Jurnal Muara
Sains, Teknologi, Kedokteran, dan Ilmu Kesehatan
Vol. 1, No. 1, April 2017: hlm 318-327

ISSN 2579-6402 (Versi Cetak)
ISSN-L 2579-6410 (Versi Elektronik)

APLIKASI GRAF DALAM REKAYASA PERANGKAT LUNAK

Tri Sutrisno
Jurusan Teknik Informatika, Universitas Tarumanagara
tris@fti.untar.ac.id

ABSTRAK

Dalam makalah ini akan dibahas terkait peranan graf dalam rekayasa perangkat lunak, khususnya dalam pengujian progam. Pengujian progam yang digunakan adalah pengujian White Boxmenggunakan pengujian jalur dasar (basis path testing). Langkah pertama adalah menggambarkan graf alir (flow graph) berdasarkan algoritma struktur progam, dimana aliran kontrol logika dimodelkan dengan graf berarah dengan simpul menyatakan pernyataan atau kondisi yang dievaluasi dan sisi menyatakan aliran kontrol logika ke pernyataan atau kondisi berikutnya.Kedua, menentukan kompleksitas logis dari struktur progam dengan menentukan banyaknya jalur dasar dalam himpunan basis progam.Ketiga, mendefinisikan kasus uji untuk setiap jalur dasar yang telah ditentukan.Pada bagian akhir makalah ini, diberikan hasil percobaan pengujian jalur dasar yang diaplikasikan pada algoritma suatu progam.

Kata kunci: Graf alir, Jalur dasar (*basis path*), Pengujianan *White Box* dengan jalur dasar.

1. PENDAHULUAN

Adanya kebutuhan manusia dengan segala tuntutannya, sehingga perlu mendesain suatu objek atau proses dalam komputer dengan sekumpulan instruksi sehingga memungkinkan komputer dapat melakukan tugas tertentu, guna membantu menyelesaikan pekerjaa manusia. Hal ini melatarbelakangi pengembangan ataurekayasa perangkat lunak.

Rekayasa perangkat lunak adalah suatu proses dimana kebutuhan pemakai diterjemahkan menjadi produk perangkat lunak. Proses ini mencakup aktivitas penerjemahan kebutuhan pemakai menjadi kebutuhan perangkat lunak, transformasi perangkat lunak menjadi desain, penerapan desain menjadi kode progam, dan uji coba kode progam. Ada dua metode pendekatan sistematik untuk uji coba kode progam yaitu pengujian *white box* dan pengujian *black box.*Pada penelitian ini hanya akan dibahas pengujian *white box*menggunakan pengujian jalur dasar (*basis path testing*).

Pengujian jalur dasar merupakan metode uji coba yang didasarkan pada pengecekan rancangan secara detail, aliran kontrol logika dimodelkan dengan graf berarah dengan simpul menyatakan pernyataan atau kondisi yang dievaluasi dan sisi menyatakan aliran kontrol logika ke pernyataan atau kondisi berikutnya.Graf berarah yang mewakili aliran kontrol logika dari progam disebut graf alir (*flow graph*).Secara luas graf alir digunakan untukmenganalisis perangkat lunak dan telah dipelajari selama bertahun-tahun.

Kosaraju (1973) menyelidiki berbagai struktur control progamuntuk memahami kompleksitas dan keterbatasan komputasi.Selajutnya McCabe (1976) mendiskripsikan ukuran kompleksitas dalam teori graf dapat digunakan untuk mengukur kompleksitas logis dari struktur progam.Berdasarkan hasil penelitian tersebut, maka diperkenalkanlah pengujian jalur dasar (*basis path testing*).

Tujuan penulisan ini adalah mendiskripsikan peranan graf dalam rekayasa perangkat lunak, khususnya dalam hal pengujian progam.

2. METODE PENELITIAN

Metode penelitian yang digunakan dalam memecahkan masalah diatas adalah dengan menggunakan langkah-langkah sebagai berikut :

1. Studi literatur

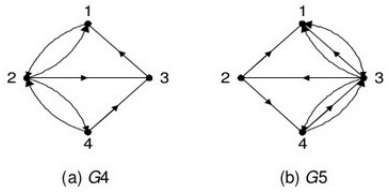
Pada tahap ini dilakukan pengumpulan literatur-literatur yang berhubungan dengan aplikasi graf dalam rekayasa perangkat lunak.

