PROJEK AKHIR

STRUKTUR DATA

DISUSUN OLEH :

Azhar Fikri Haryodwiseno	123220042
Almer Farand Rafael	123220040
Febrian Chrisna A.	123220051
Naufal Rafid Muhammad Faddila	123220052

PROGRAM STUDI INFORMATIKA
JURUSAN INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
YOGYAKARTA

2023

1. **Deskripsi Proyek**

   Sistem ini akan memanfaatkan struktur data PTB untuk menyimpan informasi pasien rumah sakit, seperti nama pasien, id pasien, Golongan pasien, lama rawat inap, dan total harga rawat inap. Selain itu, pohon telusur biner akan digunakan untuk menyimpan data pasien berdasarkan urutan ID pasien.

Data harga rawat inap:

Golongan	Harga
A	Rp150,000 /malam
B	Rp200,000 /malam
C	Rp250,000 /malam
D	Rp300,000 /malam
E	Rp350,000 /malam

2. **Fitur Proyek**
   - Input nama pasien, ID pasien, golongan, dan lama rawat inap dalam struktur hash.
   - Tampilkan data pasien dengan total harga rawat inap menggunakan struktur hash.
   - Pencarian data pasien berdasarkan rentang nama pasien menggunakan pohon telusur biner.
     Misal: mencari nama pasien dalam rentang (A & C) maka program akan menampilkan daftar nama pasien dalam rentang tersebut.

   - Penghapusan data menggunakan struktur hash.
   - Data yang dihapus dapat dilihat kembali dalam history pasien. (in-order, post-order, pre-order)

