `programStack.cpp` menggunakan modul `stack.cpp` yang telah dibuat di sub-direktori `modules` sebelumnya. Program ini berjalan dari perspektif penerima pegawai atau HRD yang dapat menerima, memproses, dan membuang surat lamaran. Selain itu, terdapat `programQueue.cpp` yang juga menerapkan modul `queue.cpp` dan berjalan dari perspektif pelamar pekerjaan. Program ini dapat memberikan antrian kepada pelamar dan mengambil surat lamaran dari pelamar. Terakhir, `programUserManager.cpp` memanggil modul `userManager.cpp`, program ini mengimplementasikan operasi CRUD terhadap pengguna, dan dalam prosesnya juga menggunakan metode aritmatika untuk mengatur pilihan menu yang berubah sesuai jumlah pengguna yang tersedia. Terdapat juga file `programInfo.cpp` yang berisi informasi tentang program utama ini, termasuk pembuat program dan penjelasan singkat tentang setiap peran pengguna dalam program. Semua program yang tersebut dibuat dalam tampilan CLI (Command Line Interface).

Pada program yang kami buat, data di program stack dan queue dapat terhubung satu sama lain, dimana aliran datanya pertama-tama saat program queue memberikan antrian (menjalankan method `enqueue`), maka data lamaran hanya akan disimpan secara temporary, namun ketika data lamaran di ambil atau mengambil surat lamaran (menjalankan method `dequeue`), maka data surat lamaran tersebut akan dimasukan ke variabel yang selanjutnya akan direturn ketika program queue ditutup dan ditampung di file `main.cpp`. Selanjutnya, ketika login sebagai user stack, saat dilakukan pemanggilan function `programStack()`, diperlukan 2 parameter atau argumen, yaitu 2 buah vector yang bisa berisi data baru (hasil return program queue) dan data sisa (hasil return program stack), akan dilakukan pengecekan apakah data data tersebut kosong atau tidak, jika tidak maka data tersebut akan langsung ditambahkan kedalam stack, sehingga data yang direturn dari program queue bisa masuk ke program stack. Di program stack sendiri kita bisa menerima surat lamaran tambahan (menjalankan method `push`) serta membaca surat lamaran lalu memprosesnya (menjalankan method `pop`). Saat program stack akan diakhiri, data yang masih tersimpan di stack akan di return lalu nantinya disimpan di sebuah variabel di file `main.cpp`, nantinya data tersebut masih bisa diakses oleh program stack, selama data tersebut tidak di clear.

Selanjutnya, dalam sub-direktori src terdapat sub-sub-direktori DB yang berisi database pengguna lokal. Database ini disimpan dalam format file .csv dengan nama users.csv. Setiap baris dalam database berisi username, password, dan peran pengguna. Database lokal ini akan dipanggil dalam skema login dan program pengelola pengguna.

Terakhir, dalam sub-direktori src terdapat sub-sub-direktori utils yang berisi fungsi-fungsi utilitas tambahan yang sering digunakan dalam program. File header dataType.h berisi definisi tipe data khusus yang sering digunakan dalam program. Di dalamnya terdapat dua tipe data khusus yang dibuat dengan struktur (struct), yaitu tipe data UserData untuk menyimpan data pengguna, dan tipe data LamaranData untuk menyimpan data lamaran. Di samping tipe data khusus, dalam sub-sub-direktori utils terdapat juga file header utilityFunctions.h yang berisi fungsi-fungsi utilitas seperti clearScreen untuk membersihkan tampilan layar, garis untuk membuat garis pemisah, dan getInput untuk memproses masukan dari pengguna.

Struktur program ini dirancang untuk mengorganisasi dan memisahkan setiap komponen dalam modul-modul yang terpisah. Dengan demikian, program dapat dikembangkan dan dikelola secara modular, mempermudah pemeliharaan, kolaboratif, dan meningkatkan keterbacaan serta reusabilitas kode.



Gambar 2 - Struktur Direktori Program

Dalam pembuatan program, kami menggunakan platform github untuk berkolaborasi dan mengelola kode program, sehingga kami bisa mengerjakan program ini secara remote karena dengan github, kode yang dibuat disimpan pada penyimpanan berbasis cloud sehingga kita bisa mengakses kode satu sama lain. Berikut ini link menuju repositori github yang kami gunakan : [link github](#)

2.2. Direktori Root

```
.\main.cpp
#include <iostream>
#include <vector>

using namespace std;

#include "../src/components/programStack.cpp"
#include "../src/components/programQueue.cpp"
#include "../src/components/programUserman.cpp"
#include "../src/components/programInfo.cpp"

#include "../src/utills/dataType.h"
#include "../src/utills/utilityFunctions.h"

int main() {
    init();

    string username, password;
    bool auth;

    vector<string> dataPelamar; // menampung hasil return dari program queue
    vector<string> sisaPelamar; // menampung hasil return dari program stack

    int index;

    while(true){
        clrscr();
        cout << endl << "\tSelamat Datang Di Program Lamaran" << endl
             << garis("=", 50) << endl
             << "Menu Utama: " << endl
             << "1) Login" << endl
             << "2) Info" << endl
             << "3) Keluar " << endl << endl;
        index = getInput("Pilih [1-3] >> ");

        if (index == 1) {
            clrscr();
            cout << endl << "\t" << garis("=", 15) << " LOGIN " << garis("=", 15)<<
endl
             << endl << "*Masuk menggunakan username dan password dari admin" << endl
             << garis("~", 50) << endl;
            cout << "Username : "; getline(cin, username);
            cout << "Password : "; getline(cin, password);

            auth = login(username, password);
            if (!auth) {
                cout << "Tidak dapat ijin akses" << endl
                     << endl << "klik tombol apapun untuk melanjutkan...";
                getch();
                cin.clear();
                continue;
            }
        }
    }
}
```

```

    }

    cout << endl << endl << "Login Berhasil" << endl << endl;

    if (curr_user->role == "admin") {
        // admin
        programUserman();
    } else if (curr_user->role == "stack") {
        // stack user
        LamaranData temp;
        temp = programStack(dataPelamar, sisaPelamar);
        dataPelamar = temp.dataBaru;
        sisaPelamar = temp.dataSisa;
    } else if (curr_user->role == "queue") {
        // Queue User
        vector<string> temp = programQueue();
        dataPelamar.insert(dataPelamar.end(), temp.begin(), temp.end());
    }
}

else if (index == 2) {
    info();
}

else if (index == 3) {
    cout << endl << "Keluar dari program berhasil" << endl;
    cout << endl << "klik tombol apapun untuk melanjutkan...";
    getch();
    break;
}

else {
    cout << endl << "Input tidak valid" << endl;
    cout << endl << "klik tombol apapun untuk melanjutkan...";
    getch();
}
}
}
}

```

Penjelasan

File main.cpp adalah file utama dalam program penerimaan pegawai. Pada file ini, terdapat beberapa tautan ke file-file lain yang diperlukan dalam program. Pertama, file-file header yang diperlukan diimpor menggunakan perintah **#include**, seperti iostream dan vector.

Kemudian, file-file program seperti programStack.cpp, programQueue.cpp, programUserman.cpp, dan programInfo.cpp juga diimpor menggunakan perintah **#include**. Ini memungkinkan akses ke fungsi dan definisi yang ada dalam file-file tersebut.

Setelah impor modul dan file yang diperlukan, fungsi main() dimulai. Pertama, fungsi init() dipanggil untuk melakukan inisialisasi program.

Kemudian, dilakukan pengaturan variabel username, password, dan auth untuk proses login. Ada pula vektor dataPelamar dan sisaPelamar yang akan menampung hasil return dari program queue dan stack.

Selanjutnya, dilakukan loop while(true) yang akan terus berjalan sampai ada perintah keluar dari program. Di dalam loop ini, tampilan menu utama ditampilkan kepada pengguna dan pengguna diminta untuk memilih opsi.

