STRUKTUR DATA

LIST OF LIST

Rosa Ariani Sukamto

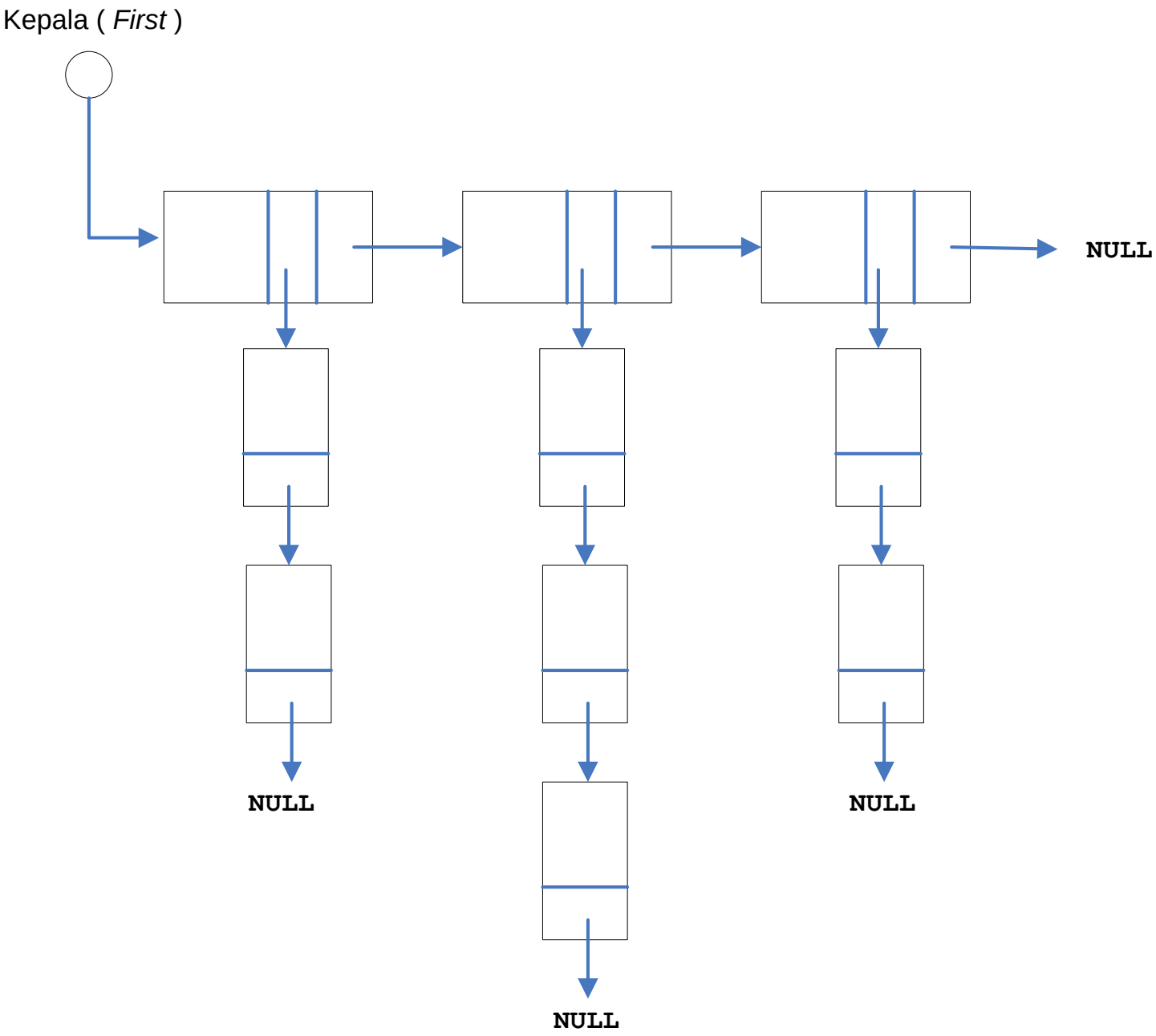
ROSA ARIANI SUKAMTO

Blog: http://hariiniadalahhadiah.wordpress.com

Facebook: https://www.facebook.com/rosa.ariani.sukamto

Email: rosa_if_itb_01@yahoo.com

LIST OF LIST



DEKLARASI ELEMEN

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>

typedef struct{
  char nim[10];
  char nama[50];
}mahasiswa;

typedef struct{
  char kode[10];
  char nilai[2];
}matKul
```

```
typedef struct eklm
   *alamatekolom;
typedef struct eklm{
  matKul elmt;
  alamatekolom next;
}eKolom;

typedef struct ebr *alamatebaris;
typedef struct ebr{
  mahasiswa elmt;
  eKolom *col;
  alamatebaris next;
}eBaris;

typedef struct{
  eBaris *first;
}list;
```

CREATE LIST

```
void createList(list *L){

   (*L).first = NULL;