3. **Source Code**

```
#include <iostream>
#include <unordered_map>
#include <list>
#include <limits>

using namespace std;
```

```
struct HargaRawatInap {
    char golongan;
    int harga;
};

struct Pasien {
    string nama;
    int id;
    char golongan;
    int lamaRawat;
    int totalHarga;
};

struct TreeNode {
    Pasien data;
    TreeNode* left;
    TreeNode* right;
};

class SistemManajemenRumahSakit {
private:
    unordered_map<int, Pasien> dataPasien;
    list<Pasien> dataPasienDihapus;
    TreeNode* pohonTelusurBiner;

public:
    SistemManajemenRumahSakit() : pohonTelusurBiner(nullptr) {}

    void tambahDataPasien() {
        system("cls");
        Pasien pasien;
        cout << "=================================================\n";
        cout << "|              Input Data Pasien                |\n";
        cout << "=================================================\n";
        cin.ignore();
        cout << "Input Nama Pasien: ";
        getline(cin, pasien.nama);
        cout << "Input ID Pasien: ";
        cin >> pasien.id;
```

```
        cout << "Input Golongan Pasien (A, B, C, D, E): ";
        cin >> pasien.golongan;
        cout << "Input Lama Rawat Inap: ";
        cin >> pasien.lamaRawat;

        pasien.totalHarga = hitungTotalHarga(pasien.golongan,
pasien.lamaRawat);

        if (dataPasien.find(pasien.id) == dataPasien.end()) {
            dataPasien[pasien.id] = pasien;

            if (pohonTelusurBiner == nullptr) {
                pohonTelusurBiner = tambahNode(pohonTelusurBiner, pasien);
            } else {
                tambahNode(pohonTelusurBiner, pasien);
            }

            cout <<
"================================================\n";
            cout << "|       Data Pasien berhasil
ditambahkan!        |\n";
            cout <<
"================================================\n";
        } else {
            cout <<
"================================================\n";
            cout << "|     Gagal menambahkan, id sudah
digunakan!     |\n";
            cout <<
"================================================\n";
        }
        system("pause");
    }

    void tampilkanDataPasien() {
        system("cls");
        tampilkanDataPasien(dataPasien);
        system("pause");
    }
```

```
    void cariDataPasien(string nama) {
        bool dataDitemukan = false;

        cout << "\nData Pasien dengan Nama '" << nama << "':\n";
        cariNode(pohonTelusurBiner, nama, dataDitemukan);

        if (!dataDitemukan) {
            cout <<
"================================================\n";
            cout << "|         Data Pasien tidak
ditemukan!           |\n";
            cout <<
"================================================\n";
        }
    }

    void cariDataPasienByHurufAwal(char hurufAwal, char hurufAkhir = 'Z')
{
        cout << "\nData Pasien dalam Rentang (" << hurufAwal << " - " <<
hurufAkhir << "):\n";
        cariNodeByHurufAwal(pohonTelusurBiner, hurufAwal, hurufAkhir);
    }

    void hapusDataPasien(int id) {
        auto it = dataPasien.find(id);
        if (it != dataPasien.end()) {
            cout <<
"================================================\n";
            cout << "Data Pasien Nama: " << it->second.nama << " & ID " <<
it->second.id <<" dihapus! "<< endl;
            dataPasienDihapus.push_back(it->second);
            dataPasien.erase(it);
            rebuildTree(dataPasien);
        } else {
            cout << "Data Pasien dengan ID " << id << " tidak ditemukan."
<< endl;
        }
    }
```

```
    void tampilkanHistoryPasien() {
        if (pohonTelusurBiner == nullptr) {
            cout << "Tree is empty.\n";
            return;
        }

        cout << "\nHistory Pasien (In-Order Traversal):\n";
        cout << "================================================\n";
        inOrderTraversal(pohonTelusurBiner);


        cout << "\nHistory Pasien (pre-Order Traversal):\n";
        cout << "================================================\n";
        preOrderTraversal(pohonTelusurBiner);


        cout << "\nHistory Pasien (post-Order Traversal):\n";
        cout << "================================================\n";
        postOrderTraversal(pohonTelusurBiner);

        cout << "\n";
        inOrderTraversalList();
        cout << "\n";
        preOrderTraversalList();
        cout << "\n";
        postOrderTraversalList();
        cout << "\n";
    }


    void cariDataPasienByHuruf(char huruf) {
        bool dataDitemukan = false;

        cout << "\nData Pasien dengan Nama Dimulai dari Huruf '" << huruf
<< "':\n";
        cariNodeByHuruf(pohonTelusurBiner, huruf, dataDitemukan);
```

```
        if (!dataDitemukan) {
            cout <<
"================================================\n";
            cout << "|         Data Pasien tidak
ditemukan!           |\n";
            cout <<
"================================================\n";
        }
    }

private:
    int hitungTotalHarga(char golongan, int lamaRawat) {
        unordered_map<char, HargaRawatInap> hargaMap = {
            {'A', {'A', 150000}},
            {'B', {'B', 200000}},
            {'C', {'C', 250000}},
            {'D', {'D', 300000}},
            {'E', {'E', 350000}}
        };

        return hargaMap[golongan].harga * lamaRawat;
    }

    void tampilkanDataPasien(const unordered_map<int, Pasien>& dataPasien)
{
        cout << "================================================\n";
        cout << "|                  Data Pasien                  |\n";
        cout << "================================================\n";
        for (auto const& entry : dataPasien) {
            Pasien pasien = entry.second;
            cout << "Nama: " << pasien.nama << endl;
            cout << "ID: " << pasien.id << endl;
            cout << "Golongan: " << pasien.golongan << endl;
            cout << "Total Harga: " << pasien.totalHarga << endl;
            cout <<
"================================================\n";
        }
