**4.1 Operasi Dasar Singly linked List 1 (Primitive List).PDF -> Halaman 28**

**Nomor 1**

1. Ilustrasi Gambar
   1. Pertama buat list bernama “mahasiswaNilai”

mahasiswaNilai

* 1. Selanjutnya buat elemen dengan banyaknya sesuai inputan user

Elemen dari list “mahasiswaNilai”, di mana elemen berupa data struct “ElemtList”

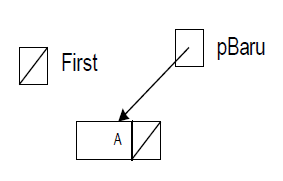
* 1. Masukkan data ke dalam elemen baru yang sudah dialokasikan sebelumnya

NPM  
Nama  
Nilai

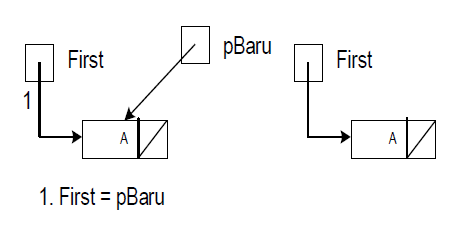
* 1. Tunjuk elemen yang sudah teralokasi dan didefinisikan sebelumnya menjadi posisi awal dari list dengan fungsi “insertFirst”

Insert first pada list kosong :

Keadaan awal

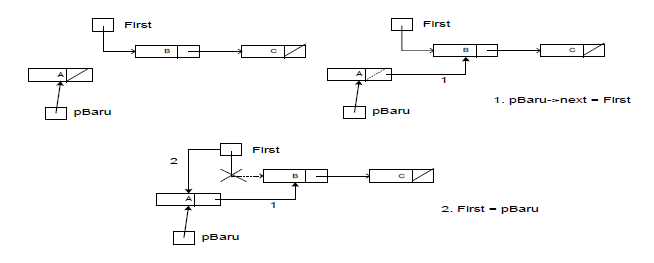


Keadaan akhir

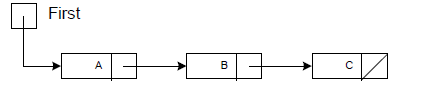


Insert first pada list yang sudah berisi :

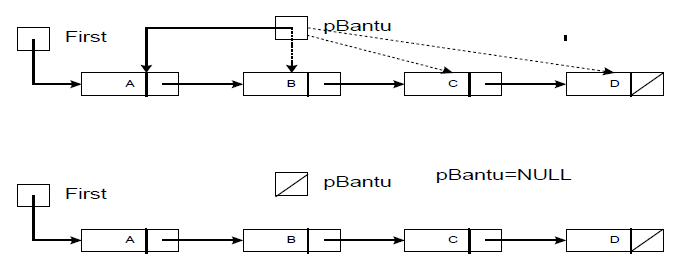
Keadaan awal



Keadaan akhir



* 1. Ulangi proses c dan d sebanyak inputan user di awal program.
  2. Terakhir, cetak data dengan menggunakan fungsi “traversal”



Fungsi traversal adalah menelusuri setiap elemen yang ada, dimulai dari elemen pertama, untuk kemudian mencetak data dari tiap elemen yang dikunjungi. Untuk itu fungsi traversal membutuhkan pointer baru yaitu “pBantu” untuk pengecekan tiap elemen yang ada pada list. Selama pointer bantuan tidak bernilai NULL, maka proses pengecekan elemen akan terus berlanjut.

1. Perintah

Buatlah fungsi untuk menampilkan mahasiswa yang lulus dan yang gagal dengan ketentuan :

Nilai >= 60 ; keterangan lulus

Nilai < 60 ; keterangan gagal

Informasi pada elemen list mahasiswa adalah NPM, nama dan nilai. Gunakan skema dasar traversal.

1. Source Code

/\*

Nama Program : Program Tugas Operasi Dasar Singly Linked List

Tanggal : 07 April 2017

Nama : Eko Fajar Putra

NPM : 140810160061

Deskripsi : Program modular tugas operasi dasar singly linked list 1 halaman 28

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

struct ElemtList{

char npm[14];

char nama[30];

int nilai;

ElemtList\* next;

};

typedef ElemtList\* pointer;

typedef pointer list;

void banyakData(int& n){

cout<<"Banyak data : "; cin>>n;

}

void createList(list& first){

first = NULL;

}

void createElmt(pointer& pBaru){

pBaru = new ElemtList; //Alokasi

cout<<"Masukkan NPM : "; cin>>pBaru->npm;

cout<<"Masukkan Nama : "; cin.ignore(); cin.getline(pBaru->nama, 28);

cout<<"Masukkan Nilai : "; cin>>pBaru->nilai;

pBaru->next = NULL;

}

void insertFirst(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen di depan dengan pBaru

if(first==NULL) //Kasus kosong

first = pBaru;

else{ //Kasus ada isi

pBaru->next = first;// 1

first = pBaru;// 2

}

}

void traversal(list first){

pointer pBantu;

if (first==NULL) {

cout<<"List kosong"<<endl;

}

else {

pBantu = first; //Catat elemen pertama

do {

cout<<"NPM : "<<pBantu->npm<<endl;

cout<<"Nama : "<<pBantu->nama<<endl;

cout<<"Nilai : "<<pBantu->nilai<<endl;

cout<<"Keterangan : ";

if(pBantu->nilai >= 60)

cout<<"Lulus\n\n";

else

cout<<"Gagal\n\n"; //Proses cetak

pBantu = pBantu->next; //Pindah ke berikutnya

} while(pBantu != NULL);

}

}

int main(){

int n;

pointer p;

list mahasiswaNilai;

createList(mahasiswaNilai); //List kosong

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(mahasiswaNilai, p);

cout<<endl;

}

cout<<"============================================\n";

cout<<"\t\tCetak Data\n";

cout<<"============================================\n";

traversal(mahasiswaNilai);

}

**Nomor 2**

1. Ilustrasi Gambar
   1. Pertama buat list bernama “listPegawai”

listPegawai

* 1. Selanjutnya buat elemen dengan banyaknya sesuai inputan user

Elemen dari list “listPegawai”, di mana elemen berupa data struct “ElemtList”

