**PENGEMBANGAN PERKAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

**TESIS**

**Karya tulis sebagai salah satu syarat**

**untuk memperoleh gelar Magister dari**

**Institut Teknologi Bandung**

**Oleh**

**HABIBIE ED DIEN**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

****

**INSTITUT TEKNOLOGI BANDUNG**

**Juli 2017**

**PENGEMBANGAN PERKAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

Tanggal Juli 2017

|  |  |
| --- | --- |
| Pembimbing Pertama  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Dr. M.M. Inggriani Liem) | Pembimbing Kedua  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Yudistira Dwi W. Asnar, S.T., Ph.D.) |

# DAFTAR ISI

HALAMAN PENGESAHAN i

[DAFTAR ISI ii](#_Toc485359567)

[DAFTAR GAMBAR DAN ILUSTRASI v](#_Toc485359568)

[DAFTAR TABEL vi](#_Toc485359569)

[Bab I Pendahuluan 1](#_Toc485359570)

[I.1 Latar Belakang 1](#_Toc485359571)

[I.2 Rumusan Masalah 5](#_Toc485359572)

[I.3 Tujuan 5](#_Toc485359573)

[I.4 Batasan Masalah 5](#_Toc485359574)

[I.5 Metodologi Penelitian 6](#_Toc485359575)

[I.6 Sistematika Penulisan 7](#_Toc485359576)

[Bab II Tinjauan Pustaka 8](#_Toc485359577)

[II.1 Struktur Data 8](#_Toc485359578)

[II.1.1 Graf 8](#_Toc485359579)

[II.1.2 Pohon 11](#_Toc485359580)

[II.2 Revolusi Teknologi Web dan *Computational Thinking* terhadap Pembelajaran 13](#_Toc485359581)

[II.3 Terminologi Visualisasi Perangkat Lunak 16](#_Toc485359582)

[II.4 Hasil Eksplorasi Perkakas Visualisasi Program 18](#_Toc485359583)

[II.5 Pentingnya Belajar Eksekusi Kode dalam Pemrograman 20](#_Toc485359584)

[II.6 Matrix Framework 21](#_Toc485359585)

[II.6.1 Visualisasi 22](#_Toc485359586)

[II.6.2 Struktur 23](#_Toc485359587)

[II.6.3 Konstruksi Visual 24](#_Toc485359588)

[II.6.4 Animasi 24](#_Toc485359589)

[II.6.5 Simulasi 25](#_Toc485359590)

[II.6.6 Tampilan Antarmuka Pengguna 25](#_Toc485359591)

[II.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi 26](#_Toc485359592)

[Bab III Analisis Masalah dan Perumusan Skema Graf 27](#_Toc485359593)

[III.1 Gambaran Umum Fitur Perkakas Online Python Tutor 27](#_Toc485359594)

[III.1.1 Arsitektur Perkakas OPT 29](#_Toc485359595)

[III.1.2 Komponen Perkakas OPT 30](#_Toc485359596)

[III.1.3 Proses Membentuk Galur Eksekusi Program 30](#_Toc485359597)

[III.1.4 Format Galur Eksekusi Program 31](#_Toc485359598)

[III.1.5 Fitur Pustaka *Data-Driven Documents* (D3) 33](#_Toc485359599)

[III.2 Analisis Masalah Perkakas 33](#_Toc485359600)

[III.2.1 Analisis Masalah dan Peluang Pemecahan 34](#_Toc485359601)

[III.2.2 Analisis Optimasi Perkakas 34](#_Toc485359602)

[III.3 Perumusan Skema Graf 34](#_Toc485359603)

[III.3.1 Menggunakan Format Galur Eksekusi 34](#_Toc485359604)

[III.3.2 Pembaruan Format Galur Eksekusi Program 34](#_Toc485359605)

[III.4 Penunjang Pengembangan Perkakas 35](#_Toc485359606)

[III.4.1 Kebutuhan Perangkat Lunak 35](#_Toc485359607)

[III.4.2 Kebutuhan Perangkat Keras 35](#_Toc485359608)

[Bab IV Pengembangan dan Implementasi Perkakas 36](#_Toc485359609)

[IV.1 Perbaikan Arsitektur Perkakas 36](#_Toc485359610)

[IV.2 Format Galur Eksekusi Kode 37](#_Toc485359611)

[IV.3. Implementasi Perkakas 37](#_Toc485359612)

[Bab V Pengujian dan Evaluasi Perkakas 38](#_Toc485359613)

[V.1 Pengujian Perkakas 38](#_Toc485359614)

[V.1.1 Simulasi Mengajar / Studi Kasus 38](#_Toc485359615)

[V.1.2 Pengujian Secara Luring dan Daring 38](#_Toc485359616)

[V.1.3 Wawancara Mendalam 38](#_Toc485359617)

[V.2 Evaluasi Perkakas 38](#_Toc485359618)

[V.2.1 Tujuan Pengembangan Perkakas 38](#_Toc485359619)

[V.2.2 Indikator Keberhasilan Perkakas 38](#_Toc485359620)

[V.2.3 Perbandingan Perkakas Lama dan Baru 38](#_Toc485359621)

[V.2.4 Kelebihan dan Kekurangan Perkakas 38](#_Toc485359622)

[Bab VI Kesimpulan dan Saran 39](#_Toc485359623)

[VI.1 Kesimpulan 39](#_Toc485359624)

[VI.2 Saran 39](#_Toc485359625)

[DAFTAR PUSTAKA 40](#_Toc485359626)

# DAFTAR GAMBAR DAN ILUSTRASI

[Gambar II.1 (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969) 8](#_Toc485359913)

[Gambar II.2 Model visual graf secara umum: (a) Pohon dan (b) Graf 13](#_Toc485359914)

[Gambar II.3 Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004) ..... 23](#_Toc485359915)

[Gambar II.4 Turunan hirarki *concept interfaces Matrix* (Korhonen dkk., 2004) ...... 24](#_Toc485359916)

[Gambar II.5 Proses konstruksi representasi visual (Korhonen dkk., 2004) 24](#_Toc485359917)

[Gambar III.1 Tampilan Antarmuka OPT untuk Bahasa Pemrograman C 27](#_Toc485359918)

[Gambar III.2 Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman 28](#_Toc485359919)

[Gambar III.3 Tampilan Simulasi dan Visualisasi Kode C Pointer 28](#_Toc485359920)

[Gambar III.4 Arsitektur OPT untuk visualisasi C dan C++ 29](#_Toc485359921)

[Gambar III.5 Ilustrasi Sederhana Proses Perkakas OPT 30](#_Toc485359922)

[Gambar III.6 Ilustrasi Format Galur Eksekusi Program 32](#_Toc485359923)

[Gambar III.7 Peran Utama D3 sebagai *Framework* Visualisasi 33](#_Toc485359924)

[Gambar IV.1 Perbaikan Arsitektur untuk Pengembangan Perkakas OPT 36](#_Toc485359925)

# DAFTAR TABEL

[Tabel II.1 Daftar nama algoritma graf dan model visualnya (Valiente, 2002) .....10](#_Toc485360096)

[Tabel II.2 Daftar nama algoritma pohon dan model visualnya (Valiente, 2002) .....11](#_Toc485360097)

[Tabel II.3 Perkembangan perkakas VP untuk bahasa pemrograman C/C++ 19](#_Toc485360098)

[Tabel III.1 Hasil Analisis dan Peluang Pemecahan Masalah 34](#_Toc485360099)

# Bab I Pendahuluan

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian tesis, batasan masalah, metodologi penelitian, dan sistematika penulisan laporan tesis.

## I.1 Latar Belakang

Graf adalah salah satu pokok bahasan perkuliahan Algoritma Pemrograman dan Struktur Data dalam bidang informatika atau ilmu komputer (Valiente, 2002). Graf merupakan topik penting yang selalu ada dalam perkuliahan Struktur Data. Representasi objek visual dari graf dinyatakan sebagai noktah, bulatan, atau titik, sedangkan hubungan antar objek dinyatakan dengan garis. Graf bermanfaat untuk memodelkan suatu masalah dengan cara merepresentasikan objek-objek diskrit dan hubungan antar objek tersebut. Contoh manfaat graf dalam pemodelan dapat dilihat pada jaringan komunikasi, jaringan komputer, analisis algoritma dan peta. Graf dalam pemrograman digunakan untuk memodelkan sesuatu yang abstrak, seperti aliran kerja dalam proyek, perencanaan dan manajemen proyek, pencarian rute terpendek, perpindahan dalam permainan (*game*) serta dalam bidang kecerdasan buatan (*artificial intelligence*).