    }
```

```
    TreeNode* tambahNode(TreeNode* root, Pasien data) {
        if (root == nullptr) {
            TreeNode* newNode = new TreeNode;
            newNode->data = data;
            newNode->left = newNode->right = nullptr;
            return newNode;
        }

        if (data.id < root->data.id) {
            root->left = tambahNode(root->left, data);
        } else {
            root->right = tambahNode(root->right, data);
        }

        return root;
    }

    void cariNode(TreeNode* root, string nama, bool& dataDitemukan) {
        if (root == nullptr) {
            return;
        }

        if (root->data.nama.find(nama) != string::npos) {
            dataDitemukan = true;
            cout <<
"==================================================\n";
            cout << "Nama: " << root->data.nama << endl;
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
        }

        cariNode(root->left, nama, dataDitemukan);
        cariNode(root->right, nama, dataDitemukan);
    }
```

```
    void cariNodeByHurufAwal(TreeNode* root, char hurufAwal, char
hurufAkhir) {
        if (root == nullptr) {
            return;
        }

        if (root->data.nama[0] >= hurufAwal && root->data.nama[0] <=
hurufAkhir) {
            cout <<
"================================================\n";
            cout << "Nama: " << root->data.nama << endl;
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
        }

        if (root->data.nama[0] >= hurufAwal) {
            cariNodeByHurufAwal(root->left, hurufAwal, hurufAkhir);
        }

        if (root->data.nama[0] <= hurufAkhir) {
            cariNodeByHurufAwal(root->right, hurufAwal, hurufAkhir);
        }
    }

    void cariNodeByHuruf(TreeNode* root, char huruf, bool& dataDitemukan)
{
        if (root == nullptr) {
            return;
        }

        if (root->data.nama[0] == huruf) {
            dataDitemukan = true;
            cout <<
"================================================\n";
            cout << "Nama: " << root->data.nama << endl;
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
```

```
            cout <<
"================================================\n";
        }

        cariNodeByHuruf(root->left, huruf, dataDitemukan);
        cariNodeByHuruf(root->right, huruf, dataDitemukan);
    }

    void inOrderTraversal(TreeNode* root) {
        if (root != nullptr) {
            inOrderTraversal(root->left);
            cout << "Nama: " << root->data.nama << endl;
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
            cout <<
"================================================\n";
            inOrderTraversal(root->right);
        }
    }

    void preOrderTraversal(TreeNode* root) {
        if (root != nullptr) {
            cout << "Nama: " << root->data.nama << endl;
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
            cout <<
"================================================\n";
            preOrderTraversal(root->left);
            preOrderTraversal(root->right);
        }
    }

    void postOrderTraversal(TreeNode* root) {
        if (root != nullptr) {
            postOrderTraversal(root->left);
            postOrderTraversal(root->right);
            cout << "Nama: " << root->data.nama << endl;
```

```
            cout << "ID: " << root->data.id << endl;
            cout << "Golongan: " << root->data.golongan << endl;
            cout << "Total Harga: " << root->data.totalHarga << endl;
            cout <<
"================================================\n";
        }
    }

    void rebuildTree(const unordered_map<int, Pasien>& dataPasien) {
        // Free the memory of the old tree
        deleteTree(pohonTelusurBiner);
        // Create a new tree
        pohonTelusurBiner = nullptr;
        for (auto const& entry : dataPasien) {
            pohonTelusurBiner = tambahNode(pohonTelusurBiner,
entry.second);
        }
    }

    // Function to delete the tree and free memory
    void deleteTree(TreeNode* root) {
        if (root != nullptr) {
            deleteTree(root->left);
            deleteTree(root->right);
            delete root;
        }
    }

    void inOrderTraversalList() {
        cout << "In-Order Traversal Data Pasien yang Dihapus:\n";
        cout << "================================================\n";
        for (const auto& pasien : dataPasienDihapus) {
            cout << "Nama: " << pasien.nama << endl;
            cout << "ID: " << pasien.id << endl;
            cout << "Golongan: " << pasien.golongan << endl;
            cout << "Total Harga: " << pasien.totalHarga << "\n";
            cout <<
"================================================\n";
        }
```

```
    }

    void preOrderTraversalList() {
        cout << "Pre-Order Traversal Data Pasien yang Dihapus:\n";
        cout << "==================================================\n";
        for (const auto& pasien : dataPasienDihapus) {
            cout << "Nama: " << pasien.nama << endl;
            cout << "ID: " << pasien.id << endl;
            cout << "Golongan: " << pasien.golongan << endl;
            cout << "Total Harga: " << pasien.totalHarga << "\n";
            cout <<
"==================================================\n";
        }

    }

    void postOrderTraversalList() {
        cout << "Post-Order Traversal Data Pasien yang Dihapus:\n";
        cout << "==================================================\n";
        for (const auto& pasien : dataPasienDihapus) {
            cout << "Nama: " << pasien.nama << endl;
            cout << "ID: " << pasien.id << endl;
            cout << "Golongan: " << pasien.golongan << endl;
            cout << "Total Harga: " << pasien.totalHarga << "\n";
            cout <<
"==================================================\n";
        }

    }
};

int main() {
    SistemManajemenRumahSakit sistem;
    int pilihan;

    do {
        system("cls");
        cout << "==================================================\n";
```