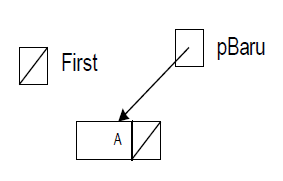
* 1. Masukkan data ke dalam elemen baru yang sudah dialokasikan sebelumnya

NIP  
Nama  
Alamat  
Gol  
Gaji

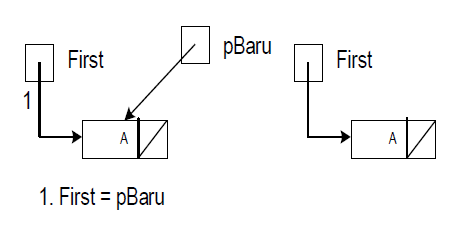
* 1. Tunjuk elemen yang sudah teralokasi dan didefinisikan sebelumnya menjadi posisi awal dari list dengan fungsi “insertFirst”

Insert first pada list kosong :

Keadaan awal

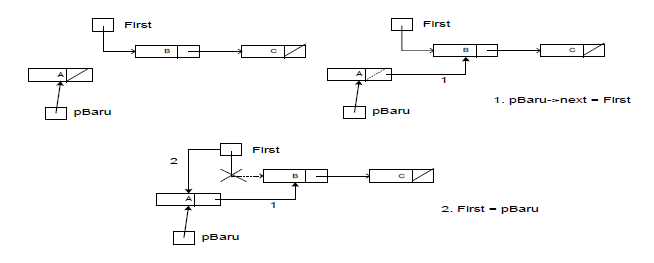


Keadaan akhir

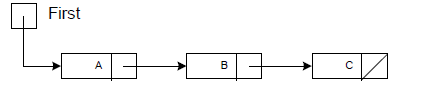


Insert first pada list yang sudah berisi :

Keadaan awal



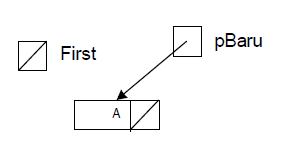
Keadaan akhir



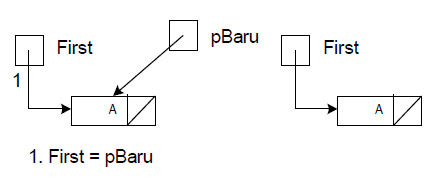
* 1. Tunjuk elemen yang sudah teralokasi dan didefinisikan sebelumnya menjadi posisi akhir dari list dengan fungsi “insertLast”

Insert last pada list kosong :

Keadaan awal

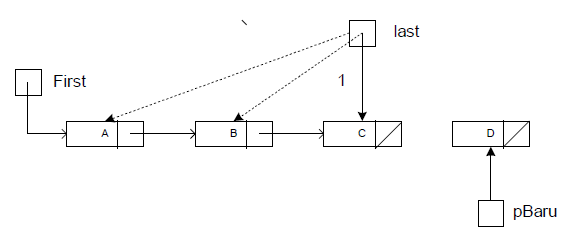


Keadaan akhir

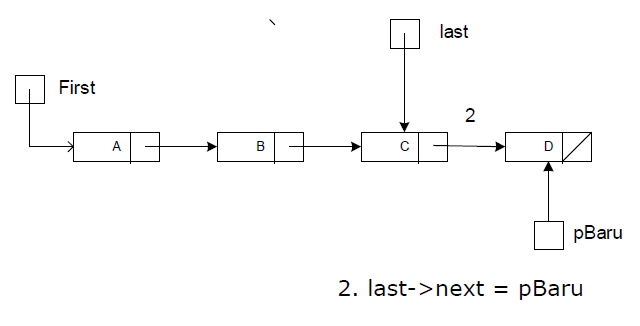


Insert last pada list yang sudah berisi :

Keadaan awal

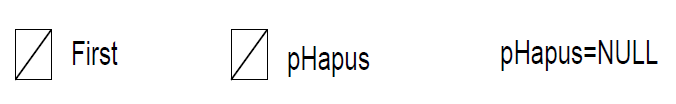


Keadaan akhir



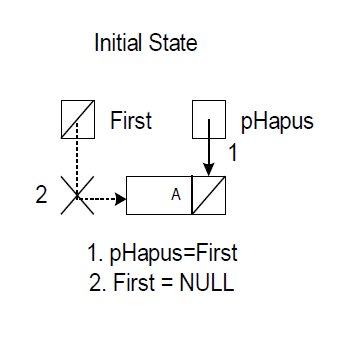
* 1. Hapus elemen yang ada di depan. Elemen yang akan dihapus adalah elemen yang ditunjuk oleh First menggunakan fungsi “deleteFirst”

Delete first pada list kosong :

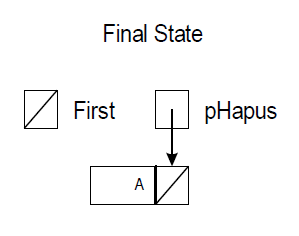


Delete first pada kasus 1 elemen :

Keadaan awal

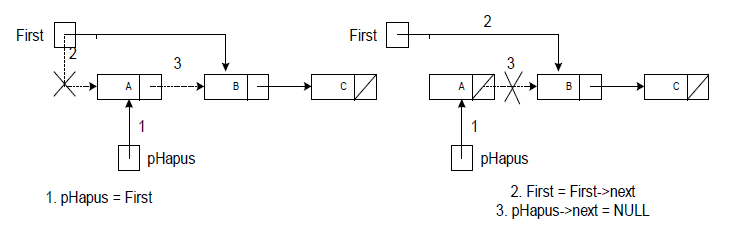


Keadaan akhir

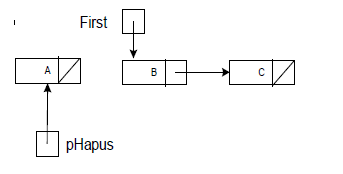


Delete first pada kasus > 1 elemen :

Keadaan awal

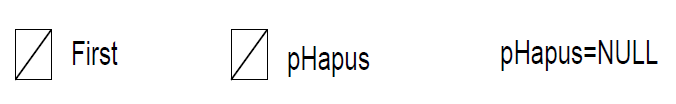


Keadaan akhir



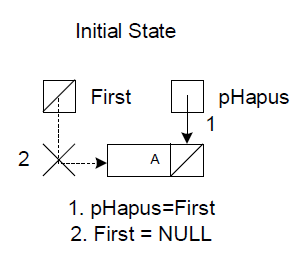
* 1. Hapus elemen yang ada di belakang. Elemen yang akan dihapus adalah elemen yang nextnya bernilai NULL.

