

**LAPORAN PROJEK AKHIR**

**SISTEM PENDATAAN KUNJUNGAN LAPAS**

MATAKULIAH:

ALGORITMA PEMROGRAMAN LANJUT

Disusun Oleh: Kelompok / 5 / 2022

Bayu Setiawan / 2109106026 (Ketua) Alfi Nor Ihsan / 2109106018

Adlina Safa Sephia Putri / 2109106021

Muhammad Nandaarjuna Fadhillah / 2109106028

Asisten:

Fayza Virdana Addiza

2009106004

**PROGRAM STUDI INFORMATIKA**

**JURUSAN INFORMATIKA FAKULTAS TEKNIK**

**UNIVERSITAS MULAWARMAN**

**SAMARINDA**

**2022**

# KATA PENGANTAR

Dengan memanjatkan segala puji syukur ke hadirat Tuhan Yang Maha Esa. Karena rahmat, hidayah, serta karunia-Nya yang melimpah, penulis dapat menyelesaikan tugas menyusun laporan yang bertema “Sistem Pendataan Kunjungan Lapas” dengan tepat waktu yang telah ditentukan.

Laporan ini disusun untuk memenuhi Projek akhir Mata Kuliah Algoritma Pemrograman Lanjut semester ini. Selain itu, kami berharap dengan penyusunan Laporan ini dapat menambah wawasan bagi para pembaca dan juga bagi penulis.

Laporan ini menjelaskan tentang “Program Sistem Pendataan Kunjungan Lapas” yang kelompok kami rancang dengan bahasa Pemrograman C++. Dengan tujuan pembuatan sebagai suatu sistem yang mengatur serta melakukan pendataan kunjungan pada sebuah Lembaga permasyarakatan, yang mana program tersebut dapat memuat data berupa informasi tentang penambahan data kunjungan, informasi pengunjung ataupun narapidana, serta tanggal, waktu dan ruangan yang digunakan saat melaksanakan kunjungan.

Laporan ini masih jauh dari kata sempurna, karena terbatasnya pengalaman dan pengetahuan yang kami miliki. Oleh sebab itu, kami mohom maaf bila ada kesalahan dalam penyusunan laporan ini. Kami berharap makalah ini dapat memberi manfaat dan menambah wawasan baik penulis maupun pembaca.

Dalam penyusunan makalah ini, penulis tidak lupa mengucapkan terimakasih pada Ibu Ir.Noviati Puspitasari, S.Kom., M.Eng, dan bapak Awang Harsa Kridalaksana, M.Kom selaku dosen Mata Kuliah Algoritma Pemrograman Lanjut. Dan juga, asisten lab Fayza Virdana Addiza selaku pembimbing projek akhir kelompok 5. Serta teman-teman yang ikut membantu dalam tugas makalah ini sehingga penulis dapat menyelesaikan makalah ini.

Samarinda, Mei 2022

Penulis

# TAKARIR

Daftar padanan kata bahasa asing dalam bahasa Indonesia yang digunakan adalah sebagai berikut :

|  |  |
| --- | --- |
| *Input* | Memasukkan |
| *Output* | Mengeluarkan |
| *Searching* | Mencari |
| *Sorting* | Mengurutkan |
| *Delete* | Menghapus |
| *Console* | Terminal untuk pengendalian |
| *Decision* | Perulangan |
| *Database* | Basis data |
| *Managemen* | Mengatur |
| *User* | Pengguna |
| *Login* | Masuk |
| *Password*  *Username* | Kata Sandi  Nama Pengguna |
| *Float* | Angka dengan titik desimal |
| *Integer*  *string* | Bilangan Bulat  Deret Simbol |

# DAFTAR ISI

[KATA PENGANTAR ii](#_Toc104503596)

[TAKARIR iii](#_Toc104503597)

[DAFTAR ISI iv](#_Toc104503598)

[DAFTAR GAMBAR v](#_Toc104503599)

[BAB I PENDAHULUAN v](#_Toc104503600)

[1.1 Latar Belakang 1](#_Toc104503601)

[1.2 Kebutuhan Fungsional 2](#_Toc104503602)

[1.3 Rumusan Masalah 2](#_Toc104503603)

[1.4 Batasan Masalah 2](#_Toc104503604)

[1.5 Tujuan 3](#_Toc104503605)

[BAB II PERANCANGAN 4](#_Toc104503606)

[2.1 Analisis Program 4](#_Toc104503607)

[2.2 Flowchart 5](#_Toc104503608)

[2.3 Konsep/Materi Praktikum yang dipakai 15](#_Toc104503609)

[BAB III HASIL DAN PEMBAHASAN 18](#_Toc104503610)

[3.1 Tampilan Program 18](#_Toc104503611)

[3.2 Source Code 27](#_Toc104503612)

[BAB IV KESIMPULAN 72](#_Toc104503613)

[4.1 Kesimpulan 72](#_Toc104503614)

[4.2 Saran 72](#_Toc104503615)

[DAFTAR PUSTAKA 73](#_Toc104503616)

[LAMPIRAN 74](#_Toc104503617)

# DAFTAR GAMBAR

**Flowchart**

[Gambar 2.2. 1 Main Menu 5](#_Toc105179282)

[Gambar 2.2. 2 Menu Tamu 6](#_Toc105179283)

[Gambar 2.2. 3 Daftar Kunjungan 6](#_Toc105179284)

[Gambar 2.2. 4 Cek Status Kunjungan 7](#_Toc105179285)

[Gambar 2.2. 5 Login 7](#_Toc105179286)

[Gambar 2.2. 6 Menu Utama 7](#_Toc105179287)

[Gambar 2.2. 7 Kunjungan Validasi 8](#_Toc105179288)

[Gambar 2.2. 8 Menu Master 8](#_Toc105179289)

[Gambar 2.2. 9 Menu Kunjungan 9](#_Toc105179290)

[Gambar 2.2. 10 Daftar Kunjungan 9](#_Toc105179291)

[Gambar 2.2. 11 Ubah Kunjungan 10](#_Toc105179292)

[Gambar 2.2. 12 Hapus Kunjungan 10](#_Toc105179293)

[Gambar 2.2. 13 Menu WBP 10](#_Toc105179294)

[Gambar 2.2. 14 Daftar WBP 11](#_Toc105179295)

[Gambar 2.2. 15 Menu Kamar 11](#_Toc105179296)

[Gambar 2.2. 16 Daftar Kamar 12](#_Toc105179297)

[Gambar 2.2. 17 Tambah Kamar 12](#_Toc105179298)

[Gambar 2.2. 18 Ubah Kamar 12](#_Toc105179299)

[Gambar 2.2. 19 Menu Operator 13](#_Toc105179300)

[Gambar 2.2. 20 Daftar Operator 13](#_Toc105179301)

[Gambar 2.2. 21 Tambah Operator 14](#_Toc105179302)

[Gambar 2.2. 22 Ubah Operator 14](#_Toc105179303)

[Gambar 2.2. 23 Hapus Operator 14](#_Toc105179304)

**Hasil Output**

[Gambar 3.1. 1 Menu Utama 18](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182232)

[Gambar 3.1. 2 Salah Input 18](#_Toc105182233)

[Gambar 3.1. 3 Inputan 19](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182234)

[Gambar 3.1. 4 Registrasi Kunjungan 19](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182235)

[Gambar 3.1. 5 Registrasi tidak terdaftar 20](#_Toc105182236)

[Gambar 3.1. 6 Pilihan 2 20](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182237)

[Gambar 3.1. 7 Status Kunjungan 21](#_Toc105182238)

[Gambar 3.1. 8 Status kunjungan Kode kunjungan 21](#_Toc105182239)

[Gambar 3.1. 9 Tidak ada kunjungan 21](#_Toc105182240)

[Gambar 3.1. 10 Daftar WBP 22](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182241)

[Gambar 3.1. 11 Pilihan Sorting WBP 22](file:///D:\LAPORAN%20PA%20KEL%205%202022.docx#_Toc105182242)

[Gambar 3.1. 12 Kata Kunci Pencarian 23](#_Toc105182243)

[Gambar 3.1. 13 Login Operator 23](#_Toc105182244)

[Gambar 3.1. 14 Menu Utama Validasi Kunjungan 24](#_Toc105182245)

[Gambar 3.1. 15 Validasi Kunjungan 24](#_Toc105182246)

[Gambar 3.1. 16 Status Kunjungan 24](#_Toc105182247)

[Gambar 3.1. 17 Jadwal ketersediaan kamar 25](#_Toc105182248)

[Gambar 3.1. 18 Data Master 25](#_Toc105182249)

[Gambar 3.1. 19 Keluar 26](#_Toc105182250)

**BAB I PENDAHULUAN**

## **Latar Belakang**

Lembaga Pemasyarakatan merupakan salah satu unit pelaksana tekhnis  
dari jajaran Kementerian Hukum dan Hak Asasi Manusia yang mempunyai  
tugas pokok melaksanakan pemasyarakatan narapidana/anak didik. Salah satu fungsi Lembaga Pemasyarakatan adalah melaksanakan fungsi pembinaan yang merupakan proses sistem pemasyarakatan sebagai realisasi pembaharuan pidana.

Selama masa pembinaan, narapidana juga memiliki hak dan kewajiban. Salah satu hak dari narapidana adalah mendapatkan kunjungan. Hal ini sesuai dengan undang-undang tentang pemasyarakatan pasal 14 ayat 8 yang berbunyi “salah satu hak darapidana yaitu menerima kunjungan dari keluarga, penasihat hukum, atau orang tertentu lainnya”.

Pada proses kunjungan tahanan dan narapidana memiliki banyak masalah maupun kekurangan. Salah satu kendala pada proses kunjungan narapidana adalah belum terorganisir dengan baik pengolahan data pengunjung, hal ini menyebabkan tidak validnya data pengunjung. Selama ini jika pengunjung ingin mengunjungi narapidana hanya mencatat identitas di buku agenda data lengkap pengunjung. Akan tetapi seringkali hanya nama yang dituliskan dibuku agenda. Sedangkan petugas tidak memeriksa atau memperhatikan hal tersebut.

Selanjutnya yang menjadi latar belakang masalah lainnya adalah tidak adanya sistem informasi yang menangani proses penjadwalan kunjungan tahanan dan narapidana. Sehingga memicu masalah yang cukup tidak terkendali. Salah satu contoh dari buruknya sistem penjadwalan kunjungan narapidana adalah penjadwalan yang berantakan, pengunjung yang tidak tahu kapan waktu dia bisa membesuk. Inilah yang menjadikan pentingnya adanya sistem penjadwalan kunjungan narapidana yang berbasis komputer.

Perkembangan Ilmu Pengetahuan dan Teknologi (IPTEK) memicu banyak kalangan untuk mencari alternatif pemecahan masalah dibidang teknologi dan sistem informasi. Penggunaan komputer sebagai alat bantu penyelesaian pekerjaan dibidang teknologi dan sistem informasi berkembang disegala bidang. Komputer dirasakan memiliki banyak keunggulan, alasannya komputer dapat diprogram sehingga dapat disesuaikan dengan keinginan pemakainya. Komputer dapat memberikan informasi yang cepat, tepat dan akurat. Komputer juga dapat mengurangi potensi terjadinya kesalahan pengolahan data dibandingkan pengolahan data secara manual.

## **Kebutuhan Fungsional**

1. Data akan diambil dari inputan yang sudah disediakan dari program
2. Data yang diinputkan berupa Data informasi pengunjung, narapidana, tanggal, waktu, dan ruangan yang digunakan dalam kunjungan.
3. Semua data yang inputkan akan ditampilkan.
4. Data yang ditampilkan di console bisa diubah, ditambahkan, dan dihapus.
5. Data yang ditampilkan dapat di-*sorting* dan di-*searching*

## **Rumusan Masalah**

Berdasarkan latar belakang yang telah dikemukakan, maka penulis merumuskan permasalahannya, sebagai berikut :

1. Bagaimana cara membuat program pendataan berbasis komputer ?
2. Bagaimana cara merancang program pendataan kunjungan Lembaga Pemasyarakatan dengan menggunakan Bahasa pemograman C++ ?
3. Bagaimana cara merancang program pendataan yang lengkap dan terorganisir ?

## **Batasan Masalah**

Agar dalam pembuatan laporan ini tidak terlalu luas tinjauannya dan tidak menyimpang dari rumusan masalah di atas, maka perlu adanya pembatasan masalah yang ditinjau.

Batasan-batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Melakukan pembuatan program untuk kepentingan pendataan kunjungan Lembaga Permyarakatan yang terorganisir.
2. Melakukan pembuatan laporan mengenai program yang mendata kunjungan Lembaga Permyarakatan.

## **Tujuan**

Dalam menulis laporan ini, Penulis memiliki beberapa tujuan yang ingin dicapai. Adapun tujuan tersebut adalah :

1. Memberikan suatu usulan rancangan sistem yang bermanfaat untuk kebutuhan pendataan.
2. Membuat layanan sistem informasi untuk pendataan kunjungan Lembaga Permasyarakatan yang terorganisir.