Pohon (*tree*) merupakan salah satu bentuk khusus dari struktur suatu graf (Valiente, 2002). Suatu graf terhubung yang setiap pasangan simpulnya hanya dapat dihubungkan oleh suatu lintasan tertentu, maka graf tersebut dinamakan pohon (*tree*). Pohon (*tree*) merupakan graf tak-berarah yang terhubung dan tidak memiliki sirkuit. Struktur pohon merupakan topik struktur data yang penting dalam informatika (Inggriani Liem, 2008). Pohon bermanfaat untuk mengorganisasi informasi berdasarkan suatu struktur logika dan memungkinkan cara akses yang khusus terhadap suatu elemen. Contoh manfaat pohon dalam memodelkan suatu masalah, seperti pohon keputusan, pohon keluarga, klasifikasi botani, pohon sintaks, dan pohon ekspresi aritmatika.

Pada abad 21 ini, memiliki kemampuan dalam pemrograman merupakan keahlian yang sangat bernilai (Helminen dan Malmi, 2010; Margulieux dkk., 2016). Oleh sebab itu banyak pelajar ingin memasuki informatika atau bidang komputer lainnya (Cetin, 2016; Piteira dan Costa, 2013). Namun, situasi tersebut tidak diimbangi dengan kualitas kemampuan pemrograman yang harus dimiliki. Untuk mempelajari pemrograman, pelajar harus memiliki pengetahuan yang baik tentang deklarasi dan prosedural program, daya ingat yang kuat, pemahaman, penyelesain masalah, abstraksi dan kemampuan logika program (Piteira dan Costa, 2013).

Riset yang ditulis oleh Piteira dan Costa pada tahun 2013 berisi survei tentang identifikasi permasalahan dalam perkuliahan pemrograman. Riset tersebut bertujuan untuk mengidentifikasi tingkat kesulitan belajar paling populer di kalangan pelajar. Aspek yang dianalisis adalah konten materi dari perkuliahan dan aspek pembelajarannya. Ternyata diperoleh data bahwa konsep pemrograman yang paling sulit adalah tipe struktur data, selain pointer dan referensi serta parameter. Struktur data merupakan subjek penting dalam bidang komputasi, namun pelajar masih mengalami kesulitan dalam memahami, mengeksekusi dan mengonsep algoritmanya (Pathania dan Singh, 2014). Struktur data dapat dibagi menjadi dua bagian, yaitu struktur data linear dan non-linear. Struktur data linear terdapat larik, antrian, tumpukan dan *linked list*, sedangkan non-linear terdapat pohon dan graf. Kemudian untuk hasil analisis dari aspek pembelajaran, bahwa yang paling signifikan pelajar sangat terbantu melalui contoh-contoh eksekusi kode program, karena merupakan implementasi dari konsep yang dipelajarinya.

Visualisasi telah banyak direkomendasikan menjadi perkakas penting untuk mengatasi kesulitan pelajar dalam memahami konsep pemrograman (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013; Gračanin dkk., 2005; Guo, 2013). Melalui interaksi visual, manusia lebih cenderung menangkap lebih banyak informasi yang diterima dibandingkan melalui indera lainnya (Ware, 2004). Oleh sebab itu visualisasi telah menjadi bagian penting dalam proses belajar mengajar pemrograman (Sorva dkk., 2013), sehingga menjadi solusi dalam proses pembelajaran (Guo, 2013). Istilah “visualisasi perangkat lunak” telah lama berkembang dan didefinisikan sebagai sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-komputer (Cetin dan Andrews-Larson, 2016). Visualisasi perangkat lunak dibagi menjadi dua, yaitu visualisasi algoritma (VA) dan visualisasi program (VP) (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013). VA berkaitan dengan abstraksi dari konsep langkah kerja perangkat lunak, sedangkan VP terkait pada cara kerja eksekusi kode program dan proses struktur datanya. Melalui VP dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari materi ajar pemrograman (Sorva, 2012).

Bahasa pemrograman C dan C++ digunakan dalam perkuliahan struktur data teknik informatika di STEI ITB (Inggriani Liem, 2008). Kedua bahasa pemrograman itu dipilih karena merupakan bahasa yang banyak digunakan dalam industri perangkat lunak saat ini. Oleh sebab itu, kedua bahasa pemrograman ini dipilih untuk dijadikan bahas ajar perkakas VP.

Teknologi web dan internet saat ini telah banyak mengubah cara belajar di dunia (Bonk, 2009). Perkembangan pesat teknologi web memberi kemudahan kepada pelajar untuk meningkatkan kemampuannya tanpa dibatasi oleh waktu dan tempat. *E-learning* menjadi portal untuk belajar bagi siapapun tanpa kendala jarak. Melalui teknologi ini, perkakas VP yang akan dikembangkan dapat dengan mudah dan langsung digunakan oleh berbagai pelajar atau pengguna di dunia.

Perkembangan VP sebanyak 40 perkakas dijelaskan pada Bab 11 disertasi Sorva (2012) sejak tahun 1983-an. Namun perkakas VP berbasis web untuk bahasa C dan C++ masih sangat langka. Umumnya pengembangan perkakas ini dibangun dengan *Java Virtual Machine* (JVM) (Helminen dan Malmi, 2010). *The Teaching Machine* (*www.theteachingmachine.org*) merupakan perkakas VP berbasis *Java Applet* web yang mendukung bahasa C++ dan Java. Kekurangannya butuh waktu dan usaha lebih bagi pelajar untuk memasang *plugin Java applet*, terutama saat digunakan di ruang laboratorium komputer yang berjumlah banyak. Alasan para pengembang menggunakan Java karena dapat beroperasi pada berbagai sistem operasi.

*Online Python Tutor* (OPT) (Guo, 2013) merupakan perkakas VP yang pada awalnya ditahun 2010 dikembangkan untuk bahasa Python. Namun saat ini telah mendukung banyak bahasa pemrograman seperti Java, C, C++, Ruby dan JavaScript. Kelebihan perkakas VP ini cukup sederhana untuk menggunakannya. Pelajar hanya mengunjungi situs *pythontutor.com* pada peramban di perangkat komputer atau piranti bergerak seperti ponsel pintar dan *tablet*. Kemudian pelajar bisa langsung mengeksekusi kode program yang sudah ada contohnya di laman tersebut atau dengan cara mengetiknya sendiri pada kode editor yang disediakan. Namun, perkakas ini masih terbatas pada visualisasi struktur data linear, sehingga untuk mendukung pembelajaran pemrograman graf belum bisa dilakukan.

*Jype* (Helminen dan Malmi, 2010; Sorva, 2012) merupakan perkakas VP yang menggunakan metode *Matrix framework* untuk memvisualkan bahasa Python secara otomatis dalam program struktur data linear dan non-linear, seperti larik dan pohon. Perkakas ini dibangun dengan Java dan beroperasi di web. Namun perkakas ini sudah tidak dikembangkan lagi dan kode sumbernya tidak ditemukan di internet (Guo, 2013; Sorva, 2012). *Matrix framework* (Korhonen dkk., 2004) dibangun berdasar empat konsep visual, yaitu *visual container*, *visual component*, *visual reference*, dan *visual data*. *Visual container* berupa struktur yang kompleks, menyimpan nilai variabel (node, indeks, dan lainnya) yang saling terhubung dengan cara tertentu. Kemudian tiap variabel di dalam *visual container* terdapat *visual component* yang berguna untuk memvisualisasikan variabel. *Visual reference* sebagai penghubung antar *visual component* dan *visual data* yang berisi tipe data primitif atau struktur data yang lebih kompleks dengan atribut-atributnya.

Saat ini perkakas VP berbasis web untuk C/C++ yang mampu melakukan visual graf belum tersedia. Sehingga pada penelitian ini, penulis akan mengembangkan perkakas VP untuk graf berbasis web dari kode sumber OPT, karena selain telah mendukung teknologi web untuk *e-learning*, perkakas ini bersifat bebas (*free*) dan bersumber terbuka (*open source*) (Guo, 2013). Agar dapat melakukan visualisasi graf secara otomatis ketika eksekusi kode program C/C++, maka perkakas VP akan dipadukan dengan metode *Matrix framework*.

## I.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan tersebut, maka diperoleh rumusan masalah sebagai berikut.

1. Apakah penggunaan *Matrix framework* cocok terhadap pengembangan perkakas VP berbasis OPT untuk graf dalam bahasa C dan C++ ?
2. Bagaimana mengembangkan perkakas VP berbasis OPT dengan menggunakan *Matrix framework* untuk graf dalam bahasa C dan C++ ?
3. Apakah penggunaan *Matrix framework* dapat meningkatkan kinerja VP ?
4. Bagaimana perkakas dapat mendeteksi adanya kode program graf atau pohon sehingga proses visualisasi sesuai dengan algoritmanya ?
5. Apakah metode *Matrix framework* dapat dikembangkan untuk mendukung visualisasi graf ?
6. Bagaimana peran VP yang dikembangkan terhadap proses pembelajaran algoritma dan pemrograman untuk graf ?

## I.3 Tujuan

Sebagai solusi dari masalah yang telah diuraikan, maka tujuan dari penelitian tesis ini adalah :

1. Menghasilkan perkakas VP berbasis web untuk bahasa C dan C++ yang dapat menunjang proses pembelajaran algoritma pemrograman untuk graf.
2. Menghasilkan modifikasi *Matrix framework* yang dapat mendukung VP untuk graf sesuai algoritma yang sedang digunakan.