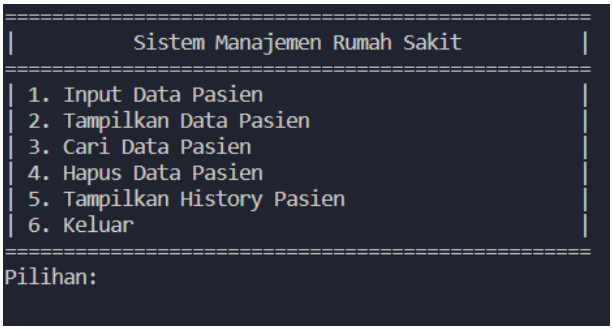
```
        cout << "|          Sistem Manajemen Rumah Sakit          |\n";
        cout << "================================================\n";
        cout << "| 1. Input Data Pasien                         |\n";
        cout << "| 2. Tampilkan Data Pasien                     |\n";
        cout << "| 3. Cari Data Pasien                          |\n";
        cout << "| 4. Hapus Data Pasien                         |\n";
        cout << "| 5. Tampilkan History Pasien                  |\n";
        cout << "| 6. Keluar                                    |\n";
        cout << "================================================\n";
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
        case 1:
            sistem.tambahDataPasien();
            break;
        case 2:
            sistem.tampilkanDataPasien();
            break;
        case 3: {
            system("cls");
            char jenisPencarian;
            cout <<
"================================================\n";
            cout << "|              Pilih Jenis
Pencarian             |\n";
            cout <<
"================================================\n";
            cout << "| N/n untuk pencarian berdasarkan
nama           |\n";
            cout << "| H/h untuk rentang
huruf                        |\n";
            cout <<
"================================================\n";
            cout << "Pilih: ";
            cin >> jenisPencarian;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            if (jenisPencarian == 'N' || jenisPencarian == 'n') {
                string nama;
```

```
                cin.ignore();
                cout <<
"================================================\n";
                cout << "Input Nama: ";
                getline(cin, nama);
                cout <<
"================================================\n";
                sistem.cariDataPasien(nama);
            } else if (jenisPencarian == 'H' || jenisPencarian == 'h') {
                char hurufAwal;
                cout <<
"================================================\n";
                cout << "Input Huruf Awal: ";
                cin >> hurufAwal;
                cout <<
"================================================\n";

                char pilihanHurufAkhir;
                cout << "Apakah Anda ingin menginput Huruf Akhir? (y/n):
";
                cin >> pilihanHurufAkhir;

                if (pilihanHurufAkhir == 'y' || pilihanHurufAkhir == 'Y')
{
                    char hurufAkhir;
                    cout <<
"================================================\n";
                    cout << "Input Huruf Akhir: ";
                    cin >> hurufAkhir;
                    cout <<
"================================================\n";
                    sistem.cariDataPasienByHurufAwal(hurufAwal,
hurufAkhir);
                } else {
                    sistem.cariDataPasienByHuruf(hurufAwal);
                }
            } else {
                cout <<
"================================================\n";
```

