STRUKTUR DATA

GRAF

Rosa Ariani Sukamto

ROSA ARIANI SUKAMTO

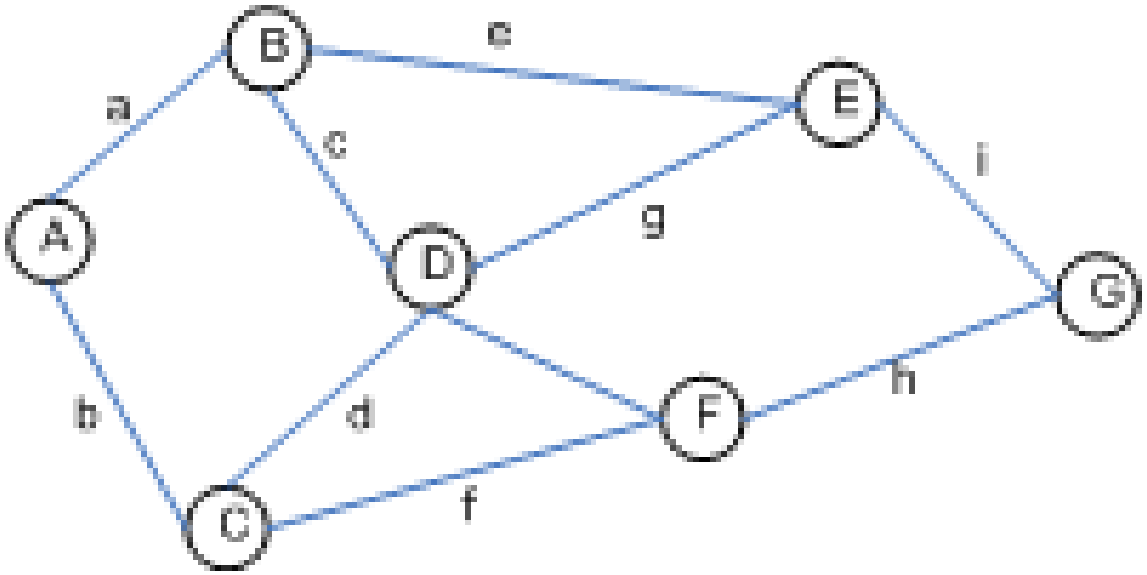
Blog: http://hariiniadalahhadiah.wordpress.com

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

Email: rosa_if_itb_01@yahoo.com

GRAF

- Graph adalah sebuah konsep struktur data yang terdiri dari kumpulan simpul (*node*) dan garis (*arc*).
- Sebuah garis harus diawali dan diakhiri dengan sebuah simpul.

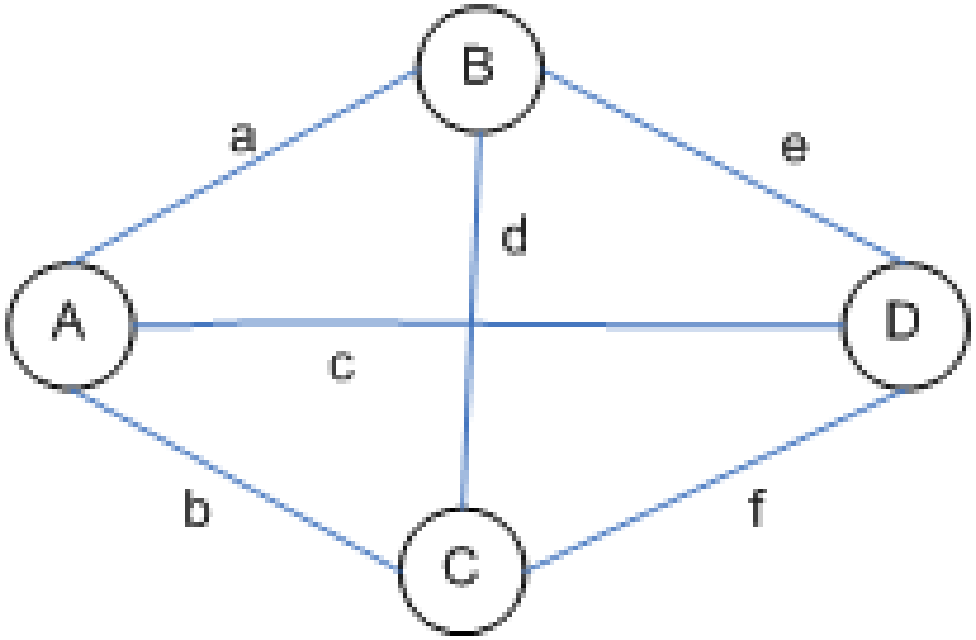


- Jalur atau arc dinyatakan dengan cara berikut:

