Laporan Resmi
Praktikum Algoritma dan Struktur Data
Single Linked List Delete (Data Int)

Dr. Tita Karlita S.Kom, M.Kom

Nama : Marits Ikmal Yasin

Kelas : 1D4 IT B

NRP : 3121600047

1. Delete Awal
   Kode :

```
#include<stdio.h>
#include<stdlib.h>

typedef struct simpul node;
struct simpul{
   int data;
   node *next;
};

node *head = NULL, *p;


void alokasi();
void akhir();
void tampil();
void menghapus_awal();
void bebaskan(node *);

int main(){
   char jawab;

   puts("Single Linked List - Insert Akhir - Delete Awal");
   do{
      fflush(stdin);
      alokasi();
      akhir();
      fflush(stdin);
      printf("Lagi (y/t)? ");
      jawab = getchar();
   }while(jawab == 'y' || jawab == 'Y');
   tampil();
   while(head != NULL){
      puts("\nMenghapus data posisi awal ...");
      menghapus_awal();
      tampil();
   }
   return 0;
}

void bebaskan(node *del){
   free(del);
   del = NULL;
}

void menghapus_awal(){
   node *hapus;

   hapus = head;
   head = hapus->next;
   bebaskan(hapus);
}

void alokasi(){
   int bilangan;

   printf("Nilai yang akan disimpan : ");
   scanf("%d",&bilangan);
   p = (node *)malloc(sizeof(node));
   if(p == NULL){
      puts("Gagal memesam alamat");
      exit(0);
```

```
    }
    else{
        p->data = bilangan;
        p->next = NULL;
    }
}

void akhir(){
    node *tail;

    if(head == NULL)
        head = p;
    else{
        tail = head;
        while(tail->next != NULL)
            tail = tail->next;
        tail->next = p;
        tail = tail->next;
    }
}

void tampil(){
    node *baca;

    baca = head;
    puts("");
    puts("Isi dari SLL");
    while(baca != NULL){
        printf("%d\n", baca->data);
        baca = baca->next;
    }
    if(head == NULL)
        puts("Kosong");
}
```

Output :

```
Single Linked List - Insert Akhir - Delete Awal
Nilai yang akan disimpan : 1
Lagi (y/t)? y
Nilai yang akan disimpan : 5
Lagi (y/t)? y
Nilai yang akan disimpan : 10
Lagi (y/t)? t

Isi dari SLL
1
5
10

Menghapus data posisi awal ...

Isi dari SLL
5
10

Menghapus data posisi awal ...

Isi dari SLL
10

Menghapus data posisi awal ...

Isi dari SLL
Kosong
```

2. Delete Akhir

Kode :

```
#include<stdio.h>
#include<stdlib.h>

typedef struct simpul node;
struct simpul{
    int data;
    node *next;
};

node *head = NULL, *p;

void alokasi();
void akhir();
void tampil();
void menghapus_akhir();
void bebaskan(node *);

int main(){
    char jawab;

    puts("Single Linked List - Insert Akhir - Delete Akhir");
    do{
        fflush(stdin);
        alokasi();
        akhir();
        fflush(stdin);
        printf("Lagi (y/t)? ");
        jawab = getchar();
    }while(jawab == 'y' || jawab == 'Y');
    tampil();
    while(head != NULL){
        puts("\nMenghapus data posisi akhir ...");
        menghapus_akhir();
        tampil();
    }
    return 0;
}
```

```
void bebaskan(node *del){
    free(del);
    del = NULL;
}

void menghapus_akhir(){
    node *hapus, *phapus;

    hapus = head;
    while(hapus->next != NULL){
        phapus = hapus;
        hapus = hapus->next;
    }
    if(hapus == head)
        head = NULL;
    else
        phapus->next = NULL;
    bebaskan(hapus);
}

void alokasi(){
    int bilangan;

    printf("Nilai yang akan disimpan : ");
    scanf("%d",&bilangan);
    p = (node *)malloc(sizeof(node));
    if(p == NULL){
        puts("Gagal memesam alamat");
        exit(0);
    }
    else{
        p->data = bilangan;
        p->next = NULL;
    }
}

void akhir(){
    node *tail;

    if(head == NULL)
```

```
        head = p;
    else{
        tail = head;
        while(tail->next != NULL)
            tail = tail->next;
        tail->next = p;
        tail = tail->next;
    }
}

void tampil(){
    node *baca;

    baca = head;
    puts("");
    puts("Isi dari SLL");
    while(baca != NULL){
        printf("%d\n", baca->data);
        baca = baca->next;
    }
    if(head == NULL)
        puts("Kosong");
}
```