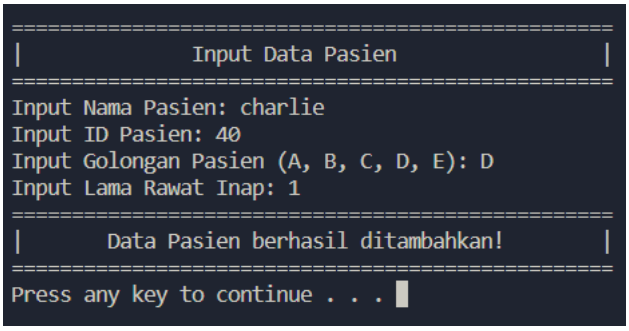
```
                cout << "|          Jenis pencarian tidak
valid!          |";
                cout <<
"================================================\n";
            }
            cout <<
"================================================\n";
            system("pause");
            break;
        }
        case 4: {
            system("cls");
            cout <<
"================================================\n";
            cout << "                Hapus Data
Pasien                \n";
            cout <<
"================================================\n";
            int id;
            cout << "Input ID Pasien yang akan dihapus: ";
            cin >> id;
            sistem.hapusDataPasien(id);
            cout <<
"================================================\n";
            system("pause");
            break;
        }
        case 5:
            system("cls");
            cout <<
"================================================\n";
            cout << "               History Data
Pasien                \n";
            cout <<
"================================================\n";
            sistem.tampilkanHistoryPasien();
            cout <<
"================================================\n";
            system("pause");
```

```
            break;
        case 6:
            cout << "Keluar dari program.\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
            break;
        }
    } while (pilihan != 6);

    return 0;
}
```