Jika opsi yang dipilih adalah 1 (login), maka pengguna diminta untuk memasukkan username dan password. Kemudian, dilakukan proses login dan otorisasi menggunakan fungsi login() dari modul userMan.cpp. Jika login berhasil, akan dilakukan pengecekan peran pengguna dan menjalankan program yang sesuai dengan perannya.

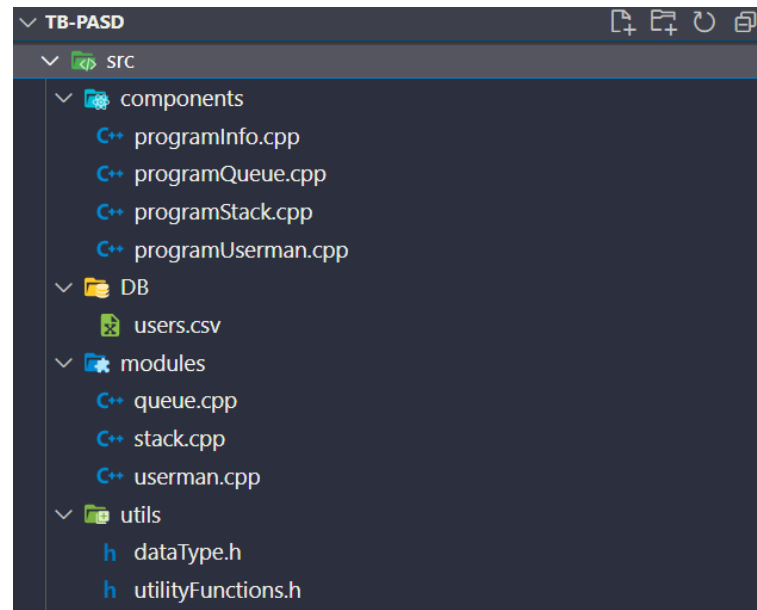
Jika peran pengguna adalah admin, maka function programUserman() dari modul programUserman.cpp akan dijalankan. Jika peran pengguna adalah stack, maka function programStack() dari modul programStack.cpp akan dijalankan dan hasil returnnya akan disimpan dalam vektor dataPelamar dan sisaPelamar. Jika peran pengguna adalah queue, maka function programQueue() dari modul programQueue.cpp akan dijalankan dan hasil returnnya akan ditambahkan ke dalam vektor dataPelamar.

Jika opsi yang dipilih adalah 2 (info), function info() akan dipanggil dari modul programInfo.cpp untuk menampilkan informasi tentang program.

Jika opsi yang dipilih adalah 3 (keluar), program akan keluar dari loop while(true) dan program akan berakhir.

Jika opsi yang dipilih tidak valid, pesan kesalahan akan ditampilkan dan pengguna diminta untuk melanjutkan dengan menekan tombol apapun.

2.3. Direktori src



Gambar 3 - Direktori src

Penjelasan

Dalam direktori src terdapat beberapa sub-direktori dan file-file yang digunakan dalam program penerimaan pegawai. Berikut adalah penjelasan mengenai isi direktori src:

1. Sub-sub-direktori "modules":
 - stack.cpp: File ini berisi implementasi algoritma dasar struktur data stack seperti push dan pop.
 - queue.cpp: File ini berisi implementasi algoritma dasar struktur data queue seperti enqueue dan dequeue.
 - userManager.cpp: File ini berisi implementasi algoritma pengelolaan pengguna (user), termasuk algoritma autentikasi login dan operasi CRUD (Create, Read, Update, Delete) terhadap database pengguna.
2. Sub-sub-direktori "components":
 - programStack.cpp: File ini menggunakan modul stack.cpp untuk membuat program yang berfokus pada penerimaan surat lamaran dan pengelolaan berkas lamaran oleh HRD.

- `programQueue.cpp`: File ini menggunakan modul `queue.cpp` untuk membuat program yang berfungsi memberikan antrian kepada pelamar pekerjaan dan mengambil surat lamaran dari pelamar.
- `programUserManager.cpp`: File ini menggunakan modul `userManager.cpp` untuk membuat program yang mengatur pengguna (user) dalam program penerimaan pegawai. Program ini juga menerapkan fitur CRUD terhadap database pengguna.
- `programInfo.cpp`: File ini berisi informasi mengenai program utama penerimaan pegawai, termasuk pembuat program dan pembahasan singkat tentang setiap peran pengguna dalam program.

3. Sub-sub-direktori "DB":

- `users.csv`: File ini merupakan database lokal pengguna (user) program yang disimpan dalam format `.csv`. File ini berisi informasi username, password, dan peran (role) dari setiap pengguna.

4. Sub-sub-direktori "utils":

- `dataType.h`: File header ini berisi definisi tipe data kustom (custom data type) yang digunakan dalam program. Salah satunya adalah tipe data `UserData` yang digunakan untuk menyimpan informasi pengguna (user).
- `utilityFunctions.h`: File header ini berisi fungsi-fungsi utilitas yang sering digunakan dalam program. Beberapa contohnya adalah fungsi `clearScreen` untuk membersihkan layar, fungsi `garis` untuk membuat garis, dan fungsi `getInput` untuk mengambil input dari pengguna dengan menangani masalah input dalam program.

Dengan adanya sub-direktori dan file-file tersebut, direktori `src` berfungsi sebagai tempat menyimpan modul-modul, program-program, database, dan utilitas yang digunakan dalam program penerimaan pegawai. Struktur direktori ini membantu dalam pengorganisasian kode dan mempermudah pemeliharaan dan pengembangan program.

2.4. Sub Direktori Modules

2.4.1. Modul stack.cpp

```
.\src\modules\stack.cpp
#include <iostream>
#include <vector>

using namespace std;

class Stack {
private:
    vector<string> stack;
public:
    void push(string data) {
        stack.push_back(data);
    }

    string pop() {
        if (stack.empty()) {
            return "";
        } else {
            string top = stack.back();
            stack.pop_back();
            return top;
        }
    }

    void clear_stack() {
        stack.clear();
    }

    void show_stack() {
        if (stack.empty()) {
            cout << "Tumpukan kosong" << endl;
        } else {
            for (int i = 0; i < stack.size(); ++i) {
                cout << i + 1 << ". " << stack[i] << endl;
            }
        }
    }
};
```

Penjelasan

File **src\modules\stack.cpp** merupakan implementasi dari struktur data stack menggunakan vector dalam bahasa C++.

Pada file ini, terdapat definisi kelas **Stack** yang memiliki beberapa metode untuk melakukan operasi pada stack. Di dalam kelas **Stack**, terdapat atribut **stack** yang merupakan sebuah vector yang digunakan untuk menyimpan elemen-elemen stack.

Metode **push(string data)** digunakan untuk menambahkan data baru ke dalam stack. Data baru tersebut akan dimasukkan ke dalam vector menggunakan fungsi **push_back()** yang merupakan fungsi dari vector.

Metode **pop()** digunakan untuk menghapus dan mengembalikan elemen teratas (top) dari stack. Jika stack kosong, maka akan dikembalikan string kosong (""). Jika stack tidak kosong, elemen teratas diambil menggunakan **back()** dan kemudian dihapus dari vector menggunakan **pop_back()**.

Metode **clear_stack()** digunakan untuk menghapus semua elemen dalam stack dengan menggunakan fungsi **clear()** dari vector.

Metode **show_stack()** digunakan untuk menampilkan isi dari stack. Jika stack kosong, akan ditampilkan pesan "Tumpukan kosong". Jika tidak kosong, akan ditampilkan elemen-elemen stack dengan nomor urut menggunakan perulangan **for**.