Delete last pada list kosong :

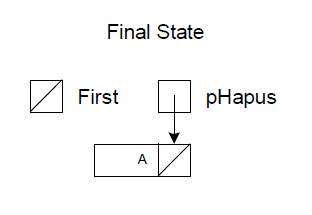


Delete first pada kasus 1 elemen :

Keadaan awal

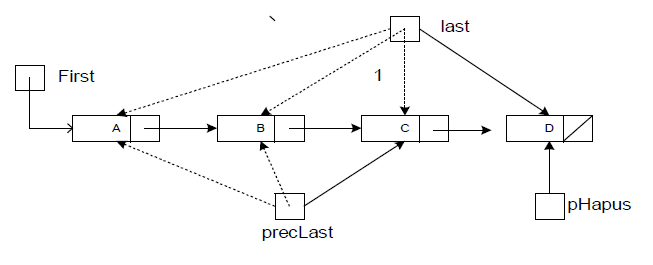


Keadaan akhir

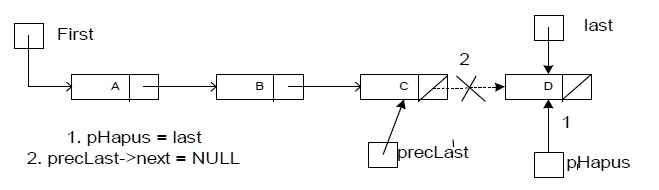


Delete first pada kasus > 1 elemen :

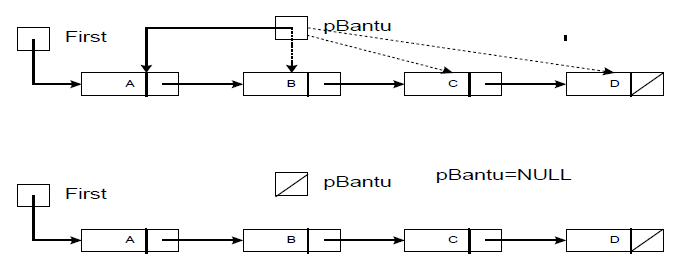
Keadaan awal



Keadaan akhir



* 1. Ulangi proses c sampai dengan g sesuai dengan pilihan inputan user di awal program.
  2. Terakhir, cetak data dengan menggunakan fungsi “traversal”



Fungsi traversal adalah menelusuri setiap elemen yang ada, dimulai dari elemen pertama, untuk kemudian mencetak data dari tiap elemen yang dikunjungi. Untuk itu fungsi traversal membutuhkan pointer baru yaitu “pBantu” untuk pengecekan tiap elemen yang ada pada list. Selama pointer bantuan tidak bernilai NULL, maka proses pengecekan elemen akan terus berlanjut.

1. Perintah

Buatlah program lengkap semua primitive list yang sudah dipelajari dengan pengelolaan menggunakan suatu menu. Kasus list yang ditangani adalah list pegawai dengan atribut NIP, nama, alamat, gol, dan gaji.

1. Source Code

/\*

Nama Program : Program Tugas Operasi Dasar Singly Linked List

Tanggal : 07 April 2017

Nama : Eko Fajar Putra

NPM : 140810160061

Deskripsi : Program modular latihan operasi dasar singly linked list 1 halaman 28 No 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <conio.h>

using namespace std;

struct ElemtList{

char nip[14];

char nama[20];

char alamat[30];

int gol;

int gaji;

ElemtList\* next;

};

typedef ElemtList\* pointer;

typedef pointer list;

void banyakData(int& n){

cout<<"Banyak data : "; cin>>n;

}

void createList(list& first){

first = NULL;

}

void createElmt(pointer& pBaru){

pBaru = new ElemtList; //Alokasi

cout<<"Masukkan NIP : "; cin>>pBaru->nip;

cout<<"Masukkan Nama : "; cin.ignore(); cin.getline(pBaru->nama, 18);

cout<<"Masukkan Alamat : "; cin.ignore(); cin.getline(pBaru->alamat, 25);

cout<<"Masukkan Golongan : "; cin>>pBaru->gol;

cout<<"Masukkan Gaji : "; cin>>pBaru->gaji;

pBaru->next = NULL;

}

void insertFirst(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen di depan denganpBaru

if(first==NULL) // kasus kosong

first = pBaru;

else{ // kasus ada isi

pBaru->next = first;// 1

first = pBaru;// 2

}

}

void traversal(list first){

pointer pBantu;

if (first==NULL) {

cout<<"List kosong"<<endl;

}

else {

pBantu = first; //Catat elemen pertama

do {

cout<<"NIP : "<<pBantu->nip<<endl;

cout<<"Nama : "<<pBantu->nama<<endl;

cout<<"Alamat : "<<pBantu->alamat<<endl;

cout<<"Golongan : "<<pBantu->gol<<endl;

cout<<"Gaji : "<<pBantu->gaji<<endl<<endl; //Proses cetak

pBantu = pBantu->next; //Pindah ke berikutnya

} while(pBantu != NULL);

}

}

void deleteFirst(list& first, pointer& pHapus){

// I.S List First mungkin kosong

// F.S. List berkurang satu di depan, yang dihapus dikembalikan

if (first==NULL){ // kosong

pHapus = NULL;

cout<<"List kosong, tidak ada yang dihapus"<<endl;

}

else if (first->next==NULL){ //isi 1 elemen

pHapus = first;

first = NULL;