2. Pendalaman materi
   Pada tahap ini dilakukan pendalaman materi dengan membaca beberapa artikel atau tutorial yang berhubungan dengan peranan graf dalam pengujian progam denganmetode pengujian *white box* menggunakan teknik pengujian jalur dasar.
3. Analisis dan perancangan pengujian progam
   Pada tahap ini dilakukan :
   a. Memberikan nomor simpul untuk perintah-perintah dalam rancangan prosedural.
   b. Menggambarkan aliran kontrol logika pada rancangan prosedural kedalam graf alir.
   c. Menentukan kompleksitas logis progam dengan *Cyclomatic Complecity*yaitu matriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu progam.
   d. Menentukan jalur dasar (*basis path*) dalam himpunan basis progam.
   e. Mendefinisikan kasus-kasus uji untuk setiap jalur dasar yang telah ditentukan.
4. Implementasi dan evaluasi progam
   Pada tahap ini akan dilakukan ujicoba yang akan mengeksekusi setiap jalur bebas dalam himpunan basis progam serta mengevaluasi apakah progam yang dihasilkan sudah berjalan dengan baik.

3. HASIL DAN PEMBAHASAN

Graf (*Graph*)

Graf adalah suatu himpunan tidak kosong dari simpul-simpul (*Vertice* atau *node*) dan sisi (*edges* atau *arcs*) yang menghubungkan simpul tersebut, ditulis dengan notasi$G=(V,E)$.Sisi pada graf dapat mempunyai orientasi arah.Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, $(u, v) = (v,u)$ adalah sisi yang sama. Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah. Pada graf berarah, (u, v) dan (v,u) menyatakan dua buah sisi yang berbeda, dengan kata lain $(u, v) \neq (v,u)$. Untuk sisi (u, v), simpul u dinamakan simpul asal (*unitial vertex*) dan simpul v dinamakan simpul terminal (*terminal vertex*).



Gambar 1. Graf berarah

Sumber: (Munir, 2014)

Graf yang dapat digambarkan pada bidang datar dengan sisi-sisi yang tidakberpotongan disebut graf planar.Representasi graf planar yang digambarkan dengan sisi-sisi yang tidak saling berpotongan disebut graf bidang.Sisi-sisi pada graf bidang membagi bidang datar menjadi beberapa wilayah (*region*) atau muka (*face*). Banyaknya wilayah pada graf bidang dapat dihitung dengan rumus Euler yaitu $n - e + f = 2$, dengan n adalah banyaknya simpul, e adalah banyaknya sisi dan f adalah banyaknya wilayah.

Salah satu istilah yang berkaitan dengan graf adalah jalur (*path*).Jalur (*path*) adalah suatu jalan yang semua sisi dalam barisan tersebut berbeda.

ISSN 2579-6402 (Versi Cetak)
ISSN-L 2579-6410 (Versi Elektronik)

Rekayasa perangkat lunak

Rekayasa perangkat lunak adalah satu bidang profesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen operasi pengembangan perangkat lunak dan manajemen kualitas.Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi.Perangkat lunak dapat berupa progam atau prosedur.Progam adalah kumpulan perintah yang dimengerti oleh komputer yang bila dieksekusi menjalankan fungsi tertentu sedangakan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi. Perangkat lunak dibuat supaya bisa digunakan (oleh pemakai) untuk membantu menyelesaikan masalah atau pekerjaan, perangkat lunak yang dibuat harus memenuhi apa yang diinginkan oleh pemakai. Oleh karena itu perlu dilakukan pengujian perangkat lunak.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. Salah satu metode yang dapat digunakan adalah metode pengujian *white box*.

Pengujian *white box*

Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan rancangan secara detail, menggunakan struktur kontrol dari rancangan progam secara procedural maka dapat membagi pengujian ke beberapa kasus uji.Penentuan kasus uji disesuaikan dengan struktur sistem, pengetahuan mengenai progam digunakan untuk mengidentifikasi kasus uji tambahan.