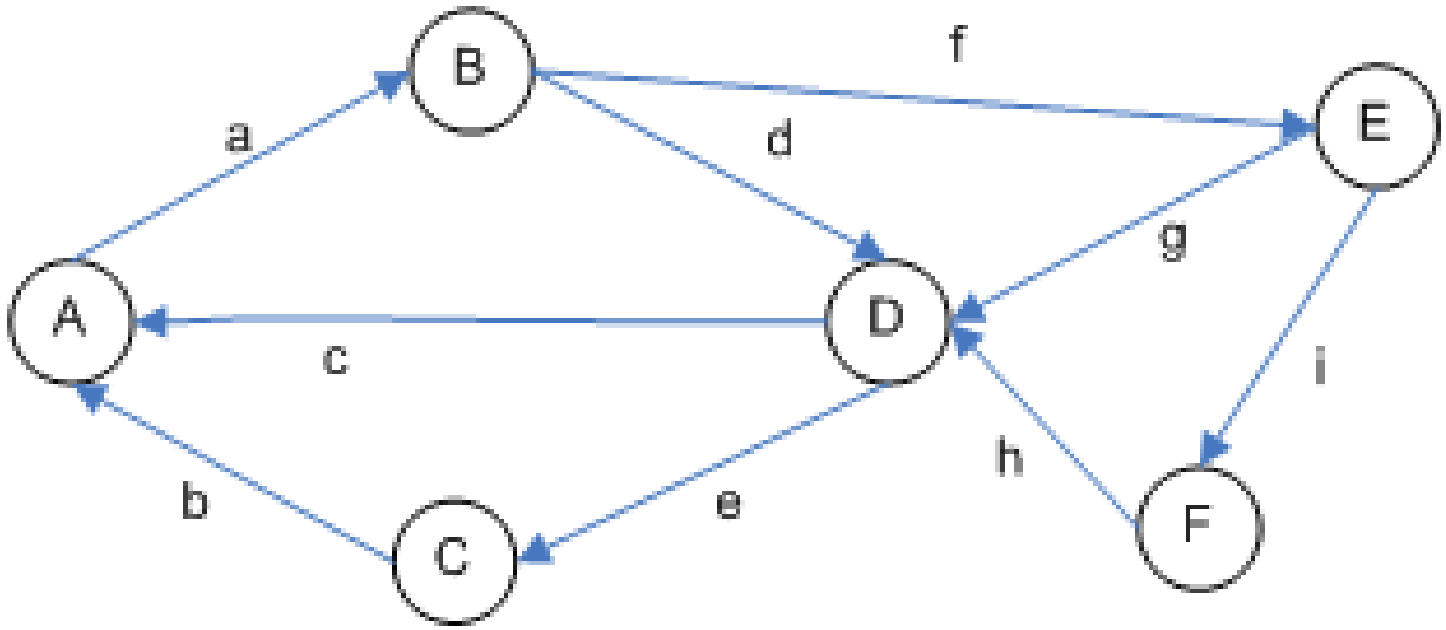
a = [A, B]

