STRUKTUR DATA

(Tugas7)

Nama : Prames Ray Lapian

NPM : 140810210059

Dikumpulkan tanggal :

17 April 2022

UNIVERSITAS PADJADJARAN

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

Program Studi INFORMATIKA

2022

```
/*  Nama Program    : Tugas7
    Nama            : Prames Ray Lapian
    NPM             : 140810210059
    Tanggal Buat    : 17 April 2022
    Deskripsi       : Buat program Linked List Circular Double untuk data
pegawai
    Lokasi File     : C:\Users\prame\Documents\PRAMES\PERKULIAHAN\SEMESTER
2\Struktur Data\TUGAS
**********************************************/

#include <iostream>
#include <string.h>
#include <iomanip>
using namespace std;

struct Pegawai
{
    char NIP[10];
    char Nama[30];
    char Alamat[30];
    string gol;
    float gaji;
};

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

typedef Node *pointer;
typedef pointer List;

void createElement(pointer& pBaru);
void insertFirst(List& first, pointer pBaru);
void insertLast(List& first, pointer pBaru);
void insertBefore(List& first, pointer pBaru, pointer pCari);
void insertAfter(List& first, pointer pBaru, pointer pCari);
void deleteFirst(List& first, pointer& pHapus);
void deleteLast(List& first, pointer& pHapus);
void deleteBefore(List& first, pointer& pHapus, pointer pCari);
void deleteAfter(List& first, pointer& pHapus, pointer pCari);
void deleteByKey(List& first, pointer& pHapus, pointer pCari, pointer
preCari);
void traversal(List first);
void linearSearch(List first, pointer& pCari, pointer& preCari, char NIP[],
int& found);
```

```
void gajiMaksimal(List first);
void gajiRata(List first);
string golPegawai(int gaji);
char menu();

int main()
{
    List first = NULL;
    pointer pBaru, pHapus, pCari, preCari;
    char keyNIP[10];
    int opsi;
    int found = 0;
    bool program = true;

    while (program)
    {
        int pil = menu();

        switch (pil)
        {
            case 1:
                createElement(pBaru);
                insertFirst(first, pBaru);
                traversal(first);
                break;

            case 2:
                createElement(pBaru);
                insertLast(first, pBaru);
                traversal(first);
                break;

            case 3:
                cout << "\nMasukkan NIP pencarian : "; cin.get(keyNIP,10);
                cin.ignore();
                linearSearch(first, pCari, preCari, keyNIP, found);

                if (found)
                {
                    cout << "\nData yang dicari telah ditemukan!" << endl;
                    createElement(pBaru);
                    insertBefore(first, pBaru, pCari);
                    traversal(first);
                }
                else
                {
                    cout << "\nData yang dicari tidak ditemukan." << endl;
                }
```

```
            break;

        case 4:
            cout << "\nMasukkan NIP pencarian : "; cin.get(keyNIP,10);
            cin.ignore();
            linearSearch(first, pCari, preCari, keyNIP, found);

            if (found)
            {
                cout << "\nData yang dicari telah ditemukan!" << endl;
                createElement(pBaru);
                insertAfter(first, pBaru, pCari);
                traversal(first);
            }
            else
            {
                cout << "\nData yang dicari tidak ditemukan." << endl;
            }
            break;

        case 5:
            deleteFirst(first, pHapus);
            cout << endl;
            traversal(first);
            break;

        case 6:
            deleteLast(first, pHapus);
            cout << endl;
            traversal(first);
            break;


        case 7:
            cout << "\nMasukkan NIP Pencarian\t: "; cin.get(keyNIP,10);
            cin.ignore();
            linearSearch(first, pCari, preCari, keyNIP, found);

            if (found)
            {
                deleteBefore(first, pHapus, pCari);
                cout << endl;
                traversal(first);
            }
            else
            {
                cout << "\nData Tidak Ditemukan!" << endl;
            }
```

```
            break;

        case 8:
            cout << "\nMasukkan NIP Pencarian\t: "; cin.get(keyNIP,10);
            cin.ignore();
            linearSearch(first, pCari, preCari, keyNIP, found);

            if (found)
            {
                deleteAfter(first, pHapus, pCari);
                cout << endl;
                traversal(first);
            }
            else
            {
                cout << "\nData Tidak Ditemukan!" << endl;