Tujuan penggunaan *white box* untuk menguji semua statement progam. Penggunaan metode pengujian *white box* dilakukan untuk :

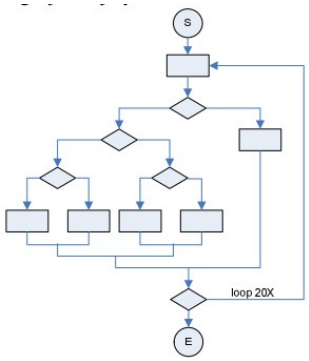
1. Memberikan jaminan bahwa semua jalurdasar (*basis path*) suatu modul digunakan minimal satu kali.
2. Menggunakan semua keputusan logis untuk semua kondisi *true* atau *false.*
3. Mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4. Menggunakan struktur data internal untuk menjamin validitas jalur keputusan.

Secara sekilas dapat diambil kesimpulan pengujian *white box* merupakan petunjuk untuk mendapatkan program yang benar secara 100%.Salah satu teknik pengujian *white box* adalah pengujian jalur dasar(*basis path testing*).

Pengujian jalur dasar (*Basis path testing*)

Pengujian jalur dasar merupakan salah satu teknik pengujian *white box*yang pertama kali diperkenalkan olehTom McCabe pada tahun 1976.Metode ini memungkinkan penguji dapat mengukur kompleksitas logis dari rancangan prosedur dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.Kasus uji dihasilkan untuk melakukan pengujian sekumpulan basis yang dijamin untuk mengeksekusi setiap perintah dalam progam, sedikitnya satu kali selama ujicoba.

Pengujian jalur dasar digunakan untuk efisiensi dan efektivitas pengujian *white box*, karena didalam pengujian *white box*tidak mungkin seluruh kemungkinan jalur dieksekusi. Sebagai ilustrasi, jika terdapat struktur progam pada gambar 2 dengan 5 kemungkinan jalur untuk setiap *loop,* maka akan terdapat 5^{20} kemungkinan jalur yang harus di uji. Jika setiap jalur membutuhkan waktu pengujian selama 1 milidetik maka total waktu yang dibutuhkan untuk pengujian sebanyak jalur tersebut adalah 3.170 tahun. Hal ini sangat tidak mungkin.Oleh karena itu, dibutuhkan sebuah teknik pengujian yang bisa mempresentasikan jalur-jalur pengujian yang ada dengan menggunakan jalur-jalur dasar sebagai jalur ujinya.



Gambar 2. Flow chart struktur progam

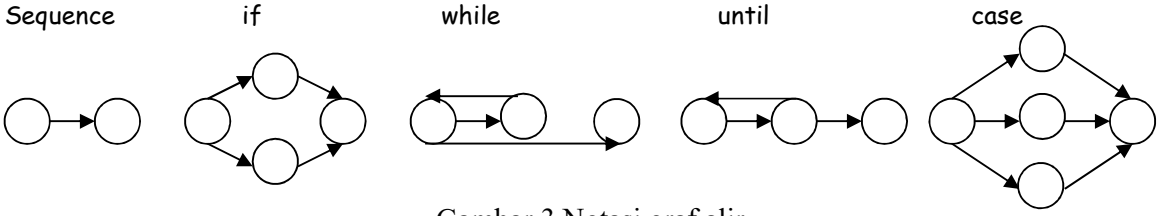
Sumber: (Lundqvist, 2005)

Langkah-langkah untuk melakukan pengujian jalur dasar (*basis path testing*) adalah :

1. Menggambarkan graf alir (*flow graph*) berdasarkan algoritma perancangan prosedur.
2. Menentukan kompleksitas siklomatis(*Cyclomatic Complexity*).
3. Menentukan jalur-jalur dasar sesuai dengan jumlah dari *Cyclomatic Complexity.*
4. Mendefinisikan kasus-kasus uji untuk setiap jalur dasar yang telah ditentukan.

Graf alir

Graf alir adalah sebuah notasi sederhana yang mempresentasikan aliran kontrol logika dari sebuah struktur progam.Gambar 3 menjelaskan notasi-notasi yang digunakan untuk mempresentasikan jenis-jenis kontrol logika yang terdapat pada suatu progam.



Gambar 3.Notasi graf alir

Sumber: (McCabe, 1976)

Notasi yang digunakan untuk menggambarkan jalur eksekusi dalam graf alir, digunakan notasi simpul untuk mempresentasikan satu atau lebih perintah prosedur atau urutan dari simbol proses dan simbol keputusan, sedangkan sisi menggambarkan aliran kontrol logika.Daerah yang dibatasi oleh sisi dan simpul disebut wilayah (*region, R*).Simpul yang memuat keputusan disebut simpul predikat (*predicate node, P*) yaitu simpul yang berisi sebuah kondisi dan ditandai dengan dua atau lebih sisi yang berasal darinya.