GRAF LENGKAP

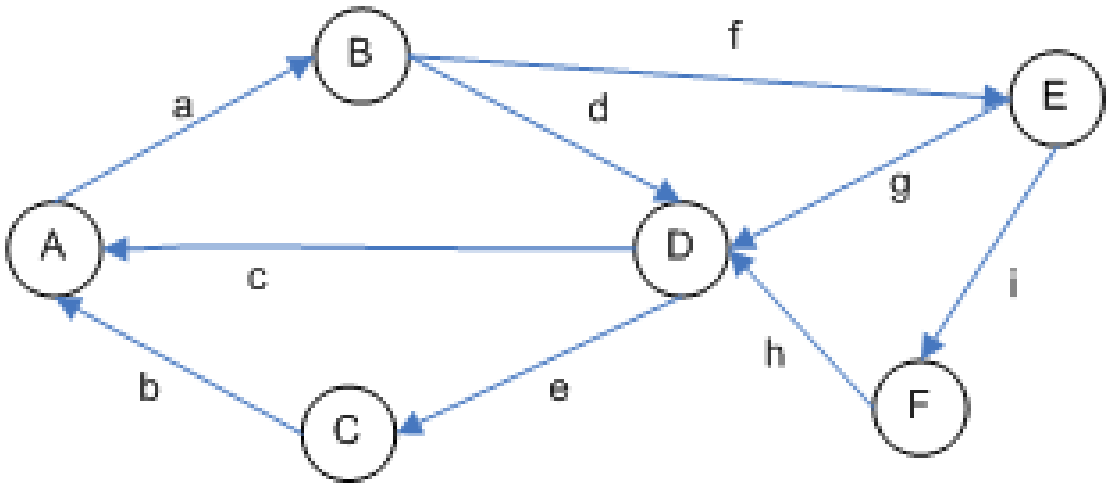
- Setiap simpul memiliki jalur ke semua simpul lainnya



GRAF BERARAH



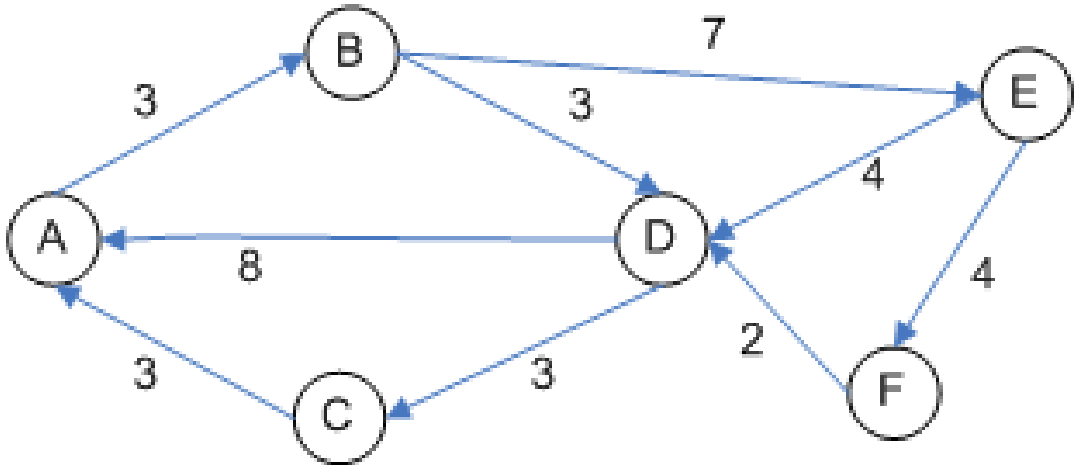
REPRESENTASI GRAF DENGAN MATRIKS TETANGGA



simpul tujuan

simpul awal	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	1	1	0
C	1	0	0	0	0	0
D	1	0	1	0	0	0
E	0	0	0	1	0	1
F	0	0	0	1	0	0

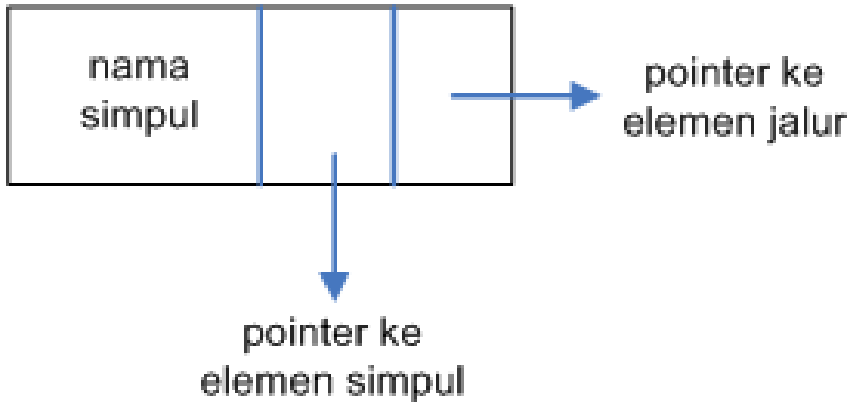
REPRESENTASI GRAF BERBOBOT DENGAN MATRIKS TETANGGA