Alasan penggunaan vector sebagai pengganti array tradisional adalah sebagai berikut:

1. Fleksibilitas ukuran: Ukuran vector dapat berubah secara dinamis saat elemen ditambahkan atau dihapus, sehingga tidak perlu menentukan ukuran awal secara eksplisit. Hal ini mempermudah dalam penggunaan stack yang dapat berubah-ubah ukurannya.
2. Akses elemen: Vector memiliki kemampuan akses elemen dengan indeks secara efisien menggunakan operator []. Ini memudahkan akses ke elemen-elemen stack berdasarkan indeks, seperti dalam metode **show_stack()**.
3. Fungsi utilitas: Vector menyediakan berbagai fungsi utilitas seperti **push_back()**, **pop_back()**, dan **clear()** yang memudahkan dalam operasi stack seperti menambahkan, menghapus, dan mengosongkan elemen-elemen stack.
4. Keamanan akses: Vector dilengkapi dengan mekanisme pengamanan akses yang memeriksa batasan-batasan indeks saat mengakses elemen-elemen. Hal ini membantu dalam mencegah kesalahan akses dan memastikan keamanan dalam penggunaan stack.

Dengan menggunakan vector, implementasi stack menjadi lebih fleksibel, efisien, dan aman dibandingkan dengan array tradisional dalam bahasa C++.

2.4.2. Modul queue.cpp

```
.\src\modules\queue.cpp
#include <iostream>
#include <deque>

using namespace std;

class Queue {
private:
    deque<string> queue;

public:
    void enqueue(string data) {
        queue.push_back(data);
    }

    string dequeue() {
        if (queue.empty()) {
            return "";
        } else {
            string front = queue.front();
            queue.pop_front();
            return front;
        }
    }

    void clear_queue() {
        queue.clear();
    }

    void show_queue() {
        if (queue.empty()) {
            cout << "Antrian kosong" << endl;
        } else {
            for (int i = 0; i < queue.size(); ++i) {
                cout << i + 1 << ". " << queue[i] << endl;
            }
        }
    }
};
```

Penjelasan

File `src\modules\queue.cpp` merupakan implementasi dari struktur data queue menggunakan library **deque (double-ended queue)** dalam bahasa C++.

Pada file ini, terdapat definisi kelas **Queue** yang memiliki beberapa metode untuk melakukan operasi pada queue. Di dalam kelas **Queue**, terdapat atribut **queue** yang merupakan sebuah deque yang digunakan untuk menyimpan elemen-elemen queue.

Metode **enqueue(string data)** digunakan untuk menambahkan data baru ke dalam queue. Data baru tersebut akan dimasukkan ke dalam deque menggunakan fungsi **push_back()** yang merupakan fungsi dari deque.

Metode **dequeue()** digunakan untuk menghapus dan mengembalikan elemen paling depan (front) dari queue. Jika queue kosong, maka akan dikembalikan string kosong (""). Jika queue tidak kosong, elemen paling depan diambil menggunakan **front()** dan kemudian dihapus dari deque menggunakan **pop_front()**.

Metode **clear_queue()** digunakan untuk menghapus semua elemen dalam queue dengan menggunakan fungsi **clear()** dari deque.

Metode **show_queue()** digunakan untuk menampilkan isi dari queue. Jika queue kosong, akan ditampilkan pesan "Antrian kosong". Jika tidak kosong, akan ditampilkan elemen-elemen queue dengan nomor urut menggunakan perulangan **for**.

Penggunaan deque sebagai struktur data untuk queue memiliki beberapa keunggulan, antara lain:

1. Penambahan dan penghapusan elemen pada kedua ujung: Deque mendukung penambahan elemen di belakang (back) dan penghapusan elemen di depan (front) dengan efisiensi yang baik. Hal ini memungkinkan implementasi queue yang efisien dengan operasi enqueue dan dequeue yang cepat.
2. Fleksibilitas ukuran: Deque dapat berubah ukurannya secara dinamis saat elemen ditambahkan atau dihapus, sehingga tidak perlu menentukan ukuran awal secara eksplisit. Ini memudahkan dalam penggunaan queue yang dapat berubah-ubah ukurannya.
3. Akses elemen: Deque memiliki kemampuan akses elemen dengan indeks menggunakan operator **[]** seperti pada array. Ini memungkinkan akses ke elemen-elemen queue berdasarkan indeks, seperti dalam metode **show_queue()**.
4. Fungsi utilitas: Deque menyediakan fungsi-fungsi utilitas seperti **push_back()**, **pop_front()**, dan **clear()** yang memudahkan dalam operasi enqueue, dequeue, dan pengosongan elemen-elemen queue.
5. Keamanan akses: Deque dilengkapi dengan mekanisme pengamanan akses yang memeriksa batasan-batasan indeks saat mengakses elemen-elemen. Hal ini membantu mencegah kesalahan akses dan memastikan keamanan dalam penggunaan queue.

2.4.3. Modul userman.cpp

```
.\src\modules\userman.cpp

#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <vector>

#include "../utils/dataType.h"
#include "../utils/utilityFunctions.h"

#define DB_NAME "../src/DB/users.csv"

using namespace std;

fstream DB_USERS;
vector<UserData> DB;
UserData *curr_user = nullptr;

void init() {
    DB.clear();

    DB_USERS.open(DB_NAME);
    string line;

    // skip 1 baris
    getline(DB_USERS, line);

    while (getline(DB_USERS, line)) {
        stringstream ss(line);
        UserData user_tmp;

        getline(ss, user_tmp.username, ',');
        getline(ss, user_tmp.password, ',');
        getline(ss, user_tmp.role);

        DB.push_back(user_tmp);
    }
    DB_USERS.close();
}

bool find_user(string username) {
    for (UserData& user : DB) {
        if (user.username == username) {
            return 1;
        }
    }
    return 0;
}

int update_user(UserData& user) {
    string username, role;

    while (true) {
        cout << "Username Baru : ";
        cin >> username;

        // mencari username yang sama
        if (!find_user(username)) {
            user.username = username;
            break;
        }
        cout << "Tidak dapat menggunakan username yang sama"
              << endl << "Coba Lagi !!!" << endl << endl;
    }
}
```

```

// new password
cout << "Password Baru : ";
cin >> user.password;

// user saat ini tidak dapat ganti role jika user saat ini sebagai admin
if (curr_user->username == user.username && user.role == "admin"){
    cout << "Gunakan admin lainya untuk mengubah role admin" << endl;
    return 0;
}
do {
    cout << endl << "pilih role (admin/stack/queue)" << endl
        << "Role Baru      : ";
    cin >> role;
    // cek role
    if (role == "admin" || role == "stack" || role == "queue"){
        user.role = role;
        break;
    }
    cout << "Pastikan anda memilih salah satu role yang telah di sediakan" <<
endl;
} while (true);
return 1;
}

void store_user() {
    DB_USERS.open(DB_NAME, ios::out | ios::trunc);
    DB_USERS << "username,password,role" << endl;

    for (UserData& user : DB) {
        DB_USERS << user.username << ",";
        DB_USERS << user.password << ",";
        DB_USERS << user.role << endl;
    }
    DB_USERS.close();
}

void display() {
    if (DB.empty()){
        cout << endl << endl << "Tidak ada user " << endl;
    } else {
        cout << endl << endl
            << "Daftar User (" << DB.size() << ")" << endl
            << garis("-", 50) << endl
            << "NO\t" << "Username\t" << "Password\t" << "Role" << endl;
        for (int i=0; i < DB.size(); i++){
            UserData& user = DB[i];
            cout << i+1 << "\t"
                << user.username << "\t\t"
                << user.password << "\t\t"
                << user.role << endl;
        }
    }
}

bool login(string username, string password) {
    for (UserData& user : DB) {
        if (user.username == username && user.password == password) {
            curr_user = &user;
            return true;
        }
    }
    return false;
}

void add_user(){
    UserData new_user;
    update_user(new_user);
    DB.push_back(new_user);
}

```

Penjelasan

File `src\modules\userman.cpp` adalah implementasi dari modul manajemen pengguna (user management) dalam program. Modul ini berfungsi untuk melakukan operasi terkait pengguna, seperti inisialisasi pengguna, pencarian pengguna, pembaruan pengguna, penyimpanan pengguna, penampilan daftar pengguna, autentikasi pengguna, dan lain-lain.