Output :

```
Single Linked List - Insert Akhir - Delete Akhir
Nilai yang akan disimpan : 1
Lagi (y/t)? y
Nilai yang akan disimpan : 8
Lagi (y/t)? y
Nilai yang akan disimpan : 30
Lagi (y/t)? t

Isi dari SLL
1
8
30

Menghapus data posisi akhir ...

Isi dari SLL
1
8

Menghapus data posisi akhir ...

Isi dari SLL
1

Menghapus data posisi akhir ...

Isi dari SLL
Kosong
```

3. Delete Tertentu

Kode :

```
#include<stdio.h>
#include<stdlib.h>

typedef struct simpul node;
struct simpul{
    int data;
    node *next;
};

node *head = NULL, *p;

void alokasi();
void akhir();
void tampil();
void menghapus_tertentu();
void bebaskan(node *);

int main(){
    char jawab;

    puts("Single Linked List - Insert Akhir - Delete Node Tertentu");
    do{
        fflush(stdin);
        alokasi();
        akhir();
        fflush(stdin);
        printf("Lagi (y/t)? ");
        jawab = getchar();
    }while(jawab == 'y' || jawab == 'Y');
    tampil();
    while(head != NULL){
        menghapus_tertentu();
        tampil();
    }
    return 0;
}

void bebaskan(node *del){
    free(del);
    del = NULL;
}

void menghapus_tertentu(){
    int key;
    node *hapus, *phapus;

    printf("Masukkan Node berapa yang ingin dihapus : ");
    scanf("%d", &key);
    hapus = head;
    if(hapus->data == key){
        head = hapus->next;
    }
    else{
        while(hapus->data != key){
            if(hapus->next != NULL){
                phapus = hapus;
                hapus = hapus->next;
            }
            else{
                printf("%d tidak ada di dalam SLL\n", key);
                exit(0);
```

```
            }
        }
        phapus->next = hapus->next;
    }
    bebaskan(hapus);
}

void alokasi(){
    int bilangan;

    printf("Nilai yang akan disimpan : ");
    scanf("%d",&bilangan);
    p = (node *)malloc(sizeof(node));
    if(p == NULL){
        puts("Gagal memesam alamat");
        exit(0);
    }
    else{
        p->data = bilangan;
        p->next = NULL;
    }
}

void akhir(){
    node *tail;

    if(head == NULL)
        head = p;
    else{
        tail = head;
        while(tail->next != NULL)
            tail = tail->next;
        tail->next = p;
        tail = tail->next;
    }
}

void tampil(){
    node *baca;

    baca = head;
    puts("");
    puts("Isi dari SLL");
    while(baca != NULL){
        printf("%d\n", baca->data);
        baca = baca->next;
    }
    if(head == NULL)
        puts("Kosong");
}
```

Output :

```
Single Linked List - Insert Akhir - Delete Node Tertentu
Nilai yang akan disimpan : 1
Lagi (y/t)? y
Nilai yang akan disimpan : 30
Lagi (y/t)? y
Nilai yang akan disimpan : 5
Lagi (y/t)? t

Isi dari SLL
1
30
5
Masukkan Node berapa yang ingin dihapus : 1

Isi dari SLL
30
5
Masukkan Node berapa yang ingin dihapus : 5

Isi dari SLL
30
Masukkan Node berapa yang ingin dihapus : 2
2 tidak ada di dalam SLL
```

