

LAPORAN TUGAS BESAR
ALGORITMA DAN STRUKTUR DATA
SISTEM PARKIR UNIVERSITAS PERTAMINA



Oleh:

Farhan Dwi Septian	(105221036)
Rachmayuki Rohairunnisa	(105221016)
Selvi Indriani	(105221017)
Muhammad Afrizal Hanif	(105221006)

Fakultas Sains dan Ilmu Komputer

Universitas Pertamina

2022

DAFTAR ISI

DAFTAR ISI.....	i
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan Program.....	1
BAB II METODE	2
2.1 Metode Percabangan	2
2.2 Metode Perulangan.....	2
2.3 Metode Struct	3
2.4 Metode Pointer	4
2.5 Metode Linked List	4
2.6 Metode Tree	5
2.7 Metode Queue	6
BAB III PEMBAHASAN	7
3.1. Fitur buat akun	7
3.2. Fitur login.....	7
2.1. Enqueue.....	8
2.2. Dequeue	8
2.3. Display List	9
2.4. Display Tree.....	10
2.5. Search.....	10
2.6. Delete (Get Out).....	11
BAB IV DOKUMENTASI.....	12
BAB V KESIMPULAN DAN KEKURANGAN	16
5.1. KESIMPULAN	16
5.2. KEKURANGAN.....	16

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sistem parkir di Universitas Pertamina saat ini masih menggunakan tenaga kerja manusia atau secara manual sehingga bertolak belakang dengan perkembangan teknologi di era sekarang. Sistem parkir di universitas pertamina saat ini menggunakan catatan yang ditulis tangan dalam penginputan data kendaraan yang masuk dan menggunakan kartu sebagai nomor antriannya, cara ini terkesan tidak efisien dengan jumlah mahasiswa di universitas pertamina saat ini dan membutuhkan tenaga kerja yang lebih banyak. Oleh karena itu, kelompok kami membuat program sistem parkir secara digital di Universitas Pertamina dengan mengambil nomor antrian dengan sistem digital dalam menginput data kendaraannya.

1.2. Tujuan Program

Adapun tujuan dari pembuatan program sistem parkir di Universitas Pertamina ini adalah :

- 1.2.1. untuk membuat suatu sistem parkir sebagai salah satu fasilitas kampus yang dapat mengelola pemarkiran kendaraan berbasis teknologi, agar penempatan lokasi parkir kendaraan dapat menjadi lebih baik dan teratur.
- 1.2.2. Agar dapat mengetahui informasi parkir yang lebih akurat dan membuat sistem dengan cepat dan mudah diakses oleh seluruh karyawan petugas parkir dan pengguna.

BAB II

METODE

2.1 Metode Percabangan

Percabangan merupakan algoritma pemrograman yang dimana dilakukannya pemilihan dari beberapa perintah yang dijalankan sesuai dengan kondisi tertentu sampai kondisi tersebut terpenuhi. Kondisi tersebut harus menghasilkan benar/salah. Terdapat beberapa macam percabangan, yaitu:

- 2.1.1. Percabangan If merupakan percabangan yang hanya mempunyai satu pilihan saat kondisi tersebut benar/terpenuhi.
- 2.1.2. Percabangan If-else merupakan percabangan yang mempunyai dua pilihan. Pilihan pertama digunakan untuk kondisi yang terpenuhi, dan kondisi kedua untuk kondisi yang tidak terpenuhi (else).
- 2.1.3. Percabangan Switch-case merupakan percabangan yang mempunyai lebih dari dua pilihan sesuai keinginan. Pada praktikum kali ini, saya menggunakan percabangan if dan if-else dan switch-case.

2.2 Metode Perulangan

Perulangan merupakan pengeksekusian kode yang dapat dilakukan secara berulang-ulang kali sejumlah n kali sampai kondisi tersebut terpenuhi atau bernilai benar. Perulangan dalam program berfungsi untuk mengulang program tanpa dideklarasikan satu persatu dan untuk mengefisiensikan program. Perulangan terdiri dari dua jenis:

- 2.2.1. Counted Loop merupakan perulangan yang sudah diketahui berapa banyak kali program akan mengulang. Perulangan yang termasuk perulangan counted loop adalah perulangan for. Perulangan for digunakan untuk mengulang kondisi yang telah diketahui sebanyak apa perulangannya sampai kondisi tersebut

terpenuhi. Struktur perulangan for lebih efisien karena susunannya lebih sederhana. Looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama kondisi terpenuhi, maka pernyataan akan terus dieksekusi

2.2.2. Uncounted Loop merupakan perulangan yang belum diketahui berapa kali program harus mengulang. Perulangan yang termasuk dalam perulangan uncounted loop adalah perulangan while dan do-while.

- Perulangan While

Perulangan while merupakan perulangan yang mengulang suatu kondisi sampai kondisinya terpenuhi (tidak sama dengan nol). Seleksi kondisi pada while dilakukan sebelum bagian perulangan, Oleh karena itu ada kemungkinan bagian perulangan pada while tidak dijalankan sama sekali, yaitu kalau kondisi yang pertama kali bernilai salah.

