PERTEMUAN 14

TEKNIK PENGUJIAN PERANGKAT LUNAK

TESTING

- Pengujian perangkat lunak adalah proses menjalankan dan mengevaluasi sebuah perangkat lunak secara manual maupun otomatis untuk menguji apakah perangkat lunak sudah memenuhi persyaratan atau belum, atau untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.
- Pengujian merupakan suatu tahapan pengerjaan yang bertujuan mencari kesalahan program. Kesalahan yang terjadi selama proses pengembangan perangkat lunak akan mengakibatkan bertambahnya waktu untuk menyelesaikan pekerjaan tersebut.
- Pengujian hendaknya dilakukan pada setiap tahap pengembangan yaitu mulai dari tahap analisis kebutuhan sampai dengan tahap perawatan.

KENAPA HARUS DIUJI ?

- Kita bukan seorang programmer yang cukup baik
- Kita mungkin tidak dapat cukup berkonsentrasi untuk menghindari kesalahan
- Kita terkadang lupa menggunakan pemrograman terstruktur secara penuh, perancangan atas-bawah untuk mendapatkan solusi
- Kita kadang buruk dalam mengerjakan sesuatu
- Kita seharusnya dapat membedakan apa yang dikatakan programmer lain atau pelanggan dan apa yang sebenarnya mereka pikirkan
- Kita seharusnya merasa bersalah apabila seseorang harus menguji koding kita
- Pengujian merupakan suatu perizinan terhadap kesalahan

PRINSIP PENGUJIAN

Beberapa prinsip pengujian yang harus diperhatikan (diusulkan oleh Davis):

1. Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan.
2. Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
3. Prinsip Pareto berlaku untuk pengujian PL.
4. Pengujian harus mulai "dari yg kecil" s/d "yang besar".
5. Pengujian yg mendalam tidak mungkin.
6. Paling efektif, pengujian dilakukan oleh pihak ketiga yang independen

SASARAN PENGUJIAN

- Glen Mayers menyatakan sejumlah aturan yang dapat dipandang sebagai sasaran dari pengujian
  - Pengujian perangkat lunak adalah suatu proses pengeksekusian program dengan tujuan menemukan kesalahan (*error*)
  - Pengujian *(Test case)* yang baik adalah yang mempunyai probabilitas yang tinggi untuk menemukan error yang tak diketemukan
  - Pengujian yang sukses adalah pengujian yang dapat menemukan kesalahan (*error*) yang tidak ditemukan sebelumnya

TUJUAN PENGUJIAN

Tujuan yang diinginkan dari pelaksanaan pengujian perangkat lunak adalah :

- Menilai apakah perangkat lunak yang dikembangkan telah memenuhi kebutuhan pemakai.
- Menilai apakah tahap pengembangan perangkat lunak telah sesuai dengan metodologi yang digunakan.
- Membuat dokumentasi hasil pengujian yang menginformasikan kesesuaian perangkat lunak yang diuji dengan spesifikasi yang telah ditentukan.

TEST ABILITAS

Testabilitas Perangkat Lunak adalah seberapa mudah sebuah program komputer dapat diuji. Karena pengujian sangat sulit, perlu diketahui apa yang dapat dilakukan untuk membuatnya menjadi mudah.

Karakteristik PL yang Di Uji

- OPERABILITAS
- OBSERVABILITAS
- KONTROLABILITAS
- DEKOMPOSABILITAS
- KESEDERHANAAN
- STABILITAS
- KEMAMPUAN DIPAHAMI

ATRIBUT PENGUJIAN YANG BAIK

- Memiliki probabilitas yg tinggi menemukan kesalahan.
- Tidak redundan.
- Harusnya ‘jenis terbaik’.
- Tidak boleh terlalu sederhana atau terlalu kompleks

DESAIN TEST CASE

Metode Desain Test case menyediakan pendekatan sistematis untuk uji coba. Dan menyediakan kemungkinan yang cukup tinggi untuk menemukan kesalahan.

