Tugas Praktikum Algoritma dan Struktur Data

Nama : Agil Deriansyah Hasan
Nim : 4522210125

Dosen Pengajar :

Dra.SRI REZEKI CANDRA NURSARI,M.Kom
Prak. Algoritma dan Struktur Data - I

S1-Teknik Informatika
Fakultas Teknik
Universitas Pancasila 2023/2024

pasd12-2

pasd12-2.cpp pasd12-3.cpp

```cpp
#include <iostream>
using namespace std;

struct node {
    int info;
    struct node *next;
};

class stack {
    struct node *top;
public:
    stack();
    void push(int);
    int pop();
    bool isempty();
    void display();
};

stack::stack() {
    top = NULL;
}

void stack::push(int data) {
    node *p = new node;
    if (p == NULL) {
        cout << "Memori Penuh" << endl;
        exit(0);
    }
    p->info = data;
    p->next = top;
    top = p;
}

int stack::pop() {
    if (top == NULL) {
        cout << "Stack Kosong" << endl;
        return -1;
    }
    node *temp = top;
    top = top->next;
    int value = temp->info;
    delete temp;
    return value;
}

bool stack::isempty() {
    return (top == NULL);
}

```

pasd12-2.cpp pasd12-3.cpp

```cpp
void stack::display() {
    if (top == NULL) {
        cout << "Tidak Ada Tampilan" << endl;
        return;
    }
    cout << "Isi Stack:" << endl;
    node *p = top;
    while (p != NULL) {
        cout << p->info << endl;
        p = p->next;
    }
}

class graph {
private:
    int n;
    int **A;
public:
    graph(int size = 2);
    ~graph();
    bool isconnected(int, int);
    void addedge(int x, int y);
    void dfs(int, int);
};

graph::graph(int size) {
    if (size < 2) n = 2;
    else n = size;
    A = new int*[n];
    for (int i = 0; i < n; ++i)
        A[i] = new int[n];
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            A[i][j] = 0;
}

graph::~graph() {
    for (int i = 0; i < n; ++i)
        delete[] A[i];
    delete[] A;
}

bool graph::isconnected(int x, int y) {
    return (A[x - 1][y - 1] == 1);
}

void graph::addedge(int x, int y) {
    A[x - 1][y - 1] = A[y - 1][x - 1] = 1;
}

```

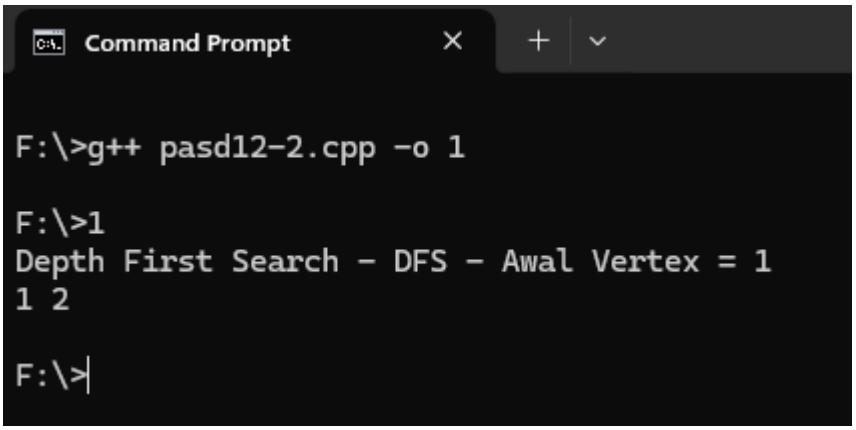
pasd12-2.cpp pasd12-3.cpp

```cpp

void graph::dfs(int x, int required) {
    stack s;
    bool *visited = new bool[n + 1];
    for (int i = 0; i <= n; i++)
        visited[i] = false;
    s.push(x);
    visited[x] = true;
    cout << "Depth First Search - DFS - Awal Vertex = " << x << endl;
    while (!s.isempty()) {
        int k = s.pop();
        cout << k << " ";
        if (k == required) break;
        for (int i = n; i >= 1; --i) {
            if (isconnected(k, i) && !visited[i]) {
                s.push(i);
                visited[i] = true;
            }
        }
    }
    cout << endl;
    delete[] visited;
}

int main() {
    graph g(8);
    g.addedge(1, 2);
    g.addedge(1, 3);
    g.addedge(1, 4);
    g.addedge(2, 5);
    g.addedge(2, 6);
    g.addedge(4, 7);
    g.addedge(2, 8);
    g.dfs(1, 2);
    return 0;
}