- Perulangan Do-While

Perulangan Do-While hampir sama dengan perulangan while, hanya saja pada proses perulangannya do-while, Seleksi berada dibawah kondisi, walaupun kondisi yang didefinisikan tidak terpenuhi (bernilai salah). Oleh karena itu, minimal akan terjadi satu kali perulangan dalam do-while.

2.3 Metode Struct

Struct merupakan tipe data bentukan yang berisi sekumpulan variabel -variabel yang bernaung dalam suatu nama yang sama. Dalam penggunaan struct bisa berisi variabelvariabel bertipe data yang sama maupun berbeda. Element struct atau field merupakan variable-variabel yang menjadi anggota struct. Struct digunakan untuk mengelompokkan beberapa informasi yang saling berkaitan menjadi sebuah kesatuan.

2.4 Metode Pointer

Pointer merupakan sebuah variable penunjuk yang berisikan alamat memori (bukan nilai) dari sesuatu variable lain atau dengan kata lain dapat dikatakan bahwa pointer adalah suatu variabel penunjuk ke alamat memori tertentu. Pointer tidak memegang nilai, tetapi memegang alamat dari variable lain. Didalam pointer terdapat dua bagian, yaitu pointer itu sendiri yang memegang alamat dan alamat tersebut menunjuk kesuatu nilai. Pendeklarasian pointer dan untuk menampilkan nilai yang ditunjuk variable pointer dengan menggunakan bintang (*) sebelum nama pointer. Dan untuk menampilkan alamat tempat penyimpanan nilai yang ditunjuk oleh variable pointer, menggunakan operator dan (&).

2.5 Metode Linked List

Linked List merupakan salah satu bentuk struktur data yang berisi sekumpulan data atau node yang tersusun secara sequential, saling berhubungan atau sambung menyambung, dinamis dan tidak terbatas. Linked list saling terhubung dengan variable pointer. Masing-masing data dalam linked list disebut dengan node yang menempati alokasi memory secara dinamis dan biasanya berupa struct yang terdiri dari beberapa field.

2.5.1. Single Linked list.

Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya dan juga memiliki field yang berisi data. Akhir linked list ditandai dengan node terakhir akan menunjuk ke null yang akan digunakan sebagai kondisi berhenti saat pembacaan linked list.

2.5.2. Single Circular Linked List.

Merupakan single linked list yang pointer nextnya menunjuk pada dirinya sendiri, jika terdiri dari beberapa node, maka pointer next pada node terakhirnya akan menunjuk ke pointer node terdepannya. Pada single circular linked list dibutuhkan sebuah kait untuk menghubungkan node-node data yang ada, dimana pada node terakhirnya yang semula menunjuk pada NULL diganti dengan menunjuk pada nilai terdepannya. Single circular linked list hanya mempunyai satu variabel pointer. Fungsi dari single circular linked list apabila ingin mengakses suatu tempat melalui linked list tanpa melalui head/kepala.

2.5.3. Double Linked List.

Double linked list merupakan linked list dengan dua buah pointer yang digunakan jika operasi terhadap elemen predesesor. Node dalam double linked list memiliki tiga field, field pointer yang pertama menunjuk ke node berikutnya, field pointer yang kedua menunjuk ke node sebelumnya, dan field pointer lainnya berisi data dari node tersebut. Sehingga, double linked list ini menunjuk ke dua arah atau bolak-balik. Pointer node awal dan akhir akan menunjuk ke NULL.

2.6 Metode Tree

Kumpulan node yang saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layaknya struktur sebuah pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (one-to-many) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node dari atas ke bawah. Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (one-to-many) dan tidak linier antara elemen-elemennya.

2.6.1. Binary Tree

Tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal dua sub pohon dan kedua subpohon harus terpisah.

2.6.2. Full Binary Tree

Semua node, kecuali leaf pasti memiliki 2 anak dan tiap subpohon

memiliki panjang path yang sama.

2.6.3. Complete Binary Tree

Tree yang mirip dengan full binary tree, tapi tiap subtree boleh memiliki panjang path yang berbeda dan tiap node (kecuali leaf) memiliki 2 anak.

2.6.4. Skewed Binary Tree

Binary tree yang semua nodenya (kecuali leaf) hanya memiliki satu anak

2.6.5. Binary Search Tree

Memiliki sifat yang mirip dengan Binary Tree biasa dengan tambahan syarat, Subpohon kiri hanya berisi node dengan nilai lebih kecil dari root. Subpohon kanan hanya berisi node dengan nilai lebih besar dari root, Subpohon kiri dan kanan masing-masing juga harus berupa Binary Search Tree dan Tidak boleh ada node duplikat.