# BAB II PERANCANGAN

## **2.1 Analisis Program**

Saat pengguna menjalankan programnya, maka terdapat tampilan menu. Setelah itu, pengguna bisa memilih antara :

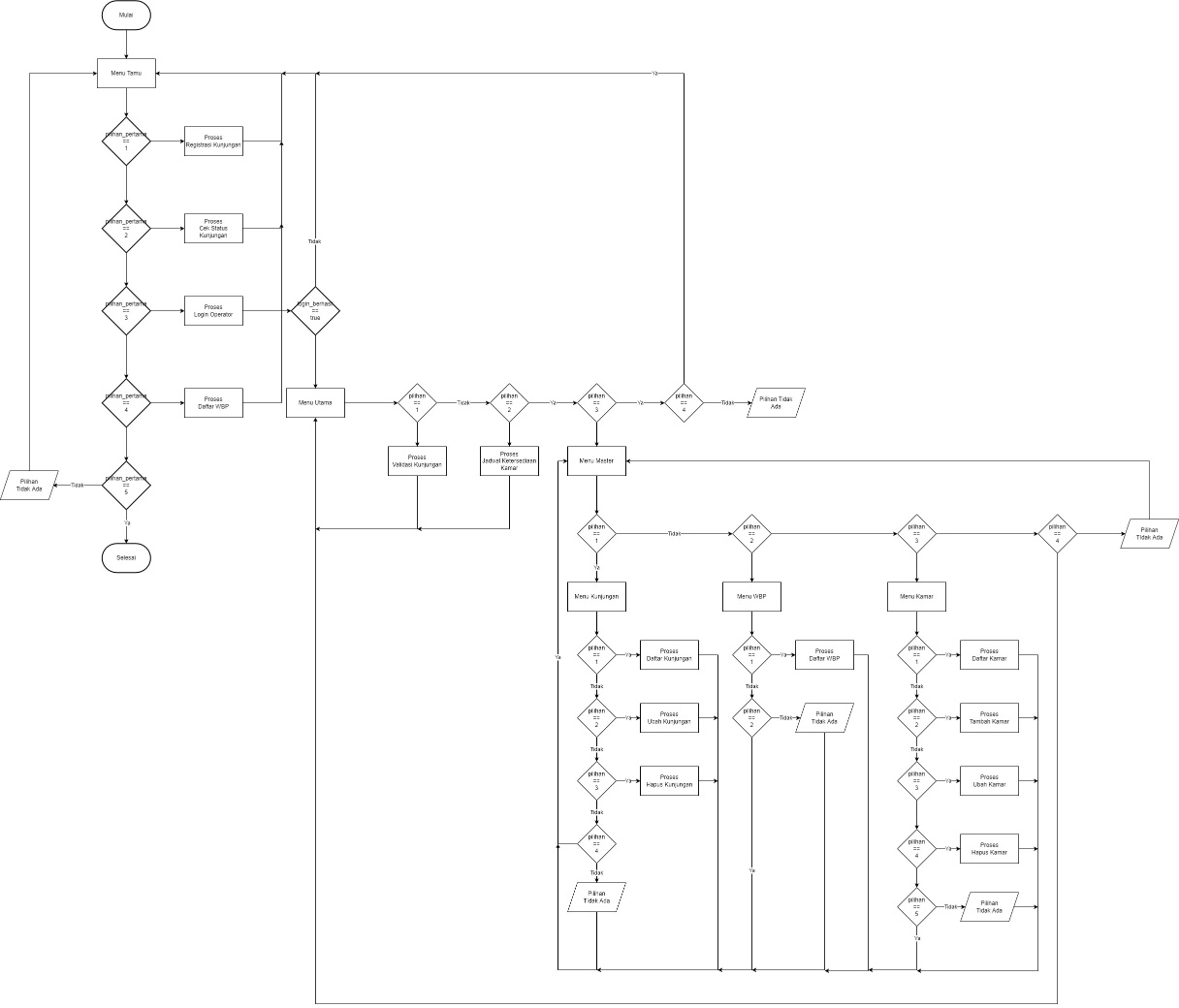
1. Registrasi kunjungan
2. Cek status kunjungan
3. Daftar WBP
4. Masuk sebagai operator
5. Keluar

Jika user memilih registrasi kunjungan maka user akan diminta untuk memasukkan data NIK dan kode WBP yang akan dikunjungi, serta nama pengguna, di mana terdapat syarat di mana kunjungan hanya dapat dilakukan dalam 1 hari. Pada tampilan cek status, pengguna akan di minta untuk memasukkan NIK ataupun kode kunjungan yang akan di proses sehingga menampilkan status dari kunjungan tersebut.

Pada tampilan data WBP, penggunaan akan melihat seluruh WBP yang tersedia dengan pilihan pengurutan dan pencarian data dari data terbesar ke data terkecil ataupun sebaliknya dengan pembanding berdasarkan kode WBP ataupun Nama WBP, Mereset proses serta keluar dari menu data WBP.

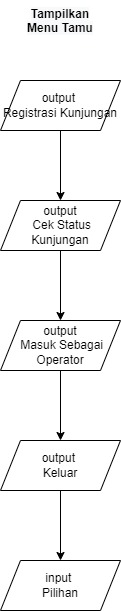
Pada menu tampilan operator bertugas untuk melakukan validasi dari proses registrasi yang di lakukan pengunjung, melihat jadwal ketersediaan kamar, berperan sebagai data master yang mengatur data berupa data kunjungan, WBP dan kamar serta menu keluar yang akan mengembalikan operator kembali ke menu utama. Adapun operator perlu melakukan login terlebih dahulu hingga username dan password sesuai dan operator akan dapat mengatur berbagai data sebelumnya.

## **2.2 Flowchart**



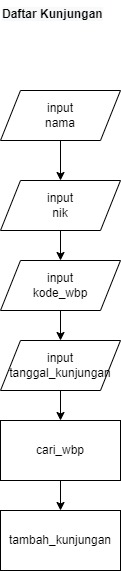
Gambar 2.2. 1 Main Menu

Menampilkan Menu Tamu



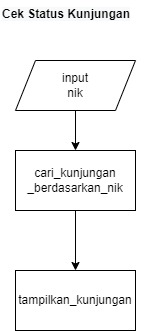
Gambar 2.2. 2 Menu Tamu

Jika memilih menu pilihan pertama, maka akan ditampilkan menu daftar kunjungan



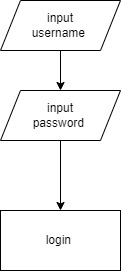
Gambar 2.2. 3 Daftar Kunjungan

Jika memilih pilihan kedua, maka akan ditampilkan menu cek status kunjungan



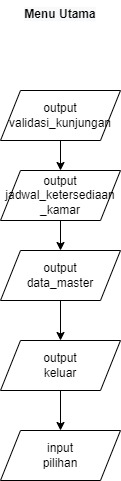
Gambar 2.2. 4 Cek Status Kunjungan

Jika memilih pilihan ketiga, maka operator diminta untuk login terlebih dahulu



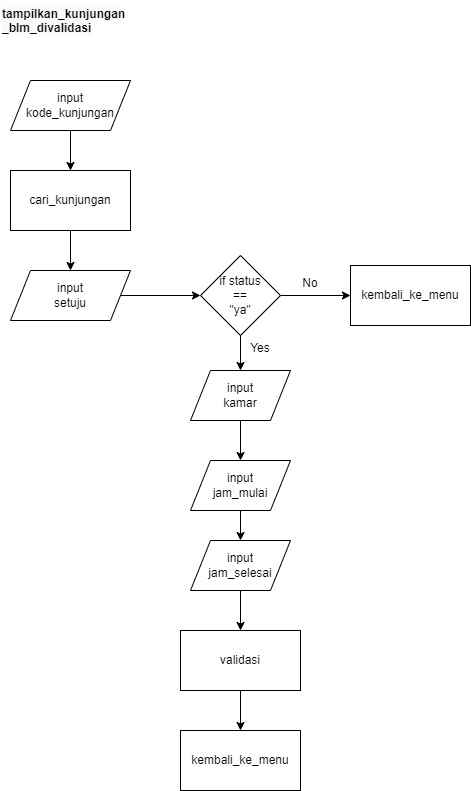
Gambar 2.2. 5 Login

Saat selesai login, maka akan ditampilkan menu utama



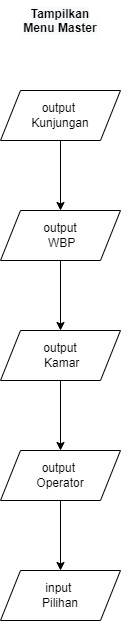
Gambar 2.2. 6 Menu Utama

Jika memilih pilihan satu menu utama maka akan menampilkan validasi kunjungan



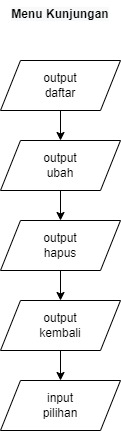
Gambar 2.2. 7 Kunjungan Validasi

Jika memilih pilihan ketiga, maka akan menampilkan menu master



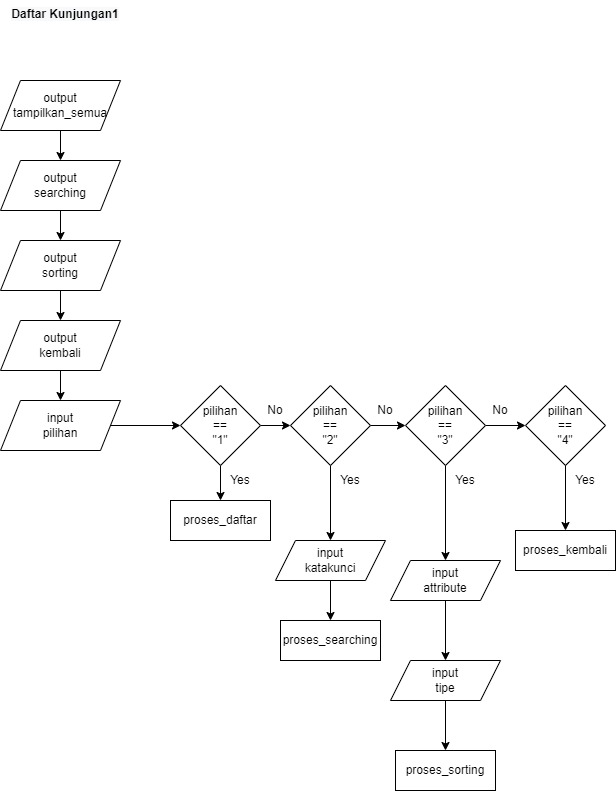
Gambar 2.2. 8 Menu Master

Jika memilih pilihan pertama menu master, maka akan ditampilkan menu kunjungan

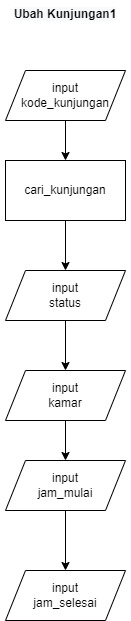


Gambar 2.2. 9 Menu Kunjungan

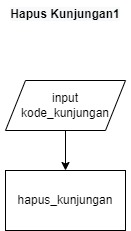
Dalam menu kunjungan terdapat beberapa pilihan menu, diantaranya :



