**LAPORAN PRAKTIKUM**

**SESI III**

**PRAKTIKUM COMP6362 – DATA STRUCTURES**

**KELAS BC20**



Oleh :

2440059495 – Bismo Agung Tri Achmad Bramantyo

**SEMESTER GENAP 2020/2021  
BINA NUSANTARA UNIVERSITY**

**MALANG**

## **Kode Program**

|  |
| --- |
| Source Code |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  struct visitor{    char name[20];  char address[10];  int age;  int number;  struct visitor \*next;    } \*head, \*current, \*index, \*newnode, \*pPre;  int regist = 0;  void insert();  void delete\_node();  void sort\_ascending\_age();  void sort\_descending\_numb();  void sort\_descending\_age();  void sort\_ascending\_numb();  void display();  void insert(){    int A;  char N[20], AD[20];  newnode = (struct visitor\*)malloc(sizeof(struct visitor));    printf("Total Registration in this day : %d \n", regist);  printf("Input Name : ");  scanf("%[^\n]", &N); fflush(stdin);  printf("Input Address: ");  scanf("%[^\n]", &AD);  printf("Input Age : ");  scanf("%d", &A);    newnode->age = A;  strcpy(newnode->name, N);  strcpy(newnode->address, AD);  newnode->number = regist + 1;    if(head == NULL){  head = newnode;  newnode->next = head;  }  else{    current = head;  while(current->next != head){  current = current->next;  }    current->next = newnode;  newnode->next = head;  }  getchar();  regist = regist + 1;  printf("\n-----Success-----\n");  getchar();  system("cls");  }  void delete\_node(int a){    current = head;  while(current->number != a){    if (current->next == head){  printf("Sorry Data Not Found\n");  getchar();  system("cls");  break;  }    pPre = current;  current = current->next;    }  if (current->next == head){  head = NULL;  free(current);  printf("\nSuccess\n");  return;  }    if (current == head){  pPre = head;  while (pPre->next != head)  pPre = pPre->next;  head = current->next;  pPre->next = head;  free(current);  printf("\nSuccess\n");      }  else if (current->next == head && current == head){  pPre->next = head;  free(current);  printf("\nSuccess\n");  }  else{  pPre->next = current->next;  free(current);  printf("\nSuccess\n");  }      }  void sort\_ascending\_age(){      index = NULL;  current = head;  index = current->next;  int temp;  char temp1[100];    if(head == NULL){    getchar();  printf("\nThere is No List Available\n");  getchar();  system("cls");  }  else{  do{    while (index != head){  if (current->age < index->age){    temp = current->age;  current->age = index->age;  index->age = temp;    temp = current->number;  current->number = index->number;  index->number = temp;    strcpy(temp1, current->name);  strcpy(current->name, index->name);  strcpy(index->name, temp1);  strcpy(temp1, current->address);  strcpy(current->address, index->address);  strcpy(index->address, temp1);    }  index = index->next;  }  current = current->next;  }  while(current->next != head);  }  }  void sort\_descending\_age(){    index = NULL;  current = head;index = current->next;  int temp;  char temp1[100];    if(head == NULL){    getchar();  printf("\nThere is No List Available\n");  getchar();  system("cls");  }  else{  do{    while (index != head){  if (current->age > index->age){    temp = current->age;  current->age = index->age;  index->age = temp;    temp = current->number;  current->number = index->number;  index->number = temp;    strcpy(temp1, current->name);  strcpy(current->name, index->name);  strcpy(index->name, temp1);  strcpy(temp1, current->address);  strcpy(current->address, index->address);  strcpy(index->address, temp1);    }  index = index->next;  }  current = current->next;  }  while(current->next != head);  }  }  void sort\_ascending\_numb(){    index = NULL;  current = head;  index = current->next;  int temp;  char temp1[100];    if(head == NULL){    getchar();  printf("\nThere is No List Available\n");  getchar();  system("cls");  }  else{  do{  while (index != head){  if (current->number < index->number){    temp = current->age;  current->age = index->age;  index->age = temp;    temp = current->number;  current->number = index->number;  index->number = temp;    strcpy(temp1, current->name);  strcpy(current->name, index->name);  strcpy(index->name, temp1);  strcpy(temp1, current->address);  strcpy(current->address, index->address);  strcpy(index->address, temp1);    }  index = index->next;  }  current = current->next;  }  while(current->next != head);  }  }  void sort\_descending\_numb(){    index = NULL;  current = head;  int temp;  char temp1[100];    if(head == NULL){    getchar();  printf("\nThere is No List Available\n");  getchar();  