Pada file ini, terdapat beberapa fungsi dan variabel yang digunakan untuk mengelola pengguna dalam program.

Variabel **DB_USERS** merupakan objek **fstream** yang digunakan untuk membaca dan menulis ke file yang berisi informasi pengguna. Variabel **DB** merupakan vektor **UserData** yang menyimpan informasi pengguna yang telah dimuat dari file.

Fungsi **init()** digunakan untuk menginisialisasi program dengan memuat informasi pengguna dari file ke dalam vektor **DB**. Informasi pengguna tersebut dibaca dari file CSV menggunakan objek **fstream** dan disimpan dalam objek **UserData** sebelum dimasukkan ke dalam vektor **DB**.

Fungsi **find_user(string username)** digunakan untuk mencari pengguna berdasarkan username dalam vektor **DB**. Fungsi ini akan mengembalikan nilai **true** jika pengguna ditemukan dan **false** jika tidak ditemukan.

Fungsi **update_user(UserData& user)** digunakan untuk memperbarui informasi pengguna yang diberikan sebagai argumen. Fungsi ini akan meminta pengguna memasukkan username baru yang unik, password baru, dan role baru. Jika pengguna mencoba mengubah role menjadi "admin" saat ini, maka pesan kesalahan akan ditampilkan. Fungsi ini mengembalikan nilai **1** jika pembaruan berhasil dan **0** jika tidak berhasil.

Fungsi **store_user()** digunakan untuk menyimpan informasi pengguna dari vektor **DB** ke dalam file CSV. Fungsi ini membuka file dengan mode write dan truncate (**ios::out | ios::trunc**), menuliskan header kolom, dan menuliskan informasi pengguna satu per satu dari vektor **DB** ke dalam file.

Fungsi **display()** digunakan untuk menampilkan daftar pengguna yang ada dalam vektor **DB**. Jika vektor **DB** kosong, akan ditampilkan pesan bahwa tidak ada

pengguna. Jika vektor **DB** tidak kosong, akan ditampilkan nomor urut, username, password, dan role pengguna dalam format tabel.

Fungsi **login(string username, string password)** digunakan untuk melakukan proses autentikasi pengguna berdasarkan username dan password yang diberikan. Fungsi ini membandingkan username dan password dengan informasi pengguna dalam vektor **DB**. Jika autentikasi berhasil, variabel **curr_user** akan menunjuk ke pengguna yang berhasil login, dan fungsi ini mengembalikan nilai **true**. Jika autentikasi gagal, fungsi ini mengembalikan nilai **false**.

Fungsi **add_user()** digunakan untuk menambahkan pengguna baru ke dalam vektor **DB**. Fungsi ini membuat objek **UserData** baru dan memanggil fungsi **update_user()** untuk meminta input informasi pengguna baru. Setelah informasi pengguna baru ditambahkan, objek **UserData** tersebut dimasukkan ke dalam vektor **DB**.

File **src\modules\userman.cpp** berperan penting dalam manajemen pengguna dalam program. Dengan menggunakan fungsi-fungsi yang ada pada file ini, program dapat melakukan operasi seperti inisialisasi pengguna, autentikasi, pembaruan informasi pengguna, penyimpanan informasi pengguna, dan penampilan daftar pengguna.

2.5. Sub Direktori Components

2.5.1. Modul programStack.cpp

```
.\src\components\programStack.cpp
#include <iostream>
#include <vector>
#include <conio.h>
#include <string>

using namespace std;

// import file tambahan
#include "../modules/stack.cpp"
#include "../utils/utilityFunctions.h"
#include "../utils/dataType.h"

// program Penerima Pegawai sebagai program menerapkan stack
LamaranData programStack(vector<string> lamaranBaru, vector<string> lamaranSisa) {
    // inisialisasi objek stack
    Stack programStack;

    // pembuatan variabel
    LamaranData dataLamaran;
    dataLamaran.dataBaru = lamaranBaru; // Penampung data pelamar dari program queue
    dataLamaran.dataSisa = lamaranSisa; // Penampung sisa pelamar sebelumnya
```

```

int pilihan; // penampung menu pilihan user
string pelamar, confirmClear; // penampung nama pelamar jika ingin menambahkan

/*
cek data pelamar dari program queue
jika tidak kosong maka semua data dimasukkan ke objek stack
*/

if (!dataLamaran.dataSisa.empty()) {
    for (int i = 0; i < dataLamaran.dataSisa.size(); i++) {
        programStack.push(dataLamaran.dataSisa[i]);
    }
}

if (!dataLamaran.dataBaru.empty()) {
    for (int i = 0; i < dataLamaran.dataBaru.size(); i++) {
        programStack.push(dataLamaran.dataBaru[i]);
        dataLamaran.dataSisa.push_back(dataLamaran.dataBaru[i]);
    }
    dataLamaran.dataBaru.clear();
}

// menu program utama
do {
    clrscr(); // Persihkan layar setiap kali program stack dimulai
    cout << garis("=", 50) << endl
    << "\tProgram Penerimaan Pegawai" << endl
    << garis("=", 50) << endl;

    cout << "\nPosisi anda sebagai penerima lamaran.\nSilahkan pilih menu dibawah ini" << endl << endl;

    cout << "Menu Utama : " << endl
    << "  1. Terima Surat Lamaran" << endl
    << "  2. Baca Surat Lamaran" << endl
    << "  3. Buang Semua Surat Lamaran" << endl
    << "  4. Logout" << endl;

    // menampilkan seluruh pelamar
    cout << endl << garis("-", 50) << endl << endl
    << "Surat lamaran yang ada : " << endl;
    programStack.show_stack();

    cout << endl << endl << garis("-", 50) << endl;

    pilihan = getInput("Pilihan menu >> ");

    switch (pilihan) {
        // kondisi user memilih untuk menerima surat lamaran baru
        case 1:
            // masukan nama pelamar
            cout << endl << "Nama pelamar : "; getline(cin, pelamar);

            // menambahkan pelamar baru ke objek stack
            programStack.push(pelamar);
            dataLamaran.dataSisa.push_back(pelamar);

            // pesan validasi bahwa pelamar baru telah ditambahkan
            cout << "Surat lamaran dari " << pelamar << " sudah disimpan ke tumpukan"
            << endl;
            break;

        case 2:
            // menampung data pelamar yang sudah dibaca
            pelamar = programStack.pop();

            /*
            pengecekan apakah data pelamar kosong atau tidak

```

```

        jika tidak kosong menampilkan pesan validasi bahwa data sudah dibaca
        jika kosong menampilkan pemberitahuan bahwa tidak ada surat lamaran untuk
        dibaca
    */
    if (pelamar != "") {
        cout << endl << "Surat lamaran dari " << pelamar << " sudah dibaca"
<< endl;
        dataLamaran.dataSisa.pop_back();
    } else {
        cout << endl << "Tidak ada surat lamaran untuk dibaca" << endl;
    }
    break;

    case 3:
        // konfirmasi bahwa data dalam stack akan dihapus
        cout << endl << "Data dalam tumpukan akan dihapus, apakah anda yakin? ";
    cin >> confirmClear;
    if (confirmClear == "Y" || confirmClear == "y") {
        programStack.clear_stack();
        dataLamaran.dataSisa.clear();
        cout << endl << "Tumpukan lamaran berhasil dihapus" << endl;
    } else {
        cout << endl << "Menghapus tumpukan lamaran dibatalkan" << endl;
    }
    break;

    case 4:
        // pemberitahuan bahwa berhasil logout
        cout << endl << "Logout berhasil." << endl;
        break;

    default:
        cout << endl << "Input tidak valid. Silakan coba lagi." << endl;
        break;
}

cout << endl << "klik tombol apapun untuk melanjutkan...";
getch();
} while (pilihan != 4);
return dataLamaran;
}

```

Penjelasan

Kode ini merupakan implementasi dari program Penerimaan Pegawai yang bertujuan untuk mengatur tumpukan surat lamaran. Kode ini terletak di direktori ``src/components/programStack.cpp``.