## I.4 Batasan Masalah

Latar belakang dan rumusan masalah telah diuraikan, maka untuk memfokuskan kegiatan penelitian tesis ini diperlukan batasan, sehingga cakupan pembahasan dan penyelesaian tidak meluas. Adapun batasan masalah pada tesis ini adalah :

1. Pengembangan perkakas VP fokus terhadap visual graf.
2. Implementasi perkakas VP berdasar kode sumber dari OPT.
3. Bahasa pemrograman yang dapat dieksekusi adalah C dan C++.
4. Kode program yang dapat divisualisasi minimal memiliki syarat terbentuknya data objek dalam graf.

## I.5 Metodologi Penelitian

Penelitian dimulai dengan studi literatur, eksplorasi, dan identifikasi masalah untuk memformulasikan pertanyaan-pertanyaan yang berkaitan dengan permasalahan belajar algoritma pemrograman graf dan perkembangan perkakas VP. Tinjauan pustaka dari penelitian sebelumnya, buku, jurnal dan artikel ilmiah, dan situs internet yang berelasi dengan pengembangan perkakas VP serta konsep dalam pembelajaran algoritma pemrograman graf, sehingga penelitian pada tesis ini akan dilaksanakan melalui beberapa tahapan sebagai berikut.

1. Studi Literatur dan Eksplorasi

Tahap ini dilakukan pendalaman mengenai konsep umum pembelajaran pemrograman, pedagogi, VP, proses cara berpikir komputer (*computational thinking*) (Sorva, 2012), serta implementasi dari perkakas yang telah dibangun. Studi literatur menghasilkan kesimpulan awal yang dapat dijadikan landasan dalam perumusan masalah dan pengembangan perkakas. Kemudian dilakukan eksplorasi berbagai perkakas VP yang telah ada.

1. Analisis Masalah dan Perumusan Skema Graf

Tahap ini dilakukan kajian terhadap masalah yang diteliti kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Selanjutnya, akan dirumuskan suatu skema untuk melakukan eksplorasi kode program dan mengonversi kode tersebut menjadi model animasi dalam sebuah tampilan grafis diagram pohon dan graf.

1. Pengembangan Perkakas

Hasil rancangan sistem pada tahap sebelumnya akan diimplementasikan dalam pembangunan perkakas VP berbasis web, kemudian akan dilakukan pengujian dan evaluasi dari hasil implementasi tersebut.

1. Pengujian dan Evaluasi Perkakas

Hasil pengembangan perkakas perlu diuji dengan memasukkan berbagai jenis kode program dan tipe data, kemudian menganalisis hasil dari visualisasi tersebut. Selain itu, perkakas perlu dievaluasi secara empiris dengan bantuan responden untuk menilai efektivitas perkakas dalam membantu pemahaman terhadap kode program yang sedang dipelajarinya.

## I.6 Sistematika Penulisan

Penulisan laporan hasil penelitian tesis ini dibagi menjadi enam bab, yaitu:

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, serta metodologi yang digunakan pada pengerjaan tesis, serta sistematika pembahasan dari laporan tesis.

1. Bab II Tinjauan Pustaka

Bab ini berisi uraian tentang konsep belajar mengajar pemrograman, peran dan perkembangan perkakas VP dari hasil penelitian sebelumnya yang berkaitan dengan masalah yang dikaji, sehingga memberikan gambaran perkembangan terhadap masalah yang akan diteliti.

1. Bab III Analisis Masalah dan Perumusan Skema Graf

Bab ini berisi analisis terhadap masalah yang akan diteliti, kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan, sehingga untuk mendukung hipotesis tersebut, dibangun pula rancangan dari sistem yang akan dikembangkan.

1. Bab IV Pengembangan Perkakas

Bab ini menjelaskan proses pengembangan dan implementasi perkakas VP berdasarkan rancangan yang telah dirumuskan pada Bab III sebelumnya.

1. Bab V Pengujian dan Evaluasi Perkakas

Bab ini menguraikan proses dan hasil pengujian serta evaluasi dari perkakas yang dikembangkan pada tesis ini. Pengujian dilakukan untuk membuktikan bahwa perkakas berhasil dikembangkan lebih baik dibandingkan sebelumnya. Evaluasi perkakas dilakukan untuk menilai efektivitas dan kualitas dari visualisasi untuk mendukung pemahaman dalam pembelajaran algoritma pemrograman.

1. Bab VI Penutup

Bab ini berisi kesimpulan dalam tesis serta saran untuk pengembangan perkakas lebih lanjut dari penelitian tesis ini.

# Bab II Tinjauan Pustaka

Pada bab ini dijelaskan mengenai ruang lingkup struktur data graf dan algoritmanya, terminologi visualisasi perangkat lunak, peran perkakas VP dalam proses pembelajaran pemrograman, dan skema detail dari *Matrix framework*.

## II.1 Struktur Data

Graf dan pohon merupakan salah satu pokok bahasan dalam Matematika Diskrit dan konsep dasar yang harus dipahami oleh pelajar dalam bidang informatika atau ilmu komputer. Dalam perkuliahan Algoritma Pemrograman dan Struktur Data, terdapat pula topik tentang graf, namun lebih cenderung kepada implementasi terhadap pemrogramannya. Karena memang pelajar bidang informatika atau ilmu komputer dituntut untuk mampu membuat kode program dalam menyelesaikan suatu permasalahan tertentu.

Berikut ini akan dijelaskan tentang sejarah graf dan pohon beserta algoritma dan model visualnya.

### II.1.1 Graf

Sejarah mencatat bahwa model graf diterapkan pertama kali sekitar Abad 17 pada tahun 1736, yaitu permasalahan jembatan Königsberg (lihat Gambar II.1). Kota Königsberg (sebelah timur Prussia, Jerman sekarang) yang berubah nama menjadi kota Kaliningrad. Di kota itu terdapat sungai Pregal yang mengalir mengelilingi pulau Kneiphof lalu bercabang menjadi dua buah anak sungai (Rinaldi Munir, 2010).

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

1. (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969)

Pada Gambar II.1(a) terdapat tujuh buah jembatan yang saling menghubungkan antar daratan yang dialiri sungai tersebut. Permasalahannya adalah apakah mungkin dapat melalui ketujuh jembatan tersebut tepat satu kali dan kembali ke tempat semula ?. Beberapa penduduk kota tersebut sepakat bahwa tidak mungkin mereka dapat melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan. Tetapi mereka tidak dapat menjelaskan secara ilmiah alasannya. Euler adalah seorang matematikawan Swiss yang berhasil menemukan jawaban terhadap masalah tersebut pada tahun 1736. Ia memberikan bukti sederhana dengan memodelkan masalah ini ke dalam model graf. Titik (noktah) menyatakan daratan yang dihubungkan oleh jembatan, yang disebut simpul (*vertex*). Garis merepresentasikan jembatan, yang disebut sisi (*edge*). Setiap titik terdapat label huruf A, B, C, dan D. Euler membuat model graf seperti terlihat pada Gambar II.1(b).

Euler mengemukakan jawabannya bahwa orang tidak mungkin melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Derajat merupakan banyak garis yang bersisian dengan noktah. Contoh simpul C memiliki derajat tiga karena ada tiga buah garis yang bersisian dengannya, simpul B dan D juga berderajat tiga, sedangkan simpul A berderajat lima. Karena semua simpul tidak berderajat genap, maka tidak mungkin dilakukan perjalanan berupa sirkuit (yang selanjutnya disebut sirkuit Euler) pada graf tersebut.

Graf memiliki berbagai macam algoritma dan model visualnya masing-masing, sehingga perlu dijabarkan lebih rinci agar mengetahui konsep dasar terhadap proses kerja dari perkakas VP yang akan dibangun. Macam algoritma graf yang digunakan dalam bidang komputer beserta model visualnya dapat dilihat pada Tabel II.1 berikut. Algoritma dan model visual graf yang dijelaskan berikut merupakan yang paling sering digunakan dalam perkuliahan Struktur Data dan Algoritma Pemrograman di STEI ITB (Inggriani Liem, 2008).

1. Daftar nama algoritma graf dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Graf** | **Model Visual** |
| --- | --- | --- |
| 1 | Breadth First Search (BFS) |  |
| 2 | Depth First Search (DFS) |  |
| 3 | Prim |  |
| 4 | Djikstra |  |
| 5 | Bellman-Ford | http://2.bp.blogspot.com/-GG_lKtfmonw/UYi73BodmvI/AAAAAAAAAjw/UGlBfBpEKI4/s400/nodenode.png |
| 6 | Floyd-Warshall |  |
| 7 | Kruskal |  |

### II.1.2 Pohon

Pohon (Rinaldi Munir, 2010) merupakan graf terhubung yang tidak mengandung sirkuit. Teori pohon (*tree*) merupakan konsep paling penting, karena memiliki banyak terapan di berbagai bidang ilmu komputer maupun bidang lainnya. Konsep ini telah banyak dikaji secara intensif sebagai objek matematika dan terapan dalam kehidupan manusia sehari-hari. Banyak orang menggunakan pohon untuk membuat hirarki seperti silsilah keluarga, struktur organisasi, organisasi pertandingan olahraga, dan lain-lain. Pohon parsing (*parse tree*) digunakan oleh para ahli bahasa untuk menguraikan kalimat.