Gambar 2.2. 10 Daftar Kunjungan 1

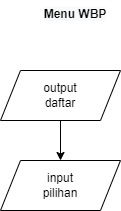


Gambar 2.2. 11 Ubah Kunjungan



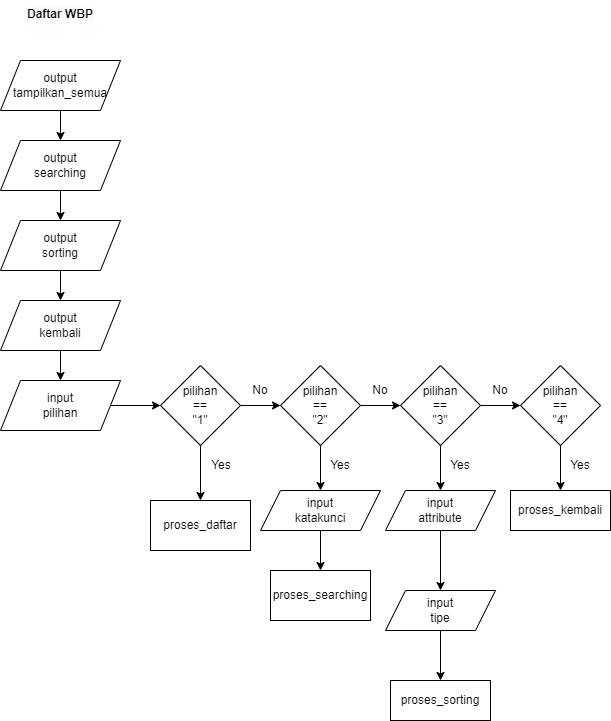
Gambar 2.2. 12 Hapus Kunjungan

Jika memilih pilihan kedua pada menu master, maka akan ditampilkan menu WBP



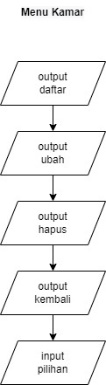
Gambar 2.2. 13 Menu WBP

Jika memilih pilihan pertama menu WBP, maka akan ditampilkan menu daftar WBP



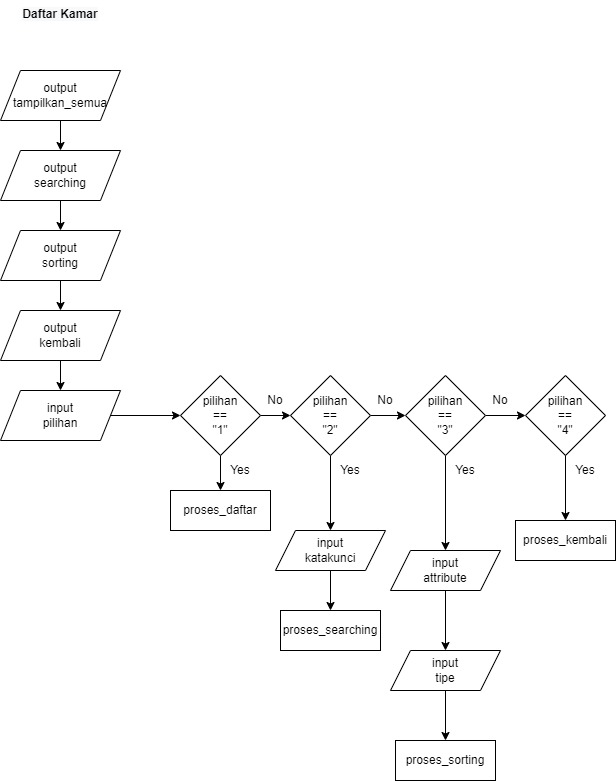
Gambar 2.2. 14 Daftar WBP

Jika memilih pilihan ketiga menu master, maka akan ditampilkan menu kamar

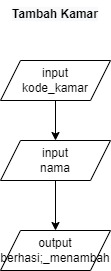


Gambar 2.2. 15 Menu Kamar

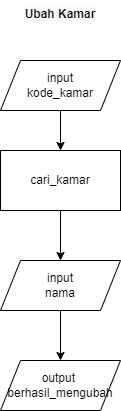
Dalam menu kamar terdapat beberapa pilihan menu, diantaranya :



Gambar 2.2. 16 Daftar Kamar

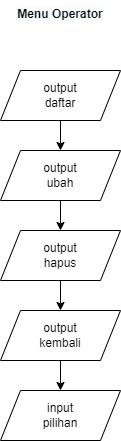


Gambar 2.2. 17 Tambah Kamar



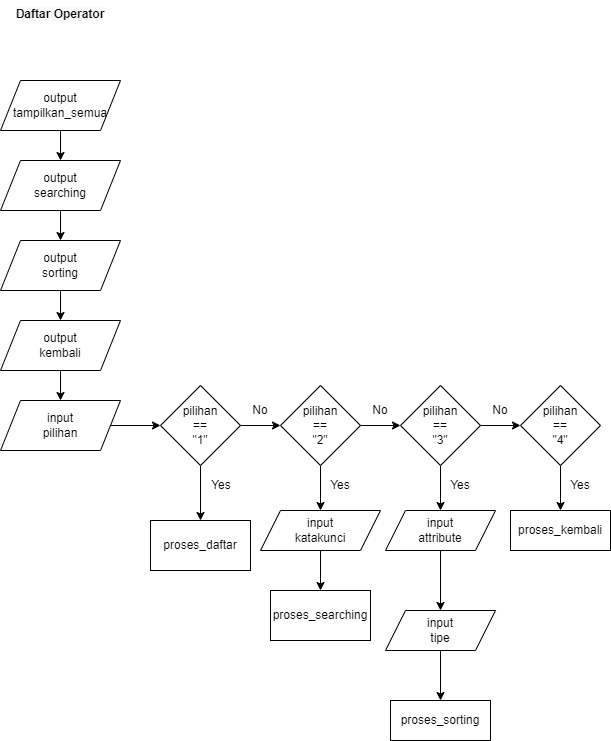
Gambar 2.2. 18 Ubah Kamar

Menu Operator

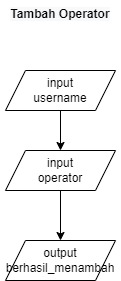


Gambar 2.2. 19 Menu Operator

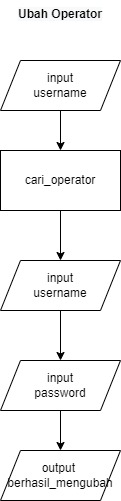
Dalam menu operaror terdapat pilihan menu, diantarnya :



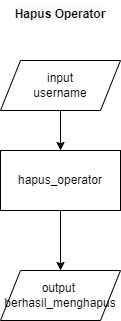
Gambar 2.2. 20 Daftar Operator



Gambar 2.2. 21 Tambah Operator



Gambar 2.2. 22 Ubah Operator



Gambar 2.2. 23 Hapus Operator

## **2.3 Konsep/Materi Praktikum yang dipakai**

Konsep yang kami gunakan dalam pembuatan laporan ini sesuai dengan persyaratan yang terdapat pada modul diantaranya :

1. Fungsi dasar : fungsi yang pada umumnya digunakan pada bahasa pemograman C++ adalah **#include <iostream>** yang merupakan arahan preprosesor yang mencakup konten file header C ++ standar iostream. Lalu ada **using namespace std;** yang digunakan untuk mengimpor keseluruhan “ namespace std “ ke namespace program saat ini. Selain itu terdapat juga **int main(){ }** baris yang mendeklarasi sebuah fungsi utama yang dimana tempat perintah eksekusi program dijalankan, fungsi ini dinamakan “main”. pada satu program , hanya boleh ada satu fungsi utama**.** Lalu kelompok kami juga memakai fungsi  **cout dan cin** yang merupakan sebuah perintah untuk masukan dan menampilkan pesan pada layer.
2. Tipe data : tipe data yang kami gunakan adalah tipe data primitive, tipe data kolektif, dan juga tipe data abstrak. Tipe data primitif adalah tipe data dasar yang digunakan untuk operasi dasar. Contohnya int, boolean, float, double, char, dll. Masingmasing tipe data mempunyai karakteristik dan range masing. Sedangkan tipe data kolektif adalah tipe data yang berupa rangkaian atau kumpulan data yang berindeks. Terdapat 3 jenis tipe data kolektif :

- Array

- List

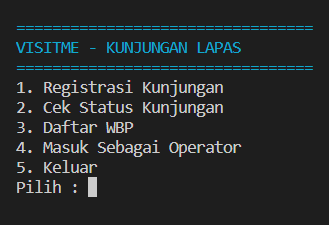
- Map

Lalu terdapat juga tipe data abstrak yang merupakan struct. Struct adalah sekumpulan variable bertipe data abstrak yang dinyatakan dengan sebuah nama dan bisa diciptakan secara dinamis.

1. Variable : Kelompok kami juga menggunakan variable yang merupakan placeholder atau wadah dalam memori (RAM) untuk menyimpan nilai format dekalrasi variable pada bahasa C++:
2. Operator : kelompok kami menggunakan operator perbandingan yang merupakan dua ekspresi yang dapat dibandingkan dengan menggunakan *Relational and Comparison Operators*.
3. Decision : Pada program kami menggunakan percabangan switch. Switch merupakan percabangan kode program dimana kita membandingkan isi sebuah variable dengan beberapa nilai.
4. Fungsi Perulangan : kami juga menggunakan perulangan while. Perulangan while adalah salah satu jenis perulangan di bahsa pemogrman C++ yang digunakan untuk melakukan perulangan dengan proses yang belum diketahui jumlahnya.
5. Array : Dalam program kami juga menggunakan Array. Array merupakan struktur data statis yang menyimpan sekumpulan elemen (data) dengan tipe data yang sama. Setiap elemen array dapat diakses langsung melalui indeks aray. Indeks tersebut memiliki tipe data yang menyatakan keterurutan misalnya integer atau karakter.
6. Pointer : Pointer adalah variabel yang berisi alamat memory sebagai nilainya dan berbeda dengan variabel biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variable yang mempunyai nilai tertentu.
7. Pada program kami juga menggunakan fungsi dan prosedur. Fungsi sendiri merupakan sebuah potongan kode yang ditujukan untuk menjalankan tugas yang spesifik, tugas tersebut membutuhkan informasi input yang disebut parameter dan biasanya suatu fungsi akan mengembalikan sebuah nilai (return value). Sedangkan prosedur adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai subprogram (program bagian).
8. Struct : Struct adalah pengelompokan variabel-variabel yang bernaung dalam satu nama yang sama. Berbeda dengan array yang berisi sekumpulan variabel-variabel yang bertipe sama dalam satu nama.
9. Sorting : Pada program kami menggunakan bubble sorting. Bubble Sort adalah metode pengurutan algoritma dengan cara melakukan penukaran data secara terus menerus sampai bisa dipastikan dalam suatu iterasi tertentu tidak ada lagi perubahan/penukaran. Algoritma ini menggunakan perbandingan dalam operasi antar elemennya.
10. Searching : Untuk pencarian data, kami menggunakan Linear search. Linear Search merupakan sebuah teknik pencarian data dengan menelusuri semua data satu per satu. Apabila ditemukan kecocokan data maka program akan mengembalikan output, jika tidak pencarian akan terus berlanjut hingga akhir dari array tersebut.
11. File : File digunakan agar hasil dari program yang tereksekusi dapat disimpan secara permanen di dalam perangkat penyimpanan mana pun, bukan sementara menghilang ketika program ditutup atau dihentikan. Tujuannya adalah agar data yang sudah tersimpan dapat digunakan kembali. Untuk memanipulasi file, biasa adalah file .txt. ada beberapa tipe data file.
12. Rekursif : Rekursi adalah algoritma yang melakukan pemanggilan terhadap dirinya sendiri. Algoritma yang bersifat rekursi disebut rekursif.
13. Vector : Vector pada C++ (atau biasa disebut menggunakan **std::vector**) adalah Array dinamis, yakni array yang proses memungkinkan proses insert dan delete element pada bagian tengah array dan “seakan-akan” mengubah ukuran array tersebut.
14. File Header : pada program kami menggunakan beberapa file header, diantaranya table.h
15. Map : pada program yang dibuat, kami juga menggunakan struktur data map yang mirip dengan array namun dengan index yang memungkinkan untuk berupa tipe data selain integer (mirip dengan dictionary di Python). Pada map, indeks tersebut diberi nama “key”.
16. Initializer\_List : pada sebuah object memiliki banyak cara, salah satunya adalah menggunakan constructor. Constructor memberi kemudahan dan fleksibilitas untuk melakukan inisialisasi pada object.

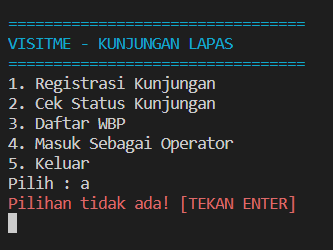
# BAB III HASIL DAN PEMBAHASAN

## **3.1 Tampilan Program**

1. Tampilan menu utama

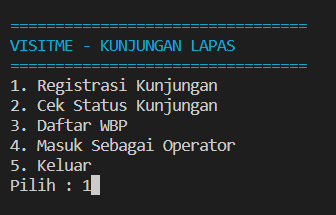
Gambar 3.1. Menu Utama

Pada saat pertama kali pengguna menjalankan program, pengguna akan melihat judul program yang merupakan tampilan utama dari program yang memberikan pilihan kepada pengguna untuk melakukan proses registrasi kunjungan untuk mendaftarkan diri sebagai pengunjung, mengecek status kunjungan, daftar WBP, serta login sebagai orang operator serta keluar dari program.



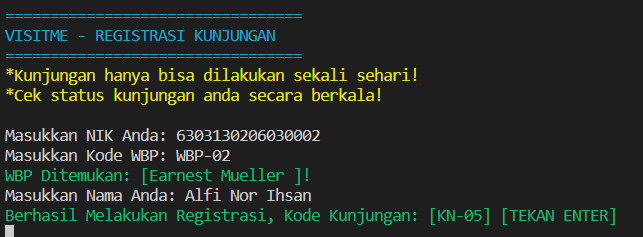
Gambar 3.1. 2 Salah Input

Dan saat pengguna memasukkan inputan yang salah, maka program yang akan mendeteksi kesalahan tersebut dan pengguna di minta menekan *enter* untuk melanjutkan program. Namun jika inputan yang di masukkan pengguna sesuai, maka akan di lanjutkan ke proses selanjutnya.