}

else { // isi > 1 elemen

pHapus = first; //Tahap 1

first = first->next; //Tahap 2

pHapus->next = NULL; //Tahap 3

}

}

void insertLast(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen dibelakang(sesudah last)

pointer last; //Last untuk mencatat elemen terakhir

if (first==NULL){ //Kosong

first = pBaru;

}

else { //Ada isi

last = first; //Menemukan elemen terakhir

while (last->next != NULL){

last = last->next;

}

last->next = pBaru;// sambungkan

}

}

void deleteLast(list& first, pointer& pHapus){

// I.S List First mungkin kosong

// F.S. List berkurang satu di belakang, yang dihapus dikembalikan

pointer last, precLast;

cout<<endl;

if (first==NULL){ // kosong

pHapus = NULL;

cout<<"List kosong, tidak ada yang dihapus"<<endl;

}

else if (first->next==NULL){ //Isi 1 elemen

pHapus = first;

first = NULL; //List jadi kosong

}

else { //Isi > 1 elemen

last = first; //Menemukan elemen terakhir

precLast = NULL;

while (last->next != NULL){

precLast = last; //Preclast mencatat yg akan ditinggalkan Last

last = last->next; //Last berpindah

}

pHapus = last;

precLast->next = NULL;

}

}

int main(){

int n, pilih;

char jawab;

pointer p;

list listPegawai;

createList(listPegawai); //List kosong

do{

system("cls");

cout<<"============================================\n";

cout<<"\t\tProgram List Pegawai\n";

cout<<"============================================\n";

cout<<"1.\tInput dengan Insert First - Delete First - Insert First\n";

cout<<"2.\tInput dengan Insert Last - Delete Last - Insert Last\n";

cout<<"3.\tInput dengan Insert First - Delete Last - Insert First\n";

cout<<"4.\tInput dengan Insert Last - Delete First - Insert Last\n";

cout<<"5.\tInput dengan Insert First - Delete First - Insert Last\n";

cout<<"6.\tInput dengan Insert Last - Delete Last - Insert First\n";

cout<<"7.\tShow data dengan Traversal\n\n";

cout<<"Pilihan (1-7) : "; cin>>pilih;

switch(pilih){

case 1:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 2:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete Last"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 3:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete First"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 4:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 5:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 6:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete First"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 7:

system("cls");

cout<<"============================================\n";

cout<<"\t\tCetak Data\n";

cout<<"============================================\n";

traversal(listPegawai);

break;

default:

cout<<"Pilihan tidak tersedia";

getch();

system("cls");

break;

};

cout<<"Kembali ke menu utama? (Y/N)"; cin>>jawab;

} while(jawab == 'y' || jawab == 'Y');

cout<<"Selesai";

}

**4.2 Operasi Dasar Singly linked List 2 (Primitive List).PDF -> Halaman 11**

**Nomor 1**

1. Ilustrasi Gambar
   1. Pertama buat list bernama “listPegawai”

listPegawai

* 1. Selanjutnya buat elemen dengan banyaknya sesuai inputan user

Elemen dari list “listPegawai”, di mana elemen berupa data struct “ElemtList”

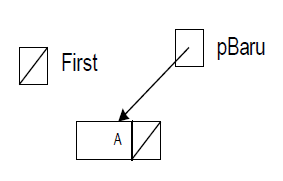
* 1. Masukkan data ke dalam elemen baru yang sudah dialokasikan sebelumnya

nip  
nama  
alamat  
gol  
gaji

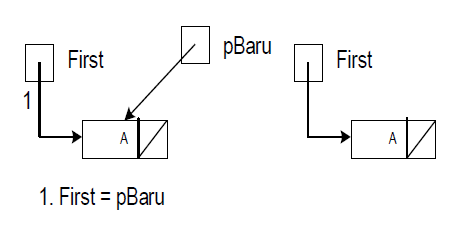
* 1. Tunjuk elemen yang sudah teralokasi dan didefinisikan sebelumnya menjadi posisi awal dari list dengan fungsi “insertFirst”

Insert first pada list kosong :

Keadaan awal

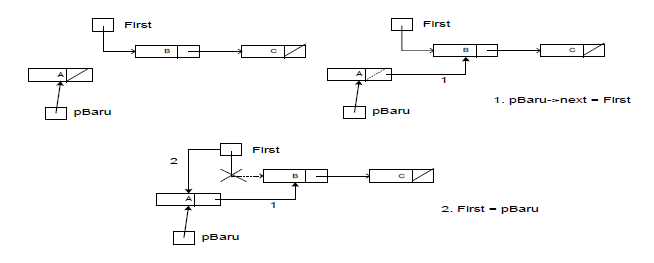


Keadaan akhir

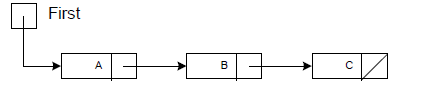


Insert first pada list yang sudah berisi :

Keadaan awal

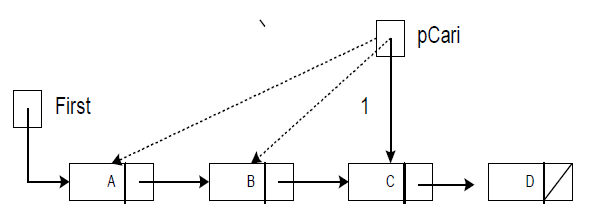


Keadaan akhir

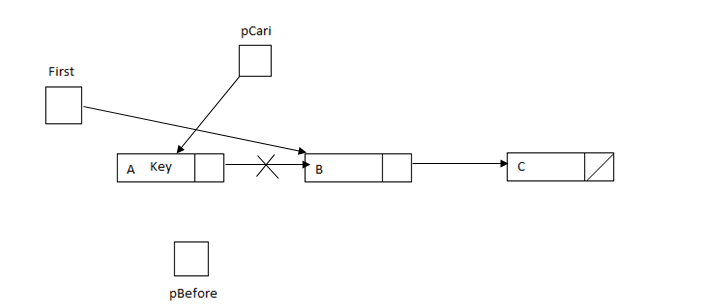


* 1. Pencarian dilakukan untuk mencari isi dari sebuah elemen sesuai dengan keyword yang diinputkan oleh user.