Sejak tahun 1857, istilah pohon telah lama digunakan oleh matematikawan asal Inggris bernama Arthur Cayley. Ia memanfaatkan pohon untuk menghitung jumlah senyawa kimia. Pohon sebagai struktur data rekursif yang merupakan bagian penting dari perkuliahan Struktur Data.

Untuk pembahasan macam algoritma pohon yang digunakan dalam bidang komputer beserta model visualnya dapat dilihat pada Tabel II.2 berikut ini.

1. Daftar nama algoritma pohon dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Pohon** | **Model Visual** |
| --- | --- | --- |
| 1 | Binary Search Tree (BST) |  |
| 2 | Digital Search Tree |  |
| 3 | Radix Tree | https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/An_example_of_how_to_find_a_string_in_a_Patricia_trie.png/220px-An_example_of_how_to_find_a_string_in_a_Patricia_trie.png |
| 4 | AVL (Adelson-Velskii dan Landis) *Tree*; terdiri dari *Single Rotation* dan *Double Rotation* |  |
| 5 | Red-Black Tree | Diagram of binary tree. The black root node has two red children and four black grandchildren. The child nodes of the grandchildren are black nil pointers or red nodes with black nil pointers. |
| 6 | Search Tree Traversal; terdiri dari *pre-order*, *in-order*, dan *post-order* | Hasil gambar untuk tree traversal algorithm |
| 7 | Splay Tree |  |

Dari penjelasan berbagai macam algoritma dan model-model visual graf tersebut, dapat ditarik kesimpulan secara umum representasi visualisasi yang dapat dilakukan hanya dua bagian besar yaitu model visual pohon dapat dilihat pada Gambar II.2 (a) dan model visual graf pada Gambar II.2 (b).

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

1. Model visual graf secara umum: (a) Pohon dan (b) Graf

## II.2 Revolusi Teknologi Web dan *Computational Thinking* terhadap Pembelajaran

Bonk (2009) menulis sebuah buku berjudul “*The World Is Open: How Web Technology Is Revolutionizing Education*” yang berisi tentang teknologi web dan internet yang mengubah cara belajar di dunia pada awal abad 21 ini. Internet telah membuka peluang besar bagi seluruh manusia di bumi. Melalui teknologi web setiap orang dapat belajar di mana dan kapan pun. Pembelajaran elektronik (*e-learning*) telah menjadi bagian penting dari setiap komunitas belajar.

Saat ini keuntungan bagi para pengajar untuk berbagi ilmu bukan merupakan hal yang sulit lagi. Secara nyata kehadiran internet telah mengubah cara berbagi kepada para pelajar di dunia, kelas, sekolah, kampus, dan masih banyak sisi yang berpotensi terkena dampak positifnya. Saat ini dengan internet atau *online sharing* dapat mempengaruhi setiap orang di mana pun dan kapan pun di dunia ini. Khususnya istilah berbagi telah populer dalam lingkungan pembelajaran melalui teknologi daring secara kolaboratif.

Abad 21 ini telah membawa informasi ke revolusi dunia digital. Masyarakat umum harus dapat beradaptasi dengan penggunaan komputer, ponsel pintar, dan internet. Dengan kata lain, mereka harus dapat mencapai kemampuan berpikir yang disebut sebagai *computational thinking* (Wang, 2015).

*Computational thinking* (CT) merupakan keterampilan mental untuk menerapkan konsep-konsep dasar dan penalaran dalam berpikir, yang berasal dari digital komputer modern, *computer science*, yang mencangkup segala aspek kehidupan, termasuk aktivitas manusia sehari-hari. CT merupakan cara berpikir yang terinspirasi dari cara kerja komputer dan teknologi informasi, dan berbagai keunggulannya, batasan-batasannya, serta permasalahan yang muncul. CT juga memberikan peluang untuk para peneliti tetap berpikir kritis dengan pertanyaan-pertanyaan seperti “bagaimana jika kita mengotomatisasikan hal ini?”, “instruksi dan tindakan apa yang perlu dilakukan terhadap pembelajaran ini kepada anak-anak?”, “seberapa efesienkah metode ini?”, dan “apa dampak negatifnya?”.

CT dapat memperluas daya berpikir, membantu dalam penyelesaian masalah, meningkatkan efisiensi, menghindari kesalahan, dan mengantisipasi terjadinya keterbatasan dalam berpikir, baik dalam berinteraksi dan komunikasi dengan manusia atau mesin. CT dapat membuat manusia lebih sukses bahkan dalam menjalani kehidupan sehari-harinya (Wang, 2015).

Untuk dapat memahami CT, tidak diperlukan menjadi seorang komputer saintis atau insinyur. Dari sudut pandang pengguna awam, CT dapat memberikan tampilan secara terstruktur, mudah, dan sederhana sesuai dengan topik komputer yang akan dipelajarinya. Asumsinya, pengguna memiliki sedikit pengetahuan dasar tentang *computer science* atau pemrograman. Karena ketika ditampilkan sebuah perangkat keras, perangkat lunak, representasi data, algoritma, sistem, keamanan, jaringan, situs web, dan konsep lainnya dalam segala aspek komputer, CT akan secara luas mengimplementasikan konsep-konsep dan keterampilan khusus untuk menjelaskan bagaimana atau di mana mereka dapat menerapkannya dalam kehidupan.

Semua komputer itu bodoh, ia hanya mampu mengolah data dalam bentuk angka-angka biner atau bit saja (Wang, 2015). Setiap bit direpresentasikan sebagai angka nol atau satu. Komputer mengeksekusi semua instruksi yang diprogram, namun hal itu tetap saja direpresentasikan sebagai 0 dan 1. Walaupun demikian, komputer merupakan mesin yang bersifat universal, dapat melakukan tugas-tugas apapun sesuai dengan instruksi program. Untuk memaksimalkan kinerja komputer maka harus diprogram, dikontrol, dan dibuat bekerja sesuai dengan yang diharapkan. Semua itu dapat dilakukan dengan memahami konsep dasar yang disebut sebagai *computational thinking*.

Beberapa aspek penting dari CT adalah sebagai berikut (Wang, 2015):

* **Penyederhanaan melalui abstraksi**. Abstraksi adalah teknik untuk mengurangi kompleksitas dengan cara mengabaikan hal-hal detail yang tidak penting dan fokus terhadap yang penting. Sebagai contoh, sopir melihat mobil langsung terlintas untuk bagaimana mengendarainya, bukan berpikir bagaimana mobil itu bekerja atau dibuat.
* **Kelebihan dalam hal automatisasi**. Menyusun hal-hal penting sehingga mudah untuk dimanajemen dan diotomatisasi. Seperti mengerjakan prosedur secara sistematis, menyusun algoritma, melakukan tugas-tugas secara terstruktur sehingga dapat meningkatkan efisiensi dan produktifitas.
* **Iterasi dan perulangan.** Merupakan teknik strategis untuk mengimplementasikan secara berulang dengan tingkat keberhasilan yang sama dan mengeksekusi beberapa langkah untuk menyelesaikan masalah.
* **Ketelitian dalam penglihatan dan pikiran**. Perubahan angka 0 ke angka 1, atau huruf kapital, dapat berpengaruh besar terhadap instruksi program. Dibutuhkan penglihatan yang teliti, pikiran yang cerdas, dan pendekatan yang cermat. Jika mengabaikan hal-hal tersebut, maka akan berakibat fatal.
* **Ketelitian dalam komunikasi melalui instruksi program**. Komputer tidak dapat langsung memahami apa yang kita maksud, namun dapat melalui kalimat atau perkataan. Oleh karena itu, memberikan instruksi program harus teliti dan lengkap. Karena ketidakjelasan instruksi tidak dapat ditoleransi dan harus dibuat secara eksplisit.
* **Logika deduktif**. Merupakan logika yang kaku atau statis. Hal ini karena tidak mempertimbangkan perasaan atau emosional manusia.
* **Kreatif dalam penyelesaian masalah**. Sebuah instruksi program dibuat untuk menyelesaikan berbagai macam tugas. Komputer tidak seperti manusia yang dapat menyelesaikan masalah berdasarkan pengalaman dan keilmuannya. Sebuah kode program pada tingkatan komputer langsung dieksekusi tanpa mempertimbangkan efeknya, sehingga pengguna harus kreatif memanfaatkan kemampuan komputer yang tersedia.
* **Mengantisipasi masalah**. Otomatisasi oleh komputer tidak selalu menjadi hal praktis dan teliti. Diperlukan rencana antisipasi untuk hal-hal di luar dugaan, seperti kerusakan sistem, tidak adanya filter dalam proses data, *error* atau *malfunction*, dan sebagainya. Berkaitan dengan CT, maka diperlukan rencana kontinjensi untuk hal-hal di luar dugaan tersebut agar tidak menjadi jebakan terhadap dirinya.