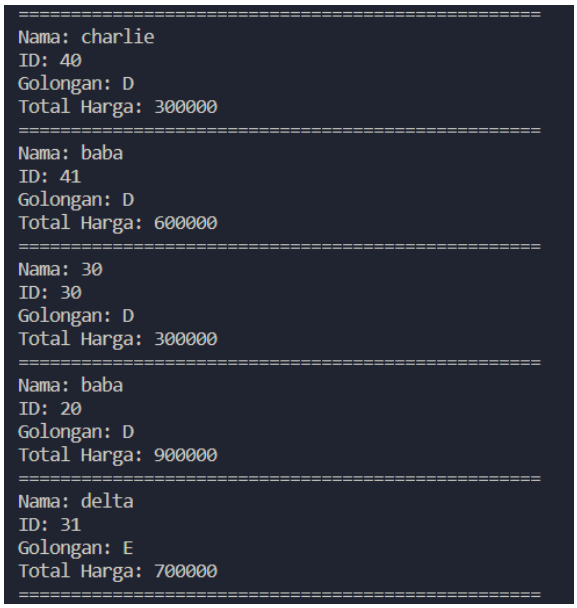
4. Screenshot Hasil Source Code



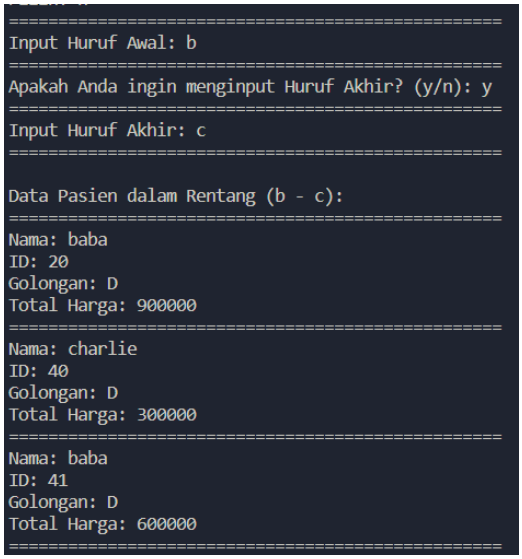
Gambar 4.1 Menu utama program



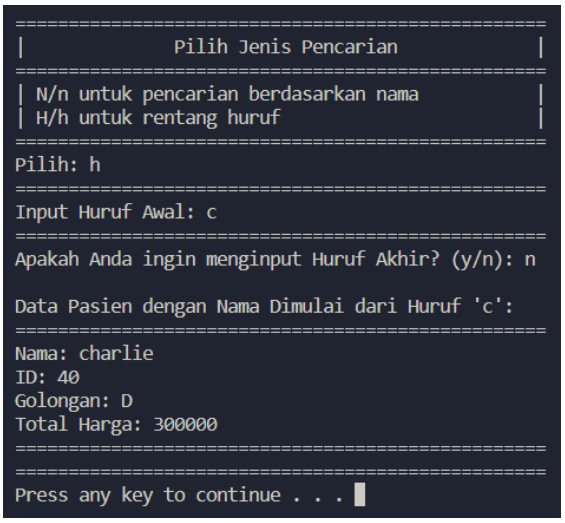
Gambar 4.2 Menu tambah data pasien



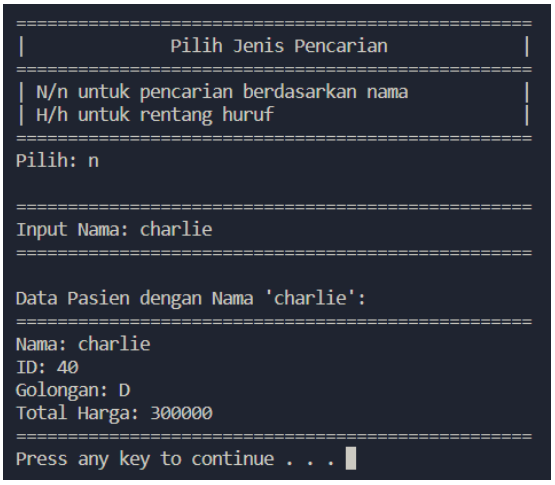
Gambar 4.3 Menu tampil data pasien



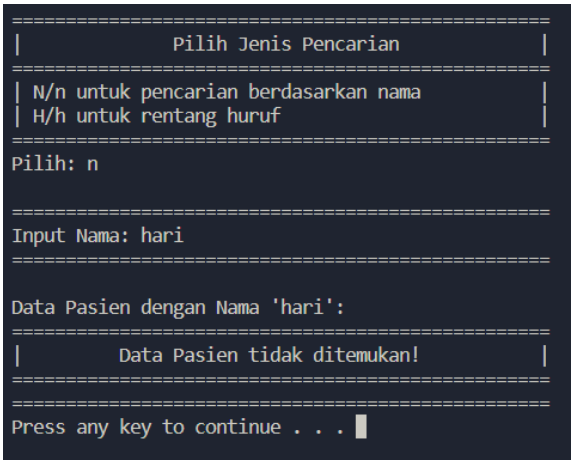
Gambar 4.4 Cari data pasien dalam rentang huruf awal b sampai c



Gambar 4.5 Cari data pasien dengan huruf awal c



Gambar 4.6 Cari data pasien dengan nama pasien ‘charlie’



Gambar 4.7 Hasil apabila data pasien tidak ada ketika mencari data pasien

```
==============================================
                Hapus Data Pasien
==============================================
Input ID Pasien yang akan dihapus: 20
==============================================
Data Pasien Nama: baba & ID 20 dihapus!
==============================================
Press any key to continue . . .
```