2 macam test case :

1. **Pengetahuan tentang fungsi yang spesifik dari produk yang telah dirancang untuk diperlihatkan**, test dapat dilakukan untuk menilai masing-masing fungsi apakah telah berjalan sebagaimana yang diharapkan.
2. **Pengetahuan tentang cara kerja dari produk**, test dapat dilakukan untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya.

PERANCANGAN TEST CASE

Dua macam pendekatan test yaitu :

1. Black Box Testing

Test case ini bertujuan untuk menunjukkan fungsi Perangkat Lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

2. White Box Testing (Structural Testing)

Adalah meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal path (jalur logika) perangkat lunak akan ditest dengan menyediakan test case yang akan mengerjakan kumpulan kondisi dan atau pengulangan secara spesifik. Secara sekilas dapat diambil

WHITE BOX TESTING

1. Basis Path Testing

- Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan.

- Test case yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

- Contoh dari Basis Path Testing :
  - Notasi Diagram Alir
  - Cyclomatic Complexity
  - Graph Metrix

WHITE BOX TESTING (2)

2. Loop Testing

- Loop Sederhana

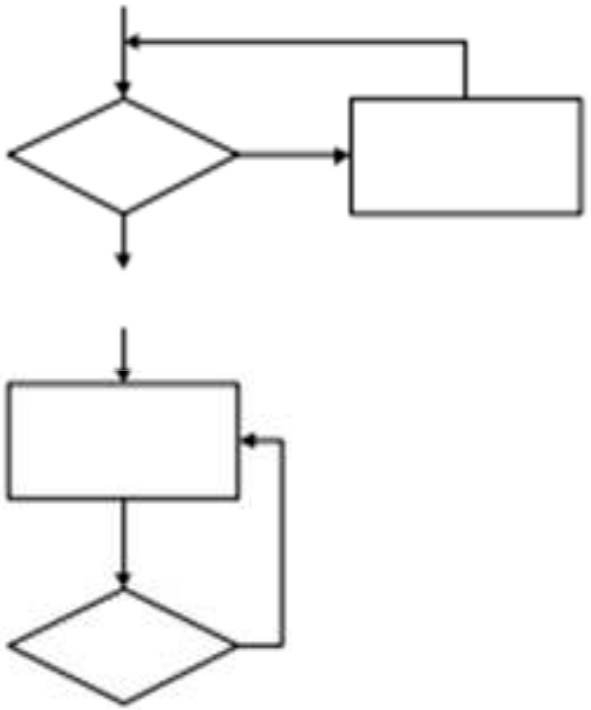
  Pengujian loop sederhana dilakukan dgn mudah, dimana n jumlah maksimum yg diijinkan melewati loop tsb.

  a. 1.Lewati loop secara keseluruhan

  b. 2.Hanya satu yg dapat melewati loop

  c. 3. m dapat melewati loop dimana m< n

- Loop terangkai
- Loop tidak terstruktur



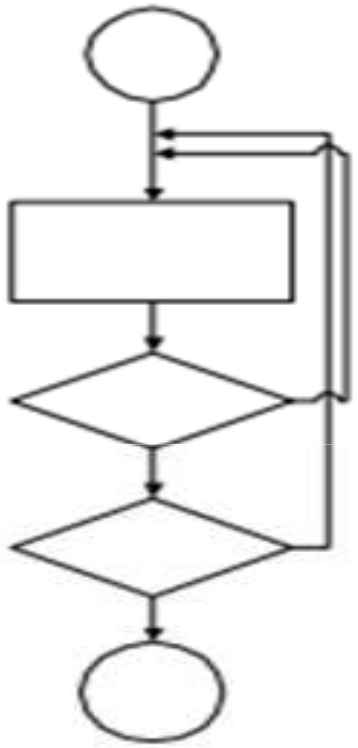
Loop sederhana

WHITE BOX TESTING (3)

- **Loop Tersarang**

  Pengujian loop ini menggunakan pendekatan loop sederhana. Petunjuk pengujian loop tersarang :

  a. Dimulai dari loop paling dalam dan atur semua loop ke nilai minimum.
  b. Kerjakan dgn prinsip loop sederhana untuk loop yg paling dalam sementara tahan loop yg di luar pada parameter terkecil (nilai kounter terkecil)
  c. Kemudian lanjutkan untuk loop yg diatasnya.
  d. Teruskan sampai semua loop selesai di uji.