1. Tampilan registrasi

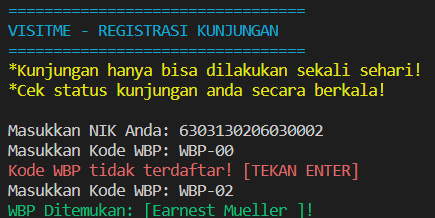
Gambar 3.1. Inputan

Pada saat user memilih untuk melakukan registrasi, maka user akan diminta memasukkan data berupa NIK dan nama kode WBP yang akan di kunjungi, serta nama pengguna, di mana terdapat syarat di mana kunjungan hanya dapat di lakukan sekali dalam 1 hari.

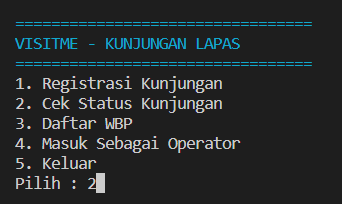


Gambar 3.1. Registrasi Kunjungan

Adapun saat pengguna salah melakukan input data kode WBP, Maka user akan di minta memasukkan data kode WBP yang benar hingga data sesuai dengan data kode WBP yang ada dan data telah terdaftar.

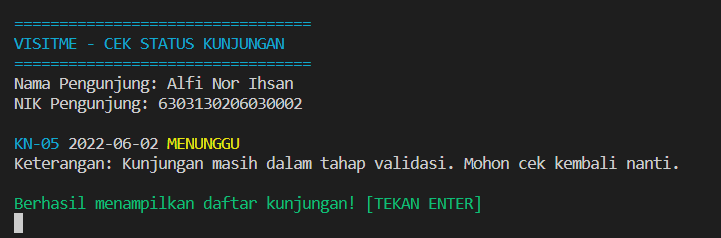


Gambar 3.1. 5 Registrasi tidak terdaftar

1. Tampilan cek status

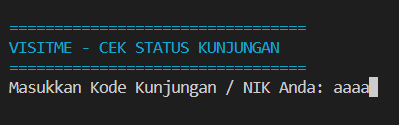
Gambar 3.1. Pilihan 2

Pada tampilan cek status, pengguna akan di minta untuk memasukkan NIK ataupun kode kunjungan yang akan di proses sehingga menampilkan status dari kunjungan tersebut.

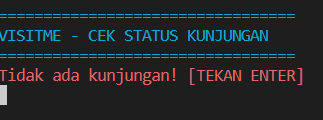


Gambar 3.1. 7 Status Kunjungan

Dan saat pengguna memasukkan data yang salah, maka pengguna akan di minta untuk mengulangi *inputan* dengan cara mengembalikan pengguna ke menu utama.

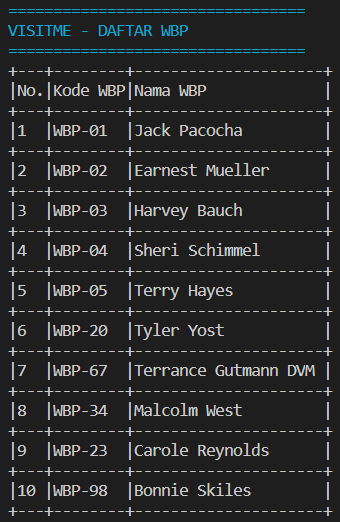


Gambar 3.1. 8 Status kunjungan Kode kunjungan



Gambar 3.1. 9 Tidak ada kunjungan

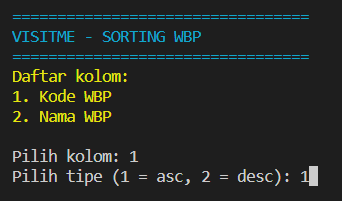
1. Tampilan data WBP



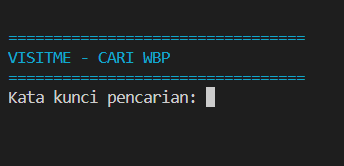
Gambar 3.1. Daftar WBP



Pada tampilan data WBP, penggunaan akan melihat seluruh WBP yang tersedia dengan pilihan pengurutan dan pencarian data dari data terbesar ke data terkecil ataupun sebaliknya dengan pembanding berdasarkan kode WBP ataupun Nama WBP, Mereset proses serta keluar dari menu data WBP.



Gambar 3.1. Pilihan Sorting WBP

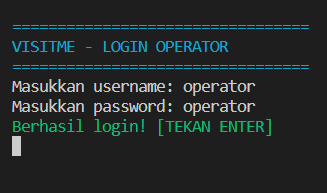
****

Gambar 3.1. 12 Kata Kunci Pencarian

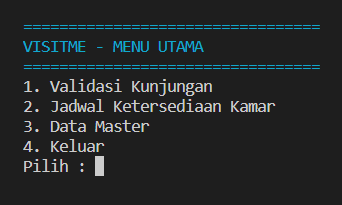
1. Tampilan masuk sebagai operator

Pada menu tampilan ini, operator bertugas untuk melakukan validasi dari proses registrasi yang di lakukan pengunjung, melihat jadwal ketersediaan kamar, berperan sebagai data master yang mengatur data berupa data kunjungan, WBP dan kamar serta menu keluar yang akan mengembalikan operator kembali ke menu utama.

Adapun operator perlu melakukan login terlebih dahulu hingga username dan password sesuai dan operator akan dapat mengatur berbagai data sebelumnya.

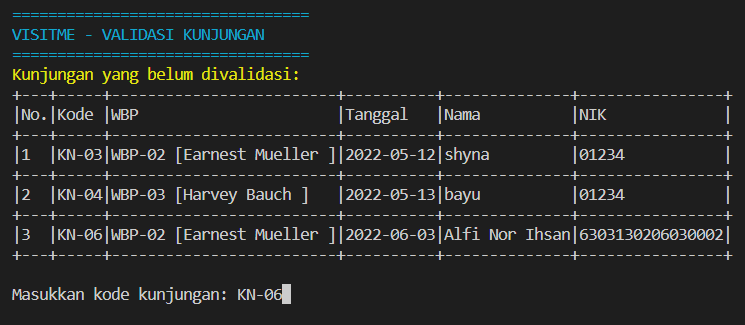


Gambar 3.1. 13 Login Operator

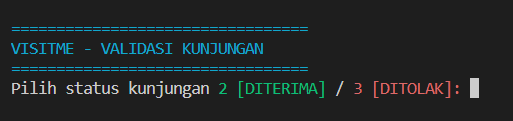


Gambar 3.1. 14 Menu Utama Validasi Kunjungan

Pada bagian validasi kunjungan, operator di tampilkan daftar permintaan kunjungan dan akan di tujukan pada pilihan untuk memberikan izin ataupun menolak permintaan kunjungan dan jika operator mengizinkan kunjungan, maka operator akan melanjutkan proses hingga pengunjung dapat melakukan kunjungan.



Gambar 3.1. 15 Validasi Kunjungan



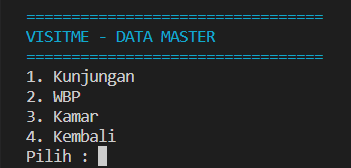
Gambar 3.1. 16 Status Kunjungan

Selanjutnya pada menu pilihan jadwal ketersediaan kamar, operator akan di berikan daftar kamar yang tersedia dan dapat di pakai.



Gambar 3.1. 17 Jadwal ketersediaan kamar

Dan pada tampilan menu data master, operator dapat melakukan banyak hal seperti yang dapat di lakukan pengguna biasa, namun sebagai operator.



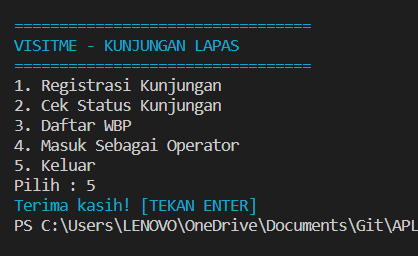
Gambar 3.1. 18 Data Master

Serta terkahir yaitu menu tampilan kembali, merupakan menu yang akan mengembalikan operator ke tampilan menu utama.

1. Tampilan keluar dari program

Tampilan keluar dari program merupakan menu yang di gunakan

untuk keluar dari program dan mengakhiri program tersebut.



Gambar 3.1. 19 Keluar

## **Source Code**

1. Program utama

|  |
| --- |
| #include <iostream>  #include "../include/utility.h"  #include "../include/kamar.h"  #include "../include/wbp.h"  #include "../include/kunjungan.h"  #include "../include/menu.h"  #include "../include/operator.h"  using namespace std;  int main() {    bool is\_running = true;    bool is\_wbp\_list\_running = false;    bool is\_main\_running = false;    bool is\_master\_running = false;    bool is\_login = false;    int guess\_choice, master\_choice , main\_choice, wbp\_list\_choice;    while(is\_running) {      guess\_choice = menu::guess();      switch (guess\_choice) {        case 1:          kunjungan::create();          break;        case 2:          kunjungan::check();          break;        case 3:          is\_wbp\_list\_running = true;          wbp::list();          while(is\_wbp\_list\_running) {            wbp\_list\_choice = menu::searching();;            switch (wbp\_list\_choice) {              case 1:                wbp::sort();                break;              case 2:                wbp::search();                break;              case 3:                wbp::list();                break;              case 4:                is\_wbp\_list\_running = false;                break;            }          }          break;        case 4:          is\_login = op::login();          if(is\_login) {            utility::notify("success", "Berhasil login!");            is\_main\_running = true;            while(is\_main\_running) {              main\_choice = menu::main();              switch (main\_choice) {                case 1:                  kunjungan::validate();                  break;                case 2:                  kamar::schedule();                  break;                case 3:                  is\_master\_running = true;                  while(is\_master\_running) {                    master\_choice = menu::master();                    switch (master\_choice) {                      case 1:                        kunjungan::index();                        break;                      case 2:                        wbp::index();                        break;                      case 3:                        kamar::index();                        break;                      case 4:                        is\_master\_running = false;                        break;                    }                  }                  break;                case 4:                  is\_main\_running = false;                  utility::notify("info", "Terima kasih!");                  break;              }            }          } else {            utility::notify("error", "Akun tidak terdaftar!");          }          break;        case 5:          is\_running = false;          utility::notify("info", "Terima kasih!");        default:          if(guess\_choice != 5) {            utility::notify("error", "Pilihan tidak ada!");          }          break;      }    }    return 0;  } |

1. Include

* data.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <string>  #include <map>  using namespace std;  namespace data {    map<string, string> colors = {      { "black", "\033[30m" },      { "red", "\033[31m" },      { "green", "\033[32m" },      { "yellow", "\033[33m" },      { "blue", "\033[34m" },      { "cyan", "\033[36m" },      { "white", "\033[37m" }    };    map<string, int> status\_kunjungan = {      { "menunggu", 1 },      { "diterima", 2 },      { "ditolak", 3 },    };    map<int, string> label\_status\_kunjungan = {      { 1, "MENUNGGU" },      { 2, "DITERIMA" },      { 3, "DITOLAK " },    };    map<int, string> color\_status\_kunjungan = {      { 1, "yellow" },      { 2, "green" },      { 3, "red" },    };    map<string, string> operational = {      { "start", "09:00" },      { "end", "16:00" }    };  } |