Beberapa panduan penting dalam penggambaran sebuah graf alir adalah :

1. Serangkaian perintah prosedur dan sebuah pernyataan kondisi bisa direpresentasikan dalam sebuah simpul.
2. Sebuah sisi harus berakhir pada sebuah simpul, walaupun simpul tersebut tidak mempresentasikan sebuah pernyataan apapun.
3. Penghitungan banyaknya wilayah melibatkan sebuah wilayah yang berada di luar graf alir.
4. Kondisi majemuk (*compound decision*) adalah kondisi yang memuat operator Boolean (AND, OR, NAND dan NOR) tidak boleh direpresentasikan dalam sebuah simpul.

ISSN 2579-6402 (Versi Cetak)
ISSN-L 2579-6410 (Versi Elektronik)

Kompleksitas siklomatis(*Cyclomatic complexity*)

Kompleksitas siklomatis(*Cyclomatic complexity, V(G)*)adalah matriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu progam. Bila matriks ini digunakan dalam konteks metode pengujian basis path, maka hasil dari kompleksitas siklomatis digunakan untuk menentukan banyaknya jalur dasar (*basis path*) dalam himpunan basis suatu progam dan memberi batas atas untuk sejumlah uji coba yang harus dilakukan untuk memastikan bahwa seluruh perintah telah dieksekusi sedikitnya satu kali. Secara matematis, *V(G)* dapat ditentukan dari salah satu cara beberapa pendekatan berikut.

1. *V(G)* = banyaknya wilayah (*R*)
2. *V(G)* = *E* – *N* + 2
3. *V(G)* = *P* + 2

Jalur dasar (*Basis path*)

Jalur dasar (*basis path*) adalah sebuah jalur pada progam yang memuat paling sedikit sebuah perintah procedur.Dalam konteks graf alir, maka sebuah jalur disebut sebagai jalur dasar jika jalur tersebut memiliki paling tidak satu buah simpul yang belum pernah dilalui oleh jalur yang sudah didefinisikan sebelumnya.Banyaknya jalur dasar yang teridentifikasi harus sesuai dengan nilai dari *Cyclomatic Complexity*.

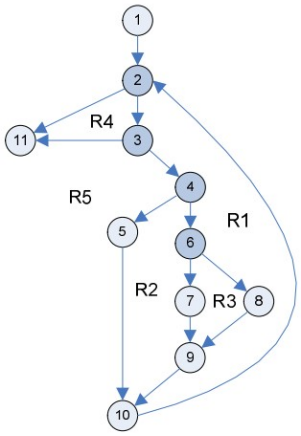
Penentuan kasus uji

Kasus uji adalah suatu kondisi yang dibuat untuk menjamin bahwa sebuah jalur dasar dapat diuji. Sebuah kasus uji harus memuat kondisi atau data awal, prosedur uji dan hasil yang diharapkan dari sebuah jalur dasar yang akan diuji. Sebuah jalur dasar mungkin tidak bisa diuji secara mandiri, tetapi harus menjadi bagian dari kasus uji jalur dasar yang lain.

Aplikasi 1

Uji coba untuk mengaplikasikan teknik pengujian jalur dasar (*basis path testing*) akan digunakan algoritma dari sebuah prosedur pencarian biner (*binary search*).

Pertama dilakukan identifikasi simpul pada algoritma, diperlihatkan pada gambar 5.



Gambar 5.Identifkasi simpul Gambar 6. Graf alir

Sumber:(Kurniawan, 2007)Sumber : (Kurniawan, 2007)

Selanjutnya, menggambarkan graf alir berdasarkan hasil identifikasi simpul, diperlihatkan pada gambar 6.Berdasarkan graf alir tersebut didapatkan banyaknya wilayah adalah 5 yaitu R1, R2, R3, R4, dan R5. Banyaknya simpul predikat (*predicate node*) adalah 4 yaitu 2, 3, 4 dan 6. Banyaknya simpul adalah 11 dan sisi sebanyak 14, sehingga nilai kompleksitas logisnya adalah

