STRUKTUR DATA
Tugas pegawai dg doubly linked list

NAMA: Bagas Diatama Wardoyo
NPM: 140810230061

Dikumpulkan tanggal :
28 April 2024

Universitas Padjadjaran
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
Program Studi S-1 Teknik Informatika
2024

Source Code:

```
/* Nama program : DataPegawai_Doubly
   Nama         : Bagas Diatama Wardoyo
   NPM          : 140810230061
   Tanggal buat : 28/04/2024
   Deskripsi    : Tugas Doubly Linked List
****************************************************************
*******/
#include <iostream>
#include <string>

using namespace std;

struct Pegawai
{
    string NIP, nama, gaji;
    int golongan;
};

struct Node
{
    Pegawai data;
    Node *next;
    Node *prev;
};

typedef Node *Pointer;
typedef Pointer List;

string golonganGaji(int gol);
void menu(); //Menampilkan menu
void createList(Pointer &head); // Membuat list
void createElement(Pointer &pBaru); // Membuat element yang
akan ditambahkan ke dalam list
void insertFirstDoubly(List &Head, Pointer pBaru); //
Menyisipkan element ke depan list
void insertLastDoubly(List &Head, Pointer pBaru); //
Menyisipkan element ke akhir list
void linearSearch(List Head, string key, bool &found, Pointer
&pCari); // Mencari data dengan kunci NIP
```

```
void insertAfterDoubly(List &Head, Pointer pCari, Pointer
pBaru); // Menambahkan element setelah data dengan NIP yang
diinginkan
void insertBeforeDoubly(List &Head, Pointer pCari, Pointer
pBaru); // Menambahkan element sebelum data dengan NIP yang
diinginkan
void deleteFirstDoubly(List &Head, Pointer &pHapus); //
Menghapus element pertama list
void deleteLastDoubly(List &Head, Pointer &pHapus); //
Menghapus element terakhir list
void deleteAfterDoubly(List &Head, Pointer pCari, Pointer
&pHapus); // Menghapus element setelah element yang diinginkan
void deleteBeforeDoubly(List &Head, Pointer pCari, Pointer
&pHapus); // Menghapus element sebelum element yang diinginkan
void delete_pCari(List &Head, Pointer pCari, Pointer &pHapus);
// Mencari element dengan key NIP
void traversal(List Head); // Menampilkan seluruh list

main()
{
    List head, newNode;
    Pointer pHapus, pCari;
    string key;
    bool found;
    bool loop = 1;
    int option;
    while (loop)
    {
        menu();
        cin >> option;
        switch (option)
        {
        case 1:
            createList(head);
            cout << "List Berhasil Dibuat\n";
            break;
        case 2:
            createElement(newNode);
            cout << "Element List Berhasil Dibuat\n";
            break;
        case 3:
            insertFirstDoubly(head, newNode);
            cout << "Node berhasil disisipkan di awal\n";
            break;
        case 4:
            insertLastDoubly(head, newNode);
            cout << "Node berhasil disisipkan di akhir\n";
            break;
        case 5:
            cout << "Node akan disisipkan setelah NIP:  ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            insertAfterDoubly(head, pCari, newNode);
            cout << "Node telah disisipkan setelah NIP " << key
<< endl;
            break;
        case 6:
            cout << "Node akan disisipkan sebelum NIP:  ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            insertBeforeDoubly(head, pCari, newNode);
            cout << "Node telah disisipkan sebelum NIP " << key
<< endl;
            break;
        case 7:
            cout << "Node akan dihapus setelah NIP:  ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            deleteAfterDoubly(head, pCari, pHapus);
            cout << "Node telah dihapus setelah NIP " << key <<
endl;
            break;
        case 8:
            cout << "Node akan dihapus sebelum NIP:  ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            deleteBeforeDoubly(head, pCari, pHapus);
            cout << "Node telah dihapus sebelum NIP " << key <<
endl;
            break;
        case 9:
            cout << "Node akan dihapus merupakan NIP:  ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            delete_pCari(head, pCari, pHapus);
            cout << "Node telah NIP telah dihapus " << key <<
endl;
            break;
        case 10:
            cout << "Masukkan data yang akan dicari (NIP): ";
            cin >> key;
            linearSearch(head, key, found, pCari);
            if (found == 1)
            {
                cout << pCari->data.NIP << "Ditemukkan\n";
            }
            else
            {
                cout << "Data tidak ditemukkan\n";
            }
        case 11:
            traversal(head);
            break;
        default:
            loop = 0;
            break;
        }
    }
}

void menu()
{
    cout << "\nMenu:" << endl;
    cout << "1. Create List" << endl;
    cout << "2. Create Elemen" << endl;
    cout << "3. Insert First" << endl;
    cout << "4. Insert Last" << endl; // Menampilkan menu
dengan pilihan (Create List, Create Elemen, Insert Head, Delete
Head, Traversal, Keluar)
    cout << "5. Insert After" << endl;
```

```cpp