* kamar.h

|  |
| --- |
| #pragma once  #include <stdio.h>  #include <fstream>  #include <iostream>  #include <vector>  #include <sstream>  #include <string>  #include <tuple>  #include "./utility.h"  #include "./struct.h"  #include "./table.h"  #include "./menu.h"  using namespace std;  namespace kamar {      const string PATH = "../files/kamar.csv";    const int TABLE\_COLUMNS\_LENGTH = 3;    string TABLE\_COLUMNS[] = {"No.", "Kode Kamar", "Nama Kamar"};      string path() {      return PATH;    }    void list() {      utility::header("VISITME - DAFTAR KAMAR");      vector<vector<string>> content = utility::list(PATH);      TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, content.size(), TABLE\_COLUMNS, content);      cout << table;    }    structure::kamar get(vector<string> data) {      structure::kamar kamar;      kamar.kode = data[0];      kamar.nama = data[1];      return kamar;    }    void store(structure::kamar kamar) {      fstream file;      file.open(PATH, ios::app);      file<< kamar.kode << ","          << kamar.nama << "\n";      file.close();    }    void create() {      structure::kamar kamar;      utility::header("VISITME - TAMBAH KAMAR");      cout << "Kode Kamar: "; cin >> kamar.kode;      cout << "Nama Kamar: "; cin >> kamar.nama;      vector<vector<string>> list = utility::search(PATH, { 0 }, kamar.kode);      if(list.size() < 1) {        kamar::store(kamar);        utility::notify("success", "Kamar berhasil ditambahkan!");      } else {        utility::notify("error", "Kamar dengan kode berikut sudah ada!");      }    }    void update(string identifier, structure::kamar kamar) {      string data[] = {kamar.kode, kamar.nama};      utility::update(PATH, 0, 2, identifier, data);    }    void edit() {      string code;      utility::header("VISITME - UBAH KAMAR");      cout << "Masukkan Kode Kamar: "; cin >> code;        vector<vector<string>> list = utility::search(PATH, { 0 }, code);      if(list.size() > 0) {        utility::notify("success", "Kamar ditemukan");        utility::header("VISITME - UBAH KAMAR");        vector<string> old\_kamar = list.front();        structure::kamar new\_kamar;        utility::cout("yellow", "\*Apabila tidak ada perubahan maka isi dengan '-'!");        cout << "Nama Kamar [" + old\_kamar[1] + "]: "; cin >> new\_kamar.nama;        new\_kamar.kode = old\_kamar[1];        new\_kamar.nama = new\_kamar.nama != "-" ? new\_kamar.nama : old\_kamar[1];        kamar::update(code, new\_kamar);        utility::notify("success", "Kamar berhasil diubah!");      } else {        utility::notify("error", "Kamar dengan kode tersebut tidak ada!");      }    }    void destroy() {      string code;      bool is\_confirmed;      utility::header("VISITME - HAPUS KAMAR");      cout << "Kode Kamar: "; cin >> code;        vector<vector<string>> list = utility::search(PATH, { 0 }, code);      if(list.size() > 0) {        is\_confirmed = utility::confirm("kamar");        if(is\_confirmed) {          utility::destroy(PATH, 0, 2, code);          utility::notify("success", "Kamar berhasil dihapus!");        }      } else {        utility::notify("error", "Kamar dengan kode tersebut tidak ada!");      }    }    void sort() {      int column, type;      utility::header("VISITME - SORTING KAMAR");      tie(column, type) = menu::sorting(TABLE\_COLUMNS, TABLE\_COLUMNS\_LENGTH);      column = column - 1;      if(column >= 0 && column < TABLE\_COLUMNS\_LENGTH) {        vector<vector<string>> list = utility::sort(PATH, column, type);        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Pilihan kolom tidak ada!");      }    }    void search() {      string keyword;      utility::header("VISITME - CARI KAMAR");      cout << "Kata kunci pencarian: "; cin >> keyword;      vector<vector<string>> list = utility::search(PATH, { 0 }, keyword, true);      if(list.size() > 0) {        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Data tidak ditemukan!");      }    }    void schedule() {      string today = utility::today();      string start\_time = data::operational["start"];      string end\_time = data::operational["end"];      vector<vector<string>> list = utility::list(PATH);      int total\_hours = stoi(end\_time) - stoi(start\_time);      vector<string> hours;      hours.push\_back(start\_time);      int cur\_hour = stoi(start\_time);      for(int hourIdx; hourIdx < total\_hours; hourIdx++) {        cur\_hour = cur\_hour + 1;        string hour = cur\_hour < 10 ? "0" + to\_string(cur\_hour) : to\_string(cur\_hour);        hours.push\_back(hour + ":00");      }      hours.push\_back(end\_time);        string columns[20];      columns[0] = "No.";      columns[1] = "Nama Kamar";      for(int colIdx = 0; colIdx < hours.size(); colIdx++) {        columns[2 + colIdx] = hours[colIdx];      }      vector<vector<string>> today\_visits = utility::search("../files/kunjungan.csv", { 2 }, today);      vector<vector<string>> schedules;      for(int roomIdx = 0; roomIdx < list.size(); roomIdx++) {        vector<string> schedule;        structure::kamar kamar = kamar::get(list[roomIdx]);          schedule.push\_back(kamar.nama);        vector<vector<string>> this\_room\_visits;        for(int visitIdx = 0; visitIdx < today\_visits.size(); visitIdx++) {          vector<string> today\_visit = today\_visits[visitIdx];          if(today\_visit[5] == "2" && today\_visit[8] == kamar.kode) {            this\_room\_visits.push\_back(today\_visit);          }        }        for(int colIdx = 0; colIdx < hours.size(); colIdx++) {          bool is\_found = false;          string hour = hours[colIdx];            for(int visitIdx = 0; visitIdx < this\_room\_visits.size(); visitIdx++) {            vector<string> visit = this\_room\_visits[visitIdx];            bool isInsideStart = utility::isTimeLater(hour, visit[6]);            bool isInsideEnd = utility::isTimeLater(visit[7], hour);            if(isInsideStart && isInsideEnd) {              schedule.push\_back(visit[0]);              is\_found = true;            }          }          if(!is\_found) {            schedule.push\_back("-");          }        }        schedules.push\_back(schedule);      }      utility::header("VISITME - JADWAL KETERSEDIAAN KAMAR");      utility::cout("yellow", "Tanggal Hari Ini: " + today);      TextTable table = utility::table(hours.size() + 1, schedules.size(), columns, schedules);      cout << table;      cout << endl;      utility::notify("success", "Untuk Kembali");    }    void index() {      bool is\_running = true;      bool is\_list\_running = false;      int choice, list\_choice;      while(is\_running) {        choice = menu::kamar();        switch (choice) {          case 1:            is\_list\_running = true;            kamar::list();            while(is\_list\_running) {              list\_choice = menu::searching();              switch (list\_choice) {                case 1:                  kamar::sort();                  break;                case 2:                  kamar::search();                  break;                case 3:                  kamar::list();                  break;                case 4:                  is\_list\_running = false;                  break;              }            }            break;          case 2:            kamar::create();            break;          case 3:            kamar::edit();            break;          case 4:            kamar::destroy();            break;          case 5:            is\_running = false;            break;          default:            utility::notify("error", "Pilihan tidak ada!");            break;        }      }    }  } |