4. Menu

Kode :

```
#include<stdio.h>
#include<stdlib.h>

typedef struct simpul node;
struct simpul{
   int data;
   node *next;
};

node *head = NULL, *p;

void menu();
void masukkan();
void menghapus();
void alokasi();
void awal();
void akhir();
void setelah();
void sebelum();
void menghapus_awal();
void menghapus_akhir();
void menghapus_tertentu();
void tampil();
void bebaskan(node *);

int main(){
   menu();

   return 0;

}

void menu(){
   int pilihan;

   do{
      puts("\nMenu SLL");
      puts("1. Insert");
      puts("2. Delete");
```

```
        puts("3. Keluar");
        printf("Masukkan Pilihan Anda : ");
        scanf("%d", &pilihan);
        if(head == NULL && pilihan == 2)
           puts("Tidak Bisa Melakukan Delete karena Data Kosong");
        else{
            switch(pilihan){
              case 1 :
                 masukkan();
                 break;
              case 2 :
                 menghapus();
                 break;
              case 3 :
                 exit(0);
                 break;
              default :
                 puts("Pilih yang benar");
           }
        }
        tampil();
    }while(pilihan != 3);
}

void masukkan(){
    int pilih;

    puts("Menu Insert");
    puts("1. Awal");
    puts("2. Akhir");
    puts("3. Sebelum");
    puts("4. Sesudah");
insert:
    printf("Masukkan Pilihan Anda : ");
    scanf("%d",&pilih);
    if(head == NULL && pilih == 3 || head == NULL && pilih == 4){
       puts("Data masih kosong tidak bisa melakukan insert sebelum atau sesudah");
       goto insert;
    }
    else{
        switch(pilih){
           case 1 :
              alokasi();
              awal();
              break;
           case 2 :
              alokasi();
              akhir();
              break;
           case 3 :
              alokasi();
              sebelum();
              break;
           case 4 :
              alokasi();
              setelah();
              break;
           default :
              puts("Pilih yang benar");
              goto insert;
        }
    }
}
```

```
void menghapus(){
    int jawab;

    puts("Menu Delete");
    puts("1. Awal");
    puts("2. Akhir");
    puts("3. Tertentu");
hapus:
    printf("Masukkan Pilihan Anda : ");
    scanf("%d",&jawab);
    switch(jawab){
        case 1 :
            menghapus_awal();
            break;
        case 2 :
            menghapus_akhir();
            break;
        case 3 :
            menghapus_tertentu();
            break;
        default :
            puts("Pilih yang benar");
            goto hapus;
    }

}

void menghapus_tertentu(){
    int key;
    node *hapus, *phapus;

input:
    printf("Masukkan Node berapa yang ingin dihapus : ");
    scanf("%d", &key);
    hapus = head;
    if(hapus->data == key){
        head = hapus->next;
    }
    else{
        while(hapus->data != key){
            if(hapus->next != NULL){
                phapus = hapus;
                hapus = hapus->next;
            }
            else{
                printf("%d tidak ada di dalam SLL\n", key);
                goto input;
            }
        }
        phapus->next = hapus->next;
    }
    bebaskan(hapus);
}

void menghapus_akhir(){
    node *hapus, *phapus;

    hapus = head;
    while(hapus->next != NULL){
        phapus = hapus;
        hapus = hapus->next;
    }
```

```
    if(hapus == head)
        head = NULL;
    else
        phapus->next = NULL;
    bebaskan(hapus);
}

void menghapus_awal(){
    node *hapus;

    hapus = head;
    head = hapus->next;
    bebaskan(hapus);
}

void bebaskan(node *del){
    free(del);
    del = NULL;
}

void alokasi(){
    int bilangan;

    printf("Nilai yang akan disimpan : ");
    scanf("%d",&bilangan);
    p = (node *)malloc(sizeof(node));
    if(p == NULL){
        puts("Gagal memesam alamat");
        exit(0);
    }
    else{
        p->data = bilangan;
        p->next = NULL;
    }
}

void awal(){
    if(head != NULL)
        p->next = head;
    head = p;
}

void akhir(){
    node *tail;

    if(head == NULL)
        head = p;
    else{
        tail = head;
        while(tail->next != NULL)
            tail = tail->next;
        tail->next = p;
        tail = tail->next;
    }
}

void setelah(){
    int key;
    node *after;

sesudah:
    printf("Disisipkan setelah data berapa ? ");
    scanf("%d",&key);
```

```
    after = head;
    while(after->data != key){
        if(after->next == NULL){
            printf("%d tidak ditemukan di dalam SLL\n",key);
            goto sesudah;
        }
        else{
            after = after->next;
        }
    }
        p->next = after->next;
        after->next = p;
}

void sebelum(){
    int key;
    node *bef, *pbef;

before:
    printf("Disisipkan sebelum data berapa ? ");
    scanf("%d",&key);
    bef = head;
    while(bef->data != key){
        if(bef->next == NULL){
            printf("%d tidak ditemukan di dalam SLL\n",key);
            goto before;
        }
        else{
            pbef = bef;
            bef = bef->next;
        }
    }
    if(bef == head){
        p->next = bef;
        head = p;
    }
    else{
        p->next = bef;
        pbef->next = p;
    }
}

void tampil(){
    node *baca;

    baca = head;
    puts("");
    puts("Data yang ada dalam SLL");
    while(baca != NULL){
        printf("%d\n",baca->data);
        baca = baca->next;
    }
    if(head == NULL)
        puts("Kosong\n");
}
```