```



Pseudocode :

```
Kamus/Dekalrasi Variabel fungsi stack
-

Algoritma/Deskripsi fungsi stack
top = NULL

Kamus/Deklarasi Variabel fungsi push
data = int

Algoritma/Deskripsi fungsi push(data)
node *p = new node
if (p == NULL)
exit(0)
endif
p->info = data
p->next = top
top = p

Kamus/Deklarasi variabel Fungsi pop
value = int

Algoritma/Deskripsi fungsi pop
if (top == NULL)
        return -1
endif
node *temp = top
top = top->next
value = temp->info
delete temp
return value
```

```
Kamus/Deklarasi Variabel fungsi isempty
-
Algoritma/Deskripsi fungsi
return (top==NULL)

Kamus/Deklarasi Variabel fungsi display
-
Algoritma/Deskripsi fungsi display
if (top == NULL)
      return
endif
   node *p = top
   while (p != NULL)
      cout << p->info << endl;
      p = p->next
        endwhile

Kamus/Deklarasi Variabel fungsi graph
size,i,j,[n]=int

Algoritma/Deskripsi fungsi graph
if (size < 2) n = 2
   else n = size
   A = new int*[n]
   for (int i = 0; i < n; ++i)
      A[i] = new int[n]
   for (int i = 0; i < n; ++i)
      for (int j = 0; j < n; ++j)
         A[i][j] = 0

Kamus/Dekalrasi Variabel fungsi graph
-
Algortima/Deskripsi fungsi graph
or (int i = 0; i < n; ++i)
      delete[] A[i];
   delete[] A
```

```
Kamus/Deklarasi Variabel fungsi isconnected
x,y = int

Algoritma/Deskripsi fungsi isconnected(x,y)
A[x - 1][y - 1] = A[y - 1][x - 1] = 1

Kamus/Deklarasi Variabel fungsi dfs
x,requred,i,k = int
visited = bool

Algoritma/Deskripsi fungsi dfs(x,required)
*visited = new bool[n + 1]
   for (int i = 0; i <= n; i++)
      visited[i] = false
   s.push(x)
   visited[x] = true
   print x
   while (!s.isempty())
      int k = s.pop()
      print k
      if (k == required) break
      for (int i = n; i >= 1; --i)
         if (isconnected(k, i) && !visited[i])
            s.push(i)
            visited[i] = true
endif
endfor
endwhile
delete[] visited
```

```
Kamus/Deklarasi Variabel fungsi utama
info,pop,n,**A,size = int
*next,*top = node
isempty = bool

Algoritma/Deskripsi fungsi utama
struct node { info, &next}
struct stack {top}
public : class stack{stack(), push(int), pop(),
isempty(), display()}
private : struct graph{n,**A}
public : struct graph{ graph(size = 2), ~graph(),
isconnected, addedge(int x, int y), dfs
graph g(8);
   g.addedge(1, 2);
   g.addedge(1, 3);
   g.addedge(1, 4);
   g.addedge(2, 5);
   g.addedge(2, 6);
   g.addedge(4, 7);
   g.addedge(2, 8);
   g.dfs(1, 2)
```

Algoritma :

1. Membuat fungsi stack
2. top==NULL
3. Membuat fungsi push(data)
4. node *p=new node
5. Jika (p==NULL) maka kerjakan baris 6
6. exit(0)
7. p->info=data
8. p->next=top
9. top=p
10. Membuat fungsi pop
11. Jika (top==NULL) maka kerjakan baris 12
12. return -1
13. node *temp = top
14. top = top->next
15. value = temp->info
16. delete temp
17. return value
18. Membuat fungsi isempty
19. return(top==NULL)
20. Membuat fungsi display
21. Jika(top==NULL) maka kerjakan baris 22
22. return
23. node *p=top
24. Selama (p != NULL) maka kerjakan baris 25 s.d 26
25. p->info
26. p=p->next
27. Membuat fungsi graph(size)
28. Jika(size<2)
29. n=2
30. n=size
31. A=new[n]
32. Selama (i=0)
33. A[i]=new[n]
34. i++
34. Selama(i=0)
35. Selama (j=0)
36. i++
37. j++
38. Membuat fungsi ~graph
39. Selama (i=0)
40. delete[] A[i]
41. delete[] A
42. i++
43. Membuat fungsi isconnected(x,y)
44. return (A[x - 1][y - 1] == 1)
45. Membuat fungsi addedge(x,y)
46. A[x - 1][y - 1] = A[y - 1][x - 1] = 1
47. Membuat fungsi dfs(x,required)
48. stack s
49. *visited = new [n+1]
50. Selama (i=0)
51. visited[i]=false
52. s.push(x)
53. visited[x]=true
54. Mencetak/Menampilkan Nilai x
55. Selama (!s.isempty()) maka kerjakan baris 56 s.d 63
56. k=s.pop
57. Mencetak/Menampilkan Nilai k
58. Jika(k==required)
59. break