1. *V(G)* = banyaknya wilayah (*R*) = 5
2. *V(G)* = *E* – *N* + 2 = 14 – 11 + 2 = 5
3. *V(G)* = *P* + 1 = 4 + 1 = 5

Berdasarkan nilai *V(G)* maka banyaknya jalur dasar yang harus di idenfikasi adalah sebanyak 5 buah, yaitu :

1. Jalur dasar 1: *1-2-11*
2. Jalur dasar 2: 1-2-*3*-11
3. Jalur dasar 3: 1-2-3-*4-5-10*-2-…
4. Jalur dasar 4: 1-2-3-4-*6-7-9*-10-2-…
5. Jalur dasar 5: 1-2-3-4-6-*8*-9-10-2-…

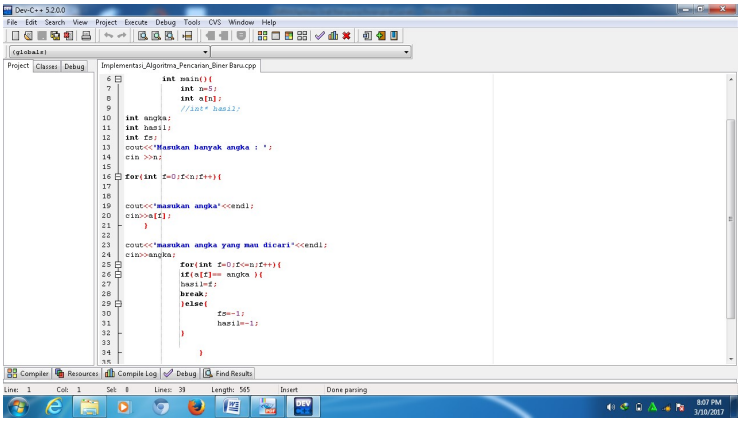
Nomor simpul yang dituliskan tebal dan miring pada jalur dasar meunjukkan simpul-simpul yang belum pernah dilewati oleh jalur dasar sebelumnya.Sehingga, dapat dilihat bahwa setiap simpul paling tidak sudah terlewati minimal sekali.

Selanjutnya, mendefinisikan kasus-kasus uji berdasarkan jalur dasar yang ada:

1. Kasus uji jalur dasar 1: 1-2-11
   tabInt = array integer kosong, *n* = 0.
   X = nilai integer sembarang.
   Expected result = idX bernilai -1.
2. Kasus uji jalur dasar 2: 1-2-3-11
   tabInt = array integer berisi *n* data terurut menaik.
   X = bukan anggota dari tabInt.
   Expected result = idX bernilai -1.
3. Kasus uji jalur dasar 3: 1-2-3-4-5-10-2
   tabInt = array integer berisi *n* data terurut menaik.
   X = anggota dari tabInt pada urutan pertengahan ((*n*+1)/2).
   Expected result = idX bernilai ((*n*+1)/2) – 1.
4. Kasus uji jalur dasar 4: 1-2-3-4-6-7-9-10-2-…
   tabInt = array integer berisi *n* data terurut menaik.
   X = bisa anggota dari tabInt atau bukan, *X* < nilai tabInt pada urutan pertengahan ((*n*+1)/2).
   Expected result = idX bernilai -1 jika *X* bukan anggota tabInt; idX bernilai antara 0 sampai ((*n*+1)/2) – 1 jika *X* anggota tabInt.
   Keterangan : jalur ini tidak bisa diuji mandiri, harus menjadi bagian dari pengujian jalur dasar 3.
5. Kasus uji jalur dasar 5: 1-2-3-4-6-8-9-10-2-…
   tabInt = array integer berisi *n* data terurut menaik.
   X = bisa anggota dari tabInt atau bukan, *X* > nilai tabInt pada urutan pertengahan ((*n*+1)/2).
   Expected result = idX bernilai -1 jika *X* bukan anggota tabInt; idX bernilai antara ((*n*+1)/2) – 1 sampai (*n* – 1) jika *X* anggota tabInt.
   Keterangan : jalur ini tidak bisa diuji mandiri, harus menjadi bagian dari pengujian jalur dasar 3.