Pada awal kode, beberapa header file seperti `iostream`, `vector`, `conio.h`, dan `string` disertakan. Selain itu, file-module ``stack.cpp``, ``utilityFunctions.h``, dan ``dataType.h`` juga disertakan.

Fungsi utama dalam kode ini adalah ``programStack()``. Fungsi ini menerima dua vector sebagai argumen, yaitu ``lamaranBaru`` dan ``lamaranSisa``, yang merepresentasikan surat lamaran baru dan surat lamaran yang belum diproses sebelumnya. Fungsi ini mengembalikan sebuah objek ``LamaranData`` yang berisi informasi tentang surat lamaran yang telah diproses.

Program ini menggunakan objek dari kelas `Stack` untuk merepresentasikan tumpukan surat lamaran. Pertama, surat lamaran yang belum diproses (`lamaranSisa`) ditambahkan ke dalam tumpukan menggunakan fungsi `push()`. Selanjutnya, surat lamaran baru (`lamaranBaru`) juga ditambahkan ke dalam tumpukan dan `lamaranBaru` disimpan dalam `lamaranSisa`. Setelah itu, `lamaranBaru` dikosongkan.

Program berjalan dalam loop do-while yang terus berjalan selama pilihan pengguna tidak sama dengan 4 (Logout). Setiap iterasi loop, layar dikosongkan dan judul program ditampilkan.

Pengguna diberikan beberapa opsi menu, antara lain "Terima Surat Lamaran", "Baca Surat Lamaran", "Buang Semua Surat Lamaran", dan "Logout". Jika pengguna memilih "Terima Surat Lamaran", program meminta nama pelamar dan menambahkannya ke dalam tumpukan menggunakan fungsi `push()`. Jika pengguna memilih "Baca Surat Lamaran", program mengambil nama pelamar dari tumpukan menggunakan fungsi `pop()`. Jika tumpukan tidak kosong, nama pelamar dihapus dari `lamaranSisa`. Jika pengguna memilih "Buang Semua Surat Lamaran", program menghapus semua data dalam tumpukan dan `lamaranSisa`. Jika pengguna memilih "Logout", program menampilkan pesan logout.

Jika pengguna memilih opsi menu yang tidak valid, program menampilkan pesan kesalahan. Setiap iterasi loop, program menunggu input pengguna sebelum melanjutkan ke iterasi selanjutnya. Setelah loop selesai, program mengembalikan objek `LamaranData` yang berisi `lamaranBaru` dan `lamaranSisa`.

2.5.2. Modul programQueue.cpp

```
.\src\components\programQueue.cpp
#include <iostream>
#include <vector>
#include <conio.h>
#include <string>

// import file tambahan
#include "../modules/queue.cpp"
#include "../utils/utilityFunctions.h"

using namespace std;

// program Pegawai sebagai program menerapkan queue
vector<string> programQueue() {
    // inisialisasi objek queue
    Queue programQueue;
```

```

// pembuatan variabel
vector<string> lamaran; // Penampung data lamaran yang dibawa saat login
int pilihan; // penampung menu pilihan user
string pelamar, confirmLogout = "y"; // penampung nama pelamar yang akan
dimasukan ke antrian

// menu program utama
do {
    clrscr(); // Bersihkan layar setiap kali program queue dimulai

    cout << garis("=", 50) << endl
    << "\tProgram Antrian Calon Pegawai" << endl
    << garis("=", 50) << endl;

    cout << "\nPosisi anda sebagai pengatur pelamar pekerjaan.\nSilahkan pilih
menu dibawah ini" << endl << endl;

    cout << "Menu Utama: " << endl
    << " 1. Berikan antrian" << endl
    << " 2. Ambil Surat Lamaran" << endl
    << " 3. Logout" << endl;

    // menampilkan seluruh pelamar
    cout << endl << garis("-", 50) << endl << endl
    << "Antrian Pelamar : " << endl;
    programQueue.show_queue();

    cout << endl << endl
    << garis("-", 50) << endl;

    pilihan = getInput("Pilih Menu >> ");

    switch (pilihan)
    {
        // kondisi user memilih untuk antrian lamaran
        case 1:
            // masukan nama pelamar
            cout << endl << "Nama pelamar : "; getline(cin, pelamar);

            // menambahkan pelamar baru ke objek queue
            programQueue.enqueue(pelamar);

            // pesan validasi bahwa antrian pelamar baru telah ditambahkan
            cout << "Pelamar bernama " << pelamar << " sudah mengantri, silahkan
menunggu." << endl;
            break;

        case 2:
            // menampung data pelamar yang sudah memberikan surat lamaran
            pelamar = programQueue.dequeue();

            /*
            pengecekan apakah data pelamar kosong atau tidak
            jika tidak kosong menampilkan pesan validasi bahwa data sudah diberikan
            jika kosong menampilkan pemberitahuan bahwa tidak ada surat lamaran untuk
dibaca
            */
            if (pelamar != "") {
                cout << endl << "Surat lamaran dari " << pelamar << " sudah diambil"
<< endl;
                lamaran.push_back(pelamar);
            } else {
                cout << endl << "Tidak ada surat lamaran untuk diambil" << endl;
            }
            break;
    }
}

```

```

        case 3:
            cout << endl << "Jika anda logout, antrian pelamar yang belum memberikan surat lamaran akan dihapus." << endl
            << "Apakah anda akan keluar? "; cin >> confirmLogout;
            if (confirmLogout == "Y" || confirmLogout == "y") {
                // pemberitahuan bahwa berhasil logout
                cout << endl << "Logout berhasil, dan semua antrian dihapus" << endl;
                programQueue.clear_queue();
            } else {
                cout << "Batal logout";
            }
            break;

        default:
            cout << endl << "Input tidak valid. Silakan coba lagi." << endl;
            // break;
    }

    cout << endl << "klik tombol apapun untuk melanjutkan...";
    getch();
} while (pilihan != 3 && confirmLogout != "Y" || confirmLogout != "y");

return lamaran;
}

```

Penjelasan

Kode ini adalah implementasi dari program Antrian Calon Pegawai yang bertujuan untuk mengatur antrian pelamar pekerjaan. Kode ini terletak di direktori **src/components/programQueue.cpp**.

Program dimulai dengan menyertakan beberapa header file yang diperlukan, seperti `iostream`, `vector`, `conio.h`, dan `string`. Selain itu, file-module `queue.cpp` dan `utilityFunctions.h` juga disertakan.

Fungsi utama dalam kode ini adalah `programQueue()`. Di dalamnya, ada sebuah objek dari kelas `Queue` yang digunakan untuk merepresentasikan antrian pelamar. Ada juga `vector` `lamaran` yang digunakan untuk menyimpan surat lamaran yang telah diambil.

Program berjalan dalam loop `do-while` yang terus berjalan selama pilihan pengguna tidak sama dengan 3 (Logout) dan konfirmasi logout tidak sama dengan "Y" atau "y". Setiap iterasi loop, layar dikosongkan dan judul program ditampilkan.

Pengguna diberikan beberapa opsi menu, antara lain "Berikan antrian", "Ambil Surat Lamaran", dan "Logout". Jika pengguna memilih "Berikan antrian", program meminta nama pelamar dan menambahkannya ke dalam antrian menggunakan fungsi `enqueue()`. Jika pengguna memilih "Ambil Surat Lamaran", program mengambil nama pelamar dari antrian menggunakan fungsi `dequeue()`.