            }
            break;

        case 9:
            cout << "\nMasukkan NIP Pencarian\t: "; cin.get(keyNIP,10);
            cin.ignore();
            linearSearch(first, pCari, preCari, keyNIP, found);

            if (found)
            {
                deleteByKey(first, pHapus, pCari, preCari);
                cout << endl;
                traversal(first);
            }
            else
            {
                cout << "\nData Tidak Ditemukan!" << endl;
            }

            break;

        case 10:
            traversal(first);
            break;

        case 11:
            gajiMaksimal(first);
            break;

        case 12:
            gajiRata(first);
            break;
```

```
            default:
                cout << "\nPilihan Tidak Tersedia." << endl;
                break;
        }

        cout << "\nIngin terus menggunakan program?" << endl
             << "1.YA                              " << endl
             << "2.TIDAK                           " << endl
             << "Pilihan\t: "; cin >> opsi; cin.ignore();

        if (opsi == 1)
        {
            program = true;
        }
        else if (opsi == 2)
        {
            program = false;
            cout << "\nTerima kasih!" << endl;
        }
        else
        {
            program = false;
            cout << "\nPilihan Tidak Tersedia" << endl;
        }
    }
}

void linearSearch(List first, pointer& pCari, pointer& preCari, char NIP[],
int& found)
{
    found = 0;
    pCari = first;

    do
    {
        if (strcmp(pCari->info.NIP, NIP) == 0)
        {
            found = 1;
            break;
        }
            preCari = pCari;
            pCari = pCari -> next;
    } while (pCari != first);
}

void createElement(pointer& pBaru)
```

```
{
    pBaru = new Node;

    cout << "\nData pegawai yang ingin ditambahkan:" << endl;
    cout << "NIP    : "; cin.get(pBaru -> info.NIP,10); cin.ignore();
    cout << "Nama   : "; cin.get(pBaru -> info.Nama,30); cin.ignore();
    cout << "Alamat : "; cin.get(pBaru -> info.Alamat,30); cin.ignore();
    cout << "Gaji   : "; cin >> pBaru -> info.gaji; cin.ignore();

    pBaru -> info.gol = golPegawai(pBaru -> info.gaji);
    pBaru -> next = NULL;
}

void insertFirst(List& first, pointer pBaru)
{
    if (first == NULL)
    {
        first = pBaru;
    }
    else
    {
        pointer last = first;

        while (last != first)
        {
            last = last->next;
        }

        pBaru->next = first;
        pBaru -> prev = last;
        last -> next = pBaru;
        first->prev = pBaru;
        first = pBaru;
    }
}

void insertLast(List& first, pointer pBaru)
{
    if (first == NULL)
    {
        first = pBaru;
    }
    else
    {
        pointer last = first;

        while (last -> next != first)
        {
```

```
            last = last -> next;
        }

        pBaru -> prev = last;
        pBaru -> next = first;
        last -> next = pBaru;
        first -> prev = pBaru;
    }
}

void insertAfter(List& first, pointer pBaru, pointer pCari)
{
    if (pCari -> next == first)
    {
        insertLast(first, pBaru);
    }
    else
    {
        pBaru -> next = pCari -> next;
        pBaru -> prev = pCari;
        pBaru -> next -> prev = pBaru;
        pCari -> next = pBaru;
    }
}

void insertBefore(List& first, pointer pBaru, pointer pCari)
{
    pointer last = first;

    while (last -> next != first)
    {
        last = last -> next;
    }

    if (pCari -> prev == last)
    {
        insertFirst(first, pBaru);
    }
    else
    {
        pBaru -> next = pCari;
        pBaru -> prev = pCari -> prev;
        pCari -> prev -> next = pBaru;
        pCari -> prev = pBaru;
    }
}

void deleteFirst(List& first, pointer& pHapus)
```

```
{
    if (first == NULL)
    {
        pHapus = NULL;
    }
    else if (first -> next == first)
    {
        pHapus = first;
        first = NULL;
    }
    else
    {
        pointer last = first;

        while (last -> next != first)
        {
            last = last -> next;
        }

        pHapus = first;
        first = first -> next;
        first -> prev = last;
        last -> next = first;
        pHapus -> next = NULL;
        pHapus -> prev = NULL;
    }
}

void deleteLast(List& first, pointer& pHapus)
{
    if (first == NULL)
    {
        pHapus = NULL;
    }
    else if (first -> next == first)
    {
        pHapus = first;
        first = NULL;
    }
    else
    {
        pointer last = first;

        while (last -> next != first)
        {
            last = last -> next;
        }
```

```