Aspek-aspek tersebut hanya beberapa ide utama dari CT, karena sebenarnya CT lebih banyak membuka konsep-konsep dan jalan berpikir yang lebih penting. Melalui peningkatan pemahaman terhadap komputer, maka proses CT telah memberikan berbagai sudut pandang yang berbeda kepada setiap individu.

## II.3 Terminologi Visualisasi Perangkat Lunak

Visualisasi perangkat lunak (*Software Visualization*) merupakan bidang aktif dalam riset dan pengembangan sistem. Berbagai sistem visualisasi perangkat lunak bermunculan untuk digunakan dengan tujuan tertentu dan terus berkembang setiap tahunnya (Sorva, 2012). Gračanin dkk., (2005) mendefinisikan visualisasi perangkat lunak (*Software Visualization*) sebagai suatu bidang untuk menginvestigasi dengan pendekatan dan teknik tertentu yang bertujuan dalam merepresentasikan grafis algoritma secara statis dan dinamis, program (*code*), dan data yang diproses. Visualisasi perangkat lunak memiliki tujuan utama untuk menganalisis program dan pengembangan; untuk meningkatkan pemahaman terhadap konsep yang tak nampak dan cara kerja perangkat lunak. Tantangan utamanya adalah mencari langkah efektif dalam pemetaan berbagai aspek perangkat lunak untuk direpresentasikan secara grafis menggunakan metafora visual.

Istilah “visualisasi perangkat lunak” telah lama berkembang dan didefinisikan sebagai sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-komputer (Cetin dan Andrews-Larson, 2016). Visualisasi perangkat lunak dibagi menjadi dua, yaitu visualisasi algoritma (VA) dan visualisasi program (VP) (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013). VA berkaitan dengan abstraksi dari konsep langkah kerja perangkat lunak, sedangkan VP terkait pada cara kerja eksekusi kode program dan proses struktur datanya. Melalui perkakas VP dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari materi ajar (Sorva, 2012).

Visualisasi secara intuitif tampaknya telah menjadi bagian dari sarana pembelajaran yang menarik (Sorva, 2012). Berbagai literatur mendukung pernyataan tersebut (Cetin dan Andrews-Larson, 2016; Guo, 2013; Pathania dan Singh, 2014; Sorva dkk., 2013; Sorva dan Sirkiä, 2010). Penelitian terkait banyak yang mengusulkan visualisasi digunakan dalam lingkungan belajar. Hampir setiap teori pembelajaran merekomendasikan visualisasi sebagai sarana untuk memberikan pemahaman lebih baik kepada pembacanya (Clark dan Mayer, 2011; Mayer, 2014).

Penggunaan sarana visual sebagai alat bantu belajar telah menjadi budaya di seluruh dunia, baik itu dalam pembelajaran komputer atau lainnya. Berdasarkan salah satu survei bahwa penggunaan visualisasi dalam perkuliahan informatika atau ilmu komputer hampir dilakukan setiap hari (Sorva, 2012).

Banyak perkakas VP dibuat karena termotivasi dengan munculnya kesulitan belajar untuk memahami konsep algoritma dan pemrograman. Beberapa kesulitan yang ada diantaranya sebagai berikut (Sorva dkk., 2013).

1. **Persepsi statis terhadap pemrograman**. Umumnya para pelajar memahami konsep program tertentu (seperti objek, rekursi) hanya merupakan bagian dari kode program, bukan memahaminya sebagai suatu komponen aktif yang utuh saat eksekusi kode program.
2. **Kesulitan memahami cara kerja mesin komputer**. Hambatan utama bagi pemula adalah memahami notasi mesin (Boulay, 1986), yaitu sebuah abstraksi dari komputer dalam eksekusi kode program dengan bahasa pemrograman tertentu. Bahasa pemrograman dan paradigma yang berbeda dapat mengasosiasikan dengan notasi mesin yang berbeda pula.
3. **Salah paham terhadap konsep dasar pemrograman**. Berbagai literatur pendidikan komputasi terdapat banyak laporan tentang kesalahpahaman pelajar yang dilakukan saat mempelajari konsep pemrograman (Clancy dkk., 2003; Sorva, 2012). Permasalahannya menyangkut eksekusi kode program yang tersembunyi dalam notasi mesin dan konsep algoritma yang tidak jelas dalam kode program, seperti referensi variabel dan *pointer*, objek, rekursi, variabel kontrol perulangan, dan sebagainya.
4. **Kesulitan penelusuran dan langkah program**. Kesulitan yang dialami pelajar dengan menelusuri program langkah demi langkah telah banyak dilaporkan di berbagai studi multinasional dan literatur (Sorva dkk., 2013). Laporan tersebut menyatakan bahwa pelajar gagal memahami pernyataan program secara sekuensial dan belum mampu memahami proses pertukaran nilai variabel dalam tiga baris sederhana.

## II.4 Hasil Eksplorasi Perkakas Visualisasi Program

Tinjauan pustaka terhadap penelitian mengenai pengembangan perkakas VP dilakukan untuk memperoleh gambaran mengenai konsep VP dalam representasi visual serta mempelajari arsitektur perkakas yang telah dikembangkan. Kemudian dilakukan eksplorasi yang bertujuan untuk menghindari duplikasi riset yang dilakukan oleh peneliti lain, sehingga dapat diperoleh sebuah ide baru yang belum pernah dikembangkan sebelumnya.

Bahasa pemrograman C dan C++ digunakan dalam perkuliahan struktur data teknik informatika di STEI ITB (Inggriani Liem, 2008). Kedua bahasa pemrograman itu dipilih karena merupakan bahasa yang banyak digunakan dalam industri perangkat lunak saat ini. Oleh sebab itu, kedua bahasa pemrograman ini dipilih untuk dijadikan bahas ajar perkakas VP.

Pada Tabel II.3 menunjukkan perkembangan perkakas VP khusus untuk bahasa pemrograman C dan C++ yang masih aktif pada terakhir dekade tahun ini. Sorva (2012) menjelaskan secara rinci perkembangan 40 perkakas VP pada Bab 11 disertasinya. Dari semua perkembangan perkakas yang masih aktif itu, dipilih hanya perkakas VP yang mendukung bahasa pemrograman C dan C++ untuk ditampilkan pada Tabel II.3.

1. Perkembangan perkakas VP untuk bahasa pemrograman C/C++

| **No.** | **Tahun** | **Nama** | **Dukungan Bahasa** | **Visual Graf** | **Berbasis Web** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1996 | GRASP / jGRASP | Ada, Java, C, C++, Objective-C, VHDL | Binary Tree | - | Aktif |
| 2 | 2000 | The Teaching Machine | C++, Java | **-** | **√** | Aktif |
| 3 | 2002 | PlanAni | Pascal, Java, C, Python | - | - | Aktif ? |
| 4 | 2003 | Jeliot 2000 / Jeliot 3 | Java, C, Python | - | - | Aktif |
| 5 | 2004 | OGRE | C++ | - | - | Non-aktif? |
| 6 | 2005 | VIP | C++ | - | - | Aktif |
| 7 | 2005 | ViLLE | Java, C++, Python, PHP, JavaScript, pseudocode | - | - | Aktif |
| 8 | 2010 | Online Python Tutor | Python, Java, C, C++, Ruby, JavaScript | **-** | **√** | Aktif |
| 9 | 2011 | CSmart | C | - | - | Aktif |

Hasil eksplorasi perkembangan perkakas VP sejak tahun 1996 hingga terakhir tahun 2011 tampak pada Tabel II.3. Terdapat tujuh perkakas yang masih aktif dan dua perkakas yang statusnya tidak jelas. Hanya dua perkakas yang mendukung teknologi web. Perkakas *The Teaching Machine* (*www.theteachingmachine.org*) memang masih aktif dalam pengembangan. Kekurangannya, perkakas ini membutuhkan verifikasi *Plugin Java* yang terpasang di peramban pengguna. Hal ini sangat berbeda dengan perkakas OPT yang dapat langsung dikunjungi (*pythontutor.com*) dan digunakan secara langsung dari peramban pengguna.

Sebuah studi yang dilakukan oleh Ben-Bassat Levy dkk. (Sorva, 2012) dalam perkuliahan pengantar algoritma pemrograman, para pelajar menggunakan perkakas VP bernama *Jeliot 2000* dalam kurun waktu beberapa tahun. Dianalisis bahwa pelajar yang rutin menggunakannya memiliki dampak yang cukup signifikan dalam peningkatan belajar pemrograman, namun muncul istilah “*middle effect*”.