60. Selama (i=n) maka kerjakan baris 61 s.d 64
61. Jika (isconnected(k, i) && !visited[i]) maka kerjakan baris 62 s.d 63
62. s.push[i]
63. visited[i]=true
64. i–
65. delete[] visited
66. Membuat fungsi utama
67. Deklarasi struktur(struct node{info,*next)
68. Deklarasi struktur(struct stack{ *top}
69. Membuat class stack
70. Mendeklarasikan clas stack dengan kata kunci public
71. {stack(), push(int), pop(), isempty(), display()
72. Membuat class graph
73. Mendeklarasikan class graph dengan kata kunci private
74. { n, **A}
75. Mendeklarasikan class graph dengan kata kunci public
76. {graph(size = 2), ~graph(), isconnected(int, int), addedge(x,y), dfs|
77. graph g(8)
78. g.addedge(1, 2)
79. g.addedge(1, 3)
80. g.addedge(1, 4)
81. g.addedge(2, 5)
82. g.addedge(2, 6)
83. g.addedge(4, 7)
84. g.addedge(2, 8)
85. g.dfs(1, 2)
86. selesai

pasd12-2.cpp pasd12-3.cpp

```cpp
#include <iostream>
using namespace std;

typedef struct simpul *altsimpul;
typedef struct jalur *altjalur;

typedef struct simpul {
    char kontainersimpul;
    altsimpul nextsimpul;
    altjalur arc;
} cansimpul;

typedef struct jalur {
    int kontainerjalur;
    altjalur nextjalur;
    cansimpul *tujuan;
} canjalur;

typedef struct {
    cansimpul* first;
} graph;

void simpulbaru(graph *G, char c) {
    cansimpul *baru = new cansimpul;
    baru->kontainersimpul = c;
    baru->nextsimpul = NULL;
    baru->arc = NULL;
    if ((*G).first == NULL) {
        (*G).first = baru;
    } else {
        cansimpul *last = (*G).first;
        while (last->nextsimpul != NULL) {
            last = last->nextsimpul;
        }
        last->nextsimpul = baru;
    }
}

void tambahjalur(cansimpul *awal, cansimpul *tujuan, int beban) {
    canjalur *baru = new canjalur;
    baru->kontainerjalur = beban;
    baru->nextjalur = NULL;
    baru->tujuan = tujuan;
    if (awal->arc == NULL) {
        awal->arc = baru;
    } else {
        canjalur *last = awal->arc;
        while (last->nextjalur != NULL) {
            last = last->nextjalur;
        }
        last->nextjalur = baru;
    }
}
```

pasd12-2.cpp pasd12-3.cpp

```cpp
}

cansimpul* findsimpul(char c, graph G) {
    cansimpul *hasil = NULL;
    cansimpul *bantu = G.first;
    bool ketemu = false;
    while ((bantu != NULL) && (ketemu == false)) {
        if (bantu->kontainersimpul == c) {
            hasil = bantu;
            ketemu = true;
        } else {
            bantu = bantu->nextsimpul;
        }
    }
    return hasil;
}

void deljalur(char ctujuan, cansimpul *awal) {
    canjalur *hapus = awal->arc;
    if (hapus != NULL) {
        canjalur *prev = NULL;
        bool ketemu = false;
        while ((hapus != NULL) && (ketemu == false)) {
            if (hapus->tujuan->kontainersimpul == ctujuan) {
                ketemu = true;
            } else {
                prev = hapus;
                hapus = hapus->nextjalur;
            }
        }
        if (ketemu) {
            if (prev == NULL) {
                awal->arc = hapus->nextjalur;
            } else {
                prev->nextjalur = hapus->nextjalur;
            }
            delete hapus;
        }
    }
}

void tampilkanGraph(graph G) {
    cansimpul *simpul = G.first;
    while (simpul != NULL) {
        cout << "Simpul " << simpul->kontainersimpul << " terhubung dengan: ";
        canjalur *jalur = simpul->arc;
        while (jalur != NULL) {
            cout << jalur->tujuan->kontainersimpul << " (beban: " << jalur->kontainerjalur << "), ";
            jalur = jalur->nextjalur;
        }
        cout << endl;
        simpul = simpul->nextsimpul;
    }
}
```