2.7 Metode Queue

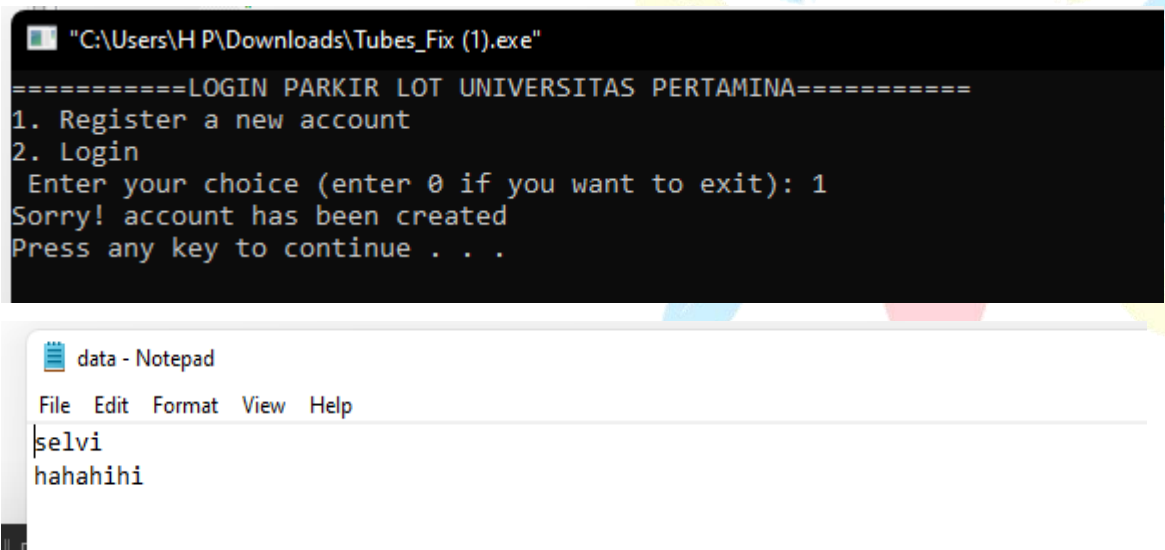
Queue atau antrian adalah suatu kumpulan data yang penambahan elemennya hanya bisa dilakukan pada suatu ujung (disebut dengan sisi belakang atau rear), dan penghapusan atau pengambilan elemen dilakukan lewat ujung yang lain (disebut dengan sisi depan atau front). Jika tumpukan dikenal dengan menggunakan prinsip LIFO (Last In First Out), maka pada antrian prinsip yang digunakan adalah FIFO yaitu (First In First Out).

BAB III

PEMBAHASAN

3.1. Fitur buat akun

program sistem parkir akan berjalan dengan membuat akun terlebih dahulu. admin atau petugas parkir di universitas pertamina akan diminta untuk membuat akun dengan memasukkan username dan password terlebih dahulu. password dan username yang telah di input akan digunakan di fitur selanjutnya yaitu fitur login. user name dan password akan tersimpan di file data dan bisa diubah juga dihapus dan membuat username dan password baru.



The image shows two screenshots. The top screenshot is a command prompt window titled "C:\Users\H P\Downloads\Tubes_Fix (1).exe". It displays the following text: "=====LOGIN PARKIR LOT UNIVERSITAS PERTAMINA===== 1. Register a new account 2. Login Enter your choice (enter 0 if you want to exit): 1 Sorry! account has been created Press any key to continue . . .". The bottom screenshot is a Notepad window titled "data - Notepad". It shows a menu bar with "File", "Edit", "Format", "View", and "Help". The text "selvi" and "hahaha" is entered on two separate lines.

3.2. Fitur login

menjalankan program sistem parkir hanya bisa dilakukan dengan login dan menginput username dan password yang benar. jika user menginput username atau password yang salah maka tidak akan bisa menuju ke fitur program utama hingga user menginput username dan password yang benar.

```
"C:\Users\H P\Downloads\Tubes_Fix (1).exe"
=====LOGIN PARKIR LOT UNIVERSITAS PERTAMINA=====
1. Register a new account
2. Login
Enter your choice (enter 0 if you want to exit): 2
Enter username: selvi
Enter password: hahahihi
Login successful! ^-^
Press any key to continue . . .
```

2.1. Enqueue

Setelah menjalankan sistem login dan berhasil login, program akan menampilkan pilihan menu. pada pilihan menu, terdapat fitur Enqueue yang akan menambahkan antrian untuk mendapatkan nomor antrian parkir di universitas pertamina, dan program akan mencatat antrian yang telah terinput. Pada fitur ini, antrian akan auto increment, dimana inputan antrian akan bertambah secara otomatis.

```
"C:\Users\H P\Downloads\Tubes_Fix (1).exe"
Queue
=====
Filled Queue Number: [ 1 ]
=====
=====PROGRAM SISTEM PARKIR UNIVERSITAS PERTAMINA=====
1. Enqueue A New Parking Queue
2. Dequeue Parking Queue To Parking Lot
3. Display List of Vehicles Entering the Parking Lot
4. Display Tree Parking Lot
5. Search Vehicle
6. Get Out of the Parking Lot
7. Exit
=====
Enter your choice: 1
Queue Has Been Added
Press any key to continue . . .

=====
Filled Queue Number: [ 1 2 3 4 5 ]
=====
```

2.2. Dequeue

Setelah mengambil nomor antrian pada fitur enqueue, user bisa lanjut memilih fitur berikutnya yaitu dequeue. program akan meminta user untuk menambahkan inputan berupa jenis kendaraan yang ingin diparkirkan, nomor plat kendaraan, dan menginput berat kendaraan. lalu data yang telah di input akan ditampilkan sesuai dengan nomor urut antrian yang telah di ambil pada fitur enqueue. Data yang telah di input akan disimpan dan dicetak kedalam file antrian, file tersebut bisa dilihat kembali jika terjadi masalah pada kendaraan yang parkir di universitas pertamina

```

=====
Nomor Antrian      Nomor Plat      ID Tingkat      ID Tree      Jenis Kendaraan
=====
1                  B 7536 HI      0                4231         beat
=====
Enter Vehicle Type: CRV
Enter Vehicle Plate Number: B 45 SL
Enter Vehicle Weight: 3651
=====