Loop tersarang

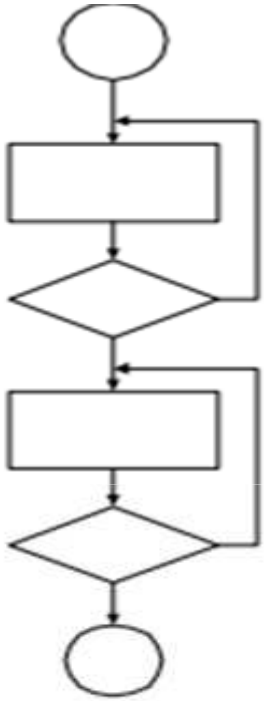
WHITE BOX TESTING (4)

- **Loop Terangkai**

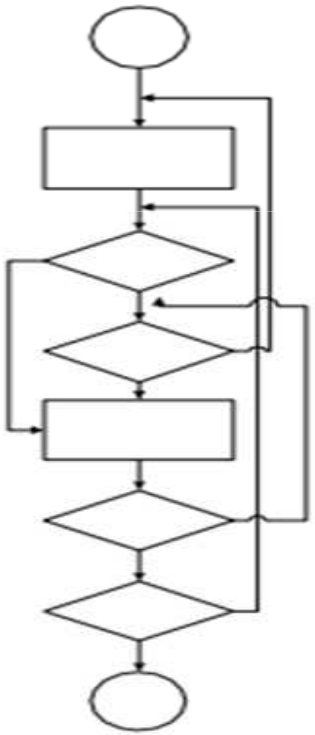
  Pengujian loop ini menggunakan pendekatan loop sederhana. Bila masing-masing loop independen, tetapi bila dua loop dirangkai dan pencacah loop 1 digunakan sebagai harga awal loop 2 maka loop tsb jadi tidak independen, dan di rekomendasikan ke loop tersarang

- **Loop Tidak Terstruktur**

  Kapan saja memungkinkan, loop ini didisain kembali agar mencerminkan penggunaan komsepsi pemrograman tertruktur.