    cout << "6. Insert Before" << endl;
    cout << "7. Delete After" << endl;
    cout << "8. Delete Before" << endl;
    cout << "9. Delete pCari" << endl;
    cout << "10. Cari Data" << endl;
    cout << "11. Traversal" << endl;
    cout << "0. Keluar" << endl;
    cout << "Masukkan Pilihan >";
}
string golonganGaji(int gol)
{
    string gaji;
    if (gol == 1)
    {
        gaji = "2.5 juta";
    }
    else if (gol == 2)
    {
        gaji = "3.5 juta";
    }
    else if (gol == 3)
    {
        gaji = "5 juta";
    }
    else if (gol == 4)
    {
        gaji = "7.5 juta";
    }
    return gaji;
}
void createList(Pointer &Head)
{
    Head = nullptr;
}
void createElement(Pointer &pBaru)
{
    pBaru = new Node;
    cout << "Masukkan NIP       :   ";
    cin >> pBaru->data.NIP;
    cin.ignore();
    cout << "Masukkan Nama      :   ";
    getline(cin, pBaru->data.nama);
    cout << "Masukkan Golongan  :   ";
    cin >> pBaru->data.golongan;
    pBaru->data.gaji = golonganGaji(pBaru->data.golongan);
    pBaru->next = nullptr;
    pBaru->prev = nullptr;
}
void insertFirstDoubly(List &Head, Pointer pBaru)
{
    if (Head == nullptr)
    {
        Head = pBaru;
    }
    else
    {
        pBaru->next = Head;
        Head->prev = pBaru;
        Head = pBaru;
    }
}
void insertLastDoubly(List &Head, Pointer pBaru)
{
    if (Head == nullptr)
    {
        Head = pBaru;
    }
    else
    {
        Pointer pHelp = Head;
        while (Head->next != nullptr)
        {
            pHelp = Head->next;
        }
        pHelp->next = pBaru;
        pBaru->prev = pHelp;
    }
}
void insertAfterDoubly(List &Head, Pointer pCari, Pointer
pBaru)
```

```
{
    if (pCari->next == nullptr)
    {
        insertLastDoubly(Head, pBaru);
    }
    else
    {
        pBaru->next = pCari->next;
        pBaru->prev = pCari;
        pBaru->next->prev = pBaru;
        pCari->next = pBaru;
    }
}
void insertBeforeDoubly(List &Head, Pointer pCari, Pointer
pBaru)
{
    if (pCari == Head)
    {
        insertFirstDoubly(Head, pBaru);
    }
    else
    {
        pBaru->next = pCari;
        pBaru->prev = pCari->prev;
        pBaru->prev->next = pBaru;
        pCari->prev = pBaru;
    }
}
void deleteFirstDoubly(List &Head, Pointer &pHapus)
{
    if (Head == nullptr)
    {
        pHapus = Head;
    }
    else if (Head->prev == nullptr)
    {
        pHapus = Head;
        Head = nullptr;
    }
    else
    {
        pHapus = Head;
        Head = pHapus->next;
        pHapus->next = nullptr;
        Head->prev = nullptr;
    }
}
void deleteLastDoubly(List &Head, Pointer &pHapus)
{
    if (Head == nullptr)
    {
        pHapus = Head;
    }
    else if (Head->prev == nullptr)
    {
        pHapus = Head;
        Head = nullptr;
    }
    else
    {
        Pointer pHelp = Head;
        while (pHelp->next != nullptr)
        {
            pHelp = pHelp->next;
        }
        pHapus = pHelp;
        pHelp->prev->next = nullptr;
        pHelp->prev = nullptr;
    }
}
void deleteAfterDoubly(List &Head, Pointer pCari, Pointer
&pHapus)
{
    if (pCari->next == nullptr)
    {
        cout << "Tidak ada yang bisa dihapus\n";
    }
    else if (pCari->next->next == nullptr)
    {
        deleteLastDoubly(Head, pHapus);
```

```cpp
    }
    else
    {
        pHapus = pCari->next;
        pCari->next = pHapus->next;
        pHapus->next->prev = pCari;
        pHapus->next = nullptr;
        pHapus->prev = nullptr;
    }
}
void deleteBeforeDoubly(List &Head, Pointer pCari, Pointer
&pHapus)
{
    pHapus = pCari->prev;
    if (pHapus == nullptr)
    {
        cout << "Tidak ada elemen yang bisa dihapus\n";
    }
    else if (pHapus->prev->prev == nullptr)
    {
        deleteFirstDoubly(Head, pHapus);
    }
    else
    {
        pHapus->prev->next = pCari;
        pCari->prev = pHapus->prev;
        pCari->next = nullptr;
        pCari->prev = nullptr;
    }
}
void delete_pCari(List &Head, Pointer pCari, Pointer &pHapus)
{
    pHapus = pCari;
    if (pCari == Head)
    {
        deleteFirstDoubly(Head, pHapus);
    }
    else if (pCari->next == nullptr)
    {
        deleteLastDoubly(Head, pHapus);
    }
    else
    {
        pCari->prev->next = pCari->next;
        pCari->next->prev = pCari->prev;
        pCari->next = nullptr;
        pCari->next = nullptr;
    }
}
void linearSearch(List Head, string key, bool &found, Pointer
&pCari)
{
    found = 0;
    pCari = Head;
    while (found == 0 && pCari != nullptr)
    {
        if (pCari->data.NIP == key)
        {
            found = 1;
        }
        else
        {
            pCari = pCari->next;
        }
    }
}
void traversal(List Head)
{
    Pointer pHelp = Head;
    cout << "NIP\tNama\tGolongan\tGaji\n";
    cout << "-----------------------------------\n";
    do
    {
        cout << pHelp->data.NIP << "\t" << pHelp->data.nama <<
"\t\t" << pHelp->data.golongan << "\t" << pHelp->data.gaji <<
"\n";
        pHelp = pHelp->next;
    } while (pHelp != nullptr);
    cout << "-----------------------------------\n";
}
```