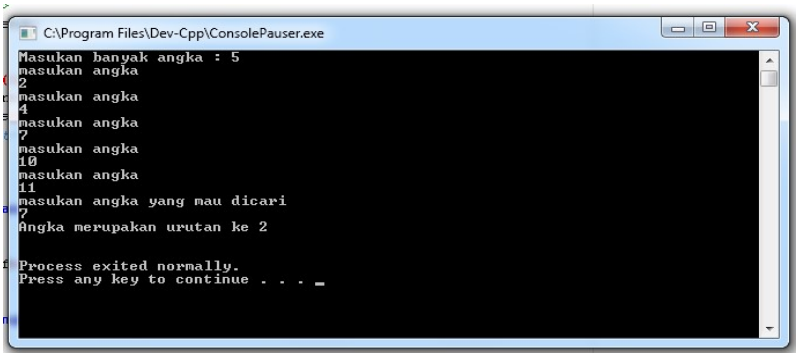
Contoh implementasi algoritma pencarian biner dalam bahasa *C* diperlihatkan pada gambar berikut.

ISSN 2579-6402 (Versi Cetak)
ISSN-L 2579-6410 (Versi Elektronik)



Gambar 7.Pencarian biner

Sumber: Hasil implementasi



Gambar 8.Hasil progam

Sumber : Hasil implementasi

Pengujian jalur dasar bisa dilakukan dengan menggunakan nilai riil untuk tabInt dan *X* yang ditentukan berdasarkan aturan umum yang ada pada kasus-kasus uji yang telah didefinisikan.Tabel berikut memperlihatkan contoh nilai-nilai tabInt dan *X* yang bisa digunakan dalam pengujian.

Tabel 1. Hasil uji jalur dasar

Jalur	***tabInt***	***N***	***X***	***Exp. Result, idx***
1	{}	0	12	-1
2	{2,4,7,10,11}	5	1	-1
3	{2,4,7,10,11}	5	7	2
4	{2,4,7,10,11}	5	4	1
5	{2,4,7,10,11}	5	10	3

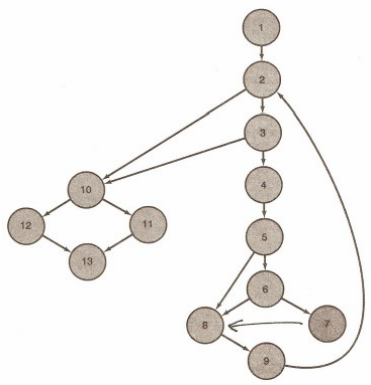
Sumber: Hasil implementasi

Aplikasi 2

Uji coba berbasis alur yang ke 2 coba diaplikasikan pada algoritma mencari nilai rata-rata.

```
PROCEDURE average;
    * This procedure computes the average of 100 or fewer
      numbers that lie between bounding values; it also computes the
      sum and the total number valid.
    INTERFACE RETURNS average, total.input, total.valid;
    INTERFACE ACCEPTS value, minimum, maximum;
    TYPE value[1:100] IS SCALAR ARRAY;
    TYPE average, total.input, total.valid;
      minimum, maximum, sum IS SCALAR;
    TYPE i IS INTEGER;
1   i = 1;
    total.input = total.valid = 0;
    sum = 0;
    DO WHILE value[ i ] <> -999 and total.input < 100
4     increment total.input by 1;
      IF value[ i ] > = minimum AND value[ i ] < = maximum
5       THEN increment total.valid by 1;
7               sum = sum + value[ i ]
        ELSE skip
8     ENDIF
      increment i by 1;
9   ENDDO
    IF total.valid > 0
11    THEN average = sum / total.valid;
12    ELSE average = -999;
13  ENDIF
END average
```

Gambar 9.Identifikasi simpul



Gambar 10. Graf alir

Sumber: (Ayuliana, 2009)Sumber: (Ayuliana, 2009)

Berdasarkan graf alir tersebut didapatkan banyaknya wilayah adalah 6. Banyaknya simpul predikat (*predicate node*) adalah 5 yaitu 2, 3, 4 dan 6. Banyaknya simpul adalah 13 dan sisi sebanyak 17, sehingga nilai kompleksitas logisnya adalah

1. $V(G)$ = banyaknya wilayah (R) = 6
2. $V(G) = E - N + 2 = 17 - 13 + 2 = 6$
3. $V(G) = P + 1 = 5 + 1 = 6$