Jika antrian tidak kosong, nama pelamar ditambahkan ke dalam vector lamaran. Jika pengguna memilih "Logout", program menampilkan pesan konfirmasi dan jika pengguna mengonfirmasi logout, antrian dikosongkan.

Jika pengguna memilih opsi menu yang tidak valid, program menampilkan pesan kesalahan. Setiap iterasi loop, program menunggu input pengguna sebelum melanjutkan ke iterasi selanjutnya.

Setelah loop selesai, program mereturn vector lamaran yang berisi daftar surat lamaran yang telah diambil.

2.5.3. Modul programUserManager.cpp

```
.\src\components\programUserman.cpp

#include <iostream>
#include <conio.h>
#include "../modules/userman.cpp"
#include "../utils/utilityFunctions.h"

int programUserman(){
    int index;

    if (curr_user->role != "admin"){
        return 0;
    }

    while (true){
        int size = DB.size();

        clrscr();
        cout << garis("=", 50) << endl
        << "\t\tProgram User Manager" << endl
        << garis("=", 50);

        display();// Display Users

        cout << garis("-", 50) << endl;
        cout << "pilih [ 1";
        if (size > 1)
            cout << " - " << size;
        cout << " ] untuk Update user atau" << endl;

        // Menu lainnya
        cout << endl;
        cout << " " << size + 1 << " ) Tambah User" << endl
        << " " << size + 2 << " ) Delete User" << endl
        << " " << size + 3 << " ) Keluar" << endl;
        cout << garis("-", 50) << endl;
        index = getInput("\nPilih [1 - " + to_string(size + 3) + "] >> ");

        // Keluar sebagai admin
        if (index == size + 3){
            break;
        }

        // Add User
        if (index == size + 1){
            clrscr();
```

```

        cout << "\t\tAdd User" << endl;

        add_user();
        store_user();

        cout << endl << "Update User Telah berhasil !!!!";

        cin.clear();

    }

    // sebelum update / del cek terlebih dahulu apakah ada user di database
    else if (DB.empty()){
        cout << "Harap masukan terlebih dahulu user" << endl;
    }

    // Update user
    else if (index >= 1 && index <= size){
        clrscr();
        UserData &user = DB[index - 1];

        cout << "\t\tUpdate User" << endl << endl << endl
             << "Username : " << user.username << endl
             << "Password : " << user.password << endl
             << "Role : " << user.role << endl
             << endl;

        update_user(user);
        store_user();

        cout << endl << "Update User Telah berhasil !!!!";
    }

    // Delete User
    else if (index == size + 2){
        int choice;
        choice = getInput("Pilih user [1 - " + to_string(size) + "] >> ");
        if (choice < 1 || choice > size){
            cout << endl << "Input tidak valid. Silakan coba lagi." << endl;
        } else {
            UserData &user = DB[choice - 1];

            if (user.username == curr_user->username){
                cout << endl << "Anda tidak dapat menghapus user saat ini "
                     << endl << "gunakan admin lain untuk menghapus admin";
            } else {
                DB.erase(DB.begin() + (choice - 1));
                store_user();
                cout << "Menghapus user berhasil" << endl;
            }
        }
    }
    else {
        cout << endl << "Input tidak valid. Silakan coba lagi." << endl;
    }

    cout << endl << "klik tombol apapun untuk melanjutkan...";
    getch();
    cin.clear();
}
cout << endl << "klik tombol apapun untuk melanjutkan...";
getch();
return 0;
}

```

Penjelasan

Kode ini merupakan implementasi dari program User Manager yang bertujuan untuk mengelola pengguna dalam sistem program. Lokasinya berada di **src/components/programUserman.cpp**.

Terdapat beberapa header file yang dipanggil, seperti `iostream` untuk input/output stream, `conio.h` untuk fungsi-fungsi konsol, `userman.cpp` sebagai modul pengelolaan pengguna, dan `utilityFunctions.h` yang berisi fungsi-fungsi utilitas.

Di dalamnya terdapat fungsi `programUserman()` yang menjadi inti dari program pengelola pengguna. Fungsi ini memeriksa peran (role) pengguna saat ini. Jika perannya bukan "admin", maka fungsi ini akan mengembalikan nilai 0 dan keluar dari `programUserman()`.

Kemudian, terdapat sebuah `while` loop utama yang berjalan terus menerus sampai kondisi `break` terpenuhi. Pada setiap loop, layar dikosongkan dan judul program "Program User Manager" ditampilkan.

Daftar pengguna ditampilkan menggunakan fungsi `display()`. Menu opsi juga ditampilkan, termasuk opsi untuk mengupdate pengguna, menambah pengguna, menghapus pengguna, dan keluar dari program. Pengguna diminta untuk memilih opsi menu yang tersedia. Opsi yang dipilih oleh pengguna akan diproses sesuai dengan kondisi-kondisi yang ada dalam kode:

1. Jika opsi yang dipilih adalah untuk menambah pengguna, fungsi `add_user()` akan dipanggil untuk menambah pengguna baru ke dalam database. Perubahan tersebut kemudian disimpan menggunakan fungsi `store_user()`.
2. Jika opsi yang dipilih adalah untuk mengupdate pengguna, fungsi `update_user()` akan dipanggil untuk mengupdate pengguna yang dipilih dari database. Perubahan tersebut kemudian disimpan menggunakan fungsi `store_user()`.
3. Jika opsi yang dipilih adalah untuk menghapus pengguna, pengguna akan diminta untuk memilih pengguna yang akan dihapus. Jika pengguna yang dipilih adalah pengguna yang sedang digunakan (`curr_user`), pesan khusus akan ditampilkan. Jika bukan, pengguna tersebut akan dihapus dari database menggunakan fungsi `DB.erase()`. Pesan "Menghapus user berhasil" akan ditampilkan di layar.

4. Jika opsi yang dipilih tidak valid, pesan "Input tidak valid. Silakan coba lagi." akan ditampilkan di layar.

Kode ini menggunakan modul `userman.cpp`, fungsi `display()`, dan beberapa fungsi utilitas yang didefinisikan dalam file `utilityFunctions.h` untuk mengelola pengguna dalam database.

2.5.4. Modul `programInfo.cpp`

```
.\src\components\programInfo.cpp
#include <iostream>
#include <conio.h>

#include "../utils/utilityFunctions.h"
using namespace std;

void info() {
    clrscr();
    cout << garis("=", 50) << endl
    << "\tProgram Penerimaan Pegawai" << endl
    << garis("=", 50) << endl;

    cout << endl << "dibuat oleh : " << endl
    << "1. Ahmad Nur Sahid (2206042)" << endl
    << "2. Yoga Agustiansyah (2206050)" << endl
    << garis("-", 50) << endl;

    cout << endl << "Program Utama" << endl
    << garis("~", 50) << endl;

    cout << "1. Program Admin (User Manager)" << endl
    << "    Menjalankan peran sebagai admin," << endl
    << "    yang memiliki hak akses CRUD, yaitu untuk: " << endl
    << "        - membuat user (Create)" << endl
    << "        - melihat data user (Read)" << endl
    << "        - mengubah data user (Update)" << endl
    << "        - menghapus user (Delete)" << endl
    << garis("-", 25) << endl << endl;

    cout << "2. Program Penerima Pegawai (Stack)" << endl
    << "    Menjalankan peran sebagai penerima pegawai," << endl
    << "    memiliki hak akses untuk : " << endl
    << "        - mengambil surat lamaran (Push)" << endl
    << "        - memproses surat lamaran (Pop)" << endl
    << "        - membuang surat lamaran (Clear)" << endl
    << garis("-", 25) << endl << endl;

    cout << "3. Program Pelamar Pekerjaan (Queue)" << endl
    << "    Menjalankan peran sebagai pengatur pelamar pekerjaan," << endl
    << "    memiliki hak akses untuk : " << endl
    << "        - memberikan nomor antrian (Enqueue)" << endl
    << "        - mengambil surat lamaran (Dequeue)" << endl << endl;

    cout << garis("=", 50) << endl;
    cout << endl << "Tekan tombol apapun untuk melanjutkan...";
    getch();
}
```

Penjelasan

Fungsi `info()` dalam kode tersebut digunakan untuk menampilkan informasi tentang program Penerimaan Pegawai. Pertama, kode ini membersihkan layar dan menampilkan judul program. Kemudian, informasi pembuat program ditampilkan, beserta informasi NIM pembuat.