pasd12-2.cpp pasd12-3.cpp

```cpp
    }
}

int main() {
    graph G;
    G.first = NULL;

    simpulbaru(&G, 'A');
    simpulbaru(&G, 'B');
    simpulbaru(&G, 'C');

    cansimpul *nodeA = findsimpul('A', G);
    cansimpul *nodeB = findsimpul('B', G);
    cansimpul *nodeC = findsimpul('C', G);

    tambahjalur(nodeA, nodeB, 5);
    tambahjalur(nodeA, nodeC, 10);

    cout << "Graph sebelum menghapus jalur:" << endl;
    tampilkanGraph(G);

    deljalur('B', nodeA);

    cout << "Graph setelah menghapus jalur:" << endl;
    tampilkanGraph(G);

    return 0;
}
```

```
F:\>g++ pasd12-3.cpp -o 1

F:\>1
Graph sebelum menghapus jalur:
Simpul A terhubung dengan: B (beban: 5), C (beban: 10),
Simpul B terhubung dengan:
Simpul C terhubung dengan:
Graph setelah menghapus jalur:
Simpul A terhubung dengan: C (beban: 10),
Simpul B terhubung dengan:
Simpul C terhubung dengan:

F:\>
```

Pseudocode :

```
Kamus/Deklarasi Variabel fungsi simpulbaru
*G = graph
c = char

Algoritma/Deskripsi fungsi simpulbaru (*G, c)
cansimpul *baru = new cansimpul;
   baru->kontainersimpul = c
   baru->nextsimpul = NULL
   baru->arc = NULL
   if ((*G).first == NULL)
      (*G).first = baru
   else
      cansimpul *last = (*G).first
      while (last->nextsimpul != NULL)
         last = last->nextsimpul
        endwhile
      last->nextsimpul = baru
endif

Kamus/Dekalrasi Variabel fugnsi tambahjalur
"awal, *tujuan = cansimpul
beban = int

Algoritma/Deskripsi fungsi tambahjalur(*awal,*tujuan,beban)
canjalur *baru = new canjalur
   baru->kontainerjalur = beban
   baru->nextjalur = NULL
   baru->tujuan = tujuan
   if (awal->arc == NULL)
      awal->arc = baru
   else
      canjalur *last = awal->arc
      while (last->nextjalur != NULL)
         last = last->nextjalur
        endwhile
      last->nextjalur = baru
endif
```

```
Kamus/Deklarasi Variabel fungsi findsimpul
c = char
ketemu = bool
G = graph

Algoritma/Deskripsi fungsi findsimpull(c,G)
cansimpul *hasil = NULL
   cansimpul *bantu = G.first
   ketemu = false
   while ((bantu != NULL) && (ketemu == false))
      if (bantu->kontainersimpul == c)
         hasil = bantu
         ketemu = true
      else
         bantu = bantu->nextsimpul
         endif
return hasil

Kamus/Deklarasi Variabel fungsi deljalur
ctujuan = char
*awal = cansimpul

Algoritma/Deskripsi fungsi deljalur(ctujuan, *awal)
canjalur *hapus = awal->arc
   if (hapus != NULL)
      canjalur *prev = NULL
      bool ketemu = false
      while ((hapus != NULL) && (ketemu == false))
         if (hapus->tujuan->kontainersimpul == ctujuan)
            ketemu = true
         else
            prev = hapus
            hapus = hapus->nextjalur
         endif
         endwhile
      if (ketemu)
         if (prev == NULL)
            awal->arc = hapus->nextjalur
         else
            prev->nextjalur = hapus->nextjalur
 delete hapus
endif endif
```

```
Kamus/Deklarasi Variabel fungsi tampilkanGraph
G = graph

Algorima/Deskripsi fungsi tampilkanGraph(G)
cansimpul *simpul = G.first
   while (simpul != NULL)
print simpul->kontainersimpul
       canjalur *jalur = simpul->arc
       while (jalur != NULL)
print jalur->tujuan->kontainersimpul
print jalur->kontainerjalur
          jalur = jalur->nextjalur
       simpul = simpul->nextsimpul

Kamus/Dekalrasi Variabel fungsi utama
kontainesimpul = char
kontainerjalur = int
G = graph
```

```
Algoritma/Deskripsi fungsi utama
typedef struct simpul *altsimpul
typedef struct jalur *altjalur
typedef struct simpul {kontainersimpul,
altsimpul nextsimpul, altjalur arc}
cansimpul
typedef struct jalur {kontainerjalur, altjalur
nextjalur,cansimpul *tujuan}
canjalur
typedef struct {cansimpul*}
firstgraph
graph G
G.first =NULL
simpulbaru(&G, 'A')
   simpulbaru(&G, 'B')
   simpulbaru(&G, 'C')
   cansimpul *nodeA = findsimpul('A', G)
   cansimpul *nodeB = findsimpul('B', G)
   cansimpul *nodeC = findsimpul('C', G)
   tambahjalur(nodeA, nodeB, 5)
   tambahjalur(nodeA, nodeC, 10)
   tampilkanGraph(G)
   deljalur('B', nodeA)
   tampilkanGraph(G)
```