system("cls");  }  else{  do{  index = current->next;  while (index != head){  if (current->number > index->number){    temp = current->age;  current->age = index->age;  index->age = temp;    temp = current->number;  current->number = index->number;  index->number = temp;    strcpy(temp1, current->name);  strcpy(current->name, index->name);  strcpy(index->name, temp1);  strcpy(temp1, current->address);  strcpy(current->address, index->address);  strcpy(index->address, temp1);    }  index = index->next;  }  current = current->next;  }  while(current->next != head);  }  }  void display(){    int sort, choice, choice2;    current = head;    if (head == NULL){  getchar();  printf("\n------There is no List Available------\n");  getchar();  system("cls");  }    else  {  printf("Choose The sorting menu : \n");  printf("1. Ascending\n");  printf("2. Descending\n");  printf("\n\nInput choice: ");  scanf("%d", &sort);  printf("-------------------------\n\n");  if (sort == 1){  printf("\nChoose the Ascending by: ");  printf("\n1. Age");  printf("\n2. Registration Number\n");  printf("\nInput Choice : ");  scanf("%d", &choice);  if (choice == 1){  if(current != NULL){  sort\_ascending\_age();  getchar();  system("cls");  do{    printf("-------------------------\n");  printf("Name : %s\n", current->name);  printf("Address: %s\n", current->address);  printf("Age : %d\n", current->age);  printf("Registration Number: %d", current->number);  current = current->next;  printf("\n-------------------------\n");    } while(current->next != head);  getchar();  system("cls");  }    }    else if(choice == 2){    if(current != NULL){  sort\_ascending\_numb();  getchar();  system("cls");    do{  printf("-------------------------\n");  printf("Name : %s\n", current->name);  printf("Address: %s\n", current->address);  printf("Age : %d\n", current->age);  printf("Registration Number: %d", current->number);  current = current->next;  printf("\n-------------------------\n");        } while(current->next != head);  getchar();  system("cls");  }    }  }    else if (sort == 2){  printf("\nChoose the Descending by: ");  printf("\n1. Age");  printf("\n2. Registration Number\n");  printf("\nInput Choice : ");  scanf("%d", &choice);  if (choice == 1){  if(current != NULL){  sort\_descending\_age();  getchar();  system("cls");  do{      printf("-------------------------\n");  printf("Name : %s\n", current->name);  printf("Address: %s\n", current->address);  printf("Age : %d\n", current->age);  printf("Registration Number: %d", current->number);  current = current->next;  printf("\n-------------------------\n");    ;  } while(current->next != head);  getchar();  system("cls");  }    }  else if(choice == 2){  sort\_descending\_numb();  if (current != NULL){    getchar();  system("cls");  do{    printf("-------------------------\n");  printf("Name : %s\n", current->name);  printf("Address: %s\n", current->address);  printf("Age : %d\n", current->age);  printf("Registration Number: %d", current->number);  current = current->next;  printf("\n-------------------------\n");    } while(current->next != head);  getchar();  system("cls");  }  }  }  }  }    void main()  {  int menu;  int numb;  while(menu != 4 && menu < 4){  printf("\t-----Governance Library-----\n");  printf("\t=============================\n");  printf("\n\n");  printf("Visitor Registration Menu:\n\n");  printf("1. Add the Data\n");  printf("2. Delete the Data\n");  printf("3. Show the Data\n");  printf("4. Exit\n");  printf("\nInput Menu[1/2/3/4]: ");  scanf("%d", &menu); fflush(stdin);  printf("\n");  switch (menu)  {  case 1:  insert();  break;    case 2:    if (head != NULL){  printf("--------------------------\n");  printf("Input the Registration Number to Delete from the list: ");  scanf("%d", &numb);  delete\_node(numb);  getchar();  system("cls");  }  else{  printf("\n------There is no List Available------\n");  getchar();  system("cls");  }  break;    case 3:  display();  break;    case 4:  printf("\nThankyou Programm Closed\n");  break;  }  }  } |

## **Deskripsi Program**

* Struct

struct visitor{

char name[20];

char address[10];

int age;

int number;

struct visitor \*next;

} \*head, \*current, \*index, \*newnode, \*pPre;

Disini saya membuat sebuah struct dengan nama visitor yang memiliki 4 variabel dan 6 pointer. Pointer head untuk menunjukan data pertama, pointer current untuk menunjukan data yang diolah, pointer index untuk pointer bantu di sorting, pointer newnode berperan sebagai temp di insert data, dan pointer pPre untuk pointer bantu di fungsi delete.

