**PRAKTIKUM ALGORITMA STRUKTUR DATA**

**TEKNIK INFORMATIKA**

**Prak-11**



Oleh :

Faathir Akbar Nugroho

4522210033

Kelas A

**Pseudocode (Nomor 02)**

**Kamus/Deklarasi Variabel Function Fatirstack::Fatirstack()**

-

**Algoritma/Deskripsi Function Fatirstack::Fatirstack()**

Fatirtop = NULL

**Kamus/Deklarasi Variabel Function Fatirstack::Fatirpush(int Fatirdata)**

**-**

**Algoritma/Deskripsi Function Fatirstack::Fatirpush(int Fatirdata)**

Fatirnode \*Fatirp

if ((Fatirp = (Fatirnode\*)malloc(sizeof(Fatirnode))) == NULL)

print("Memori Penuh")

exit(0)

endif

Fatirp = new Fatirnode

Fatirp->Fatirinfo = Fatirdata

Fatirp->Fatirnext = NULL

if (Fatirtop != NULL)

Fatirp->Fatirnext = Fatirtop

endif

Fatirtop = Fatirp

**Kamus/Deklarasi Variabel Function Fatirstack::Fatirpop()**

Fatirvalue = int

**Algoritma/Deskripsi Function Fatirstack::Fatirpop()**

struct Fatirnode \*Fatirtemp

if (Fatirtop == NULL)

print("Stack Kosong")  
else

Fatirtemp = Fatirtop

Fatirtop = Fatirtop->Fatirnext

Fatirvalue = Fatirtemp->Fatirinfo

delete Fatirtemp

endif

return Fatirvalue

**Kamus/Deklarasi Variabel Function** **Fatirstack::FatirisEmpty()**

**-**

**Algoritma/Deskripsi Function Fatirstack::FatirisEmpty()**

return (Fatirtop == NULL)

**Kamus/Deklarasi Variabel Function** **Fatirstack::Fatirdisplay()**

**-**

**Algoritma/Deskripsi Function Fatirstack::Fatirdisplay()**

struct Fatirnode \*Fatirp = Fatirtop

if (Fatirtop == NULL)

print("Tidak ada tampilan”)

else

while (Fatirp != NULL)

print(Fatirp->Fatirinfo)

Fatirp = Fatirp->Fatirnext

endwhile

**Kamus/Deklarasi Variabel Function FatirGraph::FatirGraph(int Fatirsize)**

Fatiri, Fatirj = int

**Algoritma/Deskripsi Function FatirGraph::FatirGraph(int Fatirsize)**

if (Fatirsize < 2)

Fatirn = 2

else

Fatirn = Fatirsize

endif

FatirA = new int \*[Fatirn]

for (Fatiri = 0; Fatiri < Fatirn; ++Fatiri)

FatirA[Fatiri] = new int[Fatirn]

endfor

for (Fatiri = 0; Fatiri < Fatirn; ++Fatiri)

for (Fatirj = 0; Fatirj < Fatirn; ++Fatirj)

FatirA[Fatiri][Fatirj] = 0

endfor

endfor

**Kamus/Deklarasi Variabel Function** **FatirGraph::~FatirGraph()**

**-**

**Algoritma/Deskripsi Function** **FatirGraph::~FatirGraph()**

for (int Fatiri = 0; Fatiri < Fatirn; ++Fatiri)

delete[] FatirA[Fatiri]

delete[] FatirA

endfor

**Kamus/Deklarasi Variabel Function FatirGraph::FatirisConnected(int Fatirx, int Fatiry)**

**-**

**Algoritma/Deskripsi Function FatirGraph::FatirisConnected(int Fatirx, int Fatiry)**

return (FatirA[Fatirx - 1][Fatiry - 1] == 1)

**Kamus/Deklarasi Variabel Function** **FatirGraph::FatiraddEdge(int Fatirx, int Fatiry)**

**-**

**Algoritma/Deskripsi Function FatirGraph::FatiraddEdge(int Fatirx, int Fatiry)**

FatirA[Fatirx - 1][Fatiry - 1] = FatirA[Fatiry - 1][Fatirx - 1] = 1

**Kamus/Deklarasi Variabel Function FatirGraph::FatirDFS(int Fatirx, int required)**

Fatiri = int

**Algoritma/Deskripsi Function FatirGraph::FatirDFS(int Fatirx, int required)**

Fatirstack Fatirs

bool \*Fatirvisited = new bool[Fatirn + 1]

for (Fatiri = 0; Fatiri <= Fatirn; Fatiri++)