```

Antrian - Notepad

Nomor Antrian	Nomor Plat	ID Tingkat	ID Tree	Jenis Kendaraan
1	B 7536 HI	0	4231	beat
2	B 4534 SL	1	3651	CRV
3	B 4311 JI	2	3211	HRV
4	B 8432 RE	3	2000	yamaha jupiter

2.3. Display List

Setelah menginput data pada fitur dequeue, user bisa menampilkan kembali data yang telah di input melalui fitur display. Program akan menampilkan nomor antrian pada fitur enqueue, nomor plat kendaraan, jenis kendaraan yang sudah diinputkan pada fitur dequeue, menampilkan ID tingkat dan ID tree yang merupakan inputan berat kendaraan. semua data pada tampilan display tersimpan dalam file antrian yang bisa dicetak dan diubah jika ada kesalahan.

```

Enter your choice:
3
=====Display List of Vehicles Entering the Parking Lot=====
=====
Nomor Antrian      Nomor Plat      ID Tingkat      ID Tree      Jenis Kendaraan
=====
1                  B 7536 HI      0                4231         beat
2                  B 4534 SL      1                3651         CRV
3                  B 4311 JI      2                3211         HRV
4                  B 8432 RE      3                2000         yamaha jupiter
5                  P 7483 DE      4                1000         mio
6                  B 3253 II      4                2009         scoopy
=====

```

Antrian - Notepad

Nomor Antrian	Nomor Plat	ID Tingkat	ID Tree	Jenis Kendaraan
1	B 7536 HI	0	4231	beat
2	B 4534 SL	1	3651	CRV
3	B 4311 JI	2	3211	HRV
4	B 8432 RE	3	2000	yamaha jupiter
5	P 7483 DE	4	1000	mio
6	B 3253 II	4	2009	scoopy

2.4. Display Tree

Fitur display tree dibuat untuk menampilkan jenis kendaraan dengan metode tree berdasarkan ukuran kendaraan, apabila ukuran kendaraan lebih besar daripada root maka akan diletakkan disebelah kanan, dan program akan meletakkan kendaraan di sebelah kiri apabila ukuran kendaraan lebih kecil daripada root. Kendaraan yang masuk pertama kali akan menjadi root. pada display tree juga terdapat urutan tree berdasarkan tranversal preorder, inorder, dan postorder.

```
=====
Enter your choice: 4
=====DISPLAY TREE=====

      beat
     /
    CRV
   /
  HRV
 /
scoopy
/
yamaha jupiter
/
mio

Traversal Preorder: beat CRV HRV yamaha jupiter mio scoopy
Traversal InOrder:  mio yamaha jupiter scoopy HRV CRV beat
Traversal PostOrder:  mio scoopy yamaha jupiter HRV CRV beat
Press any key to continue
```

2.5. Search

Selain fitur enqueue, dequeue, dan display, program ini juga terdapat fitur search. fitur search digunakan untuk mencari kendaraan dengan menginputkan jenis kendaraan dan nomor plat kendaraan yang ingin dicari. selanjutnya, program akan mencari sesuai dengan inputan. jika data kendaraan yang dicari tersedia di file maka akan menampilkan jenis kendaraan dan plat kendaraan tersebut, lalu fitur search ini akan menampilkan level kendaraan tersebut dalam binary tree.

```
=====
Enter your choice: 5
=====
Nomor Antrian      Nomor Plat      ID Tingkat      ID Tree      Jenis Kendaraan
1                  B 7536 HI       0                4231         beat
2                  B 4534 SL       1                3651         CRV
3                  B 4311 JI       2                3211         HRV
4                  B 8432 RE       3                2000         yamaha jupiter
5                  P 7483 DE       4                1000         mio
6                  B 3253 II       4                2009         scoopy

=====
Enter the type of vehicle you want to search for: CRV
Enter the number plate you want to search for: B 4534 SL
CRV with number plate B 4534 SL found on the levels 1
```

