**SOFTWARE TESTING**

**PENGUJIAN APLIKASI MENGGUNAKAN BLACK BOX DAN WHITE BOX TESTING**

**(Studi Kasus: Program Array)**

****

**Disusun Oleh:**

**Alfiatus Salatin (065115221)**

**Muhammad Rinaldi (065115223)**

**Fredi Utomo (065115232)**

**Krisnawati Ayu (065115239)**

**Agus Budiyanto (065115278)**

**PROGRAM STUDI ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PAKUAN**

**2017**

**BAB I**

**PENDAHULUAN**

**Latar Belakang**

Program Array atau larik adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama. Array merupakan konsep yang penting dalam pemrograman, karena array memungkinkan untuk menyimpan data maupun referensi objek dalam jumlah banyak dan terindeks.

Array menggunakan indeks integer untuk menentukan urutan elemen-elemennya diaman elemen pertamanya dimulai dari indeks 0, elemen kedua memiliki indeks 1, dan seterusnya.

Array atau larik di definisikan sebagai pemesanan alokasi memory berurutan. Semua elemen array bertipe sama. Array cocok untuk organisasi kumpulan data homogen yang ukuran atau jumlah elemen maksimumnya telah diketahui dari awal.

**Maksud dan Tujuan**

* 1. Maksud

Berdasarkan permasalahan yang dibahas, maksud dari pembahasan ini adalah untuk menganalisa program Array menggunakan white box dan black box

* 1. Tujuan

Tujuan dari penelitian ini adalah membangun sistem pengujian program array dengan mengkombinasikan metode pengujian black box dan white box.

**BAB II**

**TINJAUAN PUSTAKA**

**2.1 Pengertian Software**

Pengujian *Software* sangat diperlukan untuk memastikan software/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji *software* harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan *software* seperti halnya analisis, desain, dan pengkodean. (Shi,2010)

Pengujian software haruslah dilakukan dalam proses rekayasa perangkat lunak atau *software engineering*. Sejumlah strategi pengujian *software* telah diusulkan dalam literiur. Semuanya menyediakan template untuk pengujian bagi pembuat *software*. Dalam hal ini, semuanya harus memiliki karakteristik umum berupa (Bhat and Quadri, 2015):

1. Testing dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.
2. Teknik testing yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Testing diadakan oleh pembuat/pengembang *software* dan untuk proyek yang besar oleh group testing yang independent.
4. *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus dikomodasikan pada setiap strategi testing.

Pengujian software adalah satu elemen dari sebuah topik yang lebih luas yang sering diartikan sebagai Verifikasi dan Validasi (V&V)

* Verifikasi: menunjuk kepada kumpulan aktifitas yang memastikan bahwa software telah mengimplementasi sebuah fungsi spesifik.
* Validasi: menunjuk kepada sebuah kkumpulan berbeda dari aktivitas yang memastikan bahwa software yang telah dibangun dapat ditelusuri terhadap kebutuhan customer.

Definisi V&V meliputi banyak aktifitas SQA (*software quality assurance*) termasuk review teknis formal, kualitas dan audit konfigurasi, monitor performance. Terdapat beberapa tipe yang berbeda dalam pengujian software yang meliputi studi kelayakan dan simulasi. (Bhat and Quardi,2015):

1. Metode *software engineering* menyediakan dasar dari mutu yang mana yang akan dipakai.
2. Metode *Analysis, design and Constuction* berupa tindakan untuk meningkatkan kualitas dengan menyediakan teknik yang seragam sesuai dengan keinginan.
3. Metode *Formal Technical Reviews* menolong untuk memastikan kualitas kerja produk merupakan hasil konsekuensi dari setiap langkah *software engineering*.
4. Metode *Measurement* diberlakukan pada setiap elemen dari konfigurasi software.
5. Metode *Standards and Procedures* membantu untuk memastikan keseragaman dan formalitas dari SQA untuk menguatkan dasar “filosofi kualitas total”.
6. Metode *Testing* menyediakan cara terakhir dari tingkat kualitas mana yang dapat dicapai dan dengan praktis dapat mengetahui letak error.