}
```

COUNT ELEMENT BARIS

```
int countElementB(list L){
  int hasil = 0;

  if(L.first !=NULL){
    /*list tidak kosong*/

    eBaris *elmt;

    /*inisialisasi*/
    elmt = L.first;
```

```
while(elmt != NULL){
      /*proses*/
      hasil = hasil + 1;

      /*iterasi*/
      elmt = elmt->next;
    }

  }

  return hasil;

}
```

COUNT ELEMENT KOLOM

```
int countElementK(eBaris L){

  int hasil = 0;

  if(L.col !=NULL){
    /*list tidak kosong*/

    eKolom *elmt;

    /*inisialisasi*/
    elmt = L.col;

    while(elmt != NULL){
      /*proses*/
      hasil = hasil + 1;

      /*iterasi*/
      elmt = elmt->next;
    }

  }

  return hasil;

}
```

ADD FIRST BARIS

```
void addFirstB(char nim[], char nama[], list *L){


  eBaris *elmt;
  elmt = (eBaris *) malloc (sizeof (eBaris));
  strcpy(elmt->elmt.nim, nim);
  strcpy(elmt->elmt.nama, nama);
  elmt->col = NULL;
  if((*L).first == NULL){
    elmt->next = NULL;
  }else{
    elmt->next = (*L).first;
  }
  (*L).first = elmt;
   elmt = NULL;
}
```

ADD FIRST KOLOM

```
void addFirstK(char kode[], char nilai[], eBaris *L){


  eKolom *elmt;
  elmt = (eKolom *) malloc (sizeof (eKolom));
  strcpy(elmt->elmt.kode, kode);
  strcpy(elmt->elmt.nilai, nilai);
  if((*L).col == NULL){
    elmt->next = NULL;
  }else{
    elmt->next = (*L).col;
  }
  (*L).col = elmt;
  elmt = NULL;


}
```

ADD AFTER BARIS

```
void addAfterB(eBaris *prev, char nim[], char nama[]){


  eBaris *elmt;
  elmt = (eBaris *) malloc (sizeof (eBaris));
  strcpy(elmt->elmt.nim, nim);
  strcpy(elmt->elmt.nama, nama);
  elmt->col = NULL;
  if(prev->next == NULL){
    elmt->next = NULL;
  }else{
    elmt->next = prev->next;
  }
  prev->next = elmt;
  elmt = NULL;


}
```

ADD AFTER KOLOM

```
void addAfterK(eKolom *prev, char kode[], char nilai[]){


  eKolom *elmt;
  elmt = (eKolom *) malloc (sizeof (eKolom));
  strcpy(elmt->elmt.kode, kode);
  strcpy(elmt->elmt.nilai, nilai);
  if(prev->next == NULL){
     elmt->next = NULL;
  }else{
     elmt->next = prev->next;
  }
  prev->next = elmt;
  elmt = NULL;
}
```

ADD LAST BARIS

```
void addLastB(char nim[], char
   nama[], list *L){


  if((*L).first == NULL){
    /*jika list adalah list
   kosong*/
    addFirstB(nim, nama, L);
  }
  else{
    /*jika list tidak kosong*/
    eBaris *elmt;
    elmt = (eBaris *) malloc
   (sizeof (eBaris));
    strcpy(elmt->elmt.nim, nim);
    strcpy(elmt->elmt.nama,
   nama);
    elmt->next = NULL;
    elmt->col = NULL;
```

```
    /*mencari elemen terakhir
   list*/
    eBaris *last = (*L).first;
    while(last->next != NULL){
      /*iterasi*/
      last = last->next;
    }
    last->next = elmt;
    elmt = NULL;
  }

}
```

ADD LAST KOLOM

```
void addLastK(char kode[], char
   nilai[], eBaris *L){


  if((*L).col == NULL){
    /*jika list adalah list
   kosong*/
    addFirstK(kode, nilai, L);
  }
  else{
    /*jika list tidak kosong*/
    eKolom *elmt;
    elmt = (eKolom *) malloc
   (sizeof (eKolom));
    strcpy(elmt->elmt.kode,
   kode);
    strcpy(elmt->elmt.nilai,
   nilai);
    elmt->next = NULL;
```