2.6. Delete (Get Out)

Fitur terakhir yang tersedia yaitu Delete (get out), jika kendaraan ingin keluar dari parkir atau telah selesai parkir di universitas pertamina maka kendaraan bisa di hapus menggunakan fitur ini. untuk mengeluarkan kendaraan dari tempat parkir (tree) hanya dengan menginput plat nomor kendaraan yang digunakan. kendaraan yang telah di delete akan terhapus dari file antrian dan tidak akan ditampilkan kembali dalam display list juga dari display tree.

```
=====
Enter your choice: 6
=====
Nomor Antrian      Nomor Plat      ID Tingkat      ID Tree      Jenis Kendaraan
1                  B 7536 HI      0                4231         beat
2                  B 4534 SL      1                3651         CRV
3                  B 4311 JI      2                3211         HRV
4                  B 8432 RE      3                2000         yamaha jupiter
5                  P 7483 DE      4                1000         mio
6                  B 3253 II      4                2009         scoopy
=====
Masukkan plat nomor yang ingin dihapus: B 8432 RE
Vehicle yamaha jupiter has left the parking lot!
Value Deleted
=====
Nomor Antrian      Nomor Plat      ID Tingkat      ID Tree      Jenis Kendaraan
1                  B 7536 HI      0                4231         beat
2                  B 4534 SL      1                3651         CRV
3                  B 4311 JI      2                3211         HRV
5                  P 7483 DE      4                1000         mio
6                  B 3253 II      4                2009         scoopy
=====
Enter your choice: 4
=====DISPLAY TREE=====

      beat
    CRV
  HRV
scoopy
      mio

Traversal Preorder: beat CRV HRV scoopy mio
Traversal InOrder: mio scoopy HRV CRV beat
Traversal PostOrder: mio scoopy HRV CRV beat
=====
```

BAB IV DOKUMENTASI

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <string>
#define SPACE 10
#define limit 100
using namespace std;

struct Tree{
    string jenis_Kendaraan, plat_kendaraan;
    int antrian, id_Tree;
    Tree *right, *left;
    Tree(string a, string plat, int id){
        jenis_Kendaraan = a;
        plat_kendaraan = plat;
        id_Tree = id;
        right = left = NULL;
    }
    Tree(){
        right = left = NULL;
    }
};

Tree *Root;

struct Queue{
    int antrian;
    string jenis_Kendaraan;
    Queue *next;
    Queue *prev;
};

Queue *Head, *Tail;

int queue = 0;
string jumlah = "1";
string jenis_Kendaraan[limit], nomor_Plat[limit], nomor_Antrian[limit], ID[limit], IDTREE[limit];
char islanjut;
string Temporary;

bool login();
void menu_login();
void checkDatabase(fstream &storage);
void display_Antrian(fstream &storage);
void enqueue();
void dequeue();
bool isEmpty();
void display_Queue();
void addAntrian(fstream &storage);
void jumlah_Antrian(fstream &storage, bool cek);
void display2D(Tree *root, int space);
void Postorder(struct Tree* node);
void Inorder(struct Tree *node);
void Preorder(struct Tree *node);
void deleteAntrian(fstream &storage);
void cariKendaraan(fstream &storage);
Tree *iterativeSearch(int v);
Tree *deleteNode(Tree *r, int v);
Tree *minValueNode(Tree *node);
Tree *insert(Tree *node, string plat, string data_kendaraan, int &id, int &tingkat);
Tree *newTree(string data_kendaraan, string plat, int id);

int main(int argc, char const *argv[]){
    char is_continue;
    menu_login();
    int inputan;
    bool cek = false;
    string cari_kendaraan[limit], temp[limit];
    fstream storage;
    checkDatabase(storage);
    int j = 0, i = 0;
    while(!storage.eof()){
        getline(storage, cari_kendaraan[i], '\t');
        getline(storage, temp[j+1], '\n');
        j++;
    }
    if(i >= 0 && cari_kendaraan[0] != ""){
        cek = true;
    }
}
```

```
{
    if(Root == NULL){
        cout << "Data Tree is Still Empty, fill it in first!" << endl;
    }
    else{
        cout << "Traversal Preorder: ";
        Preorder(Root);
        cout << endl;
        cout << "Traversal InOrder: ";
        inorder(Root);
        cout << endl;
        cout << "Traversal PostOrder: ";
        Postorder(Root);
        cout << endl;
    }
    system("PAUSE");
    goto menu;
    break;
}

case SEARCH:
{
    display_Antrian(storage);
    cariKendaraan(storage);
    system("PAUSE");
    goto menu;
}

case DELETE:
{
    deleteAntrian(storage);
    system("PAUSE");
    goto menu;
}

case FINISH:
    exit(0);
default:
{
    cout << "Choose 1-7" << endl;
    break;
}
}
```

```
{
    jumlah_Antrian(storage, cek);
    storage.close();
    char loop;
    menu:
    system("cls");
    display_Queue();
    cout << "=====PROGRAM SISTEM PARKIR UNIVERSITAS PERTAMAMA===== << endl;
    cout << "1. Enqueue A New Parking Queue/2. Dequeue Parking Queue To Parking Lot/3. Display List of Vehicles Entering the Parking Lot/4. Display Tree Parking Lot/5. Search Vehicle/6. Get Out of the Parking Lot/7. Exit" << endl;
    cout << "=====Enter your choice: ";
    cin >> inputan;
    cin.ignore();
    menu_antrian[queue] = 1; DEQUEUE, DISPLAY_QUEUE, DISPLAY_TREE, SEARCH, DELETE, FINISH;
    while(inputan != FINISH){
        switch (inputan){
            case ENQUEUE: //menyimpan barang store ke Antrian, dll
            {
                enqueue();
                cout << "Queue Has Been Added" << endl;
                system("PAUSE");
                goto menu;
            }
            case DEQUEUE: //manggilin yang udah di store ke Antrian, dll nya
            {
                addAntrian(storage);
                dequeue();
                system("PAUSE");
                goto menu;
            }
            case DISPLAY_QUEUE:
            {
                cout << "=====Display List of Vehicles Entering the Parking Lot===== << endl;
                display_Antrian(storage);
                system("PAUSE");
                goto menu;
            }
            case DISPLAY_TREE:
            {
                cout << "=====DISPLAY TREE=====\\n" << endl;
                display2D(Root, 10);
                cout << endl;
            }
        }
    }
}
```