Fatirvisited[Fatiri] = false

endfor

Fatirs.Fatirpush(Fatirx)

Fatirvisited[Fatirx] = true

if (Fatirx == required)

return

endif

print(Fatirx)

while (!Fatirs.FatirisEmpty())

int Fatirk = Fatirs.Fatirpop()

if (Fatirk == required)

break

endif

print(Fatirk)

for (Fatiri = Fatirn; Fatiri >= 1; --Fatiri)

if (FatirisConnected(Fatirk, Fatiri) && !Fatirvisited[Fatiri])

Fatirs.Fatirpush(Fatiri)

Fatirvisited[Fatiri] = true

endif

endfor

endwhile

delete[] Fatirvisited

**Kamus/Deklarasi Variabel**

**-**

**Algoritma/Deskripsi**

struct Fatirnode

int Fatirinfo

struct Fatirnode \*Fatirnext

class Fatirstack

struct Fatirnode \*Fatirtop

public:

Fatirstack()

void Fatirpush(int)

int Fatirpop()

bool FatirisEmpty()

void Fatirdisplay()

class FatirGraph

private:

int Fatirn

int \*\*FatirA

public:

FatirGraph(int Fatirsize = 2)

~FatirGraph()

bool FatirisConnected(int, int)

void FatiraddEdge(int Fatirx, int Fatiry)

void FatirDFS(int Fatirx, int required)

FatirGraph Fatirg(8)

Fatirg.FatiraddEdge(1, 2)

Fatirg.FatiraddEdge(1, 3)

Fatirg.FatiraddEdge(1, 4)

Fatirg.FatiraddEdge(2, 5)

Fatirg.FatiraddEdge(3, 6)

Fatirg.FatiraddEdge(4, 7)

Fatirg.FatiraddEdge(4, 8)

Fatirg.FatirDFS(1, 8)