Loop terangkai

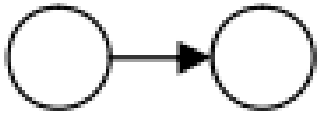


Loop tidak terstruktur

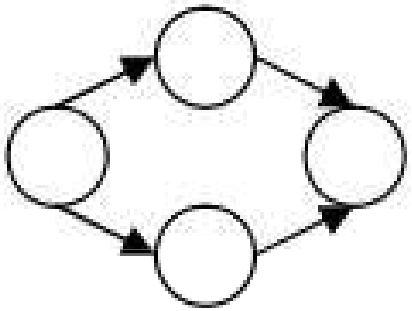
BASIS PATH TESTING

1. Notasi Diagram Alir (Program Flow Graph)

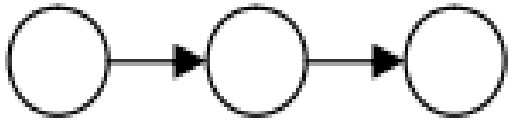
Sequence



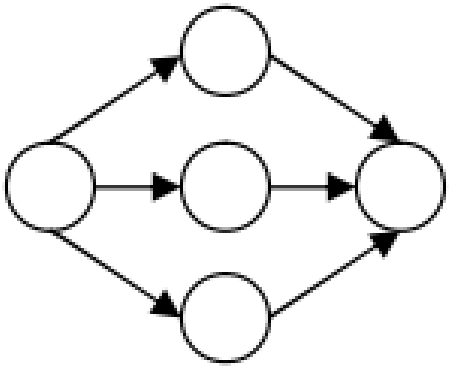
if



while