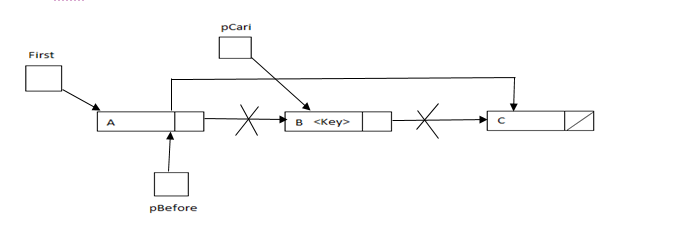
Pencarian dimulai dari elemen pertama dan dilakukan sampai data yang dicari ditemukan ATAU data dalam list sudah habis diperiksa. Atau sebaliknya dikatakan bahwa pencarian dilakukan selama data yang dicari belum ditemukan DAN data dalam list yang akan diperiksa masih ada. {Hukum de Morgan : (A or B)’ = A’ and B’}.



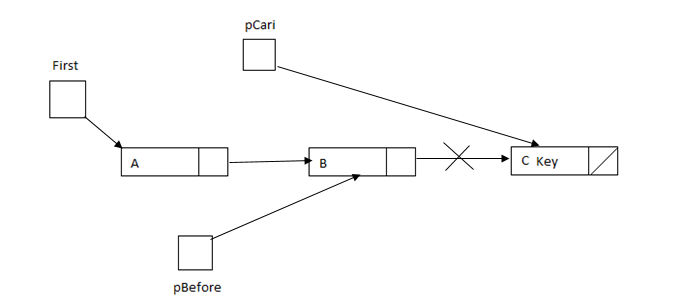
* 1. Ulangi proses c sampai d sebanyak inputan user di awal program.
  2. Lakukan proses e sebelum eksekusi fungsi “deletePCari”. Fungsi “deletePCari” memiliki 3 kemungkinan kasus yang mungkin terjadi :
     1. Elemen berada pada posisi First



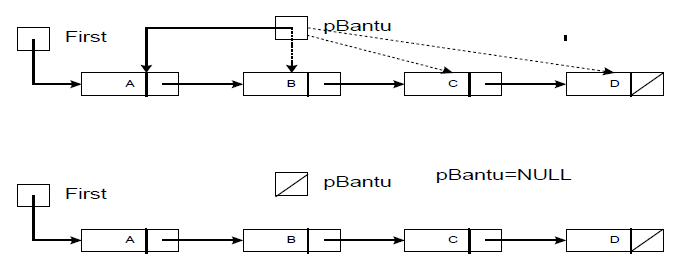
* + 1. Elemen berada pada posisi tengah



* + 1. Elemen berada pada posisi Last



* 1. Terakhir, cetak data dengan menggunakan fungsi “traversal”



Fungsi traversal adalah menelusuri setiap elemen yang ada, dimulai dari elemen pertama, untuk kemudian mencetak data dari tiap elemen yang dikunjungi. Untuk itu fungsi traversal membutuhkan pointer baru yaitu “pBantu” untuk pengecekan tiap elemen yang ada pada list. Selama pointer bantuan tidak bernilai NULL, maka proses pengecekan elemen akan terus berlanjut.

1. Perintah

Buatlah fungsi untuk melakukan penghapusan elemen yang dicari (DeletePCari). Dilakukan pencarian menggunakan Searching dan jika ditemukan maka elemen tersebut akan dihapus dari list berkait. Analisis semua kasus yang mungkin terjadi. Tambahkan pointer bantuan untuk membantu proses pemutusan elemen.

1. Source Code

/\*

Nama Program : Program Latihan Operasi Dasar Singly Linked List

Tanggal : 05 April 2017

Nama : Eko Fajar Putra

NPM : 140810160061

Deskripsi : Program modular latihan operasi dasar singly linked list 2 halaman 11 No 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <conio.h>

using namespace std;

struct ElemtList{

char nip[14];

char nama[20];

char alamat[30];

int gol;

int gaji;

ElemtList\* next;

};

typedef ElemtList\* pointer;

typedef pointer list;

void banyakData(int& n){

cout<<"Banyak data : "; cin>>n;

}

void createList(list& first){

first = NULL;

}

void createElmt(pointer& pBaru){

pBaru = new ElemtList; //Alokasi

cout<<"Masukkan NIP : "; cin>>pBaru->nip;

cout<<"Masukkan Nama : "; cin.ignore(); cin.getline(pBaru->nama, 18);

cout<<"Masukkan Alamat : "; cin.ignore(); cin.getline(pBaru->alamat, 25);

cout<<"Masukkan Golongan : "; cin>>pBaru->gol;

cout<<"Masukkan Gaji : "; cin>>pBaru->gaji;

pBaru->next = NULL;

}

void insertFirst(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen di depan denganpBaru

if(first==NULL) // kasus kosong

first = pBaru;

else{ // kasus ada isi

pBaru->next = first;// 1

first = pBaru;// 2

}

}

void traversal(list first){

pointer pBantu;

if (first==NULL) {

cout<<"List kosong"<<endl;

}

else {

pBantu = first; //Catat elemen pertama

do {

cout<<"NIP : "<<pBantu->nip<<endl;

cout<<"Nama : "<<pBantu->nama<<endl;

cout<<"Alamat : "<<pBantu->alamat<<endl;

cout<<"Golongan : "<<pBantu->gol<<endl;

cout<<"Gaji : "<<pBantu->gaji<<endl<<endl; //Proses cetak

pBantu = pBantu->next; //Pindah ke berikutnya

} while(pBantu != NULL);

}

}

void linearSearch(list First, char key[12], int& found, pointer& pCari){

found = 0; // belum ditemukan

pCari = First; // pencarian mulai dari elemen pertama

while (found==0 && pCari != NULL) {

if (strcmp(pCari->nip, key) == 0)

found=1;

else

pCari=pCari->next;