* Variabel Universal

Variabel Regist berfungsi untuk menampung jumlah registration

* Fungsi

Untuk fungsi pertama saya menggunakan fungsi insert

void insert(){

int A;

char N[20], AD[20];

newnode = (struct visitor\*)malloc(sizeof(struct visitor));

printf("Total Registration in this day : %d \n", regist);

printf("Input Name : ");

scanf("%[^\n]", &N); fflush(stdin);

printf("Input Address: ");

scanf("%[^\n]", &AD);

printf("Input Age : ");

scanf("%d", &A);

newnode->age = A;

strcpy(newnode->name, N);

strcpy(newnode->address, AD);

newnode->number = regist + 1;

if(head == NULL){

head = newnode;

newnode->next = head;

}

else{

current = head;

while(current->next != head){

current = current->next;

}

current->next = newnode;

newnode->next = head;

}

getchar();

regist = regist + 1;

printf("\n-----Success-----\n");

getchar();

system("cls");

}

Di fungsi ini ada variabel int A untuk menampung (temp) usia visitor, kemudian Variabel string AD untuk menampung alamat visitor, dan variabel string N untuk menampung nama visitor.

Lalu saya membooking memori yang akan digunakan dengan

newnode = (struct visitor\*)malloc(sizeof(struct visitor));

user akan diminta untuk menginput nama, alamat dan umur, yang akan ditampung sementara dalam variabel yang telah didefinisikan dalam fungsi ini.

Kemudian data data pada variabel tersebut akan dipindahkan ke dalam node baru dengan

newnode->age = A;

strcpy(newnode->name, N);

strcpy(newnode->address, AD);

newnode->number = regist + 1;

untuk variabel bertipe interger bisa langsung di subtitusi kan, tetapi untuk variabel string harus menggunakan strcpy(lokasi tujuan, lokasi asal). Pada data number akan diisi oleh angka dari penjumlahan regist + 1, yang nantinya akan menjadi nomor registrasi visitor.

* Percabangan kondisi

if(head == NULL){

head = newnode;

newnode->next = head;

}

Ketika pointer head menunjuk node yang kosong / tidak ada data, maka newnode akan menjadi head nya, dan poniter next dari newnode akan menuju ke head(newnode) sehingga menjadi bentuk circular linked list.

else{

current = head;

while(current->next != head){

current = current->next;

}

current->next = newnode;

newnode->next = head;

}

getchar();

regist = regist + 1;

Ketika pada head ada data maka pointer current akan menujuk pada node yang ditunjuk pointer head, lalu selama pointer data terakhir tidak mengarah ke head maka pointer next akan dicabut dari head lalu diarahkan ke newnode, lalu pointer newnode akan mengarah ke head, sehingga newnode menjadi data terakhir. Kemudian variabel regist akan ditambah nilainya 1.

Untuk fungsi kedua saya menggunakan fungsi delete

void delete\_node(int a){

current = head;

while(current->number != a){

if (current->next == head){

printf("Sorry Data Not Found\n");

getchar();

system("cls");

break;

}

pPre = current;

current = current->next;

}

if (current->next == head){

head = NULL;

free(current);

printf("\nSuccess\n");

return;

}

if (current == head){

pPre = head;

while (pPre->next != head)

pPre = pPre->next;

head = current->next;

pPre->next = head;

free(current);

printf("\nSuccess\n");

}

else if (current->next == head && current == head){

pPre->next = head;

free(current);

printf("\nSuccess\n");

}

else{

pPre->next = current->next;

free(current);

printf("\nSuccess\n");

}

}

Pointer Current akan menunjuk kepada node yang ditunjuk oleh pointer head, kemudian pada fungsi ini memiliki 1 passing parameter yaitu int a, int a disini berfungsi sebagai key / keyword yang akan dihapus. Selama data number pada node current(head) tidak sama dengan a (key) maka program akan mencari key tersebut dengan cara, ponter next akan ditujukan pada node setelah nya (current->next) kemudian node sebelum nya akan diisi oleh pointer pPre, apabila masih tidak ditemukan program akan menampilkan pesan bahwa node tidak ditemukan dan kembali ke menu awal. Selain itu, maka program akan mengecek apakah node hanya 1 atau tidak dengan cara

if (current->next == head){

head = NULL;

free(current);

printf("\nSuccess\n");

return;