* kunjungan.h

|  |
| --- |
| #pragma once  #include <stdio.h>  #include <fstream>  #include <iostream>  #include <cstdio>  #include <ctime>  #include <tuple>  #include "./utility.h"  #include "./struct.h"  #include "./wbp.h"  #include "./menu.h"  #include "./kamar.h"  using namespace std;  namespace kunjungan {    const string PATH = "../files/kunjungan.csv";    const int TABLE\_COLUMNS\_LENGTH = 11;    string TABLE\_COLUMNS[] = {"No.", "Kode", "WBP", "Tanggal", "Nama", "NIK", "Status", "Jam Mulai", "Jam Selesai", "Kode Kamar", "Catatan"};    string path() {      return PATH;    }    vector<vector<string>> formatter(vector<vector<string>> list) {      vector<vector<string>> formatted;      for(int rowIdx = 0; rowIdx < list.size(); rowIdx++) {        vector<string> row = list[rowIdx];        string col\_data;        for(int colIdx = 0; colIdx < TABLE\_COLUMNS\_LENGTH - 1; colIdx++) {          col\_data = row[colIdx];          if(colIdx == 1) {            vector<string> wbp = utility::find(wbp::path(), { 0 }, col\_data);            row[colIdx] = col\_data + " [" + wbp[1] + "]";          }          if(colIdx == 5) {            int status = stoi(col\_data);            row[colIdx] = data::label\_status\_kunjungan[status];          }          if(colIdx == 8) {            if(col\_data != "-") {              vector<string> kamar = utility::find(kamar::path(), { 0 }, col\_data);              row[colIdx] = col\_data + " [" + kamar[1] + "]";            }          }        }          formatted.push\_back(row);      }      return formatted;    }    void list() {      utility::header("VISITME - DAFTAR KUNJUNGAN");      vector<vector<string>> content = utility::list(PATH);      vector<vector<string>> list = kunjungan::formatter(content);      TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);      cout << table;    }    structure::kunjungan get(vector<string> data) {      string path\_wbp = wbp::path();      string path\_kamar = kamar::path();        structure::kunjungan kunjungan;      kunjungan.kode = data[0];      kunjungan.kode\_wbp = data[1];      kunjungan.tanggal = data[2];      kunjungan.nama\_pengunjung = data[3];      kunjungan.nik\_pengunjung = data[4];      kunjungan.status = stoi(data[5]);      kunjungan.jam\_mulai = data[6];      kunjungan.jam\_selesai = data[7];      kunjungan.kode\_kamar = data[8];      kunjungan.catatan = data[9];      vector<string> wbp = utility::find(path\_wbp , { 0 }, kunjungan.kode\_wbp);      kunjungan.wbp.kode = wbp[0];      kunjungan.wbp.nama = wbp[1];      if(kunjungan.kode\_kamar != "-") {        vector<string> kamar = utility::find(path\_kamar , { 0 }, kunjungan.kode\_kamar);        kunjungan.kamar.kode = kamar[0];        kunjungan.kamar.nama = kamar[1];      }      return kunjungan;    }    void store(structure::kunjungan kunjungan) {      fstream fout;      string today = utility::today();      vector<string> latest = utility::latest(PATH);      int number = latest.empty() ? 0 : stoi(latest[0].substr(3, latest[0].find("-")));      string next\_number = number > 9 ? to\_string(number + 1) : "0" + to\_string(number + 1);      kunjungan.kode = "KN-" + next\_number;      kunjungan.tanggal = today;      kunjungan.status = data::status\_kunjungan["menunggu"];      kunjungan.jam\_mulai = "-";      kunjungan.jam\_selesai = "-";      kunjungan.kode\_kamar = "-";      kunjungan.catatan = "-";      fout.open(PATH, ios::out | ios::app);      fout<< kunjungan.kode << ","          << kunjungan.kode\_wbp << ","          << kunjungan.tanggal << ","          << kunjungan.nama\_pengunjung << ","          << kunjungan.nik\_pengunjung << ","          << kunjungan.status << ","          << kunjungan.jam\_mulai << ","          << kunjungan.jam\_selesai << ","          << kunjungan.kode\_kamar << ","          << kunjungan.catatan << "\n";      fout.close();      utility::notify("success", "Berhasil Melakukan Registrasi, Kode Kunjungan: [" + kunjungan.kode + "]");    }    structure::wbp get\_wbp() {      vector<string> wbp;      string kode\_wbp;        cout << "Masukkan Kode WBP: "; cin >> kode\_wbp;      wbp = utility::find(wbp::path(), { 0 }, kode\_wbp);      if(wbp.empty()) {        utility::notify("error", "Kode WBP tidak terdaftar!");        get\_wbp();      }      utility::cout("green", "WBP Ditemukan: [" + wbp[1] + "]!");      structure::wbp found\_wbp = wbp::get(wbp);      return found\_wbp;    }    void create() {      bool is\_not\_exist = true;      bool check\_wbp = false;      string today = utility::today();      structure::kunjungan kunjungan;      structure::wbp wbp;      utility::header("VISITME - REGISTRASI KUNJUNGAN");      utility::cout("yellow", "\*Kunjungan hanya bisa dilakukan sekali sehari!");      utility::cout("yellow", "\*Cek status kunjungan anda secara berkala!\n");      cout << "Masukkan NIK Anda: "; cin >> kunjungan.nik\_pengunjung;      // \*\* CHECK IF USER ALREADY REGIST TODAY      vector<vector<string>> list = utility::search(PATH, { 4 }, kunjungan.nik\_pengunjung);      for(int index = 0; index < list.size(); index++) {        vector<string> row = list[index];        if(row[2] == today) {          is\_not\_exist = false;        }      }      if (is\_not\_exist) {        wbp = get\_wbp();        kunjungan.kode\_wbp = wbp.kode;        cout << "Masukkan Nama Anda: "; fflush(stdin);        getline(cin, kunjungan.nama\_pengunjung);        kunjungan::store(kunjungan);      } else {        utility::notify("error", "Anda sudah melakukan registrasi!");      }    }    void check() {      string keyword;      utility::header("VISITME - CEK STATUS KUNJUNGAN");      cout << "Masukkan Kode Kunjungan / NIK Anda: "; cin >> keyword;      vector<vector<string>> list = utility::search(PATH, { 0, 4 }, keyword);      utility::header("VISITME - CEK STATUS KUNJUNGAN");      if(list.size() > 0) {        for(int index = 0; index < list.size(); index++) {          string description;          vector<string> row = list[index];          structure::kunjungan kunjungan = kunjungan::get(row);          if(index == 0) {            cout << "Nama Pengunjung: " << kunjungan.nama\_pengunjung << endl;            cout << "NIK Pengunjung: " << kunjungan.nik\_pengunjung << endl << endl;          }          string label\_status = data::label\_status\_kunjungan[kunjungan.status];          string label\_color = data::color\_status\_kunjungan[kunjungan.status];          switch (kunjungan.status) {            case 1:              description = "Kunjungan masih dalam tahap validasi. Mohon cek kembali nanti.";              break;            case 2:              description = "Kamar: " + kunjungan.kamar.nama +  ", Jam: " + kunjungan.jam\_mulai + " - " + kunjungan.jam\_selesai;              break;            case 3:              description = kunjungan.catatan;              break;          }          utility::cout("cyan", kunjungan.kode, false);          cout << " " << kunjungan.tanggal << " ";          utility::cout(label\_color, label\_status);          cout << "Keterangan: " << description << endl << endl;        }        utility::notify("success", "Berhasil menampilkan daftar kunjungan!");      } else {        utility::notify("error", "Tidak ada kunjungan!");      }    }    vector<vector<string>> validate\_formatter(vector<vector<string>> list) {      vector<vector<string>> formatted;      for(int rowIdx = 0; rowIdx < list.size(); rowIdx++) {        vector<string> row = list[rowIdx];        vector<string> wbp = utility::find(wbp::path(), { 0 }, row[1]);        row[1] = row[1] + " [" + wbp[1] + "]";        formatted.push\_back(row);      }      return formatted;    }    tuple<structure::kamar, string, string> get\_availability(string kode\_kunjungan) {        string kode\_kamar, jam\_mulai, jam\_selesai;      string today = utility::today();      cout << "Masukkan Kode kamar: "; cin >> kode\_kamar;      vector<string> kamar = utility::find(kamar::path(), { 0 }, kode\_kamar);      // \*\* CHECK IF IS THERE A ROOM      if(kamar.empty()) {        utility::notify("error", "Kode kamar tidak terdaftar!");        get\_availability(kode\_kunjungan);      }      // \*\* GET START AND AND TIME      cout << "Masukkan jam mulai (hh:mm): "; cin >> jam\_mulai;      cout << "Masukkan jam selesai (hh:mm): "; cin >> jam\_selesai;      // \*\* GET TODAY VISITS      structure::kamar found\_kamar = kamar::get(kamar);      vector<vector<string>> today\_visits = utility::search(PATH, { 2 }, today);      if(!today\_visits.empty()) {          for(int visitIdx = 0; visitIdx < today\_visits.size(); visitIdx++) {          structure::kunjungan curr\_today\_visit = kunjungan::get(today\_visits[visitIdx]);          if(curr\_today\_visit.kode != kode\_kunjungan && curr\_today\_visit.status == 2 && curr\_today\_visit.kode\_kamar == found\_kamar.kode) {            bool isInsideStart = utility::isTimeLater(jam\_mulai, curr\_today\_visit.jam\_mulai);            bool isInsideEnd = utility::isTimeLater(curr\_today\_visit.jam\_selesai, jam\_selesai);            if(isInsideStart && isInsideEnd)  {              utility::notify("error", "Kamar sudah terisi!");              get\_availability(kode\_kunjungan);            }          }        }      }      utility::cout("green", "Kamar ditemukan dan tersedia: [" + kamar[1] + "]!");      return make\_tuple(found\_kamar, jam\_mulai, jam\_selesai);    }    void update(string identifier, structure::kunjungan kunjungan) {      string data[] = {          kunjungan.kode,          kunjungan.kode\_wbp,          kunjungan.tanggal,          kunjungan.nama\_pengunjung,          kunjungan.nik\_pengunjung,          to\_string(kunjungan.status),          kunjungan.jam\_mulai,          kunjungan.jam\_selesai,          kunjungan.kode\_kamar,          kunjungan.catatan        };        utility::update(PATH, 0, 10, identifier, data);    }    void validate() {      utility::header("VISITME - VALIDASI KUNJUNGAN");      string kode;      string validate\_table\_columns[] = {"No.", "Kode", "WBP", "Tanggal", "Nama", "NIK"};      int validate\_columns\_length = 6;      vector<vector<string>> content = utility::search(PATH, { 5 }, "1");      vector<vector<string>> list = kunjungan::validate\_formatter(content);      TextTable table = utility::table(validate\_columns\_length, list.size(), validate\_table\_columns, list);      utility::cout("yellow", "Kunjungan yang belum divalidasi: ");      cout << table << endl;      cout << "Masukkan kode kunjungan: "; cin >> kode;      vector<string> kunjungan\_raw = utility::find(PATH, { 0 }, kode);      if(!kunjungan\_raw.empty()) {        structure::kunjungan kunjungan = kunjungan::get(kunjungan\_raw);        structure::kamar kamar;        utility::notify("success", "Kunjungan ditemukan!");        utility::header("VISITME - VALIDASI KUNJUNGAN");        string status, jam\_mulai, jam\_selesai, kode\_kamar, catatan;        bool is\_status\_validate\_pass, is\_room\_not\_available = true;        while(!is\_status\_validate\_pass) {          utility::cout("white", "Pilih status kunjungan ", false);          utility::cout("green", "2 [DITERIMA] ", false);          utility::cout("white", "/", false);          utility::cout("red", " 3 [DITOLAK]: ", false);          cin >> status;          if(status == "2" || status == "3") {            is\_status\_validate\_pass = true;          }        }        if(status == "2") {          tie(kamar, jam\_mulai, jam\_selesai) = get\_availability(kode);          kunjungan.kode\_kamar = kamar.kode;          kunjungan.jam\_mulai = jam\_mulai;          kunjungan.jam\_selesai = jam\_selesai;          kunjungan.status = 2;          utility::notify("success", "Kunjungan berhasil divalidasi");        } else {          cout << "Masukkan catatan penolakan: "; cin >>catatan;          kunjungan.status = 3;          kunjungan.catatan = catatan;          utility::notify("success", "Kunjungan berhasil ditolak!");        }        kunjungan::update(kunjungan.kode, kunjungan);        } else {        utility::notify("error", "Kode Kunjungan Tidak Ada!");      }      }    void edit() {      string code;      utility::header("VISITME - UBAH KUNJUNGAN");      utility::cout("yellow", "\*Kunjungan yang bisa diubah hanya kunjungan yang sudah berlalu!");      cout << "Masukkan Kode Kunjungan: "; cin >> code;        vector<string> kunjungan\_raw = utility::find(PATH, { 0 }, code);      if(!kunjungan\_raw.empty()) {        utility::notify("success", "Kunjungan ditemukan");        utility::header("VISITME - UBAH KUNJUNGAN");        structure::kunjungan old\_kunjungan = kunjungan::get(kunjungan\_raw);        structure::kunjungan new\_kunjungan = old\_kunjungan;        utility::cout("yellow", "\*Apabila tidak ada perubahan maka isi dengan '-'!");        cout << "Nama Pengunjung [" + old\_kunjungan.nama\_pengunjung + "]: "; cin >> new\_kunjungan.nama\_pengunjung;        cout << "NIK Pengunjung [" + old\_kunjungan.nik\_pengunjung + "]: "; cin >> new\_kunjungan.nik\_pengunjung;          new\_kunjungan.nama\_pengunjung = new\_kunjungan.nama\_pengunjung != "-" ? new\_kunjungan.nama\_pengunjung : old\_kunjungan.nama\_pengunjung;        new\_kunjungan.nik\_pengunjung = new\_kunjungan.nik\_pengunjung != "-" ? new\_kunjungan.nik\_pengunjung : old\_kunjungan.nik\_pengunjung;        kunjungan::update(old\_kunjungan.kode, new\_kunjungan);        utility::notify("success", "Kunjungan berhasil diubah!");      } else {        utility::notify("error", "Kunjungan dengan kode tersebut tidak ada!");      }    }    void destroy() {      string code;      bool is\_confirmed;      utility::header("VISITME - HAPUS KUNJUNGAN");      cout << "Kode Kunjungan: "; cin >> code;        vector<vector<string>> list = utility::search(PATH, { 0 }, code);      if(list.size() > 0) {        is\_confirmed = utility::confirm("kunjungan");        if(is\_confirmed) {          utility::destroy(PATH, 0, 2, code);          utility::notify("success", "Kunjungan berhasil dihapus!");        }      } else {        utility::notify("error", "Kunjungan dengan kode tersebut tidak ada!");      }    }    void sort() {      int column, type;      utility::header("VISITME - SORTING KUNJUNGAN");      tie(column, type) = menu::sorting(TABLE\_COLUMNS, TABLE\_COLUMNS\_LENGTH);      column = column - 1;      if(column >= 0 && column < TABLE\_COLUMNS\_LENGTH) {        vector<vector<string>> list = utility::sort(PATH, column, type);        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Pilihan kolom tidak ada!");      }    }    void search() {      string keyword;      utility::header("VISITME - CARI KUNJUNGAN");      cout << "Kata kunci pencarian: "; cin >> keyword;      vector<vector<string>> list = utility::search(PATH, { 0 }, keyword, true);      if(list.size() > 0) {        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Data tidak ditemukan!");      }    }    void index() {      bool is\_running = true;      bool is\_list\_running = false;      int choice, list\_choice;      while(is\_running) {        choice = menu::kunjungan();        switch (choice) {          case 1:            is\_list\_running = true;            kunjungan::list();            while(is\_list\_running) {              list\_choice = menu::searching();              switch (list\_choice) {                case 1:                  kunjungan::sort();                  break;                case 2:                  kunjungan::search();                  break;                case 3:                  kunjungan::list();                  break;                case 4:                  is\_list\_running = false;                  break;              }            }            break;          case 2:            kunjungan::edit();            break;          case 3:            kunjungan::destroy();            break;          case 4:            is\_running = false;            break;          default:            utility::notify("error", "Pilihan tidak ada!");            break;        }      }    }  } |

* menu.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <tuple>  #include <ctype.h>  #include "../include/utility.h"  using namespace std;  namespace menu {    int check(string choice) {      try {        int int\_choice = stoi(choice);        return int\_choice;      }      catch (...) {        cin.clear();        return 0;      }    }    // \*\* FOR THE MAIN    int guess() {      string choice;      utility::header("VISITME - KUNJUNGAN LAPAS");      cout  << "1. Registrasi Kunjungan" << endl            << "2. Cek Status Kunjungan" << endl            << "3. Daftar WBP" << endl            << "4. Masuk Sebagai Operator" << endl            << "5. Keluar" << endl            << "Pilih : "; cin >> choice;      return check(choice);      }    int main() {      string choice;      utility::header("VISITME - MENU UTAMA");      cout  << "1. Validasi Kunjungan" << endl            << "2. Jadwal Ketersediaan Kamar" << endl            << "3. Data Master" << endl            << "4. Keluar" << endl            << "Pilih : "; cin >> choice;      return check(choice);    }    int master() {      string choice;      utility::header("VISITME - DATA MASTER");      cout  << "1. Kunjungan" << endl            << "2. WBP" << endl            << "3. Kamar" << endl            << "4. Kembali" << endl            << "Pilih : "; cin >> choice;      return check(choice);    }    // \*\* FOR THE ENTITIES    int kamar() {      string choice;      utility::header("VISITME - MANAJEMEN KAMAR");      cout  << "1. Daftar Kamar" << endl            << "2. Tambah Kamar" << endl            << "3. Ubah Kamar" << endl            << "4. Hapus Kamar" << endl            << "5. Kembali" << endl            << "Pilih : "; cin >> choice;      return check(choice);    }    int kunjungan() {      string choice;      utility::header("VISITME - MANAJEMEN KAMAR");      cout  << "1. Daftar Kunjungan" << endl            << "2. Ubah Kunjungan" << endl            << "3. Hapus Kunjungan" << endl            << "4. Kembali" << endl            << "Pilih : "; cin >> choice;      return check(choice);    }    int wbp() {      string choice;      utility::header("VISITME - MANAJEMEN WBP");      cout  << "1. Daftar WBP" << endl            << "2. Keluar" << endl            << "Pilih : "; cin >> choice;      return check(choice);    }    // \*\* FOR OTHERS    int searching() {      string choice;      utility::cout("black", "\nApa yang ingin anda lakukan :");      utility::cout("yellow", "1. Sorting     2. Searching    3. Reset    4. Kembali");      cout << "Pilih : "; cin >> choice;      return check(choice);    }    tuple<int, int> sorting(string columns[], int length) {      int column, type;      utility::cout("yellow", "Daftar kolom: ");      for(int i = 1; i < length; i++) {        utility::cout("yellow", to\_string(i) + ". " + columns[i]);      }      cout << "\nPilih kolom: "; cin >> column;      cout << "Pilih tipe (1 = asc, 2 = desc): "; cin >> type;      return make\_tuple(column, type);    }  } |

* operator.h

|  |
| --- |
| #include <stdio.h>  #include <fstream>  #include <iostream>  #include <vector>  #include <sstream>  #include <string>  #include <tuple>  #include "./utility.h"  #include "./struct.h"  #include "./table.h"  #include "./menu.h"  using namespace std;  namespace op {      const string PATH = "../files/operator.csv";      bool login() {      bool is\_login = false;      string username, password;      string cur\_password, cur\_username, hashed;      utility::header("VISITME - LOGIN OPERATOR");      cout << "Masukkan username: "; cin >> username;      cout << "Masukkan password: "; cin >> password;      vector<vector<string>> list = utility::list(PATH);      for(int row = 0; row < list.size(); row++) {        cur\_password = utility::toLower(list[row][1]);        cur\_username = utility::toLower(list[row][0]);        hashed = to\_string(utility::hash(password));        if(cur\_username == username && cur\_password == hashed) {          is\_login = true;        }      }      return is\_login;    }  } |