Isitilah “*middle effect*” yang dimaksud terkait dengan visualisasi dan fungsionalitasnya. Walaupun telah lama digunakan, animasi tidak membantu banyak pelajar, karena pelajar yang memiliki kapasitas intelektual cukup tinggi sudah tidak membutuhkannya, sedangkan pelajar yang memiliki kapasitas intelektual rendah akan kesulitan dalam memanfaatkan perkakas. Tetapi untuk kebanyakan pelajar umumnya sebuah model yang konkrit dapat memberikan dampak baik yang cukup signifikan dalam menggapai kesuksesan belajarnya.

## II.5 Pentingnya Belajar Eksekusi Kode dalam Pemrograman

Banyak pelajar di dunia tidak mau untuk belajar eksekusi kode program (Sorva, 2012). Padahal dalam mempelajari pemrograman, harus mempelajari eksekusi kode program agar dapat memahami algoritma dan pemrograman secara utuh. Eksekusi kode program dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari konsep algoritma (Sorva, 2012). Melalui eksekusi kode, pelajar dapat teruji tingkat pemahamannya terhadap konsep algoritma yang telah dipelajarinya. Seperti halnya mengerti dan paham tentang konsep cara berenang, namun tidak pernah praktik langsung ke dalam air untuk berenang.

Ketika sebuah solusi atau algoritma telah dibuat, maka sebuah kode program tetap harus ditulis. Saat ini komputer banyak berperan penting dalam kehidupan manusia sehingga masalah yang dihadapi menjadi lebih besar dan kompleks, yang membutuhkan pengembangan kode program lebih rumit lagi. Tujuan dari penelitian tesis ini adalah untuk menghasilkan perkakas VP yang dapat memberikan kemampuan analisis pemrograman dan algoritma secara simultan kepada para pelajar terhadap graf, sehingga mereka dapat mengembangkan program dengan kualitas dan kuantitas yang maksimum.

Dalam pembelajaran pemrograman sebagai disiplin ilmu yang memerlukan ketelitian tinggi ada dua hal penting yang saling melengkapi (Sorva, 2012), yaitu:

* Pemrograman merupakan pendekatan secara komputasi yang eksekusinya didefinisikan dalam sebuah mesin komputer. Dalam dunia nyata, intuisi pelajar sebenarnya dapat memahami eksekusi program khususnya pada sistem interaktif, seperti potongan-potongan kode program yang dibuat oleh pelajar dapat langsung terlihat hasil eksekusinya. Melalui teknik tersebut pelajar tidak lagi bingung apa yang akan terjadi pada program yang dibuatnya sehingga dapat mempercepat proses pemahamannya.
* Pemrograman merupakan pendekatan secara logika sebagai percabangan dari ilmu logika matematika. Logika bukan merupakan eksekusi program akan tetapi merupakan karakteristik dari program yang berjalan, sehingga dapat disebut sebagai abstraksi pada tingkatan tertinggi. Program merupakan konstruksi matematika yang harus sesuai dengan aturan logika yang berlaku. Pemrograman dengan pendekatan secara logika bagi pelajar lebih sulit untuk dipahami karena definisi dari rincian program setiap barisnya tidak akan ditampilkan.

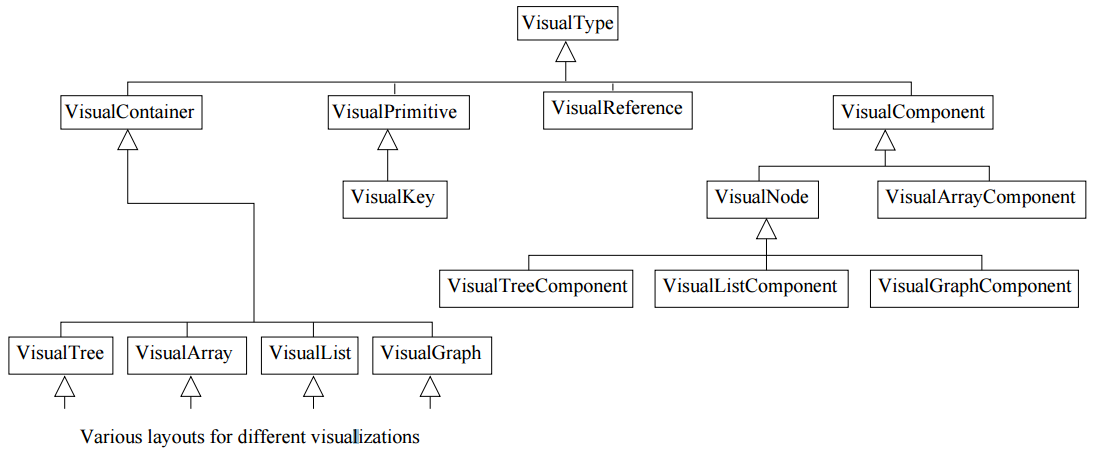
## II.6 Matrix Framework

Pada subbab ini akan dibahas konsep *Matrix framework* yang dikembangkan oleh Korhonen dkk. (2004). *Framework* ini dibangun untuk mendukung desain dan implementasi visualisasi pada tingkat abstrak. Awal pengembangannya diimplementasi menggunakan bahasa pemrograman Java. Kemudian dipadukan dengan *XML library* yang berguna untuk membaca konfigurasi berkas *Matrix*. Berikut ini akan dijelaskan enam bagian yang ada di dalam *Matrix framework*, yaitu visualisasi, struktur, konstruksi visual, animasi, simulasi, dan tampilan antarmuka pengguna.

### II.6.1 Visualisasi

Visualisasi struktur data dalam *Matrix framework* berisi empat konsep dasar visual, yaitu *visual container*, *visual component*, *visual reference* dan *visual data*. *Visual container* berupa struktur yang kompleks, menyimpan nilai variabel (node, indeks, dan lainnya) yang saling terhubung dengan cara tertentu. Kemudian tiap variabel di dalam *visual container* terdapat *visual component* yang berguna untuk memvisualisasikan variabel. *Visual reference* sebagai penghubung antar *visual component* dan *visual data* yang berisi tipe data primitif atau struktur data yang lebih kompleks dengan atribut-atributnya. Atribut kunci yang kompleks direpresentasikan menggunakan *visual container* yang bersarang di dalam *visual component*.

Pada Gambar II.3 menunjukkan hirarki *class Matrix framework* yang digunakan untuk implementasi visualisasi. *Class* pada tingkat kedua dari gambar hirarki tersebut merupakan empat konsep visual yang telah dijelaskan sebelumnya. VisualContainer berhubungan dengan *visual container*, VisualComponent dengan *visual component*, VisualReference dengan *visual reference*, dan VisualPrimitive dengan *visual data*.

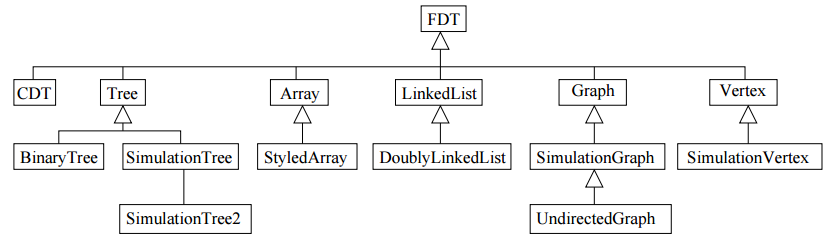


1. Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004)

Visualisasi dengan tipe data berbeda membutuhkan setiap fungsi yang berbeda pula, seperti tipe data *arrays*, graf, *list*, atau pohon. Pada kelas VisualContainer terdiri dari empat sub kelas (lihat Gambar II.3), yaitu VisualTree, VisualArray, VisualList, dan VisualGraph. Setiap sub kelas berkorespondensi dengan salah satu tipe data yang berada di atasnya. Setiap kelas menangani fungsinya masing-masing terhadap visualisasi dan manipulasi tipe data tertentu. Sebagai contoh, kelas VisualGraph berperan ketika pengguna menambahkan sisi (*edge*), sedangkan kelas VisualTree memastikan bahwa graf yang ditampilkan adalah pohon, yang berarti tidak terdapat sisi atau garis yang melintang (*cross*), atau sisi yang mengarah kembali (*back-edges*) ke noktah sebelumnya. Selain itu, pada masing-masing kelas utama terdapat satu atau lebih sub kelas yang menangani tipe data tertentu dalam menguraikan visualisasi dan menggambarkan *layout* tertentu.