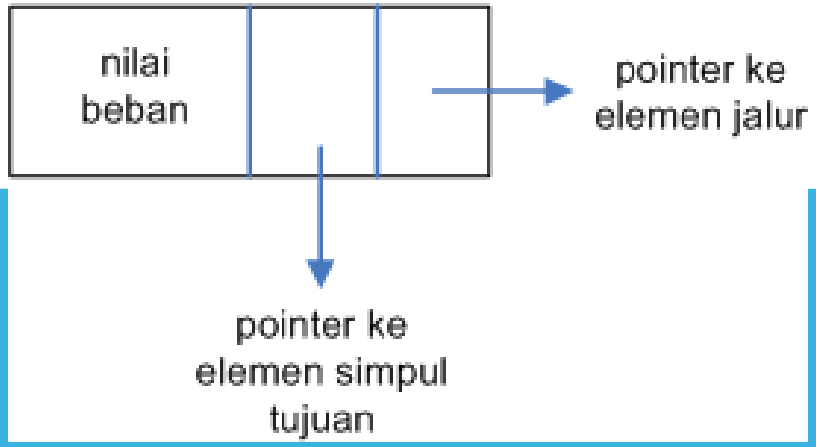


simpul tujuan

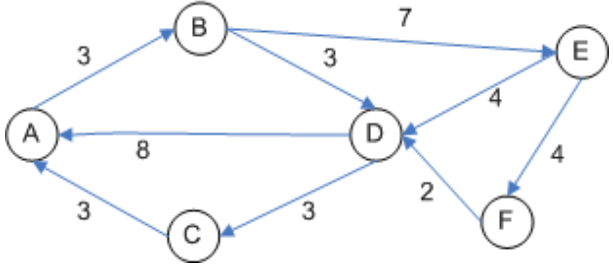
simpul awal	A	B	C	D	E	F
A	0	3	0	0	0	0
B	0	0	0	3	7	0
C	3	0	0	0	0	0
D	8	0	3	0	0	0
E	0	0	0	4	0	4
F	0	0	0	2	0	0