* struct.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <string>  using namespace std;  namespace structure {    struct kamar {      string kode;      string nama;    };    struct wbp {      string kode;      string nama;    };    struct kunjungan {      string kode;      string kode\_wbp;      string kode\_kamar;      string tanggal;      string nama\_pengunjung;      string nik\_pengunjung;      int status;      string jam\_mulai;      string jam\_selesai;      string catatan;      kamar kamar;      wbp wbp;    };  } |

* table.h

|  |
| --- |
| // https://github.com/haarcuba/cpp-text-table  #pragma once  #include <iomanip>  #include <iostream>  #include <map>  #include <string>  #include <vector>  #ifdef TEXTTABLE\_ENCODE\_MULTIBYTE\_STRINGS  #include <clocale>  #ifndef TEXTTABLE\_USE\_EN\_US\_UTF8  #define TEXTTABLE\_USE\_EN\_US\_UTF8  #endif  #endif  class TextTable {  public:    enum class Alignment { LEFT, RIGHT };    typedef std::vector<std::string> Row;    TextTable()        : \_horizontal('-'), \_vertical('|'), \_corner('+'), \_has\_ruler(true) {}    TextTable(char horizontal, char vertical, char corner)        : \_horizontal(horizontal), \_vertical(vertical), \_corner(corner),          \_has\_ruler(true) {}    explicit TextTable(char vertical)        : \_horizontal('\0'), \_vertical(vertical), \_corner('\0'),          \_has\_ruler(false) {}    void setAlignment(unsigned i, Alignment alignment) {      \_alignment[i] = alignment;    }    Alignment alignment(unsigned i) const { return \_alignment[i]; }    char vertical() const { return \_vertical; }    char horizontal() const { return \_horizontal; }    void add(const std::string& content) { \_current.push\_back(content); }    void endOfRow() {      \_rows.push\_back(\_current);      \_current.assign(0, "");    }    template <typename Iterator> void addRow(Iterator begin, Iterator end) {      for (auto i = begin; i != end; ++i) {        add(\*i);      }      endOfRow();    }    template <typename Container> void addRow(const Container& container) {      addRow(container.begin(), container.end());    }    const std::vector<Row>& rows() const { return \_rows; }    void setup() const {      determineWidths();      setupAlignment();    }    std::string ruler() const {      std::string result;      result += \_corner;      for (auto width = \_width.begin(); width != \_width.end(); ++width) {        result += repeat(\*width, \_horizontal);        result += \_corner;      }      return result;    }    int width(unsigned i) const { return \_width[i]; }    bool has\_ruler() const { return \_has\_ruler; }    int correctDistance(const std::string& string\_to\_correct) const {      return static\_cast<int>(string\_to\_correct.size()) -             static\_cast<int>(glyphLength(string\_to\_correct));    };  private:    const char \_horizontal;    const char \_vertical;    const char \_corner;    const bool \_has\_ruler;    Row \_current;    std::vector<Row> \_rows;    std::vector<unsigned> mutable \_width;    std::vector<unsigned> mutable \_utf8width;    std::map<unsigned, Alignment> mutable \_alignment;    static std::string repeat(unsigned times, char c) {      std::string result;      for (; times > 0; --times)        result += c;      return result;    }    unsigned columns() const { return \_rows[0].size(); }    unsigned glyphLength(const std::string& s) const {      unsigned int \_byteLength = s.length();  #ifdef TEXTTABLE\_ENCODE\_MULTIBYTE\_STRINGS  #ifdef TEXTTABLE\_USE\_EN\_US\_UTF8      std::setlocale(LC\_ALL, "en\_US.utf8");  #else  #error You need to specify the encoding if the TextTable library uses multybyte string encoding!  #endif      unsigned int u = 0;      const char \*c\_str = s.c\_str();      unsigned \_glyphLength = 0;      while (u < \_byteLength) {        u += std::mblen(&c\_str[u], \_byteLength - u);        \_glyphLength += 1;      }      return \_glyphLength;  #else      return \_byteLength;  #endif    }    void determineWidths() const {      if (\_rows.empty()) {        return;      }      \_width.assign(columns(), 0);      \_utf8width.assign(columns(), 0);      for (auto rowIterator = \_rows.begin(); rowIterator != \_rows.end();           ++rowIterator) {        Row const &row = \*rowIterator;        for (unsigned i = 0; i < row.size(); ++i) {          \_width[i] =              \_width[i] > glyphLength(row[i]) ? \_width[i] : glyphLength(row[i]);        }      }    }    void setupAlignment() const {      if (\_rows.empty()) {        return;      }      for (unsigned i = 0; i < columns(); ++i) {        if (\_alignment.find(i) == \_alignment.end()) {          \_alignment[i] = Alignment::LEFT;        }      }    }  };  inline std::ostream &operator<<(std::ostream &stream, const TextTable& table) {    if (table.rows().empty()) {      return stream;    }    table.setup();    if (table.has\_ruler()) {      stream << table.ruler() << "\n";    }    for (auto rowIterator = table.rows().begin();         rowIterator != table.rows().end(); ++rowIterator) {      TextTable::Row const &row = \*rowIterator;      stream << table.vertical();      for (unsigned i = 0; i < row.size(); ++i) {        auto alignment = table.alignment(i) == TextTable::Alignment::LEFT                             ? std::left                             : std::right;        // std::setw( width ) works as follows: a string which goes in the stream        // with byte length (!) l is filled with n spaces so that l+n=width. For a        // utf8 encoded string the glyph length g might be smaller than l. We need        // n spaces so that g+n=width which is equivalent to g+n+l-l=width ==> l+n        // = width+l-g l-g (that means glyph length minus byte length) has to be        // added to the width argument. l-g is computed by correctDistance.        stream << std::setw(table.width(i) + table.correctDistance(row[i]))               << alignment << row[i];        stream << table.vertical();      }      stream << "\n";      if (table.has\_ruler()) {        stream << table.ruler() << "\n";      }    }    return stream;  } |

* utility.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <conio.h>  #include <stdlib.h>  #include <stdio.h>  #include <fstream>  #include <vector>  #include <sstream>  #include <string>  #include <map>  #include <functional>  #include <algorithm>  #include <ctime>  #include "./struct.h"  #include "./data.h"  #include "./table.h"  namespace utility {    void cout(string color, string message, bool with\_endl = true) {      string hex = data::colors[color];      string new\_msg = hex + message + "\033[0m";      with\_endl ? std::cout << new\_msg << endl : std::cout << new\_msg;    }    string today() {      time\_t curr\_time;      tm \* curr\_tm;      char date\_string[100];      time(&curr\_time);      curr\_tm = localtime(&curr\_time);      strftime(date\_string, 50, "%Y-%m-%d", curr\_tm);      return string(date\_string);    }    bool isTimeLater(string first, string second) {      return stoi(first) > stoi(second) ||  stoi(first) == stoi(second);    }    bool confirm(string message, bool is\_formatted = true) {      string is\_confirmed = "t";      string new\_message = is\_formatted ? "Data " + message + " tidak akan bisa dikembalikan lagi! (y/t): " : message;      std::cout << endl;      utility::cout("red", "Apa anda yakin?");      utility::cout("red", new\_message, false);      cin >> is\_confirmed;      return is\_confirmed == "y" || is\_confirmed == "Y";    }    void header(string title) {      system("cls");      utility::cout("cyan", "\n\n=================================");      utility::cout("cyan", title);      utility::cout("cyan", "=================================");    }    string toLower(string word) {      string transformed = word;      transform(transformed.begin(), transformed.end(), transformed.begin(), ::tolower);      return transformed;    }    void notify(string type, string message, bool with\_prefix = true) {      string new\_message = with\_prefix ? message + " [TEKAN ENTER]" : message;      map<string, string> types = {        { "success", "green" },        { "error", "red" },        { "warning", "yellow" },        { "info", "cyan" },      };      utility::cout(types[type], new\_message);      getch();    }    size\_t hash(string word) {      std::hash<string> hashed;      return hashed(word);    }    vector<vector<string>> list(string path) {      fstream file;      file.open(path, ios::in);      vector<vector<string>> content;      vector<string> row;      string line, word;      if(file.is\_open()) {        while(getline(file, line)) {          row.clear();          stringstream str(line);          while(getline(str, word, ','))            row.push\_back(word);          content.push\_back(row);        }      } else {        utility::notify("error", "File tidak ada!");      }      file.close();      return content;    }    vector<string> latest(string path) {      vector<vector<string>> content = utility::list(path);      return content.back();    }    vector<vector<string>> search(string path, const std::initializer\_list<int>& fields, string keyword, bool is\_universal = false, bool is\_exact = false) {      vector<vector<string>> filtered;      vector<vector<string>> list = utility::list(path);      string compared;      bool condition;      keyword = utility::toLower(keyword);      for(int index = 0; index < list.size(); index++) {        if(is\_universal) {          for(int second\_index = 0; second\_index < list[index].size(); second\_index++) {            compared = utility::toLower(list[index][second\_index]);            if(compared.find(keyword) != string::npos) {              filtered.push\_back(list[index]);            }          }        } else {          for (auto field : fields) {            compared = utility::toLower(list[index][field]);            condition = is\_exact ? (compared == keyword) : (compared.find(keyword) != string::npos);            if(condition) {              filtered.push\_back(list[index]);            }          }        }      }      return filtered;    }    vector<string> find(string path, const std::initializer\_list<int>& fields, string keyword, bool is\_exact = true) {      vector<string> empty;      vector<vector<string>> list = utility::search(path, fields, keyword, false, is\_exact);      return list.empty() ? empty : list.back();    }    vector<vector<string>> sort(string path, int field, int type) {      vector<vector<string>> list = utility::list(path);      for(int i=0; i < list.size() - 1; i++) {        for(int j=i+1; j < list.size(); j++) {          // \* 1 = ASC, 2 = DESC          bool is\_pass = type == 1 ? list[i][field] > list[j][field] : list[i][field] < list[j][field];          if(is\_pass) {            vector<string> temp = list[j];            list[j] = list[i];            list[i] = temp;          }        }      }        return list;    }    void update(string path, int field, int field\_length, string identifier, string new\_data[]) {      fstream file;      identifier = utility::toLower(identifier);      vector<vector<string>> content = utility::list(path);      file.open(path, ios::out);      for(int row = 0; row < content.size(); row++) {        string val;        string compared = utility::toLower(content[row][field]);        if(compared != identifier) {          for(int col = 0; col < content[row].size(); col++) {            val += content[row][col] + ",";          }        } else {          for(int new\_data\_col = 0; new\_data\_col < field\_length; new\_data\_col++) {            val += new\_data[new\_data\_col] + ",";          }        }        val += "\n";        file << val;      }      file.close();    }    void destroy(string path, int field, int field\_length, string identifier) {      fstream file;        identifier = utility::toLower(identifier);      vector<vector<string>> content = utility::list(path);      file.open(path, ios::out);      for(int row = 0; row < content.size(); row++) {        string val;        string compared = utility::toLower(content[row][field]);        if(compared != identifier) {          for(int col = 0; col < content[row].size(); col++) {            val += content[row][col] + ",";          }          val += "\n";          file << val;        }      }      file.close();    }      TextTable table(int cols, int rows, string headers[], vector<vector<string>> data) {      TextTable table( '-', '|', '+' );      // SET HEADERS      for(int col = 0; col < cols; col++) {        table.add(headers[col]);      }      table.endOfRow();      // SET ROWS      for(int row = 0; row < rows; row++) {        for(int col = 0; col < cols; col++) {          string val = col != 0 ? data[row][col - 1] : to\_string(row + 1);          table.add(val);        }        table.endOfRow();      }      return table;    }  }; |