Berdasarkan nilai $V(G)$ maka banyaknya jalur dasar yang harus di idenfikasi adalah sebanyak 6 buah, yaitu :

1. Jalur dasar 1: 1 – 2 – 10 – 11 – 13
2. Jalur dasar 2: 1 – 2 – 10 – 12 – 13
3. Jalur dasar 3:1 – 2 – 3 – 10 – 11 – 13
4. Jalur dasar 4:1 – 2 – 3 – 4 – 5 – 8 – 9 – 2 – …
5. Jalur dasar 5:1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 2 – …
6. Jalur dasar 6:1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 2 – …

Tanda (…) mengidentifikasi perulangan yang diperbolehkan. Bila pengujian dapat dieksekusi pada jalur-jalur tersebut, maka setiap perintah pada progam tersebut akan dieksekusi minimal sekali dan setiap kondisi telah dieksekusi pada sisi true dan false-nya.

Selanjutnya, mendefinisikan kasus-kasus uji berdasarkan jalur dasar yang ada:

1. Kasus uji jalur dasar 1:

   nilai (k) = Input yang valid, dimana $k < i$ yang didefinisikan

   nilai (i) = -999 dimana $2 \leq i \leq 100$

   Hasil diharapkan : Nilai rata-rata yang benar berdasarkan nilai k dan total yang tepat

   Catatan : Tidak dapat diujicoba secara terpisah, harus diujikan sebagai bagian dari uji jalur 4, 5 dan 6.
2. Kasus uji jalur dasar 2:

   nilai (1) = -999

   Hasil diharapkan : Nilai rata-rata = -999; total lainnya berisi nilai awal
3. Kasus uji jalur dasar 3 :

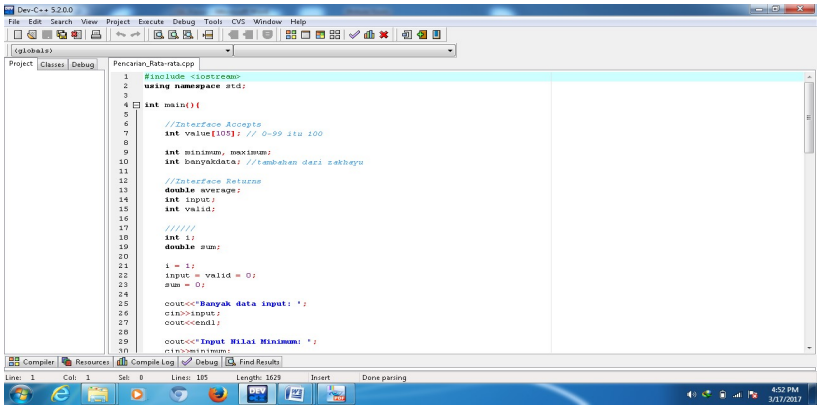
   Berusaha untuk memproses nilai 101 atau lebih.100 nilai pertama harus valid

ISSN 2579-6402 (Versi Cetak)
ISSN-L 2579-6410 (Versi Elektronik)

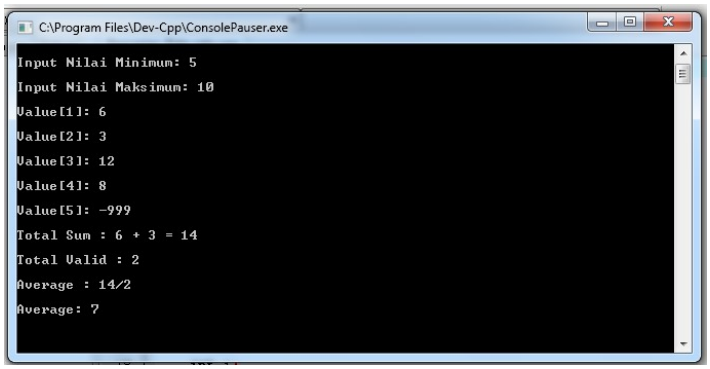
Hasil diharapkan : Sama dengan uji kasus Path 1

4. Kasus uji jalur dasar 4:
   nilai (i) : Input yang valid, dimana $i < 100$
   nilai (k)< minimum dimana $k < i$
   Hasil diharapkan : Nilai rata-rata yang benar berdasarkan nilai k dan total yang tepat
5. Kasus uji jalur dasar 5:
   nilai (i) = Input yang valid, dimana $i < 100$
   nilai (k)> maksimum dimana $k \leq i$
   Hasil diharapkan : Nilai rata-rata yang benar berdasarkan nilai n dan total yang tepat
6. Kasus uji jalur dasar 6:
   nilai (i) = Input yang valid, dimana $i < 100$
   Hasildiharapkan : Nilai rata-rata yang benar berdasarkan nilai n dan total yang tepat.