David menyerahkan satu set prinsip pengujian:

1. Semua test harus dapat dilacak ke kebutuhan pelanggan.
2. Test harus direncanakan dengan baik debelum pengujian mulai.
3. Prinsip Pareto berlaku untuk pengujian
4. 80% dari semua kesalahan yang terungkap selama pengujian akan mudah dapat dilacak dari 20% semua model program.
5. Pengujian seharusnya mulai “dari yang kecil” dan pengujian perkembangan kearah “yang besar”.
6. Pengujian menyeluruh adalah tidak mungkin. Paling efektif, pengujian harus diselenggarakan oleh suatu pihak ketiga mandiri.

Langkah-langkah pengujian software ada 4 yaitu:

1. Unit testing-testing per unit yaitu mencoba alur yang spesifik pada struktur modul kontrol untuk memastikan pelengkapan secara penuh dan pendeteksian error secara maksimum.
2. Integration testing – testing per penggabungan unit yaitu pengalamatan dari isu-isu yang diasosiasikan dengan masalah ganda pada verifikasi dan kontruksi program.
3. High-order test yaitu terjadi ketika software telah selesai diintegrasikan atau dibangun menjadi satu-tidak terpisah-pisah.
4. Validation test yaitu menyediakan jaminan akhir bahwa software memenuhi semua kebutuhan fungsional, kepribadian dan performa.

Tom Gilb menyatakan bahwa prosedur yang harus digunakan jika ingin mengimplementasikan strategi testing software yang sukses (Bhat and Quadri, 2015):

1. Menetapkan seluruh kebutuhan produk *software* dalam perhitungan sebelum memulai testing.
2. Status obyek testing harus jelas.
3. Memahami pengguna *software* dan mengembangkan sebuah profil untuk setiap kategori user.
4. Mengembangkan rencana testing yang menekankan pada “*rapid cycle testing*”.
5. Membangun *software* yang sempurna yang didesain untuk menguji dirinya sendiri.
6. Menggunakan tinjauan ulang yang formal sebagai filter sebelum pengujian.
7. Malakukan tinjauan ulang secara formal untuk menilai strategi tes dan kasus tes itu sendiri.
8. Mengembangkan pendekatan peningkatan yang berkelanjutan untuk proses testing.

Ada beberapa jenis pengujian perangkat lunak, antara lain (Khan,2011):

1. Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan program yang benar secara 100%.
2. *Black-Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.
   1. ***White Box Testing***

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau software dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang dibuat ada yang salah atau tidak. Kalau modul yang telah dan sudah dihasilkan berupa output yang tidak sesuai dengan yang diharapkan maka akan dikompilasi ulang dan dicek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan (Nidhra and Dondetti,2012).

Kasus yang sering menggunakan white box testing akan diuji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logika.
2. Pengujian keseluruhan loop yang ada susuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Kelebihan *White Box Testing* antara lain (Nidhra and Dondetti, 2012):

1. Kesalahan Logika

Menggunakan sintax ‘if’ dan sintax pengulangan. Langkah selanjutnya metode white box testing ini akan mencari dan mendeteksi segala kondisi yang dipercaya tidak sesuai dan mencari kapan suatu proses perulangan di akhir.

1. Ketidaksesuaian Asumsi

Menampilkan dan memonitor beberapa asumsi yang diyakini tidak sesuai dengan yang diharapkan atau yang ada diwujudkan, untuk selanjutnya akan dianalisa kembali dan kemudian diperbaiki.

1. Kesalahan Pengetikan

Mendeteksi dan mencari bahasa-bahasa pemrograman yang di anggap bersifat *case sensitife.*

Kelemahan *White Box Testing* adalah pada perangkat lunak yang jenisnya besar, metode *white box testing* ini dianggap boros karena melibatkan banyak sumberdaya untuk melakukannya. (Nidhra and Dondetti, 2012).

* 1. **Black box Testing**

*Black box testing* berfokus pada spesifikasi fungsional dari perangkat lunak. *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

*Black box testing* bukanlah solusi alternatif dari white box testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *white box testing.*

Black box testing cenderung untuk menemukan hal-hal berikut:

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (*interfaces errors*).
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi (*performance errors*).
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada input-input tertentu?
4. Bagaimana sekumpulan data dapat disolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metode atau teknik untuk melaksanakan black box testing, antara lain:

1. *Equivalence Partitioning*
2. *Boundary Value Analysis/Limit Testing*
3. *Comparison Testing*
4. *Sample Testing*
5. *Robustness Testing*
6. *Behavior Testing*
7. *Requirement Testing*
8. *Performance Testing*
9. Uji Kesehatan (*Endurance Testing)*
10. Uji Sebab-Akibat (*Cause-Effect Relationship Testing)*
    1. **Boundary Value Analysis/Limit Testing**