Output :

```
Menu SLL
1. Insert
2. Delete
3. Keluar
Masukkan Pilihan Anda : 1
Menu Insert
1. Awal
2. Akhir
3. Sebelum
4. Sesudah
Masukkan Pilihan Anda : 1
Nilai yang akan disimpan : 30

Data yang ada dalam SLL
30

Menu SLL
1. Insert
2. Delete
3. Keluar
Masukkan Pilihan Anda : 1
Menu Insert
1. Awal
2. Akhir
3. Sebelum
4. Sesudah
Masukkan Pilihan Anda : 2
Nilai yang akan disimpan : 24
```

```
Data yang ada dalam SLL
30
24

Menu SLL
1. Insert
2. Delete
3. Keluar
Masukkan Pilihan Anda : 2
Menu Delete
1. Awal
2. Akhir
3. Tertentu
Masukkan Pilihan Anda : 1

Data yang ada dalam SLL
24

Menu SLL
1. Insert
2. Delete
3. Keluar
Masukkan Pilihan Anda : 3
```

Analisa :

Algoritma umum pada operasi delete dalam single linked list yaitu ada 3. Pertama, pilih node yang ingin kita delete. Kedua, selamatkan node agar tidak terputus. Ketiga, bebaskan node yang ingin kita hapus dan jangan lupa me-NULL kan pointer yang digunakan untuk membebaskan node.

Dalam melakukan delete awal yang pertama kita lakukan yaitu kita harus menyiapkan sebuah pointer hapus dan kita assign head ke pointer hapus. Setelah itu kita selamatkan head dengan cara diassignkan oleh head->next. Setelah itu otomatis node masih tetap terhubung. Setelah node head dihubungkan maka langkah selanjutnya melakukan pembebasan pada node yang ditunjuk oleh hapus. Setelah alamat yang ditunjuk hapus dibebaskan jangan lupa untuk me-NULLkan hapus.

Dalam melakukan delete akhir kita memerlukan 2 buah pointer bantuan yaitu phapus dan hapus. Cara menggunakannya sama seperti algoritma di saat kita melakukan insert before. Apabila data hanya terdapat hanya satu node, maka langsung dilakukan pembebasan layaknya delete awal. Apabila data node lebih dari 1 maka hapus akan mencari data terakhir setelah itu phapus akan berada di belakang pointer hapus. Setelah node terakhir ditemukan maka kita harus menyambungkan node agar tidak terputus dengan cara, phapus->next disambungkan dengan hapus->next. Setelah aman maka dapat dilakukan pembebasan. Sama seperti sebelumnya apabila telah melakukan pembebasan maka harus me-NULLkan pointer hapus.

Dalam melakukan delete node tertentu kita memerlukan sebuah key, untuk mencari data mana yang ingin dicari. Apabila key yang dicari ada di awal data maka seperti sebelumya akan dilakukan delete awal. Jika data yang dicari belum ketemu maka pointer hapus akan mencari sampai menemukannya. Di belakang pointer hapus ada pointer phapus yang berfungsi untuk memegang node di belakang pointer hapus. Apabila data yang dicari ketemu maka lakukan penyelamatan dengan cara melakukan assign antara phapus->next dengan hapus->next. Setelah dilakukan penyelamatan maka bisa dilakukan pembebasan. Setelah dilakukan pembebasan maka jangan lupa melakukan NULL pada pointer hapus. Akan tetapi apabila data yang dicari tidak ada di dalam SLL maka program akan melakukan perintah keluar.

Setelah itu terdapat sebuah menu untuk SLL, kita dapat memilih kita mau melakukan apa. Contohnya seperti kita dapat melakukan insert dan delete. Untuk memberhentikan program dengan cara memilih opsi 3 yaitu dengan cara keluar.