```

        lanjutkan;
        cout << "Continue?(y/n) : ";
        cin >> is_continue;
        cin.ignore();
        if ( (is_continue == 'y') | (is_continue == 'Y') ){
            goto menu;
        } else if ((is_continue == 'n') | (is_continue == 'N')){
            break;
        } else {
            goto lanjutkan;
        }
    }
    return 0;
}

bool login(){
    fstream myFile;
    myFile.open("data.txt", ios::in);
    string username, password, simpan_username, simpan_password;
    cout << "Enter username: ";
    getline(cin, username);
    cout << "Enter password: ";
    getline(cin, password);

    while(!myFile.eof()){
        getline(myFile, simpan_username);
        getline(myFile, simpan_password);
    }
    if (simpan_username == username && simpan_password == password){
        return true;
    }
    return false;
}
myFile.close();
}

```

```

void menu_login(){
    string username, password;
    int pilihan;
    fstream myFile;
    myFile.open("data.txt", ios::out | ios::in);
    label:
    system("cls");
    cout << "=====LOGIN PARKIR LOT UNIVERSITAS PERTAMINA===== << endl;
    cout << "1. Register a new account\n2. Login\nEnter your choice (enter 0 if you want to exit): ";
    cin >> pilihan;
    cin.ignore();
    if (pilihan == 1){
        if(!myFile){
            myFile.close();
            myFile.open("data.txt", ios::out);
            cout << "Choose your username: ";
            getline(cin, username);
            cout << "Choose your password: ";
            getline(cin, password);
            myFile << username << endl << password;
            myFile.close();
            cout << "Yoo! New account has been created ^-^ << endl;
            system("PAUSE");
            goto label;
        }
        cout << "Sorry! account has been created << endl;
        system("PAUSE");
        goto label;
    }
    } else if (pilihan == 2){
        if(!myFile){
            cout << "Sorry! Account not yet created, please register an account!" << endl;
            system("PAUSE");
            goto label;
        }
    }
    bool status = login();
}

```

```

} else if (pilihan == 2){
    if(!myFile){
        cout << "Sorry! Account not yet created, please register an account!" << endl;
        system("PAUSE");
        goto label;
    }
    bool status = login();
    if (status == true){
        cout << "Login successful! ^-^ << endl;
        system("PAUSE");
    }
    } else {
        cout << "Login Unsuccessful! enter username and password correctly!" << endl;
        system("PAUSE");
        goto label;
    }
} else if (pilihan == 0){
    exit(0);
} else {
    cout << "Your input is wrong!" << endl;
    system("PAUSE");
    goto label;
}
}

void checkDatabase(fstream &storage){
    storage.open("Antrian.txt", ios::out | ios::in);
    // check file ada atau tidak
    if (storage.is_open()){
        cout << "database ditemukan" << endl;
        storage << "Nomor Antrian\t\t";
        storage << "Nomor Plat\t\t";
        storage << "ID Tingkat\t\t";
        storage << "ID Tree\t\t";
        storage << "Jenis Kendaraan" << endl << endl;
    }
}

```

```

} else {
    //cout << "database tidak ditemukan, buat database baru" << endl;
    storage.close();
    storage.open("Antrian.txt", ios::trunc | ios::out | ios::in);
    storage << "Nomor Antrian\t\t";
    storage << "Nomor Plat\t\t";
    storage << "ID Tingkat\t\t";
    storage << "ID Tree\t\t";
    storage << "Jenis Kendaraan" << endl << endl;
}
}

void jumlah_Antrian(fstream &storage, bool cek){
    if(cek == true){
        string temp[limit];
        string jumlah_awal[limit], nomor[limit], id_tingkat[limit], id_tree[limit], jenis[limit];
        int j = 0;
        int i = 0;
        char sampah[limit], sam[limit];
        storage.close();
        storage.open("Antrian.txt", ios::in);
        while(!storage.eof()){
            while(i < 24){
                getline(storage, temp[j]);
                j++;
            }
            getline(storage, jumlah_awal[i], '\t');
            getline(storage, temp[j], '\t');
            getline(storage, temp[j+1], '\t');
            getline(storage, nomor[i], '\t');
            getline(storage, temp[j+2], '\t');
            getline(storage, id_tingkat[i], '\t');
            getline(storage, temp[j+3], '\t');
            getline(storage, id_tree[i], '\t');
            getline(storage, temp[j+4], '\t');
            getline(storage, jenis[i], '\n');
        }
    }
}

```