*Boundary value analysis* adalah salah satu teknik *black box testing* yang melakukan pengujian pada batas atas dan batas bawah nilai yang diisikan pada aplikasi. Beberapa prinsip yang mendasari pada *boundary value analysis* (BVA) yaitu:

1. Banyak kesalahan terjadi pada kesalahan masukan.
2. BVA mengijinkan untuk menyeleksi kasus uji yang menguji batasan niali input.
3. BVA merupakan komplemen dari *equivalence partitioning.* Lebih pada memilih elemen-elemen di dalam kelas ekivalen pada bagian sisi batas dari kelas.
4. Contoh:
5. Untuk rentang yang dibatasi a dan b maka uji (a-1),a,(a+1),(b-1),b,(b+1).
6. Jika kondisi input mensyaratkan sejumlah n nilai maka uji dengan sejumlah (n-1), n dan (n+1) nilai.
7. Aplikasikan dua aturan sebelumnya pada kondisi output (buat table pengujian hasil outputnya untuk nilai maksimal dan minimal).
8. Jika struktur data internal dari program memiliki cakupan (missal: ukuran buffer, batas array) gunakan data input yang menguji batas cakupan.

Seacara umum, aplikasi BVA dapat dikerjakan secara generic. Bentuk dasar implementasi BVA adalah untuk menjaga agar satu variable berada pada nilai nominal (normal atau rata-rata) dan mengijinkan variable lain diisikan dengan nilai ekstrimnya. Nilai yang digunakan untuk menguji keekstriman data adalah:

Min -------- minimal

Min+ ------ di atas minimal

Nom ------- rata-rata

Max ------- tepat dibawah maksimum

Max ------- maksimum

Sebagai contoh, misalnya akan dientrikan data tanggal. Data tanggal memiliki tiga variable yaitu tunggal, bualan dan tahun. Maka untuk ketiga variable tersebut dapat diambil kondisi berikut:

1< tanggal < 31

1< bulan < 12

1812 < tahun < 2016

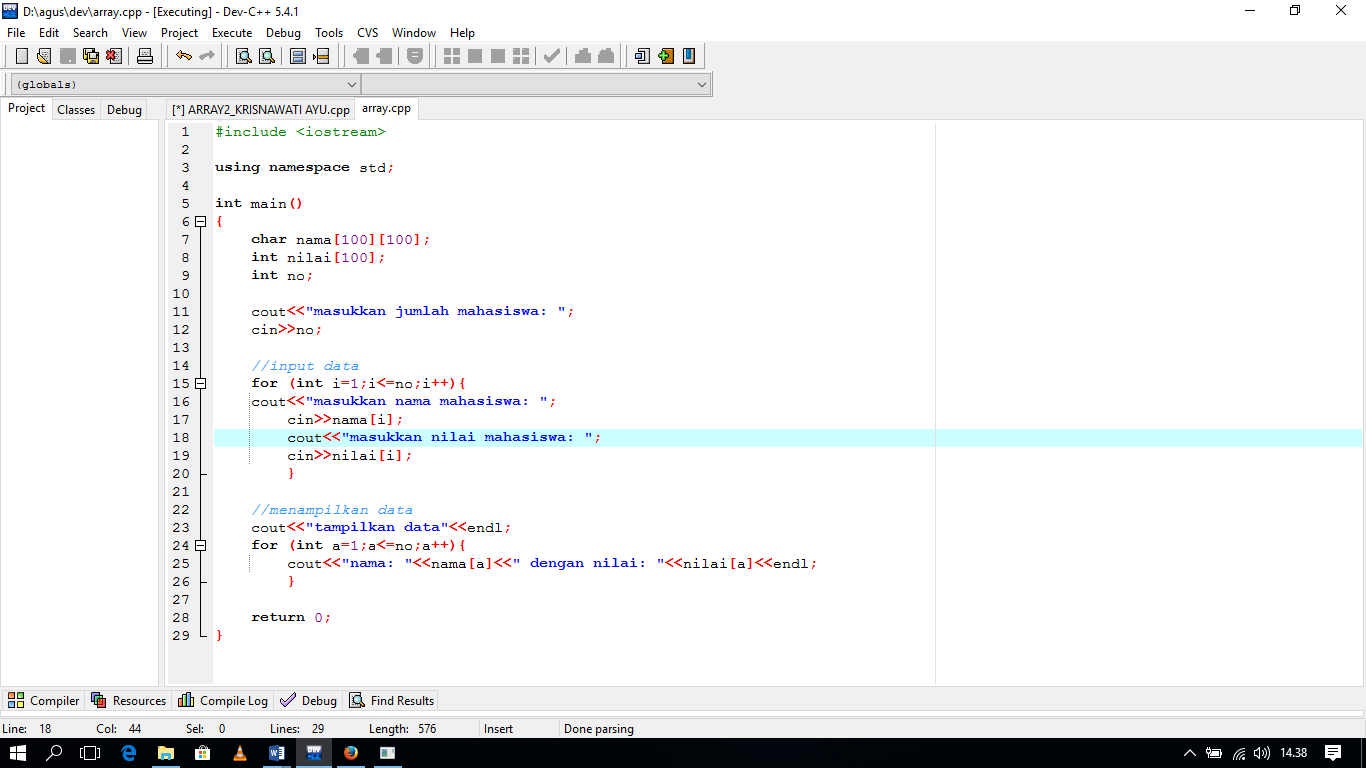
Maka untuk setiap entri data diluar angka diatas akan menampilkan pesan “Tanggal yang anda isikan salah”.