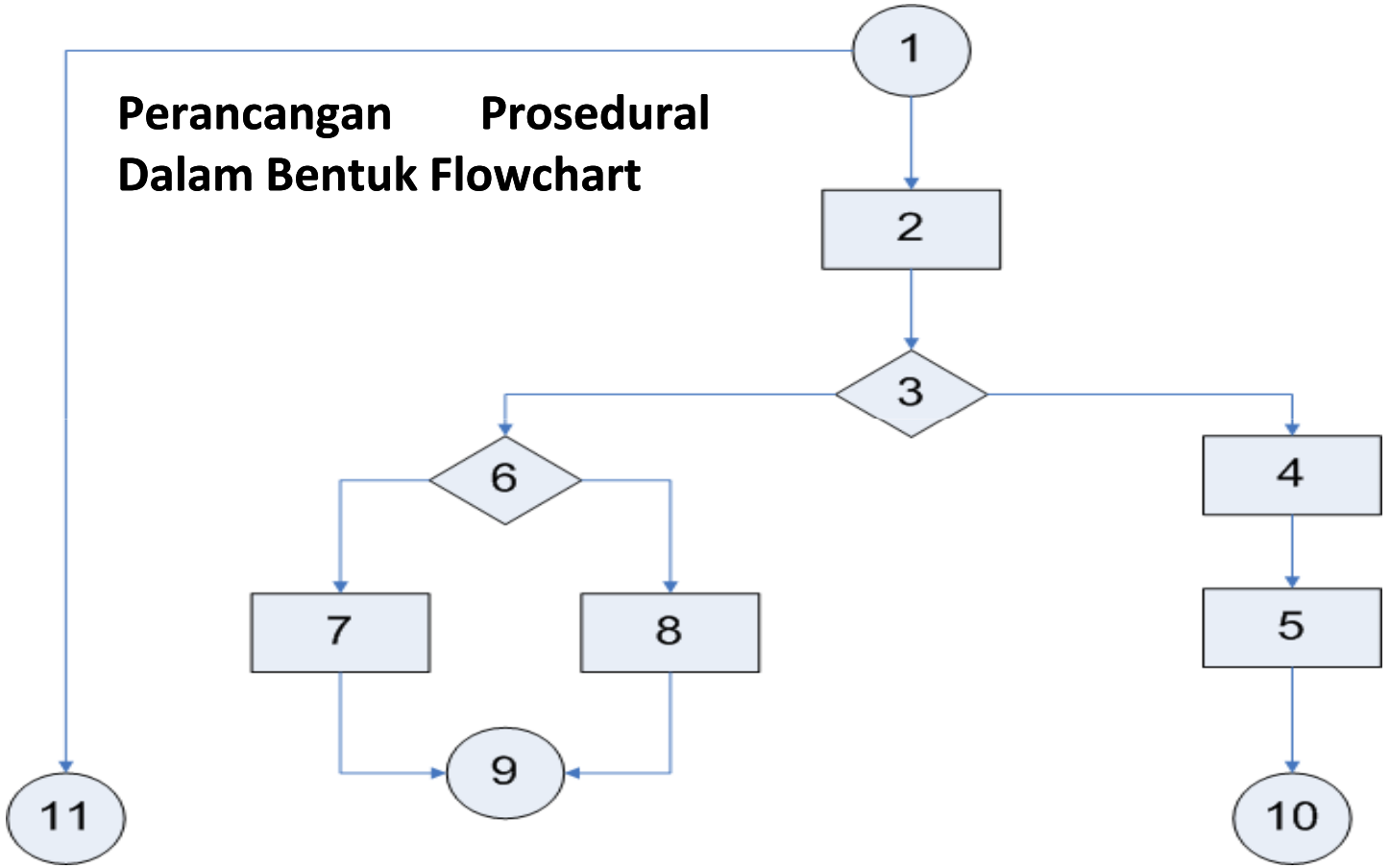


case



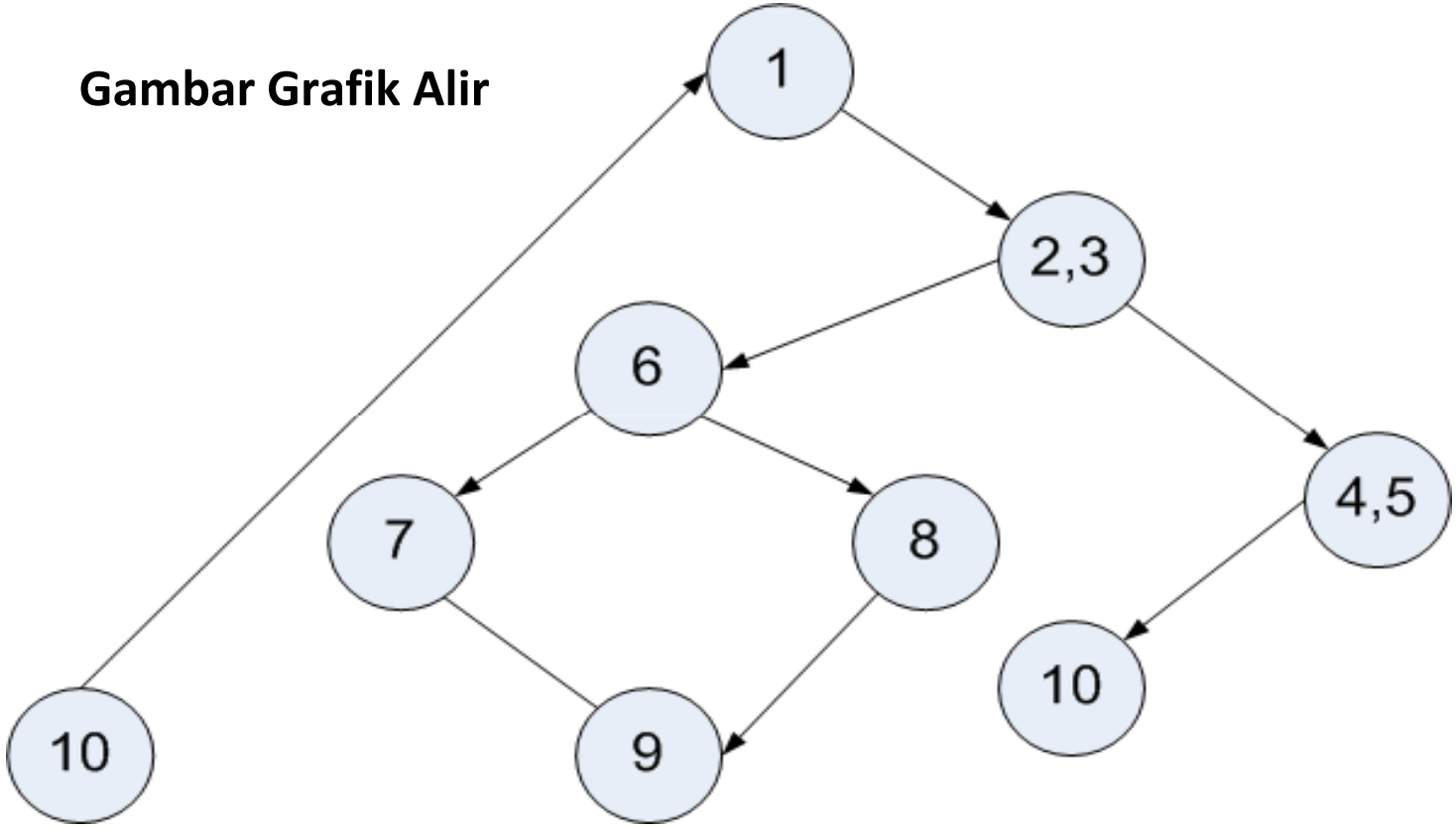
BASIS PATH TESTING (3)

Perancangan Prosedural
Dalam Bentuk Flowchart



BASIS PATH TESTING (4)

Gambar Grafik Alir



- **Lingkaran/node :**

  Menggambarkan satu/lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu node.