Algoritma :

1. Membuat fungsi simpulbaru(*G,c)
2. cansimpul *baru = new cansimpul
3. baru->kontainersimpul = c
4. baru->nextsimpul = NULL
5. baru->arc = NULL
6. Jika ((*G).first == NULL) maka kerjakan baris 7 s.d 11
7. (*G).first = baru
8. cansimpul *last = (*G).first
9. Selama(last->nextsimpul != NULL) maka kerjakan baris 10
10. last = last->nextsimpul
11. last->nextsimpul = baru
12. Membuat fungsi tambahjalur(*awal,*tujuan,beban)
13. canjalur *baru = new canjalur
14. baru->kontainerjalur = beban
15. baru->nextjalur = NULL
16. baru->tujuan = tujuan
17. Jika (awal->arc == NULL) maka kerjakan baris 18 s.d 22
18. awal->arc = baru
19. canjalur *last = awal->arc
20. selama (last->nextjalur != NULL) maka kerjakan baris 21
21. last = last->nextjalur
22. last->nextjalur = baru
23. Membuat fungsi findsimpul(c,G)
24. cansimpul *hasil = NULL
25. cansimpul *bantu = G.first
26. ketemu = false
27. Selama ((bantu != NULL) && (ketemu == false)) maka kerjkan baris 28 s.d 31
28. Jika (bantu->kontainersimpul == c) maka kerjakan baris 29 s.d 31
29. hasil = bantu
30. ketemu = true
31. bantu = bantu->nextsimpul
32. return hasil

33. Membuat fungsi deljalur(ctujuan, *awal)
34. Jika (hapus != NULL) maka kerjakan baris 35 s.d 46
35. canjalur *prev = NULL
36. ketemu = false
37. Selama ((hapus != NULL) && (ketemu == false)) maka kerjakan baris 38 s.d 46
38. Jika (hapus->tujuan->kontainersimpul == ctujuan) maka kerjakan baris 41
39. ketemu = true
40. prev = hapus
41. hapus = hapus->nextjalur 46
42. Jika (ketemu) maka kerjakan baris 43 s.d 46
43. Jika (prev == NULL) maka kerjakan baris 44 s.d 45
44. awal->arc = hapus->nextjalur
45. prev->nextjalur = hapus->nextjalur
46. delete hapus
46. Membuat fungsi tampilkanGraph(G)
47. cansimpul *simpul = G.first
48. Selama (simpul != NULL) maka kerjakan baris 49 s.d 55
49. Mencetak/Menampilkan Nilai simpul->kontainersimpul
50. canjalur *jalur = simpul->arc
51. Selama (jalur != NULL) maka kerjakan baris 52 s.d 54
52. Mencetak/Menampilkan Nilai jalur->tujuan->kontainer
53. Mencetak/Menampilkan nilai simpul jalur->kontainerjalur
54. jalur = jalur->nextjalur
55. simpul = simpul->nextsimpul
56. Membuat fungsi utama
57. Deklarsi struktur (simpul *altsimpul)
58. Deklarasi struktur (struct jalur *altjalur)
59. Dekalrasi sttruktur (struct simpul{kontainersimpul, nextsimpul,arc}
60. Membuat object cansimpul dari struktrur simpul
61. Deklarsi struktur (struct jalur{kontainerjalur, nextjalur,tujuan})
62. Membuat object canjalur dari struktur jalur
63. Deklarsi struktur(struct {first})
64. Membuat object graph dari struktur

65. G.first =NULL
66. simpulbaru(&G, 'A');
67. simpulbaru(&G, 'B');
68. simpulbaru(&G, 'C');
69. cansimpul *nodeA = findsimpul('A', G)
70. cansimpul *nodeB = findsimpul('B', G)
71. cansimpul *nodeC = findsimpul('C', G)
72. tambahjalur(nodeA, nodeB, 5)
73. tambahjalur(nodeA, nodeC, 10)
74. tampilkanGraph(G)
75. deljalur('B', nodeA)
76. tampilkanGraph(G)
77. Selesai