Gambar 4.8 Menghapus data pasien dengan id ‘20’

```
==============================================
Nama: azhar
ID: 10
Golongan: C
Total Harga: 750000
==============================================
Nama: charlie
ID: 40
Golongan: D
Total Harga: 300000
==============================================
Nama: baba
ID: 41
Golongan: D
Total Harga: 600000
==============================================
Nama: 30
ID: 30
Golongan: D
Total Harga: 300000
==============================================
Nama: delta
ID: 31
Golongan: E
Total Harga: 700000
==============================================
Press any key to continue . . .
```

Gambar 4.9 Menu tampil data pasien setelah menghapus salah satu data

```
==============================================
                Hapus Data Pasien
==============================================
Input ID Pasien yang akan dihapus: 3
Data Pasien dengan ID 3 tidak ditemukan.
==============================================
Press any key to continue . . .
```

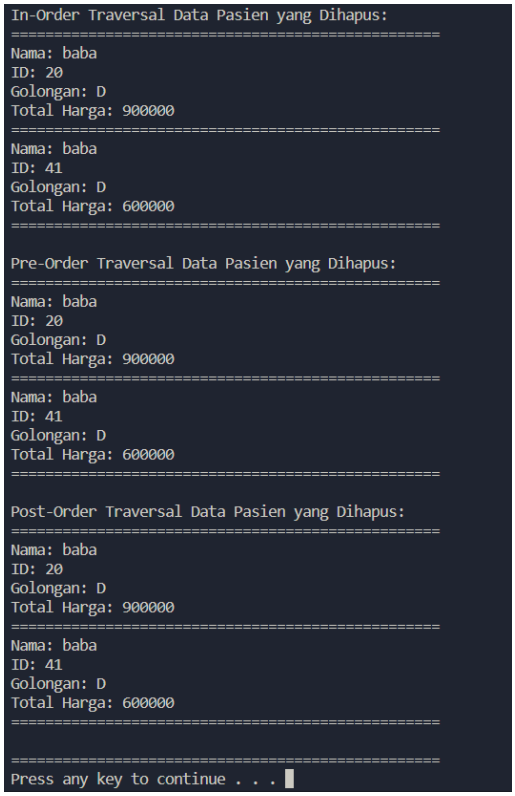
Gambar 4.10 Menghapus data pasien yang belum pernah dimasukkan

```
==============================================
              History Data Pasien
==============================================

History Pasien (In-Order Traversal):
==============================================
Nama: azhar
ID: 10
Golongan: C
Total Harga: 750000
==============================================
Nama: 30
ID: 30
Golongan: D
Total Harga: 300000
==============================================
Nama: delta
ID: 31
Golongan: E
Total Harga: 700000
==============================================
Nama: charlie
ID: 40
Golongan: D
Total Harga: 300000
==============================================

History Pasien (pre-Order Traversal):
==============================================
Nama: azhar
ID: 10
Golongan: C
Total Harga: 750000
==============================================
Nama: charlie
ID: 40
Golongan: D
Total Harga: 300000
==============================================
Nama: 30
ID: 30
Golongan: D
Total Harga: 300000
==============================================
Nama: delta
```

Gambar 4.11 Tampil history data pasien (bagian 1)

```
Nama: delta
ID: 31
Golongan: E
Total Harga: 700000
==============================================
Nama: 30
ID: 30
Golongan: D
Total Harga: 300000
==============================================
Nama: charlie
ID: 40
Golongan: D
Total Harga: 300000
==============================================
Nama: azhar
ID: 10
Golongan: C
Total Harga: 750000
==============================================

In-Order Traversal Data Pasien yang Dihapus:
==============================================
Nama: baba
ID: 20
Golongan: D
Total Harga: 900000
==============================================
Nama: baba
ID: 41
Golongan: D
Total Harga: 600000
==============================================

Pre-Order Traversal Data Pasien yang Dihapus:
==============================================
Nama: baba
ID: 20
Golongan: D
Total Harga: 900000
```

Gambar 4.11 Tampil history data pasien (bagian 2}



Gambar 4.11 Tampil history data pasien (bagian 3}