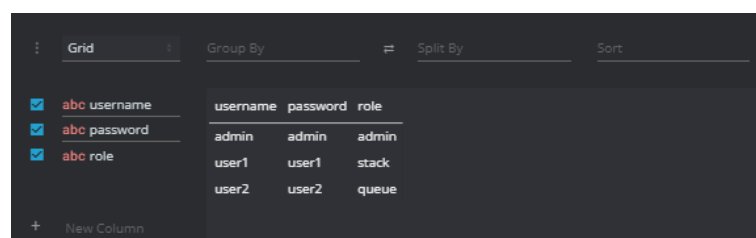
Selanjutnya, ada bagian yang disebut "Program Utama" yang menampilkan penjelasan singkat tentang tiga program yang tersedia dalam program Penerimaan Pegawai. Program pertama adalah "Program Admin (User Manager)" yang berperan sebagai admin dan memiliki hak akses CRUD (Create, Read, Update, Delete). Program kedua adalah "Program Penerima Pegawai (Stack)" yang berperan sebagai penerima lamaran pegawai dan memiliki akses untuk mengambil, memproses, dan menghapus surat lamaran menggunakan struktur data tumpukan (stack). Program ketiga adalah "Program Pelamar Pekerjaan (Queue)" yang berperan sebagai pengatur pelamar pekerjaan dan memiliki akses untuk memberikan nomor antrian dan mengambil surat lamaran menggunakan struktur data antrian (queue).

Secara keseluruhan, fungsi `info()` memberikan gambaran singkat tentang program Penerimaan Pegawai, termasuk informasi pembuatnya, program utama yang tersedia, dan peran serta hak akses dari masing-masing program tersebut.

2.6. Sub Direktori DB

.\src\DB\users.csv		
username,	password,	role
admin,	admin,	admin
user1,	user1,	stack
user2,	user2,	queue

Penjelasan



Grid	Group By	Split By	Sort
<input checked="" type="checkbox"/> abc username			
<input checked="" type="checkbox"/> abc password			
<input checked="" type="checkbox"/> abc role			
+ New Column			
username	password	role	
admin	admin	admin	
user1	user1	stack	
user2	user2	queue	

Gambar 4 - Tampilan Database

File `users.csv` merupakan file yang digunakan sebagai sumber data untuk menyimpan informasi pengguna dalam format CSV (Comma-Separated Values). File ini terletak di direktori `src\DB\users.csv`.

Struktur file `users.csv` terdiri dari tiga kolom, yaitu `username`, `password`, dan `role`. Setiap baris dalam file ini merepresentasikan satu pengguna dengan nilai-nilai yang terkait dengan kolom tersebut.

2.7. Sub Direktori Utils

2.7.1. Header `dataType.h`

```
.\src\utils\dataType.h
#ifndef DATATYPE_H
#define DATATYPE_H

#include <string>
#include <vector>

using namespace std;

struct UserData {
    string username;
    string password;
    string role;
};

struct LamaranData {
    vector<string> dataBaru;
    vector<string> dataSisa;
};

#endif
```

Penjelasan

Kode ini terletak di file `dataType.h` yang berada di direktori `src\utils`. Kode ini merupakan definisi struktur data yang digunakan dalam program.

Pertama, kita memiliki struktur `UserData` yang memiliki tiga variabel anggota: `username`, `password`, dan `role`. Struktur ini digunakan untuk menyimpan informasi pengguna, seperti nama pengguna, kata sandi, dan peran pengguna dalam program.

Selanjutnya, kita memiliki struktur `LamaranData` yang memiliki dua vektor `string`: `dataBaru` dan `dataSisa`. Struktur ini digunakan untuk menyimpan data surat lamaran. Vektor `dataBaru` menyimpan surat lamaran baru, sedangkan vektor `dataSisa` menyimpan surat lamaran yang belum diproses.

Kode ini menggunakan direktif preprosesor `#ifndef` dan `#define` untuk mencegah duplikasi inklusi file header.

2.7.2. Header `utilityFunctions.h`

```
.\src\utils\utilityFunctions.h
#ifndef UTILITYFUNCTIONS_H
#define UTILITYFUNCTIONS_H

#include <iostream>
#include <limits>

using namespace std;

string garis(string type, int jml) {
    string _garis = "";

    for (int i = 0; i < jml; i++) {
        _garis += type;
    }
    return _garis;
}

void clrscr() {
    cout << "\033c";
}

int getInput(string message) {
    int choice;
    cout << message;
    if (!(cin >> choice)) {
        cin.clear();
    }
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return choice;
}

#endif
```

Penjelasan

Kode ini terletak di file `utilityFunctions.h` yang berada di direktori `src\utils`. Kode ini berisi beberapa fungsi utilitas yang digunakan dalam program.

Pertama, kita memiliki fungsi `garis` yang menghasilkan sebuah string berupa garis dengan panjang tertentu. Fungsi ini menerima dua parameter, yaitu `type` yang menentukan karakter yang digunakan untuk membentuk garis dan `jml` yang menentukan panjang garis yang dihasilkan. Fungsi ini berguna untuk membuat tampilan garis dalam program.

Selanjutnya, kita memiliki fungsi `clrscr` yang digunakan untuk membersihkan layar konsol. Fungsi ini mengirimkan karakter escape sequence `\033c` ke output

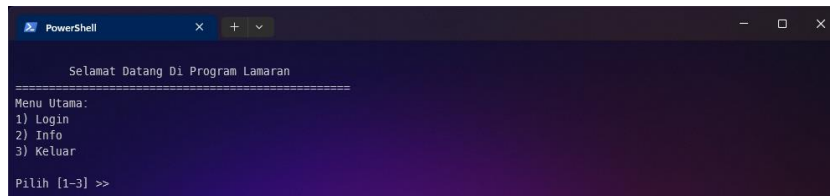
konsol untuk membersihkan layar. Fungsi ini berguna untuk memperbarui tampilan layar dalam program.

Terakhir, kita memiliki fungsi `getInput` yang digunakan untuk mendapatkan input dari pengguna dalam bentuk bilangan bulat. Fungsi ini menerima parameter `message` yang merupakan pesan yang ditampilkan kepada pengguna sebelum meminta input. Fungsi ini membaca input dari pengguna menggunakan `cin` dan mengabaikan karakter tambahan yang mungkin tersisa dalam buffer input. Fungsi ini berguna untuk mempermudah pengambilan input yang aman dan terhindar dari masalah input yang tidak valid.