}

Apabila node current (node pertama dalm kasus ini) pointer next nya menuju ke head (artinya kembali pada dirinya sendiri) maka node tersebut akan langsung di hapus. Kemudian akan diberi return agar eksekusi if dapat dihentikan

Selain itu , program akan mengecek apakah data yang dicari berada di node pertama dengan cara

if (current == head){

pPre = head;

while (pPre->next != head){

pPre = pPre->next;

head = current->next;

pPre->next = head;

free(current);

printf("\nSuccess\n");

}

}

Ketika pointer current menunjuk ke node yang sama dengan head, maka pointer pPre akan menunjuk head juga, selama pPre pointer nextnya tidak menunjuk ke head maka pointer pPre akan menunjuk ke node selanjutnya, kemudian pointer head akan menunjuk ke node yang ditunjuk oleh pointer next dari node current, kemudian pPre pointer next(node terakhir) akan menunjuk ke head, lalu current dihapus.

Selain itu program juga akan mengecek apakah data yang dicari berada di node terakhir

else if (current->next == head && current == head){

pPre->next = head;

free(current);

printf("\nSuccess\n");

}

Ketika pointer next dari node current mnuju ke head dan pointer current menunjuk node yang sama dengan head maka, pPre-> next (node sebelumnya) akan menunjuk ke head, lalu current dihapus.

Selain itu,

else{

pPre->next = current->next;

free(current);

printf("\nSuccess\n");

}

Jika data tidak berada di first node maupun last node, pointer next dari pPre akan menunjuk ke node yang ditunjuk oleh pointer next nya current.

Kemudian saya menggunakan fungsi sort,

void sort\_ascending\_age(){

index = NULL;

current = head;

index = current->next;

int temp;

char temp1[100];

if(head == NULL){

getchar();

printf("\nThere is No List Available\n");

getchar();

system("cls");

}

else{

do{

while (index != head){

if (current->age < index->age){

temp = current->age;

current->age = index->age;

index->age = temp;

temp = current->number;

current->number = index->number;

index->number = temp;

strcpy(temp1, current->name);

strcpy(current->name, index->name);

strcpy(index->name, temp1);

strcpy(temp1, current->address);

strcpy(current->address, index->address);

strcpy(index->address, temp1);

}

index = index->next;

}

current = current->next;

}

while(current->next != head);

}

}

Pointer index saya NULL kan terlebih dahulu, kemudian pointer current akan menunjuk ke node yang ditunjuk oleh pointer head, sedangkan index akan menunjuk ke node yang ditunjuk pointer next dari current(head), kemudian saya juga mendefinisikan 2 variabel tambahan untuk menampung data sementara (temp), yaitu int temp untuk menampung data age / number yang akan dipindahkan dan char temp untuk menampung data nama / alamat.

Selama node yang ditunjuk oleh pnter head tidak bernilai NULL / kosongprogram akan memandingkan nilai dan mengurutkan data, tatapi apabila node kosong, maka program akan menampilkan pesan bahwa tidak ada node yang bisa diurutkan.

Kemudian program akan membandingkan data (tergantung fungsi bisa age, bisa number) selama index tidak menunjuk ke head(karena jika index menunjuk ke head, maka data hanya ada 1) program akan memandingkan nilai current dan index, apabila memenuhi persayaratan maka data akan di swap. Disini saya menggunakan perulangan do while, program akan melakukan swap selama current->next(pointer next pada data terakhir) tidak mengarah pada head.

Untuk fungsi terakhir saya menggunakan fungsi display

void display(){

int sort, choice, choice2;

current = head;

if (head == NULL){

getchar();

printf("\n------There is no List Available------\n");

getchar();

system("cls");

}

else

{

printf("Choose The sorting menu : \n");

printf("1. Ascending\n");

printf("2. Descending\n");

printf("\n\nInput choice: ");

scanf("%d", &sort);

printf("-------------------------\n\n");

if (sort == 1){

printf("\nChoose the Ascending by: ");

printf("\n1. Age");

printf("\n2. Registration Number\n");

printf("\nInput Choice : ");

scanf("%d", &choice);

if (choice == 1){

if(current != NULL){

sort\_ascending\_age();

getchar();

system("cls");

do{

printf("-------------------------\n");

printf("Name : %s\n", current->name);

printf("Address: %s\n", current->address);

printf("Age : %d\n", current->age);

printf("Registration Number: %d", current->number);

current = current->next;

printf("\n-------------------------\n");