**BAB III**

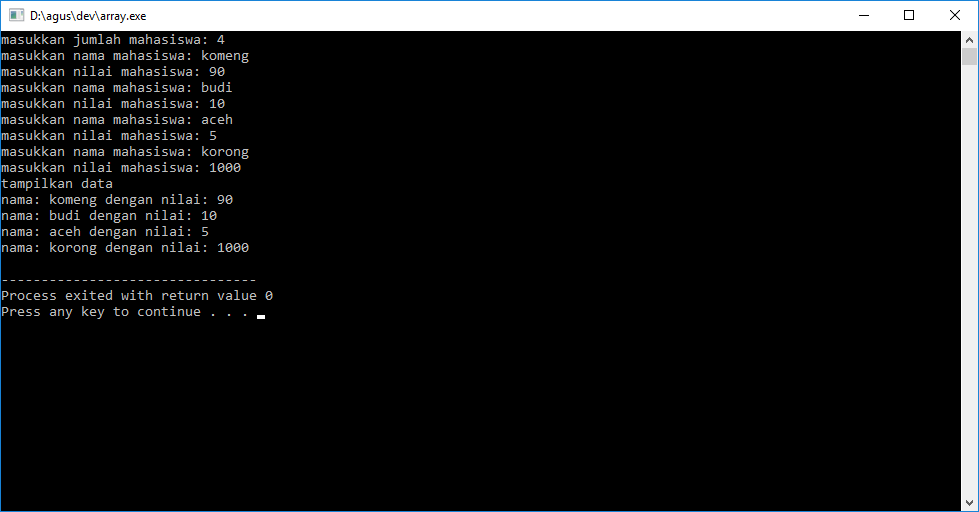
**PEMBAHASAN**

**3.1 Pengujian Black Box Testing**

Berdasarkan penjelasan sebelumnya, maka diterapkan teknik Blackbox Testing untuk menguji program yaitu “ARRAY”. Program ini sendiri memiliki fungsionalitas yang akan dibahas hasil pengujiannya pada artikel ini. Contoh program yang telah kami buat.



Pada contoh diatas, untuk memasukan nilai kepada array kita menggunakan cara pengulangan(looping) dimana akan terus berulang dan menjalankan perintah input data jika nilai variable i kurang dari sama dengan nilai variable no(jumlah mahasiswa) jika nilai variable i sudah lebih dari variable no, maka program akan menghentikan perulangan tersebut dan melanjutkan mengeksekusi kode selanjutnya .



**IV. KESIMPULAN**

Berdasarkan hasil analisin maka dapat ditarik kesimpulan bahwa:

1. Metode Blackbox Testing merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang di harapkan,
2. Setelah melakukan pengujian diketahui bahwa fungsionalitas masih bisa berjalan namun masih dapat menerima masukan data yang tidak diharapkan sehingga dapat menyebabkan data yang disimpan kurang valid.