Contoh implementasi algoritma pencarian nilai rata-rata dalam bahasa *C* diperlihatkan pada gambar berikut.



Gambar 7. Pencarian nilai rata-rata
Sumber: Hasil implementasi



Gambar 8. Hasil progam
Sumber : Hasil implementasi

Pengujian jalur dasar bisa dilakukan dengan menggunakan nilai riil untuk *value* dan nilai minimum serta nilai maksimum yang ditentukan berdasarkan aturan umum yang ada pada kasus-kasus uji yang telah didefinisikan.

Tabel 2. Hasil uji jalur dasar

Jalur	*N. min*	*N. max*	*Value*	*Total valid*	*Total*	*Rata-rata*
1	5	10	N_1=6, N_2=3, N_3=12, N_4=8, N_5=-999	2	14	7
2	5	10	N_1=-999	0	0	-999
3	5	10	N_1=6, N_2=10, N_3=8, …, N_{100}=-999	100	800	8
4	5	10	N_1=6, N_2=3, N_3=10, N_4=8, N_5=-999	3	24	8
5	5	10	N_1=6, N_2=12, N_3=10, N_4=8, N_5=-999	3	24	8
6	5	10	N_1=6, N_2=8, N_3=10, N_4=8, N_5=-999	4	32	8

Sumber: Hasil implementasi

4. KESIMPULAN DAN SARAN

Makalah ini mendiskripsikan peranan graf dalam rekayasa perangkat lunak, khususnya dalam hal pengujian progam. Teknik pengujian progam yang digunakan adalah metode *white box* dengan menggunakan pengujian jalur dasar (*basis path testing*), dari pembahasan tersebut dapat diambil kesimpulan :

1. Graf berarah digunakan untuk memodelkan aliran kontrol logika yang digunakan dalam suatu bahasa pemrogaman, simpul menyatakan pernyataan atau kondisi yang dievaluasi dan busur atau sisi menyatakan aliran kontrol logika ke pernyataan atau kondisi berikutnya.
2. Ukuran kompleksitas dalam graf dapat digunakan untuk mengukur kompleksitas logis dari struktur progam. Graf alir yang mempresentasikan aliran kontrol progam dihitung kompleksitasnya menggunakan rumus Euler, sehingga didapatbanyaknya jalur dasar dalam himpunan basis progam serta dapat menentukan batas atas untuk sejumlah uji coba yang harus dilakukan untuk memastikan bahwa seluruh perintah telah dieksekusi sedikitnya satu kali.

Penulisan berikutnya, Perlu mendiskripsikan aplikasi graf dalam pengujian *Black Box.*

5. REFERENSI

Ayuliana, 2009. Testing dan Implementasi.http://ayulianan_st.staff.gunadarma.ac.id, 12 Februari 2017.

Gold, R. (2010). "Control Flow Graphs and Code Coverage". *Int.J. Appl. Math. Comput. Sci*., 20(4), 739-749.

Kurniawan, T.A. (2007). "Pengujian Struktur Progam dengan Pengujian Jalur Dasar (Basis Path Testing) : Teori dan Aplikasi". *Jurnal EECCIS*., 1(1), 29-32.

Lundqvist, I.K. *Introduction to Computers and Progamming,* http://ocw.mit.edu/NR/rdonlyers/Aeronautics-and-Astronautics/16-01Fall-2005-Spring-2006/3AFC37EE-19C9-4D31-947B-23F51919DE20/0/13_testing.pdf, 7 Maret 2017.

McCabe, T. (1976) "A Complexity Measure". *IEEE Transactions on Software Engineering* **SE-2**(4): 308-320.

Munir, R. (2014). Matematika Diskrit (edisi ke 5). Progam Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Preesman, R.S. (2001). Sofware Engineering: A Practitioner's Approach. 4th, Mc-Graw Hill.

Sommerville, I. (1996). Sofware Engineering. 4th Addison-Wesley.