```

        getline(storage, nomor[i], '\t');
        getline(storage, temp[j+2], '\t');
        getline(storage, id_tingkat[i], '\t');
        getline(storage, temp[j+3], '\t');
        getline(storage, temp[j+4], '\t');
        getline(storage, id_tree[i], '\t');
        getline(storage, temp[j+5], '\t');
        getline(storage, temp[j+6], '\t');
        getline(storage, jenis[i], '\n');
        i++;
        j++;
    }
    for (int k = 0; k < i-1; k++){
        strcpy(sampah, id_tree[k].c_str());
        int id = atoi(sampah);
        char sam[100];
        strcpy(sam, id_tree[k].c_str());
        int tingkat = atoi(sam);
        Root = insert(Root, nomor[k], jenis[k], id, tingkat);
    }
}

char arr[100];
Temporary = jumlah_awal[i-2];
strcpy(arr, Temporary.c_str());
int sementara = atoi(arr);
sementara++;
jumlah = to_string(sementara);
}

```

```

} else {
    return;
}
}

void enqueue(){
    Queue *Temp = new Queue;
    int temp = atoi(jumlah.c_str());
    Temp->antrian = temp;
    Temp->jenis_Kendaraan = "";
    if(is_Empty()){
        Temp->next = NULL;
        Temp->prev = NULL;
        Head = Temp;
        Tail = Temp;
        temp++;
        string ToString = to_string(temp);
        jumlah = ToString;
        queue++;
    }
    } else if (queue >= limit){
        cout << "Queue Full, Please wait!" << endl;
    }
    } else {
        Temp->next = NULL;
        Temp->prev = Tail;
        Tail->next = Temp;
        Tail = Temp;
        temp++;
        string ToString = to_string(temp);
        jumlah = ToString;
        queue++;
    }
}

```



```

}

void dequeue(){
    Queue *Hapus, *Temp;
    Hapus = Head;
    if(!is_Empty()){
        cout << "The queue is still empty!" << endl;
    }else if(Head->next == NULL){
        Hapus->next = NULL;
        Hapus->prev = NULL;
        Hapus = NULL;
        Head = Hapus;
        Tail = Head;
        queue--;
    }else{
        Temp = Hapus->next;
        Hapus->next = NULL;
        Hapus->prev = NULL;
        Head = Temp;
        Head->prev = NULL;
        delete Hapus;
        queue--;
    }
}

void display_Antrian(fstream &storage){
    char loop;
    string word;
    storage.open("Antrian.txt", ios::in);
    cout << "===== " << endl;
    while(!storage.eof()){
        getline(storage, word, '\t');
        cout << word << '\t';
    }
    cout << endl;
    cout << "===== " << endl;
    storage.close();
}

```

```

}

bool is_Empty(){
    if(Head == NULL){
        return 1;
    }else{
        return 0;
    }
}

void display_Queue(){
    Queue *Temp;
    Temp = Head;
    if(!is_Empty()){
        cout << "Empty Queue!" << endl;
    }else{
        cout << "      Queue      " << endl;
        cout << "===== \n" << endl;
        cout << "Filled Queue Number: [ ";
        while(Temp != NULL){
            cout << Temp->antrian << " ";
            Temp = Temp->next;
        }
        cout << "]" << endl;
        cout << "===== \n" << endl;
    }
}

void addAntrian(fstream &storage){
    string temp[limit], nPlat[limit], jenis[limit], id[limit], idTree[limit];
    int tingkat_Antrian;
    int posisi = 0;
    do{
        if(posisi < limit){
            if(!is_Empty()){
                int i = 0, j = 0;
                display_Antrian(storage);
                storage.open("Antrian.txt", ios::in);
                while(!storage.eof()){
                    while(i < j){
                        getline(storage, temp[i]);
                        i++;
                    }
                    getline(storage, nAntrian[i], '\t');
                    getline(storage, temp[j], '\t');
                }
            }
        }
    }while(true);
}

```

```

        getline(storage, temp[j], '\t');
        getline(storage, temp[j+1], '\t');
        getline(storage, nPlat[i], '\t');
        getline(storage, temp[j+2], '\t');
        getline(storage, id[i], '\t');
        getline(storage, temp[j+3], '\t');
        getline(storage, temp[j+4], '\t');
        getline(storage, idTree[i], '\t');
        getline(storage, temp[j+5], '\t');
        getline(storage, temp[j+6], '\t');
        getline(storage, jenis[i], '\n');
        i++;
    }
    storage.close();
    storage.open("Antrian.txt", ios::out|ios::app);
    ID:
    cout << "Enter Vehicle Type: ";
    getline(cin, jenis_Kendaraan[posisi]);
    cout << "Enter Vehicle Plate Number: ";
    getline(cin, nomor_Plat[posisi]);
    cout << "Enter Vehicle Weight: ";
    getline(cin, IDTREE[posisi]);
    for(int v = 0; v < 1 - 1; v++){
        if(nPlat[v] == nomor_Plat[posisi]){
            cout << "Sorry! Number Plate Already Used!" << endl;
            system("PAUSE");
            goto ID;
        }else{
            continue;
        }
    }
    char sw[100];
    strcpy(sw, IDTREE[posisi].c_str());
    int ID = atoi(sw);
    int tingkat = 0;
    Root = Insert(Root, nomor_Plat[posisi], jenis_Kendaraan[posisi], ID, tingkat);
    string toString = to_string(Head->antrian);
    nomor_Antrian[posisi] = toString;
    storage << nomor_Antrian[posisi] << "\t\t\t";
    storage << nomor_Plat[posisi] << "\t\t";
    storage << tingkat << "\t\t\t";
    storage << IDTREE[posisi] << "\t\t\t";
    storage << jenis_Kendaraan[posisi] << "\n";
    posisi++;
    storage.close();
    system("cls");
}

```