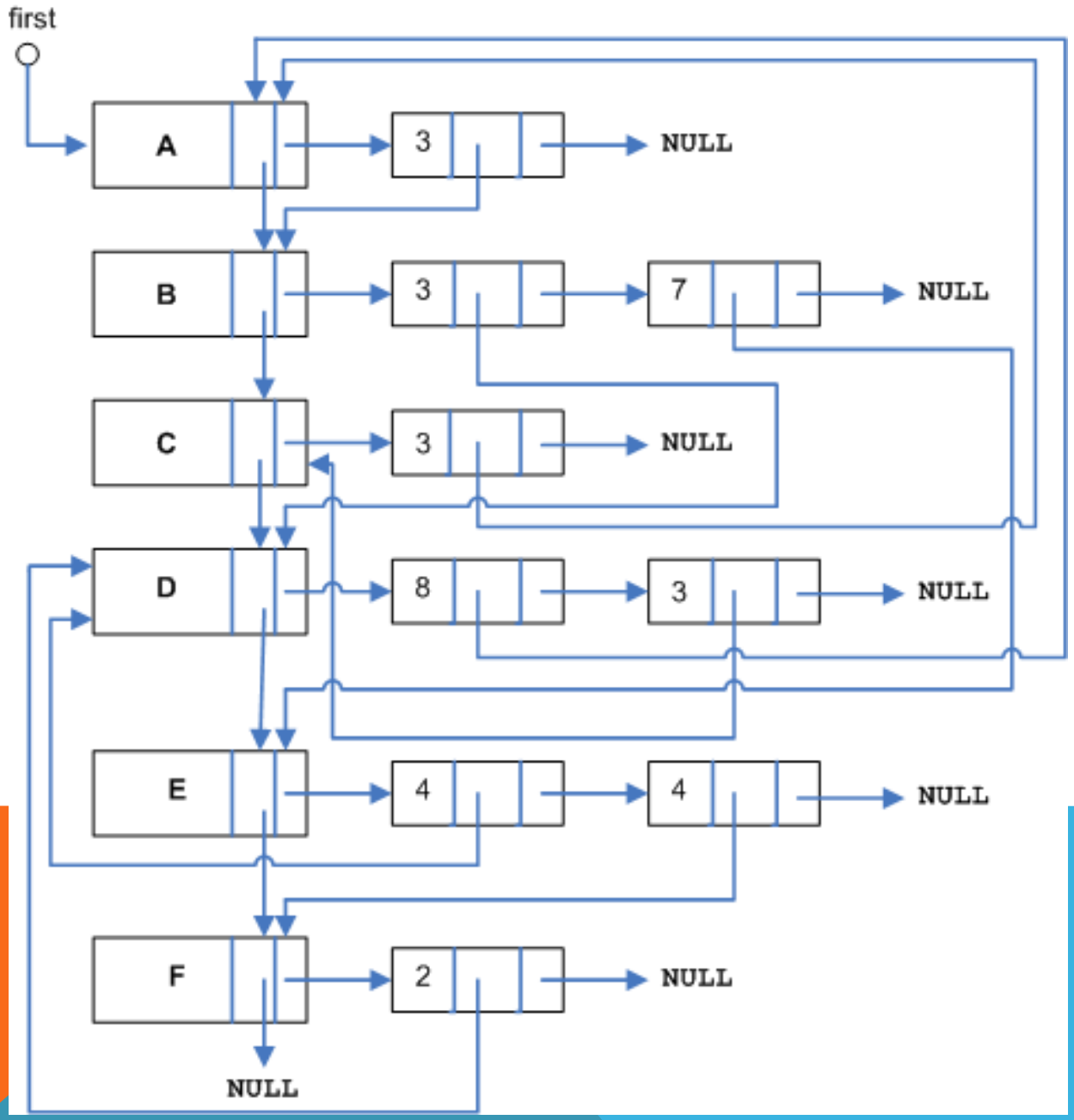
GRAF REPRESENTASI ELEMEN DINAMIS





GRAF REPRESENTASI DINAMIS





DEKLARASI ELEMEN DAN INISIALISASI

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

typedef struct smp *alamatsimpul;
typedef struct jlr *alamatjalur;

typedef struct smp{
  char info;
  alamatsimpul next;
  alamatjalur arc;
}simpul;

typedef struct jlr{
  int info;
  alamatjalur next;
  simpul *node;
}jalur;
```

```
typedef struct{
  simpul* first;
}graph;


void createEmpty(graph *G){
  (*G).first = NULL;
}
```

ADDSIMPUL

```
void addSimpul(char c, graph *G){



  simpul *node;
  node = (simpul *) malloc
   (sizeof (simpul));
    node->info = c;
    node->next = NULL;
    node->arc = NULL;


    if((*G).first == NULL){
      /*jika graph kosong*/
      (*G).first = node;
    }
```

```
    else{
      /*menambahkan simpul baru
    pada akhir graph*/


      simpul *last = (*G).first;


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


      last->next = node;


   }


}
```

ADDJALUR

```
void addJalur(simpul *tujuan, int
   beban, simpul *awal){


  jalur *arc;
  arc = (jalur *) malloc (sizeof
   (jalur));
    arc->info = beban;
    arc->next = NULL;
    arc->node = tujuan;


    if(awal->arc == NULL){
      /*jika list jalur kosong*/
      awal->arc = arc;
    }
```

```
else{
      /*menambahkan jalur baru
   pada akhir list jalur*/


      jalur *last = awal->arc;


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


      last->next = arc;


  }

}
```

DELSIMPUL

```
void delSimpul(char c, graph *G){
  simpul *elmt = (*G).first;
  if(elmt != NULL){
    simpul *prec = NULL;
    /*mencari simpul yang akan
   dihapus*/
    int ketemu = 0;
    while((elmt != NULL) &&
    (ketemu == 0)){
      if(elmt->info == c){
        ketemu = 1;
      }else{
        prec = elmt;
        elmt = elmt->next;
      }
    }
   if(ketemu == 1){
      if(prec == NULL){
        /*hapus simpul pertama*/
        (*G).first = elmt->next;
```