} while(current->next != head);

getchar();

system("cls");

}

}

else if(choice == 2){

if(current != NULL){

sort\_ascending\_numb();

getchar();

system("cls");

do{

printf("-------------------------\n");

printf("Name : %s\n", current->name);

printf("Address: %s\n", current->address);

printf("Age : %d\n", current->age);

printf("Registration Number: %d", current->number);

current = current->next;

printf("\n-------------------------\n");

} while(current->next != head);

getchar();

system("cls");

}

}

}

else if (sort == 2){

printf("\nChoose the Descending by: ");

printf("\n1. Age");

printf("\n2. Registration Number\n");

printf("\nInput Choice : ");

scanf("%d", &choice);

if (choice == 1){

if(current != NULL){

sort\_descending\_age();

getchar();

system("cls");

do{

printf("-------------------------\n");

printf("Name : %s\n", current->name);

printf("Address: %s\n", current->address);

printf("Age : %d\n", current->age);

printf("Registration Number: %d", current->number);

current = current->next;

printf("\n-------------------------\n");

;

} while(current->next != head);

getchar();

system("cls");

}

}

else if(choice == 2){

sort\_descending\_numb();

if (current != NULL){

getchar();

system("cls");

do{

printf("-------------------------\n");

printf("Name : %s\n", current->name);

printf("Address: %s\n", current->address);

printf("Age : %d\n", current->age);

printf("Registration Number: %d", current->number);

current = current->next;

printf("\n-------------------------\n");

} while(current->next != head);

getchar();

system("cls");

}

}

}

}

}

Ada 3 variabel bertipe integer, sort berfungsi untuk pemilihan sorting secara ascending atau descending sementara choice berfungsi untuk pemilihan key pada sort ascending, sedangkan choice2 berfungsi untuk pemilihan key pada sort descending, selama node belum habis (pointer next node terakhir belum menunjuk ke head)

Ketika sort bernilai 1 dan choice bernilai 1, maka program akan memanggil fungsi sort\_ascending\_age , sementara ketika choice bernilai 2 maka program akan memanggil fungsi sort\_ascending\_numb.

Ketika sort bernilai 1 dan choice 2 bernilai satu maka program akan memanggil fungsi sort\_descending\_age, namun ketika choice2 bernilai 2 maka program akan memanggil fungsi sort\_descending\_numb, setelah itu program akan mencetak hasil node yang telah diurutkan.

Void main

void main()

{

int menu;

int numb;

while(menu != 4 && menu < 4){

printf("\t-----Governance Library-----\n");

printf("\t=============================\n");

printf("\n\n");

printf("Visitor Registration Menu:\n\n");

printf("1. Add the Data\n");

printf("2. Delete the Data\n");

printf("3. Show the Data\n");

printf("4. Exit\n");

printf("\nInput Menu[1/2/3/4]: ");

scanf("%d", &menu); fflush(stdin);

printf("\n");

switch (menu)

{

case 1:

insert();

break;

case 2:

if (head != NULL){

printf("--------------------------\n");

printf("Input the Registration Number to Delete from the list: ");

scanf("%d", &numb);

delete\_node(numb);

getchar();

system("cls");

}

else{

printf("\n------There is no List Available------\n");

getchar();

system("cls");

}

break;

case 3:

display();

break;

case 4:

printf("\nThankyou Programm Closed\n");

break;

}

}

}

Untuk void main saya mendefinisikan 2 variabel yaitu menu dan numb, menu berfungsi untuk pemilihan menu dan switch case, untuk numb berfungsi menampung variabel angka yang nantinya akan dijadikan passing parameter delete.

Pertama user diminta untuk menginput menu, saat user memilih 1 maka program akan memanggil fungsi insert, ketika user memilih 2, maka selama head tidak NULL, user diminta untuk menginput key / nomor yang akan dihapus lalu program akan memanggil fungsi delete node dengan passing parameter numb yang telah diinputkan oleh user, ketika user memilih nomor 3 , maka program akan memanggil fungsi display, dan ketika user memilih 4 maka program akan menampilkan ucpan terimakasih dan program akan tertutup.

## **Bukti Presentasi**