### II.6.2 Struktur

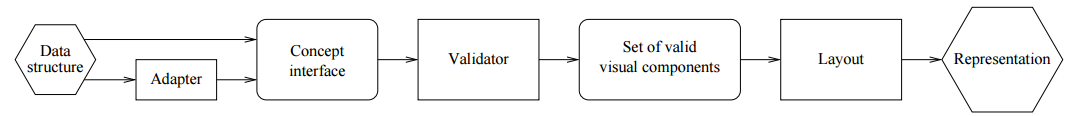
Matrix framework menggunakan sejumlah *concept interfaces* untuk membedakan antara *fundamental data types* (FDT) dan *abstract data types* (ADT). *Concept interfaces* diimplementasi menggunakan interface Java. Hirarki dari struktur *interfaces* yang digunakan dapat dilihat pada Gambar II.4. *Interfaces FDT* merupakan *superclass* (*root*) dari semua konsep *interfaces*. Sebuah objek yang dikenali oleh *Matrix framework* akan dapat divisualisasikan dari implementasi *interfaces FDT* ini. Untuk objek yang tidak dapat dikenali (bukan objek elemen *FDT*), maka akan disimpan dan divisualisasikan sebagai objek data primitif (hanya merepresentasikan sebuah *string* dari elemen tersebut).



1. Turunan hirarki *concept interfaces Matrix* (Korhonen dkk., 2004)

### II.6.3 Konstruksi Visual

Proses representasi visual untuk struktur data diilustrasikan seperti pada Gambar II.5. Proses visualisasi dibagi menjadi tiga tahap, yaitu *Adaptation*, *Validation*, dan *Laying out*. Pada proses *Adaptation* (bagian *Adapter*) terjadi pencocokan visualisasi struktur data terhadap konsep visual *interfaces*. Namun keadaan ini akan membuat sulit bagi *programmer* untuk mengimplementasi *interface* mana yang efisien sesuai kebutuhan visualnya. Oleh karena itu, sistem menyediakan *Adapter* untuk menangani visualisasi terhadap beberapa konsep struktur data seperti yang telah dijelaskan pada Gambar II.3 sebelumnya.



1. Proses konstruksi representasi visual (Korhonen dkk., 2004)

### II.6.4 Animasi

Animasi dalam *Matrix framework* dibagi menjadi dua bagian besar: *forward animation*, berfungsi untuk merepresentasikan secara visual struktur data yang akan terjadi perubahan pada langkah berikutnya, dan *backward animation*, berfungsi untuk menampilkan kembali visualisasi struktur data yang telah terjadi perubahan pada langkah sebelumnya.

Sistem dapat secara otomatis mendukung *forward animation* terhadap semua jenis bentuk struktur data terhadap *concept interfaces* yang telah dijelaskan pada Sub bab II.6.2. Untuk mendukung *backward animation* dibutuhkan fitur lain yang cukup sulit dalam memrogramnya. Sebuah animasi dalam *Matrix framework* disimpan dalm bentuk rangkaian. Sehingga walaupun pengguna menggunakan fitur *backward animation*, sistem akan mengembalikan bentuk struktur data yang relevan sesuai konfigurasi awal. Namun, hal tersebut sangat sulit menyimpan perubahannya yang dibuat secara acak sebagai data objek Java. Oleh sebab itu, variabel yang berisi struktur data harus disimpan ke objek khusus yang disebut MemoryStructure. Objek-objek tersebut juga merepresentasikan beberapa data primitif Java, objek *array*, dan mendukung penyimpanan pada tiap perubahan yang terjadi pada struktur data. Hal itu akan sulit dilakukan pada *framework* jika tanpa objek khusus tersebut untuk menggunakan fitur *backward animation*.

### II.6.5 Simulasi

Setiap operasi simulasi yang dilakukan oleh pengguna untuk menginterpretasikan sebuah visualisasi dilakukan melalui VisualType (lihat Gambar II.2). Dari VisualType ditentukan bentuk mana yang cocok dengan struktur data yang akan direpresentasikan. Kemudian akan memanggil salah satu sub metode yang berada dibawahnya. Setelah sub metode tersebut cocok dan sukses untuk representasi struktur data, VisualType akan ditandai sebagai *class* yang *invalid*. Sistem akan mengabaikan VisualType dan dilanjutkan proses ke sub metode terkait untuk dilakukan *lay out* dan merender ulang visual struktur datanya.

### II.6.6 Tampilan Antarmuka Pengguna

*Graphical user interface (GUI)* atau tampilan antarmuka pengguna merupakan bagian utama aplikasi untuk interaksi antara sistem dan pengguna menggunakan *Matrix framework*. GUI berisi beberapa bagian Panel yang berkaitan dengan fitur visualisasi struktur data atau komponen lain.

## II.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi

Studi literatur yang telah dilakukan memberikan pemahaman mendasar mengenai konsep graf dan macam algoritmanya yang telah berkembang, revolusi proses pembelajaran melalui teknologi web, terminologi visualisasi perangkat lunak, perkakas VP yang telah berkembang hingga saat ini, serta peran penting eksekusi kode program dalam pembelajaran pemrograman. Dari hasil eksplorasi perkakas VP, diperoleh bahwa tidak ada satu pun perkakas VP yang beroperasi berbasis web dengan bahasa C atau C++ yang dapat menampilkan graf. Sehingga ini menjadi peluang untuk penelitian lebih lanjut dalam pengembangannya.

OPT menjadi dasar pengembangan perkakas, karena selain telah mendukung basis web, perkakas ini juga bersifat bebas dan bersumber kode terbuka. Perkakas ini juga masih dalam proses pengembangan dan penelitian oleh pengembangnya[[1]](#footnote-1). Jadi, masih terdapat banyak peluang dan celah untuk diteliti lebih lanjut agar dapat dimanfaatkan dengan sebaik-baiknya. Mengingat teknologi web dan aplikasi piranti bergerak terus berkembang pesat hingga dekade tahun terakhir ini[[2]](#footnote-2).

# Bab III Analisis Masalah dan Perumusan Skema Graf

Bab ini menganalisis perkakas VP yang sedang dikembangkan kemudian diuraikan rumusan metode atau skema yang diperlukan untuk menerjemahkan kode program menjadi visual graf. Selanjutnya dilakukan penentuan ruang lingkup pengembangan perkakas terhadap proses visualisasi graf.

## III.1 Gambaran Umum Fitur Perkakas Online Python Tutor

*Online Python Tutor* atau disingkat OPT (Guo, 2013) merupakan perkakas VP bersifat bebas dan bersumber terbuka. Perkakas ini memiliki banyak fitur VP yang mendukung berbagai macam bahasa pemrograman, seperti Python, Java, C, C++, Ruby, JavaScript, TypeScript dan masih terus dikembangkan[[3]](#footnote-3). Ketika penelitian tesis ini dibuat, OPT telah berkembang pada versi 5 yang dirilis pada tanggal 27 Juli 2016. Penelitian aktif dilakukan pada versi 5 dengan menggunakan bahasa pemrograman *TypeScript* sebagai dasar pembangunan perkakas OPT.



1. Tampilan Antarmuka OPT untuk Bahasa Pemrograman C

Pada Gambar III.1 ditunjukkan tampilan antarmuka perkakas OPT untuk bahasa pemrograman C. Fungsi nomor 1 adalah untuk berbagi antarmuka bagi pengguna lain yang ingin belajar pemrograman secara kolaborasi. Ketika tombol tersebut diklik, tampilan tampak seperti pada Gambar III.2. Fungsi ini menggunakan kode pustaka dari *TogetherJS[[4]](#footnote-4)* yang dikembangkan oleh *Mozilla Labs* secara bebas dan terbuka. Pengguna lain dapat ikut bergabung dengan mengunjungi pranala yang dibagikan oleh pengguna pertama yang mengaktifkan fitur ini.



1. Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman

Fungsi nomor 2 pada Gambar III.1 adalah untuk memasukkan kode program. Pelajar dapat mulai belajar bahasa pemrograman dengan mengetikkan kode di form editor yang tersedia. Selanjutnya pelajar dapat menekan tombol pada nomor 3 untuk melihat simulasi dan visualisasi dari kode program yang telah dibuatnya, seperti pada Gambar III.3.



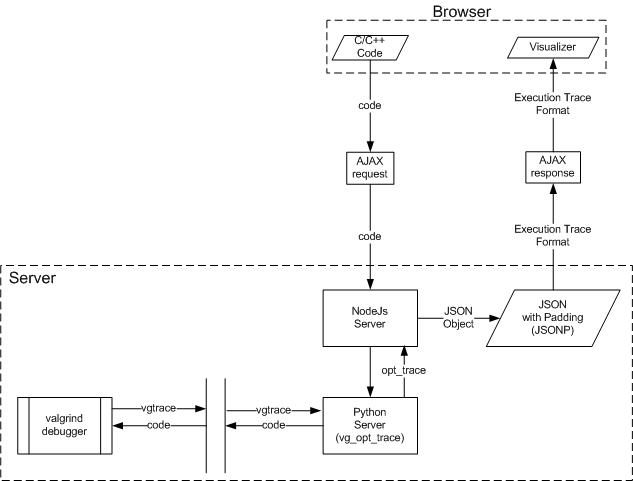
1. Tampilan Simulasi dan Visualisasi Kode C Pointer

Sebelum merumuskan metode atau skema, subbab berikut akan menguraikan arsitektur dasar pembangunan perkakas OPT yang mendukung proses VP dalam bahasa pemrograman C dan C++. Namun tidak menutup kemungkinan skema atau metode yang dirumuskan akan dapat dimplementasikan untuk bahasa pemrograman lainnya.