        pHapus = last;
        last = last -> prev;
        last -> next = first;
        first -> prev = last;
        pHapus -> next = NULL;
        pHapus -> prev = NULL;
    }
}

void deleteBefore(List& first, pointer& pHapus, pointer pCari)
{
    if (pCari -> next == NULL)
    {
        pHapus = NULL;
        cout << "Tidak ada yang dihapus" << endl;
    }
    else
    {
        pHapus = pCari -> prev;
        pHapus -> prev -> next = pCari;
        pCari -> prev = pHapus -> prev;
        pHapus -> next = NULL;
        pHapus -> prev = NULL;
    }
}

void deleteAfter(List& first, pointer& pHapus, pointer pCari)
{
    if (pCari -> next == NULL)
    {
        pHapus = NULL;
        cout << "Tidak ada yang dihapus" << endl;
    }
    else
    {
        pHapus = pCari -> next;
        pCari -> next = pHapus -> next;
        pHapus -> next -> prev = pCari;
        pHapus -> next = NULL;
        pHapus -> prev = NULL;
    }
}

void deleteByKey(List& first, pointer& pHapus, pointer pCari, pointer preCari)
{
    if (pCari == first)
    {
        deleteFirst(first, pHapus);
```

```
    }
    else if (pCari -> next == first)
    {
        deleteLast(first, pHapus);
    }
    else
    {
        deleteAfter(first, pHapus, preCari);
    }
}

void traversal(List first)
{
    if (first == NULL)
    {
        cout << "\nList kosong!" << endl;
    }
    else
    {
        pointer pBantu = first;
        cout << endl;
        cout << setw(10) << "NIP" << setw(30) << "NAMA" << setw(30) <<
"ALAMAT" << setw(10) << "GOL" << setw(15) << "GAJI" << endl;

        do
        {
            cout << setw(10) << pBantu -> info.NIP << setw(30) << pBantu ->
info.Nama << setw(30) << pBantu -> info.Alamat;
            cout << setw(10) << pBantu -> info.gol << setw(15) << fixed <<
setprecision(0)      << pBantu -> info.gaji << endl;

            pBantu = pBantu -> next;
        }
        while (pBantu != NULL);
    }
}

void gajiMaksimal(List first)
{
    pointer pBantu;
    float maksimal = 0;

    if (first == NULL)
    {
        cout << "\nList kosong!" << endl;
    }
    else
    {
```

```
        pBantu = first;

        do
        {
            if (pBantu->info.gaji > maksimal)
            {
                maksimal = pBantu -> info.gaji;
            }

            pBantu = pBantu -> next;
        }
        while (pBantu != NULL);

        cout << "\nGaji Maksimum\t: Rp" << maksimal << endl;
    }
}

void gajiRata(List first)
{
    pointer pBantu;
    float hasil, rata = 0;
    int i = 0;

    if (first == NULL)
    {
        cout << "\nList kosong!" << endl;
    }
    else
    {
        pBantu = first;

        do
        {
            rata += pBantu -> info.gaji;
            i++;
            pBantu = pBantu -> next;
            hasil = rata/i;
        }
        while (pBantu != NULL);

        cout << "\nRata-Rata Gaji\t: Rp" << hasil << endl;
    }
}

string golPegawai (int gaji)
{
    string gol;
```

```
    if (gaji <= 1000000)
    {
        gol = "1A";
    }
    else if (gaji > 1000000 && gaji <= 2000000)
    {
        gol = "1B";
    }
    else if (gaji > 2000000 && gaji <= 3000000)
    {
        gol = "2A";
    }
    else if (gaji > 3000000 && gaji <= 4000000)
    {
        gol = "2B";
    }
    else if (gaji > 4000000 && gaji <= 5000000)
    {
        gol = "3A";
    }
    else if (gaji > 5000000 && gaji <= 6000000)
    {
        gol = "3B";
    }
    else if (gaji > 6000000 && gaji <= 7000000)
    {
        gol = "4A";
    }
    else if (gaji > 7000000)
    {
        gol = "4B";
    }

    return gol;
}

char menu()
{
    int opsi;
    cout << "======================================" << endl
         << "          MENU PROGRAM PEGAWAI        " << endl
         << "======================================" << endl
         << "1.  Input Data Pertama Pegawai        " << endl
         << "2.  Input Data Terakhir Pegawai       " << endl
         << "3.  Input Data Pegawai (Before Key)   " << endl
         << "4.  Input Data Pegawai (After Key)    " << endl
         << "5.  Hapus Data Pertama Pegawai        " << endl
         << "6.  Hapus Data Terakhir Pegawai       " << endl
```

```
        << "7.  Hapus Data Pegawai (Before Key)    " << endl
        << "8.  Hapus Data Pegawai (After Key)     " << endl
        << "9.  Hapus Data Pegawai (By Key)        " << endl
        << "10. Tampilkan Seluruh List Data Pegawai" << endl
        << "11. Tampilkan Gaji Maksimum Pegawai    " << endl
        << "12. Tampilkan Rata-Rata Gaji Pegawai   " << endl << endl
        << "Pilihan\t: "; cin >> opsi; cin.ignore();

    return opsi;
}
```