}

}

void deletePCari(list& First, pointer& pCari){

if(First->next == NULL ){ //Jika First Berisi 1 Elemen

pCari = First;

First = NULL;

}

else if(pCari == First ){ //Jika Elemen Berada pada posisi First

First = First -> next;

pCari -> next = NULL;

}

else if(pCari->next == NULL){ //Jika Pencarian berada pada posisi Last

pointer pBefore;

pBefore = First ;

while (pBefore->next != pCari)

pBefore = pBefore->next;

pBefore->next = NULL;

}

else{ //Jika elemen lebih dari 1

pointer pBefore;

pBefore = First ;

while ( pBefore->next != pCari)

pBefore = pBefore->next;

pBefore->next = pCari->next ;

pCari->next = NULL ;

}

}

int main(){

int n, ketemu;

char kunci[12];

pointer p, pCari;

list listPegawai;

createList(listPegawai); //List kosong

cout<<"\t\tInsert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

cout<<"============================================\n";

cout<<"\t\tDelete pCari\n";

cout<<"============================================\n";

cout<<"NIP yang ingin di-delete : "; cin >>kunci;

linearSearch(listPegawai, kunci, ketemu, pCari);

if (ketemu) {

deletePCari(listPegawai, pCari);

}

else {

cout<<"Data tidak ditemukan"<<endl<<endl;

}

cout<<"============================================\n";

cout<<"\tData setelah delete pCari\n";

cout<<"============================================\n";

traversal(listPegawai);

cout<<"Selesai";

getch();

}

**Nomor 2**

1. Ilustrasi Gambar
   1. Pertama buat list bernama “listPegawai”

listPegawai

* 1. Selanjutnya buat elemen dengan banyaknya sesuai inputan user

Elemen dari list “listPegawai”, di mana elemen berupa data struct “ElemtList”

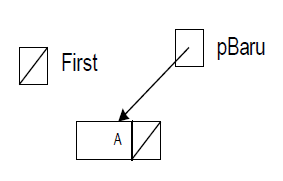
* 1. Masukkan data ke dalam elemen baru yang sudah dialokasikan sebelumnya

nip  
nama  
alamat  
gol  
gaji

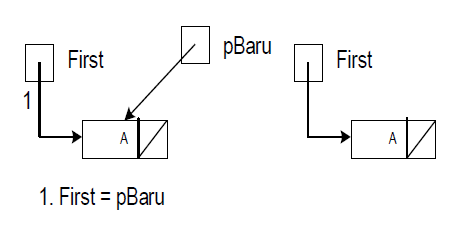
* 1. Tunjuk elemen yang sudah teralokasi dan didefinisikan sebelumnya menjadi posisi awal dari list dengan fungsi “insertFirst”

Insert first pada list kosong :

Keadaan awal

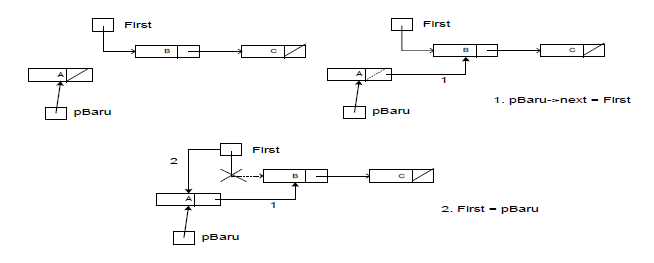


Keadaan akhir

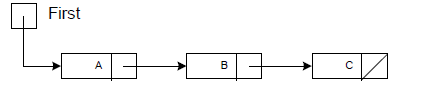


Insert first pada list yang sudah berisi :

Keadaan awal

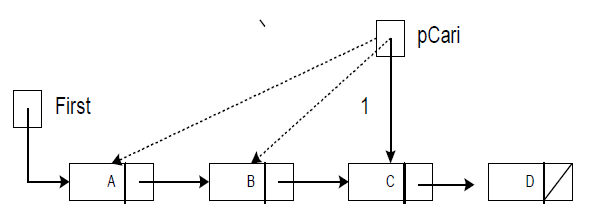


Keadaan akhir



* 1. Pencarian dilakukan untuk mencari isi dari sebuah elemen sesuai dengan keyword yang diinputkan oleh user.

Pencarian dimulai dari elemen pertama dan dilakukan sampai data yang dicari ditemukan ATAU data dalam list sudah habis diperiksa. Atau sebaliknya dikatakan bahwa pencarian dilakukan selama data yang dicari belum ditemukan DAN data dalam list yang akan diperiksa masih ada. {Hukum de Morgan : (A or B)’ = A’ and B’}.

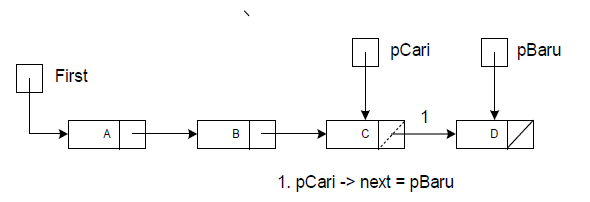


* 1. Insert After adalah proses penyisipan sebuah elemen yang baru sesudah suatu elemen tertentu. Alamat elemen dimana elemen akan disisipkan sudah diketahui terlebih dahulu. Untuk mengetahui dan mendapatkan alamat elemen tertentu bisa menggunakan fungsi pencarian linear search.

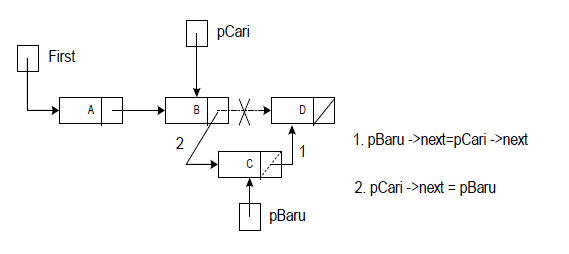
Langkah-langkah :

Temukan suatu elemen di mana elemen yang baru akan disisipkan sesudah elemen tsb. Jika ditemukan, maka ada 2 kasus yang mungkin terjadi :

* Elemen yang dicari ada di belakang sehingga akan terjadi penyisipan di belakang (insert last).