```
/*mencari elemen terakhir list*/


    eKolom *last = (*L).col;
    while(last->next != NULL){
      /*iterasi*/
      last = last->next;
    }


    last->next = elmt;
    elmt = NULL;
  }

}
```

DEL FIRST BARIS

```
void delFirstB(list *L){


  if((*L).first != NULL){
    /*jika list bukan list kosong*/
    eBaris *elmt = (*L).first;
    if(countElementB(*L) == 1){
        (*L).first = NULL;
    }else{
        (*L).first = (*L).first->next;
       elmt->next = NULL;
    }
    free(elmt);
  }


}
```

DEL FIRST KOLOM

```
void delFirstK(eBaris *L){


  if((*L).col != NULL){
    /*jika list bukan list kosong*/


    eKolom *elmt = (*L).col;
    if(countElementK(*L) == 1){
        (*L).col = NULL;
    }else{
        (*L).col = (*L).col->next;
         elmt->next = NULL;
    }
    free(elmt);
  }


}
```

DEL AFTER BARIS

```
void delAfterB(eBaris *prev){


  eBaris *elmt = prev->next;
  if(elmt->next == NULL){
     prev->next = NULL;
  }else{
      prev->next = elmt->next;
      elmt->next = NULL;
  }
  free(elmt);


}
```

DEL AFTER KOLOM

```
void delAfterK(eKolom *prev){



  eKolom *elmt = prev->next;
  if(elmt->next == NULL){
     prev->next = NULL;
  }else{
     prev->next = elmt->next;
     elmt->next = NULL;
  }
  free(elmt);
}
```

DEL LAST BARIS

```
void delLastB(list *L){


  if((*L).first != NULL){
    /*jika list tidak kosong*/
    if(countElementB(*L) == 1){
      /*list terdiri dari satu
   elemen*/
      delFirstB(L);


    }
    else{
```

```
/*mencari elemen terakhir list*/
      eBaris *last = (*L).first;
      eBaris *before_last;

      while(last->next != NULL){

        /*iterasi*/
        before_last = last;
        last = last->next;


      }

      before_last->next = NULL;
      free(last);


    }
  }
}
```

DEL LAST KOLOM

```
void delLastK(eBaris *L){

  if((*L).col != NULL){
    /*jika list tidak kosong*/
    if(countElementK(*L) == 1){
      /*list terdiri dari satu
   elemen*/
      delFirstK(L);

    }
    else{
      /*mencari elemen terakhir
    list*/
      eKolom *last = (*L).col;
      eKolom *before_last;

      while(last->next != NULL){
        /*iterasi*/
        before_last = last;
        last = last->next;

      }

      before_last->next = NULL;
      free(last);

    }
  }
}
```

PRINT ELEMENT

```
void printElement(list L){
  if(L.first != NULL){
    /*jika list tidak kosong*/
    /*inisialisasi*/
    eBaris *elmt = L.first;
    int i = 1;

    while(elmt != NULL){
      /*proses*/
      printf("elemen ke : %d\n", i);
      printf("nim : %s\n",
       elmt->elmt.nim);
      printf("nama : %s\n",
       elmt->elmt.nama);
 eKolom *eCol = elmt->col;
      while(eCol != NULL){

        printf("kode kuliah : %s\n",
         eCol->elmt.kode);
          printf("nilai : %s\n",
          eCol->elmt.nilai);
         eCol = eCol->next;
       }
       printf("------------\n");

       /*iterasi*/
     elmt = elmt->next;
       i = i + 1;
     }
   }
   else{
    /*proses jika list kosong*/
     printf("list kosong\n");
   }
}
```

DEL ALL BARIS

```
void delAllB(list *L){

 if(countElementB(*L) != 0){

  int i;

  for(i=countElementB(*L);i>=1;i--){
  /*proses menghapus elemen list*/
    delLastB(L);
  }

 }

}
```

DEL ALL KOLOM

```
void delAllK(eBaris *L){

 if(countElementK(*L) != 0){

  int i;

  for(i=countElementK(*L);i>=1;i--){
  /*proses menghapus elemen list*/
    delLastK(L);
  }

 }

}
```

MAIN

```
int main(){
  list L;
  createList(&L);
  printElement(L);

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

  addFirstB("1", "Orang_1", &L);
  addFirstK("IF40K1", "A", L.first);
  addAfterK(L.first->col, "IF40Z1",
   "A");
  addLastK("IF40Z2", "A", L.first);

  addAfterB(L.first, "2", "Orang_2");
  addFirstK("TI5141", "A",
     L.first->next);
     addLastK("IF5021", "A",
     L.first->next);

  addLastB("3", "Orang_3", &L);
  addFirstK("IF5321", "A",
     L.first->next->next);

  printElement(L);

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

  delAllB(&L);
  printElement(L);

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

  return 0;

}
```

DAFTAR PUSTAKA

S, Rosa A. dan M. Shalahuddin. 2010. Modul Pembelajaran: Struktur Data. Modula: Bandung.