```

        system("cls");
        display_Antrian(storage);
        cout << "Queue Number" << Head->antrian << " Entering the Parking Lot" << endl;
        break;
    }else{
        cout << "No Queues!" << endl;
        break;
    }
}

}while(true);
storage.close();
}

struct Tree *newTree(string data_kendaraan, string plat, int id){
    Tree *temp = new Tree;
    temp->jenis_Kendaraan = data_kendaraan;
    temp->plat_kendaraan = plat;
    temp->id_Tree = id;
    temp->left = temp->right = NULL;
    return temp;
}

struct Tree *insert(Tree *node, string plat, string data_kendaraan, int &id, int &tingkatan){
    if (node == NULL){
        return newTree(data_kendaraan, plat, id);
    }

    if (id < node->id_Tree){
        tingkatan++;
        node->left = insert(node->left, plat, data_kendaraan, id, tingkatan);
    }else if (id >= node->id_Tree){
        tingkatan++;
        node->right = insert(node->right, plat, data_kendaraan, id, tingkatan);
    }
    return node;
}

```

```

void display2D(Tree *root, int space){
    if (root == NULL){
        return;
    }
    space += SPACE;
    display2D(root->right, space);
    cout << endl;
    for (int i = SPACE; i < space; i++){
        cout << " ";
    }
    cout << root->jenis_Kendaraan << endl;
    display2D(root->left, space);
}

void inorder(struct Tree* node){
    if (node == NULL)
        return;

    /* first recur on left child */
    inorder(node->left);

    /* then print the data of node */
    cout << node->jenis_Kendaraan << " ";

    /* now recur on right child */
    inorder(node->right);
}

void Preorder(struct Tree* node){
    if (node == NULL)
        return;

    /* first print data of node */
    cout << node->jenis_Kendaraan << " ";

    /* then recur on left subtree */
    Preorder(node->left);

    /* now recur on right subtree */
    Preorder(node->right);
}

```

```

void Postorder(struct Tree* node){
    if (node == NULL)
        return;

    Postorder(node->left);
    Postorder(node->right);

    cout << node->jenis_Kendaraan << " ";
}

void cariKendaraan(fstream &storage){
    string temp[100], input_jenis, input_Plat;
    bool found = false;
    int i = 0, j = 0, h = 0, pl;
    string word;
    storage.open("Antrian.txt", ios::in|ios::out);
    while(!storage.eof()){
        while(j < >){
            getline(storage, temp[j]);
            j++;
        }
        getline(storage, nomor_Antrian[i], '\t');
        getline(storage, temp[j], '\t');
        getline(storage, temp[j+1], '\t');
        getline(storage, nomor_Plat[i], '\t');
        getline(storage, temp[j+2], '\t');
        getline(storage, ID[i], '\t');
        getline(storage, temp[j+3], '\t');
        getline(storage, temp[j+4], '\t');
        getline(storage, IDTREE[i], '\t');
        getline(storage, temp[j+5], '\t');
        getline(storage, temp[j+6], '\t');
        getline(storage, jenis_Kendaraan[i], '\n');
        i++;
    }
    cout << "Enter the type of vehicle you want to search for: ";
    getline(cin, input_jenis);
    cout << "Enter the number plate you want to search for: ";
    getline(cin, input_Plat);
    for (int q = 0; q < i-1; q++){
        if (input_jenis == jenis_Kendaraan[q] && input_Plat == nomor_Plat[q]){
            cout << input_jenis << " with number plate " << input_Plat << " found on the levels " << ID[q] << endl;
            found = true;
            break;
        }
    }
    if(found == false){
        cout << "Vehicle Not Found!" << endl;
    }
    storage.close();
}

```



Universitas
Pertamina

BAB V

KESIMPULAN DAN KEKURANGAN

5.1. KESIMPULAN

Berdasarkan program sistem parkir yang kami buat untuk area parkir Universitas Pertamina, dapat diambil kesimpulan bahwa:

- 5.1.1. Dalam program sistem parkir Universitas Pertamina dibuat menggunakan metode struct, pointer, linked list, tree, queue, dan metode file.
- 5.1.2. Pencatatan kendaraan yang ada di area parkir Universitas Pertamina sebelumnya dicatat secara manual, sehingga hal tersebut memungkinkan akan membuat data-data kendaraan akan rusak atau hilang. Oleh karena itu, dibuatkannya program sistem parkir yang data-datanya akan disimpan di file.txt dan dapat ditampilkan pada program. Dengan itu, akan lebih mudah untuk mencari dan melihat data kendaraan.
- 5.1.3. Dengan adanya program sistem parkir ini akan meningkatkan efektifitas dan efisiensi kinerja dalam pencatatan kendaraan pada area parkir Universitas Pertamina.

5.2. KEKURANGAN

Kekurangan program terletak pada segi case sensitive. Dimana inputan huruf kecil atau besar akan berpengaruh dalam pencarian atau penambahan data. Oleh karena itu, user di harap berhati-hati atas penulisan besar atau kecilnya huruf pada saat memasukkan data agar tidak terjadi kesalahan dalam pencarian data.