* Elemen yang dicari ada di tengah, sehingga elemen yang baru akan berada di tengah-tengah dua elemen.



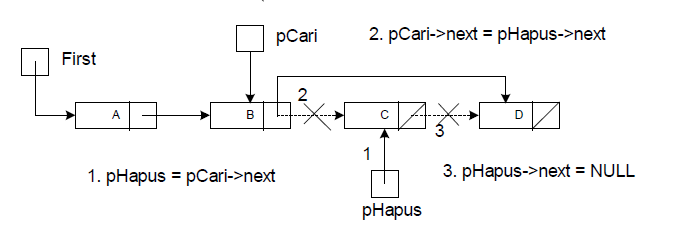
* 1. Delete After adalah proses penghapusan sebuah elemen sesudah suatu elemen tertentu. Alamat elemen dimana elemen sebelumnya berada sudah diketahui terlebih dahulu. Untuk mengetahui dan mendapatkan alamat elemen tertentu bisa menggunakan fungsi pencarian linear search.

Langkah-langkah :

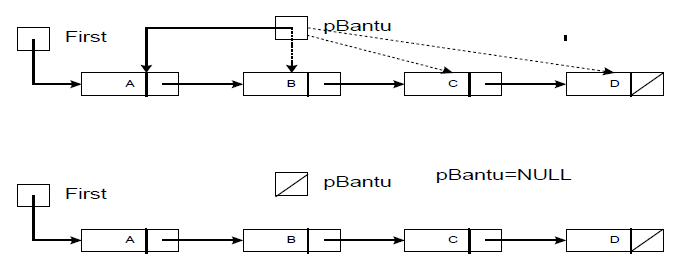
Temukan suatu elemen yang akan dihapus sesudah suatu elemen

Jika ditemukan, maka ada 3 kasus yang mungkin terjadi :

* Elemen yang dicari ada di belakang sehingga tidak akan ada yang dihapus sesudah elemen tsb.
* Elemen yang dicari ada di sebelum elemen terakhir, maka akan dilakukan penghapusan elemen paling terakhir (delete last).
* pCari berada pada selain poin a dan poin b atau yang akan dihapus berada diantara 2 buah elemen.



* 1. Ulangi proses c sampai dengan g sesuai dengan pilihan inputan user di awal program.
  2. Terakhir, cetak data dengan menggunakan fungsi “traversal”



Fungsi traversal adalah menelusuri setiap elemen yang ada, dimulai dari elemen pertama, untuk kemudian mencetak data dari tiap elemen yang dikunjungi. Untuk itu fungsi traversal membutuhkan pointer baru yaitu “pBantu” untuk pengecekan tiap elemen yang ada pada list. Selama pointer bantuan tidak bernilai NULL, maka proses pengecekan elemen akan terus berlanjut.

1. Perintah

Buatlah program lengkap semua primitive list dengan pengelolaan menggunakan suatu menu. Kasus list yang ditangani adalah list pegawai dengan atribut NIP, nama, alamat, gol, dan gaji.

1. Source Code

/\*

Nama Program : Program Latihan Operasi Dasar Singly Linked List

Tanggal : 05 April 2017

Nama : Eko Fajar Putra

NPM : 140810160061

Deskripsi : Program modular latihan operasi dasar singly linked list 2 halaman 11 No 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <conio.h>

using namespace std;

struct ElemtList{

char nip[14];

char nama[20];

char alamat[30];

int gol;

int gaji;

ElemtList\* next;

};

typedef ElemtList\* pointer;

typedef pointer list;

void banyakData(int& n){

cout<<"Banyak data : "; cin>>n;

}

void createList(list& first){

first = NULL;

}

void createElmt(pointer& pBaru){

pBaru = new ElemtList; //Alokasi

cout<<"Masukkan NIP : "; cin>>pBaru->nip;

cout<<"Masukkan Nama : "; cin.ignore(); cin.getline(pBaru->nama, 18);

cout<<"Masukkan Alamat : "; cin.ignore(); cin.getline(pBaru->alamat, 25);

cout<<"Masukkan Golongan : "; cin>>pBaru->gol;

cout<<"Masukkan Gaji : "; cin>>pBaru->gaji;

pBaru->next = NULL;

}

void insertFirst(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen di depan denganpBaru

if(first==NULL) // kasus kosong

first = pBaru;

else{ // kasus ada isi

pBaru->next = first;// 1

first = pBaru;// 2

}

}

void traversal(list first){

pointer pBantu;

if (first==NULL) {

cout<<"List kosong"<<endl;

}

else {

pBantu = first; //Catat elemen pertama

do {

cout<<"NIP : "<<pBantu->nip<<endl;

cout<<"Nama : "<<pBantu->nama<<endl;

cout<<"Alamat : "<<pBantu->alamat<<endl;

cout<<"Golongan : "<<pBantu->gol<<endl;

cout<<"Gaji : "<<pBantu->gaji<<endl<<endl; //Proses cetak

pBantu = pBantu->next; //Pindah ke berikutnya

} while(pBantu != NULL);

}

}

void deleteFirst(list& first, pointer& pHapus){

// I.S List First mungkin kosong

// F.S. List berkurang satu di depan, yang dihapus dikembalikan

if (first==NULL){ // kosong

pHapus = NULL;

cout<<"List kosong, tidak ada yang dihapus"<<endl;

}

else if (first->next==NULL){ //isi 1 elemen

pHapus = first;

first = NULL;

}

else { // isi > 1 elemen

pHapus = first; //Tahap 1

first = first->next; //Tahap 2

pHapus->next = NULL; //Tahap 3

}

}

void insertLast(list& first, pointer pBaru){

// I.S List First mungkin kosong dan pBaru sudah terdefinisi

// F.S List bertambah satu elemen dibelakang(sesudah last)

pointer last; //Last untuk mencatat elemen terakhir

if (first==NULL){ //Kosong

first = pBaru;

}

else { //Ada isi

last = first; //Menemukan elemen terakhir

while (last->next != NULL){

last = last->next;