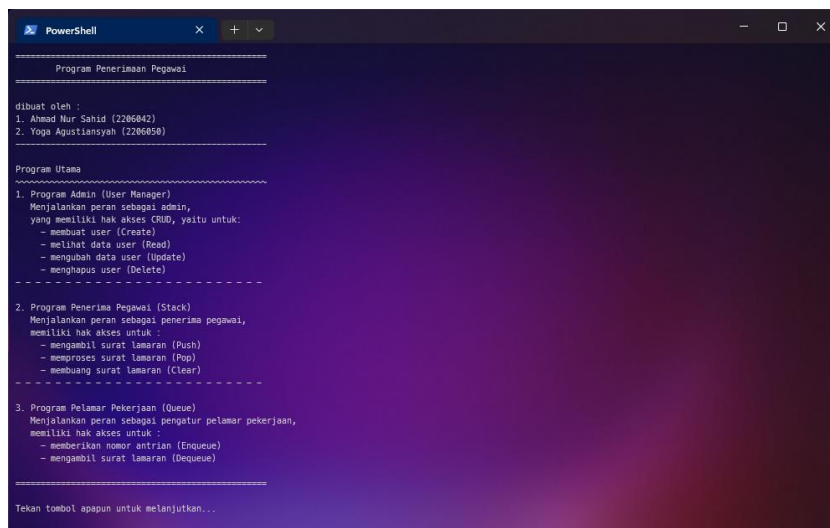
BAB III

REVIEW PENGUJI

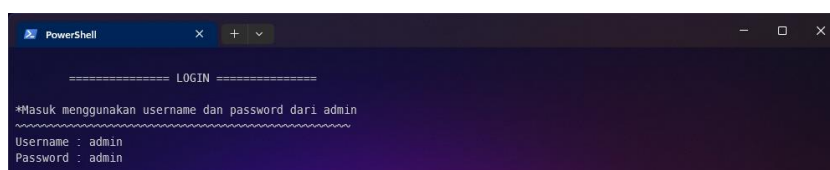
3.1. Screenshot Program



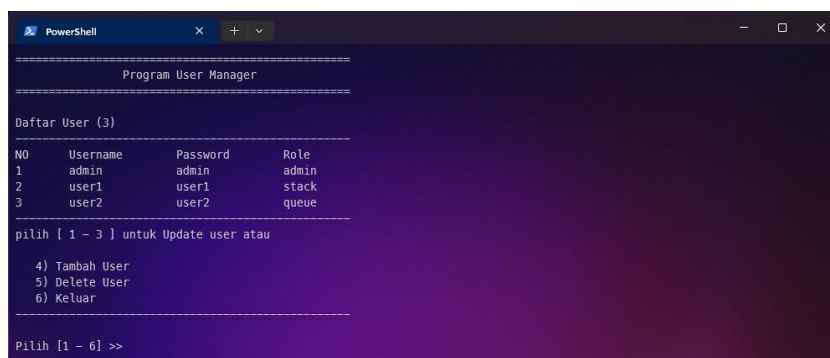
Gambar 5 - Tampilan Menu Utama Dari Program



Gambar 6 - Tampilan Program Info



Gambar 7 - Tampilan Login



Gambar 8 - Tampilan Program User Manager

```
PowerShell
Update User

Username : admin
Password : admin
Role : admin

Username Baru : ahmad
Password Baru : ahmad
Gunakan admin lainnya untuk mengubah role admin

Update User Telah berhasil !!!!
Klik tombol apapun untuk melanjutkan...
```

Gambar 9 - Tampilan Menu Update User

```
PowerShell
=====
Program Penerimaan Pegawai
=====

Posisi anda sebagai penerima lamaran.
Silahkan pilih menu dibawah ini

Menu Utama :
1. Terima Surat Lamaran
2. Baca Surat Lamaran
3. Buang Semua Surat Lamaran
4. Logout

-----

Surat lamaran yang ada :
Tumpukan kosong

-----

Pilihan menu >>
```

Gambar 10 - Tampilan Program Stack

```
PowerShell
=====
Program Antrian Calon Pegawai
=====

Posisi anda sebagai pengatur pelamar pekerjaan.
Silahkan pilih menu dibawah ini

Menu Utama:
1. Berikan antrian
2. Ambil Surat Lamaran
3. Logout

-----

Antrian Pelamar :
Antrian kosong

-----

Pilih Menu >> |
```

Gambar 11 - Tampilan Program Queue

3.2. Pertanyaan pertanyaan

1) Apakah programnya tidak menerapkan operasi aritmatika?

Ada, diterapkan di program user manager, dibagian saat menambahkan atau menghapus user, secara otomatis penomoran menu bertambah. Dikarenakan diperlukan adanya operasi aritmatika, kami menggunakan aritmatika sederhana untuk menambahkan penomoran menu.

Proses aritmatika sendiri dalam pemrograman merujuk pada serangkaian operasi matematika yang dilakukan pada data numerik. Pada dasarnya, proses ini

melibatkan penggunaan operator matematika seperti penjumlahan, pengurangan, perkalian, dan pembagian untuk melakukan perhitungan yang diperlukan.



```
31 cout << " " << size + 1 << " Tambah User" << endl // size + 1 = add
32 << " " << size + 2 << " Delete User" << endl // size + 2 = del
33 << " " << size + 3 << " Keluar" << endl; // size + 3 = exit
34 cout << garis("-", 50) << endl;
35 index = getInput("\nPilih [1 - " + to_string(size + 3) + "] >> ");
```

Gambar 12 - Implementasi Proses Aritmatika

2) apakah pakai library tambahan?

Ya, kami menggunakan library tambahan selain library standar, yaitu library:

- **vector** : sebagai pengganti array tradisional. Dengan vector pengelolaan data dalam array dapat dilakukan dengan lebih mudah dan juga meminimalisir kesalahan akses data.
- **deque** : double-ended queue dalam C++ menyediakan struktur data yang mendukung operasi penambahan dan penghapusan elemen di kedua ujungnya. Deque memungkinkan penambahan dan penghapusan elemen di awal (front) dan akhir (back) deque dengan efisiensi tinggi. Kami menggunakan library deque ini karena selama pembelajaran teori, untuk algoritma queue menggunakan library deque.

3) Apakah program stack dan queue nya sama?

Tidak, temanya sama sama penerimaan pegawai, tapi untuk programnya berbeda, program queue berjalan dari sudut pandang pelamar pekerjaan, yang memiliki pilihan menu untuk mengambil antrian dan memberikan surat lamaran. Sedangkan program stack berjalan dari sudut pandang HRD atau penerima pegawai, memiliki menu untuk menerima surat lamaran, lalu memproses surat lamaran tersebut.

BAB IV

KESIMPULAN

Dalam laporan ini, telah dibahas tentang program penerimaan pegawai yang telah dibuat dengan menggunakan algoritma dan struktur data stack dan queue. Program ini memiliki tiga bagian yang terpisah, yaitu pengelolaan berkas lamaran, pengaturan antrian pelamar, dan pengelolaan pengguna. Program ini dirancang secara modular dengan memisahkan setiap komponennya dalam modul-modul terpisah, seperti `stack.cpp`, `queue.cpp`, dan `userManager.cpp`.

Modul `stack.cpp` mengimplementasikan algoritma dasar untuk struktur data stack, termasuk operasi push dan pop. Modul `queue.cpp` mengimplementasikan algoritma dasar untuk struktur data queue, termasuk operasi enqueue dan dequeue. Sedangkan modul `userManager.cpp` mengatur pengelolaan pengguna dalam program dengan menerapkan operasi CRUD terhadap database pengguna.

Program penerimaan pegawai ini juga menggunakan beberapa sub-direktori dalam direktori `src`, seperti `components`, `utils`, dan `DB`. Sub-direktori `components` berisi program-program yang menerapkan algoritma dan struktur data stack dan queue. Sub-direktori `utils` berisi fungsi-fungsi utilitas tambahan yang sering digunakan dalam program. Sub-direktori `DB` berisi database pengguna dalam format file `.csv`.

Selain itu, dalam laporan ini juga dijelaskan mengenai struktur program utama, yang terdapat dalam file `main.cpp`. Program utama ini menggunakan modul-modul yang telah dibuat dalam sub-direktori `src` dan melakukan proses login dan otorisasi pengguna. Setelah login berhasil, program akan menjalankan program yang sesuai dengan peran pengguna, seperti `programStack.cpp`, `programQueue.cpp`, atau `programUserManager.cpp`.

Dengan adanya program penerimaan pegawai ini, diharapkan dapat membantu dalam proses pengelolaan berkas lamaran, pengaturan antrian pelamar, dan pengelolaan pengguna secara efisien. Program ini juga dirancang dengan struktur yang modular untuk mempermudah pemeliharaan dan pengembangan kode.