* wbp.h

|  |
| --- |
| #pragma once  #include <stdio.h>  #include <fstream>  #include <iostream>  #include <vector>  #include <sstream>  #include <string>  #include "./utility.h"  #include "./struct.h"  #include "./table.h"  #include "./menu.h"  using namespace std;  namespace wbp {      const string PATH = "../files/wbp.csv";    const int TABLE\_COLUMNS\_LENGTH = 3;    string TABLE\_COLUMNS[] = {"No.", "Kode WBP", "Nama WBP"};      string path() {      return PATH;    }    structure::wbp get(vector<string> data) {      structure::wbp wbp;      wbp.kode = data[0];      wbp.nama = data[1];      return wbp;    }    void list() {      utility::header("VISITME - DAFTAR WBP");      vector<vector<string>> content = utility::list(PATH);      TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, content.size(), TABLE\_COLUMNS, content);      cout << table;    }    void sort() {      int column, type;      utility::header("VISITME - SORTING WBP");      utility::cout("yellow", "Daftar kolom: ");      for(int i = 1; i < TABLE\_COLUMNS\_LENGTH; i++) {        utility::cout("yellow", to\_string(i) + ". " + TABLE\_COLUMNS[i]);      }      cout << "\nPilih kolom: "; cin >> column;      cout << "Pilih tipe (1 = asc, 2 = desc): "; cin >> type;      column = column - 1;      if(column >= 0 && column < TABLE\_COLUMNS\_LENGTH) {        vector<vector<string>> list = utility::sort(PATH, column, type);        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Pilihan kolom tidak ada!");      }    }    void search() {      string keyword;      utility::header("VISITME - CARI WBP");      cout << "Kata kunci pencarian: "; fflush(stdin);      getline(cin, keyword);      vector<vector<string>> list = utility::search(PATH, { 0 }, keyword, true);      if(list.size() > 0) {        TextTable table = utility::table(TABLE\_COLUMNS\_LENGTH, list.size(), TABLE\_COLUMNS, list);        cout << table;      } else {        utility::notify("error", "Data tidak ditemukan!");      }    }    void index() {      bool is\_running = true;      bool is\_list\_running = false;      int choice, list\_choice;      while(is\_running) {        choice = menu::wbp();        switch (choice) {          case 1:            is\_list\_running = true;            wbp::list();              while(is\_list\_running) {              list\_choice = menu::searching();              switch (list\_choice) {                case 1:                  wbp::sort();                  break;                case 2:                  wbp::search();                  break;                case 3:                  wbp::list();                  break;                case 4:                  is\_list\_running = false;                  break;              }            }            break;          case 2:            is\_running = false;            break;          default:            utility::notify("error", "Pilihan tidak ada!");            break;        }      }    }  } |

1. File

* Kamar.csv

|  |
| --- |
| KMR-1,Jasmine,  KMR-2,Orchid,  KMR-3,Irish,  KMR-4,Dahlia, |

* Kunjungan.csv

|  |
| --- |
| KN-01,WBP-01,2022-05-10,Hartono,01234,2,10.00,11.00,KMR-1,-,  KN-02,WBP-01,2022-05-11,Hartono,01234,3,-,-,-,Ruangan dipakai,  KN-03,WBP-02,2022-05-12,shyna,01234,1,-,-,-,-,  KN-04,WBP-03,2022-05-13,bayu,01234,1,-,-,-,-, |

* Operator.csv

|  |
| --- |
| operator,3060606181218273969 |

* Wbp.csv

|  |
| --- |
| WBP-01,Jack Pacocha  WBP-02,Earnest Mueller  WBP-03,Harvey Bauch  WBP-04,Sheri Schimmel  WBP-05,Terry Hayes  WBP-20,Tyler Yost  WBP-67,Terrance Gutmann DVM  WBP-34,Malcolm West  WBP-23,Carole Reynolds  WBP-98,Bonnie Skiles  WBP-85,Carl Jacobs I  WBP-88,Daniel Stokes  WBP-67,Andrew Howell  WBP-84,Dr. Kelly Hermiston  WBP-75,Elbert Johnson  WBP-63,Erika Stanton  WBP-78,Lillian Dooley  WBP-60,Ira Waelchi  WBP-80,Tasha Yost  WBP-78,Joan Durgan |

# BAB IV KESIMPULAN

## **4.1 Kesimpulan**

Program SISTEM PENDATAAN KUNJUNGAN LAPAS yang kami buat merupakan sebuah program yang di buat untuk menangani kendala pada proses kunjungan narapidana serta membuat sistem informasi yang menangani proses penjadwalan kunjungan tahanan dan data narapidana ataupun WBP. Yang mana dengan terbentuknya program ini, kami berharap pendataan kunjungan lapas menjadi lebih mudah dan efisien.

## **4.2 Saran**

Dengan terbentuknya program ini, kami sangat menyarankan pada masa yang akan datang penggunaan sistem kunjungan lapas yang teroganisir dengan baik bersama bantuan teknologi saat ini, akan mempermudah sistem pengelolaan serta pendataan pada suatu lapas menjadi lebih tertata.

# DAFTAR PUSTAKA

Andre. (2020, oktober 31). *Tutorial Belajar C++ Part 31: Percabangan Kondisi Switch Case Bahasa C++*. From duniailkom: https://www.duniailkom.com/tutorial-belajar-c-plus-plus-percabangan-kondisi-switch-case-bahasa-c-plus-plus/

Dimas, S. (2020, Januari 13). *Contoh Program C++ Perulangan While dan Do While*. From kelas\_programmer: https://kelasprogrammer.com/contoh-program-c-perulangan-while-do-while/

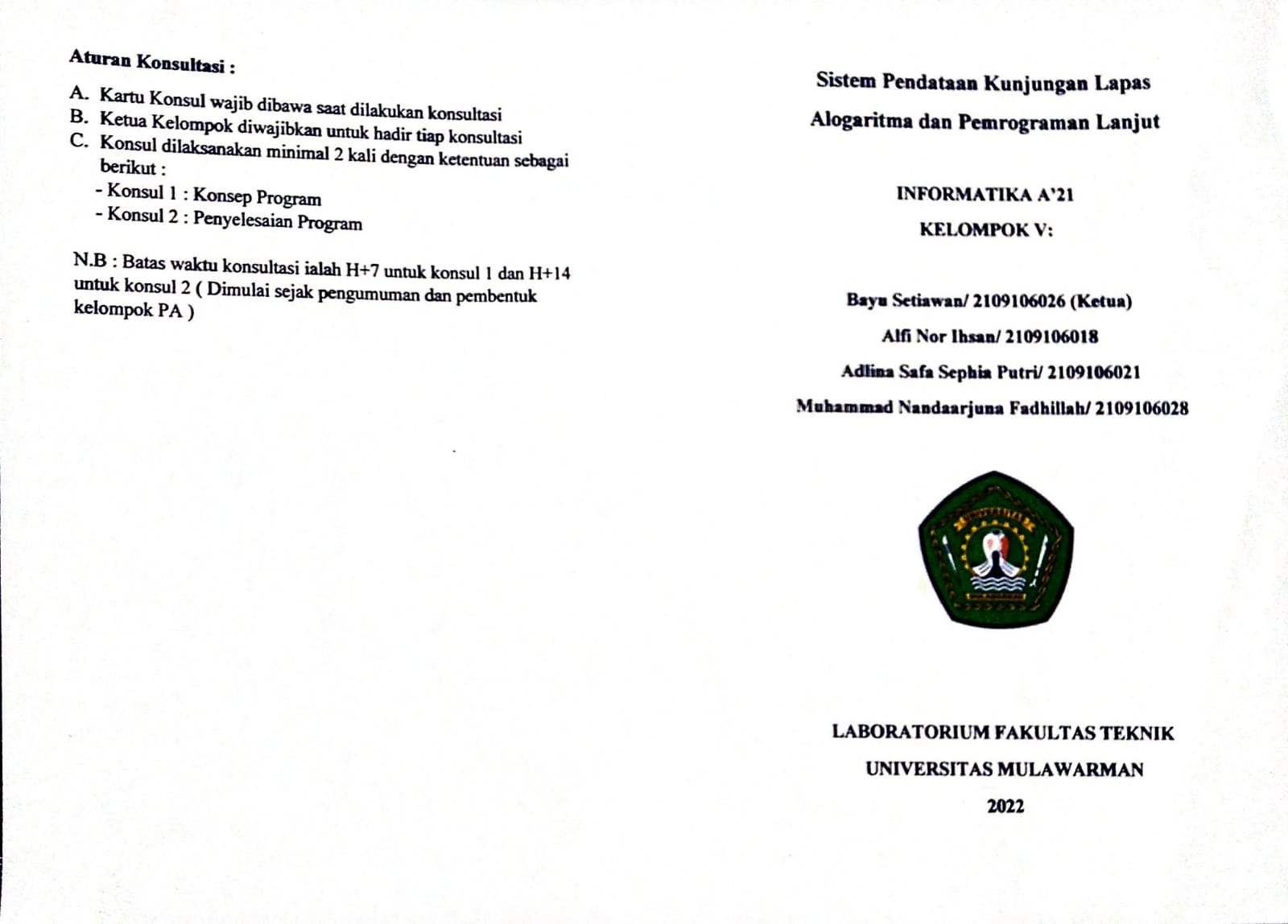
Fidelson Tanzil, S. M. (n.d.). *Linear Search* . From binus: https://socs.binus.ac.id/2019/12/26/linear-search/

haarcuba. (n.d.). *cpp-text-table*. From github: https://github.com/haarcuba/cpp-text-table

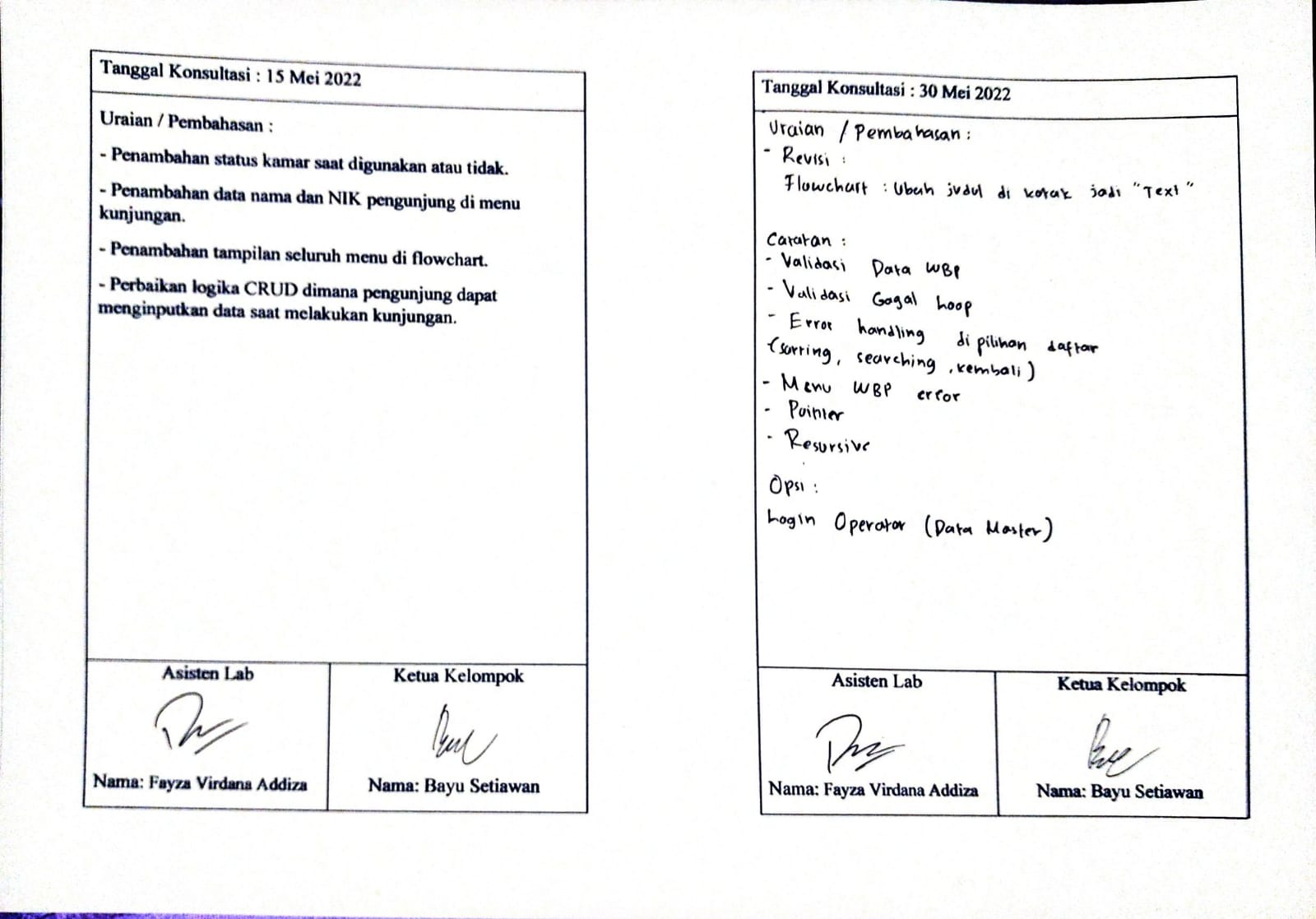
Hartanto, W. ( 2019, desember 04). *Implementasi Algoritma Bubble Sort dengan Bahasa Pemrograman Python* . From binus: https://binus.ac.id/bandung/2019/12/implementasi-algoritma-bubble-sort-dengan-bahasa-pemrograman-python/

Wahyuni, U. (2017). *Rancang Bangun Sistem Informasi Penjadwalan Kunjungan Narapidana Di Lembaga Pemasyarakatan Klas I Di Kabupaten Jeneponto Sulawesi Selatan.* Makassar: FAKULTAS SAINS DAN TEKNOLOGI UIN ALAUDDIN.

# LAMPIRAN



LAMPIRAN 1



LAMPIRAN 2