- **Tanda panah/edge :**

  Menggambarkan aliran kontrol. Setiap node harus mempunyai tujuan node

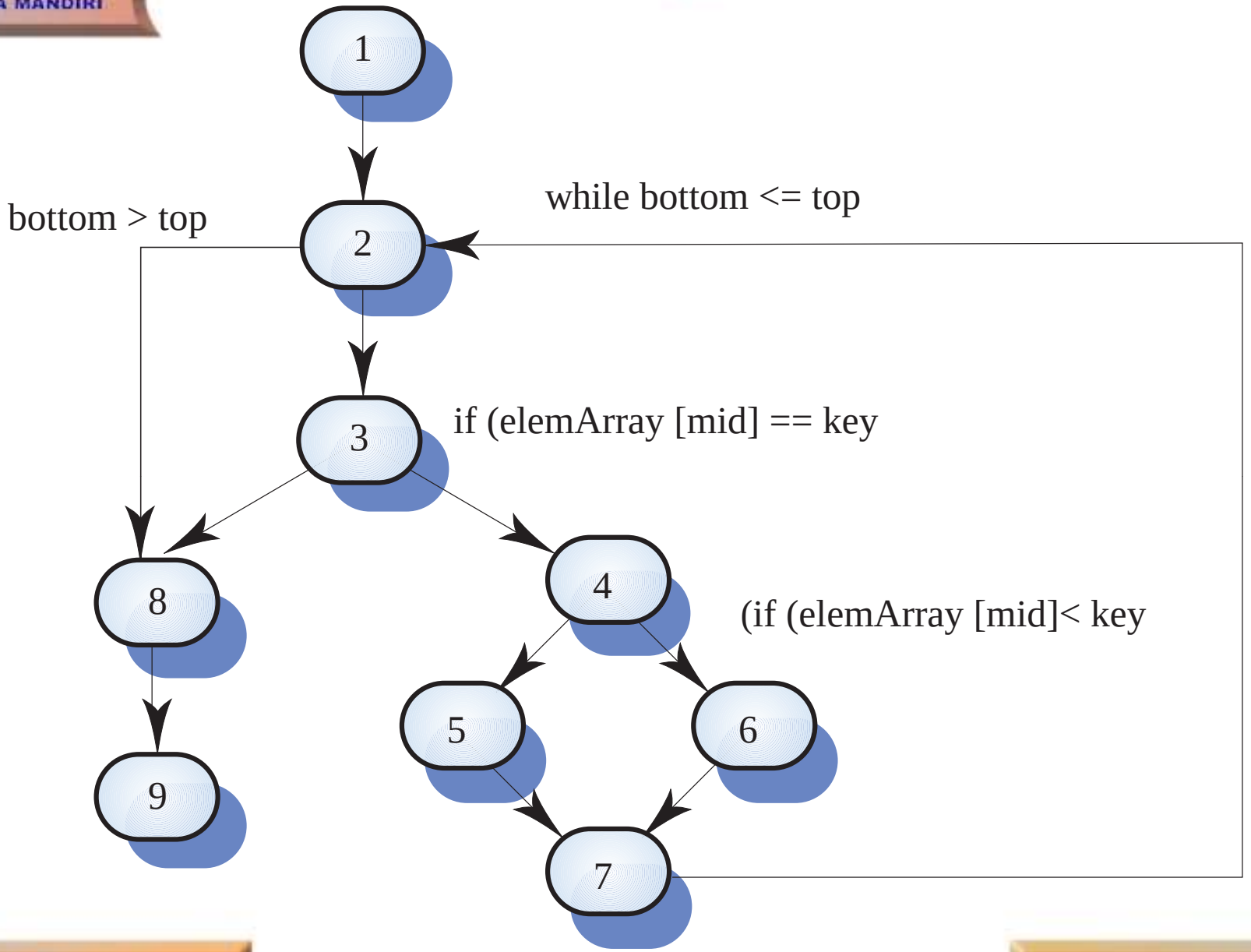
- **Region :**

  Adalah daerah yg dibatasi oleh edge dan node. Termasuk daerah diluar grafik alir.