```
}else{
        if(elmt->next == NULL){
          /*hapus simpul terakhir*/
          prec->next = NULL;
        }else{
          /*hapus simpul di tengah*/
          prec->next = elmt->next;
          elmt->next = NULL;
        }
      }
      free(elmt);
    }else{
      printf("tidak ada simpul dengan
   info karakter masukan\n");
    }
  }else{
    printf("tidak ada simpul dengan
   info karakter masukan\n");
  }
}
```

FINDSIMPUL

```
simpul* findSimpul(char c, graph
   G){


  simpul *hasil = NULL;
  simpul *node = G.first;
  int ketemu = 0;
  while((node != NULL) &&
  (ketemu == 0)){
    if(node->info == c){
      hasil = node;
      ketemu = 1;
    }
    else{
      node = node->next;
    }
  }


  return hasil;
}
```

DELJALUR

```
void delJalur(char ctujuan, simpul
   *awal){
  jalur *arc = awal->arc;
  if(arc != NULL){
    jalur *prec = NULL;
   /*mencari jalur yang akan dihapus*/
    int ketemu = 0;
    while((arc != NULL) &&
      (ketemu == 0)){
      if(arc->node->info == ctujuan){
        ketemu = 1;
      }else{
        prec = arc;
        arc = arc->next;
      }
    }
    if(ketemu == 1){
      if(prec == NULL){
        /*hapus jalur pertama*/
        awal->arc = arc->next;
```

```
}else{
        if(arc->next == NULL){
          /*hapus jalur terakhir*/
          prec->next = NULL;
        }else{
          /*hapus jalur di tengah*/
          prec->next = arc->next;
          arc->next = NULL;
        }
      }
      free(arc);
    }else{
      printf("tidak ada jalur dengan
   simpul tujuan\n");
    }
  }else{
    printf("tidak ada jalur dengan
   simpul tujuan\n");
  }
}
```

PRINTGRAF

```
void printGraph(graph G){
  simpul *node = G.first;
  if(node != NULL){
    while(node != NULL){
      printf("simpul : %c\n",
      node->info);
      jalur *arc = node->arc;
      while(arc != NULL){
        printf("  - ada jalur ke
   simpul : %c dengan beban :
   %d\n", arc->node->info, arc-
   >info);
        arc = arc->next;
      }
      node = node->next;
    }
  }else{
    printf("graph kosong\n");
  }
}
```

MAIN (1)

```
int main(){
  graph G;
  createEmpty(&G);
  addSimpul('A', &G);
  addSimpul('B', &G);
  addSimpul('C', &G);
  addSimpul('D', &G);
  addSimpul('E', &G);
  addSimpul('F', &G);
  simpul *begin = findSimpul('A', G);
  simpul *end = findSimpul('B', G);
  if((begin != NULL) &&
   (end != NULL)){
    addJalur(end, 3, begin);
  }
  begin = findSimpul('B', G);
  end = findSimpul('D', G);
```

```
 if((begin != NULL) &&
   (end != NULL)){
    addJalur(end, 3, begin);
 }
 end = findSimpul('E', G);
 if((begin != NULL) &&
 (end != NULL)){
   addJalur(end, 7, begin);
 }
 begin = findSimpul('C', G);
 end = findSimpul('A', G);
 if((begin != NULL) &&
 (end != NULL)){
   addJalur(end, 3, begin);
 }
```

MAIN (2)

```
begin = findSimpul('D', G);
if((begin != NULL) &&
(end != NULL)){
  addJalur(end, 8, begin);
}
end = findSimpul('C', G);
if((begin != NULL) &&
(end != NULL)){
  addJalur(end, 3, begin);
}
begin = findSimpul('E', G);
end = findSimpul('D', G);
if((begin != NULL) &&
(end != NULL)){
  addJalur(end, 4, begin);
}
```

```
end = findSimpul('F', G);
if((begin != NULL) &&
(end != NULL)){
  addJalur(end, 4, begin);
}
begin = findSimpul('F', G);
end = findSimpul('D', G);
if((begin != NULL) &&
(end != NULL)){
  addJalur(end, 2, begin);
}
printf("================\n");
printGraph(G);
 printf("\n================\n"
 );
```

MAIN (3)

```
  begin = findSimpul('A', G);
  if(begin != NULL){
    delJalur('B', begin);
  }

  printf("================\n");
  printf("setelah dihapus\n");
  printGraph(G);
  printf("\n================\n");

  return 0;

}
```

DAFTAR PUSTAKA

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