**Algoritma/Bahasa Natural (Nomor 02)**

1. Membuat function Fatirstack::Fatirstack()
2. Fatirtop = NULL
3. Membuat function Fatirstack::Fatirpush(int Fatirdata)
4. Mendefinisikan struct (Fatirnode \*Fatirp)
5. Jika ((Fatirp = (Fatirnode\*)malloc(sizeof(Fatirnode))) == NULL), maka kerjakan baris 6 s.d 7, kalau tidak kerjakan baris 8
6. Menampilkan (“Memori Penuh”)
7. exit(0)
8. Fatirp = new Fatirnode
9. Fatirp->Fatirinfo = Fatirdata
10. Fatirp->Fatirnext = NULL
11. Jika (Fatirtop != NULL), maka kerjakan baris 12, kalau tidak kerjakan baris 13
12. Fatirp->Fatirnext = Fatirtop
13. Fatirtop = Fatirp
14. Membuat function Fatirstack::Fatirpop()
15. Mendefinisikan struct (struct Fatirnode \*Fatirtemp)
16. Jika (Fatirtop == NULL), maka kerjakan baris 17, kalau tidak kerjakan baris 18 s.d 21
17. Menampilkan ("Stack Kosong")
18. Fatirtemp = Fatirtop
19. Fatirtop = Fatirtop->Fatirnext
20. Fatirvalue = Fatirtemp->Fatirinfo
21. delete Fatirtemp
22. return Fatirvalue
23. Membuat function Fatirstack::FatirisEmpty()
24. return (Fatirtop == NULL)
25. Membuat function Fatirstack::Fatirdisplay()
26. Mendefinisikan struct (struct Fatirnode \*Fatirp = Fatirtop)
27. Jika (Fatirtop == NULL), maka kerjakan baris 28, kalau tidak kerjakan baris 29 s.d 31
28. Menampilkan ("Tidak ada tampilan")
29. Selama (Fatirp != NULL), maka kerjakan baris 30 s.d 31, kalau tidak kerjakan baris 32
30. Menampilkan isi/nilai variabel (Fatirp->Fatirinfo)
31. Fatirp = Fatirp->Fatirnext
32. Membuat function FatirGraph::FatirGraph(int Fatirsize)
33. Jika (Fatirsize < 2), maka kerjakan baris 34, kalau tidak kerjakan baris 35
34. Fatirn = 2
35. Fatirn = Fatirsize
36. FatirA = new int \*[Fatirn]
37. Fatiri = 0
38. Selama (Fatiri < Fatirn), maka kerjakan baris 39 s.d 40, kalau tidak kerjakan baris 41
39. FatirA[Fatiri] = new int[Fatirn]
40. ++Fatiri
41. Fatiri = 0
42. Selama (Fatiri < Fatirn), maka kerjakan baris 43 s.d 47, kalau tidak kerjakan baris 48
43. Fatirj = 0
44. Selama (Fatirj < Fatirn), maka kerjakan baris 45 s.d 46, kalau tidak kerjakan baris 47
45. FatirA[Fatiri][Fatirj] = 0
46. ++Fatirj
47. ++Fatiri
48. Membuat function FatirGraph::~FatirGraph()
49. int Fatiri = 0
50. Selama (Fatiri < Fatirn), maka kerjakan baris 51 s.d 52, kalau tidak kerjakan baris 53
51. delete[] FatirA[Fatiri]
52. ++Fatiri
53. delete[] FatirA
54. Membuat function FatirGraph::FatirisConnected(int Fatirx, int Fatiry)
55. return (FatirA[Fatirx - 1][Fatiry - 1] == 1)
56. Membuat function FatirGraph::FatiraddEdge(int Fatirx, int Fatiry)
57. FatirA[Fatirx - 1][Fatiry - 1] = FatirA[Fatiry - 1][Fatirx - 1] = 1
58. Membuat function FatirGraph::FatirDFS(int Fatirx, int required)
59. Mendefinisikan struct (Fatirstack Fatirs)
60. bool \*Fatirvisited = new bool[Fatirn + 1]
61. Fatiri = 0
62. Selama (Fatiri <= Fatirn), maka kerjakan baris 63 s.d 64, kalau tidak kerjakan baris 65
63. Fatirvisited[Fatiri] = false
64. Fatiri++
65. Menampilkan pengaksesan anggota class (Fatirs.Fatirpush(Fatirx))
66. Fatirvisited[Fatirx] = true
67. Jika (Fatirx == required), maka kerjakan baris 68, kalau tidak kerjakan baris 69
68. return
69. Menampilkan isi/nilai variabel Fatirx
70. Selama (!Fatirs.FatirisEmpty()), maka kerjakan baris 71 s.d 80, kalau tidak kerjakan baris 81
71. int Fatirk = Fatirs.Fatirpop()
72. Jika (Fatirk == required), maka kerjakan baris 73, kalau tidak kerjakan baris 74
73. break
74. Menampilkan isi/nilai variabel Fatirk
75. Fatiri = Fatirn
76. Selama (Fatiri >= 1), maka kerjakan baris 77 s.d 80, kalau tidak kerjakan baris 81
77. Jika (FatirisConnected(Fatirk, Fatiri) && !Fatirvisited[Fatiri]), maka kerjakan baris 78 s.d 79, kalau tidak kerjakan baris 80
78. Fatirs.Fatirpush(Fatiri)
79. Fatirvisited[Fatiri] = true
80. --Fatiri
81. delete[] Fatirvisited
82. Mendeklarasikan struct (struct Fatirnode(int Fatirinfo, struct Fatirnode \*Fatirnext))
83. Mendeklarasikan class (class Fatirstack(struct Fatirnode \*Fatirtop))
84. Mendeklarasikan class dengan tipe akses public (class Fatirstack(Fatirstack(), void Fatirpush(int), int Fatirpop(), bool FatirisEmpty(), void Fatirdisplay()))
85. Mendeklarasikan class dengan tipe akses private (class FatirGraph(int Fatirn, int \*\*FatirA))
86. Mendeklarasikan class dengan tipe akses public (class FatirGraph(FatirGraph(int Fatirsize = 2), ~FatirGraph(), bool FatirisConnected(int, int), void FatiraddEdge(int Fatirx, int Fatiry), void FatirDFS(int Fatirx, int required)))
87. Mendefinisikan class (FatirGraph Fatirg(8))
88. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 2))
89. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 3))
90. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(1, 4))
91. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(2, 5))
92. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(3, 6))
93. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(4, 7))
94. Menampilkan pengaksesan anggota class (Fatirg.FatiraddEdge(4, 8))
95. Menampilkan pengaksesan anggota class (Fatirg.FatirDFS(1, 8))
96. Selesai

**Program (Nomor 02)**