}

last->next = pBaru;// sambungkan

}

}

void deleteLast(list& first, pointer& pHapus){

// I.S List First mungkin kosong

// F.S. List berkurang satu di belakang, yang dihapus dikembalikan

pointer last, precLast;

cout<<endl;

if (first==NULL){ // kosong

pHapus = NULL;

cout<<"List kosong, tidak ada yang dihapus"<<endl;

}

else if (first->next==NULL){ //Isi 1 elemen

pHapus = first;

first = NULL; //List jadi kosong

}

else { //Isi > 1 elemen

last = first; //Menemukan elemen terakhir

precLast = NULL;

while (last->next != NULL){

precLast = last; //Preclast mencatat yg akan ditinggalkan Last

last = last->next; //Last berpindah

}

pHapus = last;

precLast->next = NULL;

}

}

void linearSearch(list First, char key[12], int& found, pointer& pCari){

found = 0; // belum ditemukan

pCari = First; // pencarian mulai dari elemen pertama

while (found==0 && pCari != NULL) {

if (strcmp(pCari->nip, key) == 0)

found=1;

else

pCari=pCari->next;

}

}

void insertAfter(list& First,pointer pCari, pointer pBaru){

//I.S : List tidak mungkin kosong, pCari sudah diketahui, pBarusudahterdefinisi

//F.S : pBaru disisipkan sesudah pCari

if (pCari->next==NULL)// elementerakhir

pCari->next=pBaru;

else {// bukanterakhir

pBaru->next=pCari->next;

pCari->next=pBaru;

}

}

void deleteAfter(list& First, pointer pCari, pointer& pHapus){

if (pCari->next==NULL){

pHapus = NULL;

cout<<"Tidak ada yang dihapus"<<endl;

}

else {

pHapus = pCari->next;

pCari->next = pHapus->next; // NULL jika pHapus di belakang

pHapus->next = NULL;

}

}

int main(){

int n, pilih, ketemu;

char jawab;

char kunci[12];

pointer p, pCari;

list listPegawai;

createList(listPegawai); //List kosong

do{

system("cls");

cout<<"============================================\n";

cout<<"\t\tProgram List Pegawai\n";

cout<<"============================================\n";

cout<<"1.\tInput dengan Insert First - Delete First - Insert First\n";

cout<<"2.\tInput dengan Insert Last - Delete Last - Insert Last\n";

cout<<"3.\tInput dengan Insert First - Delete Last - Insert First\n";

cout<<"4.\tInput dengan Insert Last - Delete First - Insert Last\n";

cout<<"5.\tInput dengan Insert First - Delete First - Insert Last\n";

cout<<"6.\tInput dengan Insert Last - Delete Last - Insert First\n";

cout<<"7.\tPencarian NIP dengan Linear Search\n";

cout<<"8.\tInput dengan Insert After\n";

cout<<"9.\tDelete dengan Delete After\n\n";

cout<<"10.\tShow data dengan Traversal\n\n";

cout<<"Pilihan (1-10) : "; cin>>pilih;

switch(pilih){

case 1:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 2:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete Last"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 3:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete First"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 4:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 5:

system("cls");

cout<<"Insert First\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete First

cout<<"\nDelete First"<<endl;

deleteFirst(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert Last

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 6:

system("cls");

cout<<"Insert Last\n";

cout<<"============================================\n";

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertLast(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

//Delete Last

cout<<"\nDelete First"<<endl;

deleteLast(listPegawai, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl;

traversal(listPegawai);

//Insert First

banyakData(n);

for(int i=0; i<n; i++){

createElmt(p);

insertFirst(listPegawai, p);

cout<<endl;

}

traversal(listPegawai);

break;

case 7:

system("cls");

cout<<"============================================\n";

cout<<"\t\tLinear Search\n";

cout<<"============================================\n";

cout<<"NIP yang ingin dicari : "; cin >>kunci;

linearSearch(listPegawai, kunci, ketemu, pCari);

if (ketemu) {

cout<<"Ditemukan dengan informasinya\n";

cout<<"NIP\t\t: "<< pCari->nip<<endl;

cout<<"Nama\t\t: "<< pCari->nama<<endl;

cout<<"Alamat\t\t: "<< pCari->alamat<<endl;

cout<<"Gaji\t\t: "<< pCari->gaji<<endl;

cout<<"Golongan\t: "<< pCari->gol<<endl;

}

else {

cout<<"Data tidak ditemukan"<<endl<<endl;

}

break;

case 8:

system("cls");

cout<<"============================================\n";

cout<<"\t\tInsert After\n";

cout<<"============================================\n";

cout<<"NIP yang ingin dicari : "; cin >>kunci;

linearSearch(listPegawai, kunci, ketemu, pCari);

if (ketemu){

cout<<"Data ditemukan"<<endl;

createElmt(p);

cout<<endl<<endl;

insertAfter(listPegawai, pCari, p);

traversal(listPegawai);

}

else {

cout<<"Data tidak ditemukan"<<endl;

}

break;

case 9:

system("cls");

cout<<"============================================\n";

cout<<"\t\tDelete After\n";

cout<<"============================================\n";

cout<<"Masukkan kunci pencarian : "; cin>>kunci;

linearSearch(listPegawai, kunci, ketemu, pCari);

if (ketemu){

cout<<"Ditemukan yang dicari"<<endl;

deleteAfter(listPegawai, pCari, p);

cout<<"Elemen yang dihapus : "<<p->nip<<endl<<endl;

traversal(listPegawai);

}

else {

cout<<"Tidak Ditemukan yang dicari"<<endl;

}

break;

case 10:

system("cls");

cout<<"============================================\n";

cout<<"\t\tCetak Data\n";

cout<<"============================================\n";

traversal(listPegawai);

break;

default:

system("cls");

cout<<"Pilihan tidak tersedia"<<endl;

getch();

break;

};

cout<<"Kembali ke menu utama? (Y/N) : "; cin>>jawab;

} while(jawab == 'y' || jawab == 'Y');

cout<<"Selesai";

}