Program Flow Graphs

- Menggambarkan alur kontrol. Setiap cabang ditunjukkan oleh path yang terpisah dan loop ditunjukkan oleh arrows looping kembali ke loop kondisi node.
- Digunakan sebagai basis untuk menghitung cyclomatic complexity
- Cyclomatic complexity

  = Jumlah edges – Jumlah Node +2
- Cyclomatic complexity menyatakan jumlah test untuk menguji control statements



Independent Paths

- 1, 2, 3, 8, 9
- 1, 2, 3, 4, 6, 7, 2
- 1, 2, 3, 4, 5, 7, 2
- 1, 2, 3, 4, 6, 7, 2, 8, 9
- Test cases harus ditentukan sehingga semua path tersebut tereksekusi.

BLACK BOX TESTING

- Pendekatan pengujian dimana program dianggap sebagai suatu ‘black-box’ (‘kotak hitam’)
- Program test case berbasiskan spesifikasi
- Test planning dapat dimulai sejak awal proses pengembangan sistem
- Merupakan metode pelengkap *White Box Testing*. Berfokus pada kebutuhan fungsional dari PL.
- Memungkinkan perancang untuk memperoleh sekumpulan kondisi2 input yang secara penuh menguji semua kebutuhan fungsional suatu program

BLACK BOX TESTING (2)

- Black Box Testing berusaha menemukan kesalahan yang termasuk kategori di bawah ini
  - Fungsi2 yg hilang atau tidak benar
  - Kesalahan pada antarmuka
  - Kesalahan pada struktur data atau pengaksesan database ekternal
  - Kesalahan pada performance
  - Kesalahan pada inisialisasi dan terminasi

BLACK BOX TESTING (3)

Contoh ***Black Box Testing*** adalah

1. ***Equivalence Partitioning***
2. ***Boundary Value Analysis (BVA)***

   - Melengkapi Equivalence Partitioning, dengan melakukannya dari domain output BVA merupakan pilihan test case yang mengerjakan nilai yang telah ditentukan, dengan teknik perancangan test case
   - Melengkapi test case equivalence partitioning yg fokusnya pada domain input.

Hitung CC

- 1. Flowgraph mempunyai 4 region
- 2. V(G) = 11 edge - 9 node + 2 = 4
- 3. V(G) = 3 predicate node + 1 = 4

- Perlu disiapkan 4 test case untuk masing2 path.
- Buat test case yang akan mengerjakan masing-masing path pada basisset. Data yang dipilih harus tepat sehingga setiap kondisi dari predicate node dikerjakan semua.

Latihan

- Tugas 15 %

- ROCEDURE RATA-RATA
  - INTERFACE RESULT rata, total, input, total.valid
  - INTERFACE RESULT nilai, minim, max
  - TYPE NILAl (1:100) IS SCALAR ARRAY;
  - TYPE rata, total. input, total.valid, max.minim, jumlah IS SCALAR;
  - TYPE I IS INTEGER;
  - I = 1;
  - total. input = total. valid = 0;
  - jumlah = 0;
  - DO WHILE nilai(i) <> -999 .and. total.input < 100
    - tambahkan total.input dengan 1;
    - IF nilai(i) >= minimum .and. nilai(i} <=max;
    - THEN tambahkan total.valid dengan I;
    - jumlah=jumlah + nilai(i);
    - ELSE skip;
    - END IF
    - tambahkan i dengan 1;
  - ENDDO
  - IF total. valid> 0
  - THEN rata =jumlah/total. valid;
  - ELSE rata = -999;
  - ENDIF
  - END

Latihan

- Susun Flowchart,Flowgraph,hitung CC

```
00: int bsearch(int x, const int A[], int N) {
01:   int begin = 0;
02:   int end = N;
03:   while (begin < end) {
04:     int middle = begin + (end - begin) / 2;
05:     if (x <= A[middle])
06:       end = middle;
07:     else
08:       begin = middle + 1;
09:   }
10:   return begin;
11: }
```