### III.1.1 Arsitektur Perkakas OPT

Penjelasan arsitektur ini berguna untuk memberikan gambaran umum tentang cara kerja atau alur sistem didalamnya, sehingga dapat menentukan metode yang optimal. Selain itu, dapat dijadikan dasar dalam mengembangkan sistem dan menunjang modularisasi pada jangka waktu yang panjang.

Pada Gambar III.4 ditampilkan diagram alur komunikasi antara peramban milik pengguna dengan peladen. Pertukaran informasi kode program dilakukan dengan fitur permintaan *AJAX*.



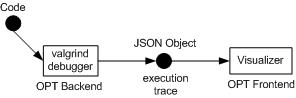
1. Arsitektur OPT untuk visualisasi C dan C++

Untuk lebih jelas tahapan proses pada Gambar III.4 di atas, berikut langkah-langkah yang terjadi ketika pengguna melakukan “Eksekusi Visualisasi” pada kode programnya.

1. Pada antarmuka peramban, sebenarnya kode tersebut dikirim sebagai tipe data *string* ke peladen dengan metode *AJAX request*.
2. Kemudian peladen mengeksekusi kode tersebut dengan *NodeJS* yang dikombinasikan dengan peladen *Python* untuk menghasilkan galur v*algrind*.
3. Peladen Python mengubah format galur eksekusi sebagai data objek *JSON* sebelum dikembalikan ke peladen *NodeJS*.
4. *JSONP* digunakan untuk memperoleh data dengan permintaan *AJAX* yang berbeda domain atau alamat IP.
5. Respon *AJAX* diterima oleh *visualizer* kemudian dibaca sesuai format yang ditentukan. Maka diperoleh visualisasi kode program yang tampil di peramban pelajar.
6. Ketika pelajar menekan tombol “*Forward*” atau “*Back*” (lihat Gambar III.3) proses visualisasi akan membaca poin indeks yang berada pada galur eksekusi sesuai dengan banyak langkah yang telah terbentuk.

### III.1.2 Komponen Perkakas OPT

Perkakas OPT dapat disederhanakan menjadi beberapa inti komponen dengan representasi tingkat tinggi seperti tampak pada Gambar III.5 berikut ini.



1. Ilustrasi Sederhana Proses Perkakas OPT

*OPT backend* berfungsi untuk respon terhadap terbentuknya galur eksekusi program. Kemudian galur eksekusi diubah menjadi standar format tertentu yang dipaketkan berbentuk objek *JSON*. *OPT frontend* (antarmuka peramban) menerjemahkan objek *JSON* menjadi visualisasi yang dikombinasikan dengan perkakas pustaka D3. Maka visualisasi dapat tampil kepada pelajar seperti pada Gambar III.3.

### III.1.3 Proses Membentuk Galur Eksekusi Program

Ketika kode program dikirim ke *OPT backend*, kode C/C++ dieksekusi dengan *valgrind framework*. Kemudian dari *valgrind* terbentuk galur yang sudah dimodifikasi oleh pengembangnya. Versi *valgrind* yang digunakan adalah 3.11.0 dengan modifikasi (versi asli bisa unduh di *www.valgrind.org*) beberapa baris kode agar dapat menghasilkan galur eksekusi program. Beberapa berkas yang dimodifikasi adalah sebagai berikut:

|  |  |
| --- | --- |
| 1 | valgrind-3.11.0\memcheck\**mc\_main.c**  (Baris ke 54, 55, 3789, 5693, 5715-5718, 7394-7406, 7646-7647, 7761-7763) |
| 2 | valgrind-3.11.0\include\**pub\_tool\_debuginfo.h**  (Baris ke 36-37, 124-132, 191, 198, 214, 285-408) |
| 3 | valgrind-3.11.0\coregrind\m\_debuginfo\**debuginfo.c**  (Baris ke 74-1458, 5043-5159, 5472, 5594-5719, 5846) |
| 4 | valgrind-3.11.0\memcheck\**mc\_translate.c**  (Baris ke 6261-6481, 6665, 6708-6716, 6795-6811) |
| 5 | valgrind-3.11.0\coregrind\m\_debuginfo\**tytypes.c**  (Baris ke 47-51, 328-864) |
| 6 | valgrind-3.11.0\coregrind\m\_debuginfo\**priv\_tytypes.h**  (Baris ke 160-165) |
| 7 | valgrind-3.11.0\**config.h** (Baris ke 41) |
| 8 | valgrind-3.11.0\memcheck\**mc\_include.h** (Baris 142-156) |

Berkas-berkas valgrind tersebut dianalisis perbedaannya dengan menggunakan perkakas *WinMerge* (*winmerge.org*). Tekniknya dengan membandingkan valgrind versi asli yang diunduh dari *www.valgrind.org*.

Untuk dapat menghasilkan galur eksekusi program, kode program dikirim ke server yang diolah oleh *ExpressJS*. Kemudian dengan program *Python* dihubungkan dengan *Pipe* ke *terminal* untuk di-*compile* dalam *valgrind debugger*. Dari sini menghasilkan galur khusus dari valgrind disebut *vgtrace*. Selanjutnya dikonversi ke bentuk *JSON* atau *JSONP* oleh *ExpressJS* dan dikembalikan sebagai respon permintaan *AJAX*. Ilustrasi lengkap dapat dilihat pada Gambar III.4

### III.1.4 Format Galur Eksekusi Program

*OPT back-end* menyimpan informasi galur eksekusi dalam standar format tertentu yang kemudian dapat digunakan untuk berinteraksi dengan *visualizer* di *OPT front-end.* Sebagai ilustrasi format galur eksekusi program yang dimaksud dapat dilihat pada Gambar III.6 berikut ini.



1. Ilustrasi Format Galur Eksekusi Program

Bagan pada Gambar III.6 di atas mengilustrasikan posisi eksekusi kode pada saat masih kosong (proses dini). Galur eksekusi program disimpan dalam bentuk objek JSON yang didefinisikan dalam dua bagian besar, yaitu:

1. “***code***” berbentuk string, yang berisi kode program dari pelajar.
2. “***trace***” berbentuk objek-objek larik, yang setiap objek merepresentasikan posisi eksekusi kode sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 1 | { |  |
| 2 | ordered\_globals: [], | // sebuah larik yang merepresen-tasikan urutan visual untuk atribut global |
| 3 | stdout: “”, | // total standar output dari kode program pada posisi eksekusi kode saat ini |
| 4 | func\_name: “<module>”, | // fungsi saat ini yang sedang diekseskusi |
| 5 | stack\_to\_render: [], | // merupakan *list* dari objek-objek, setiap objek direpresentasikan sebagai sebuah *stack frame* |
| 6 | globals: {}, | // sebuah |
| 7 | heap: {}, |  |
| 8 | line: 1, |  |
| 9 | event: “step\_line” |  |
| 10 |  |  |
| 11 | } |  |

### III.1.5 Fitur Pustaka *Data-Driven Documents* (D3)

Teknologi visualisasi utama yang digunakan oleh OPT adalah *Data-Driven Documents* (D3) (Bostock dkk., 2011) yang merupakan salah satu pustaka terpopuler untuk implementasi visualisasi berbasis web. Format galur eksekusi program dibaca oleh D3 untuk mendeklarasikan pemetaan data dan atribut yang diperlukan oleh elemen *Document Object Model* (DOM) pada laman *Hyper Text Markup Language* (HTML). Peran pustaka D3 tampak pada Gambar III.7 berikut ini.



1. Peran Utama D3 sebagai *Framework* Visualisasi

D3 sangat fleksibel terhadap abstraksi data tingkat tinggi. Pustaka ini juga mampu memvisualisasikan berbagai macam data, bahkan cocok untuk visualisasi dengan tingkat kerumitan yang tinggi.

## III.2 Analisis Masalah Perkakas

Beberapa permasalahan muncul saat mulai mengembangkan perkakas OPT dan penambahan fitur graf. Eksplorasi perkakas OPT dilakukan dengan meninjau proses kerja sistem, alur kode program dan membaca baris-baris kode pengembangnya. Versi terkahir yang dikembangkan telah bermigrasi ke bahasa pemrograman *Typescript*. Pengembang berargumen agar dapat dikembangkan lebih lanjut dengan teknik modularisasi[[5]](#footnote-5). Ini bermanfaat ketika peneliti atau pengembang lain yang ingin melanjutkan dengan penambahan fitur dapat dilakukan dengan mudah. Namun, versi terakhir belum sepenuhnya memiliki pola desain pengembangan yang standar. Selain itu, dokumentasi untuk pengembangan masih sangat terbatas dan belum diatur dengan baik.

### III.2.1 Analisis Masalah dan Peluang Pemecahan

Sebagai langkah awal pengembangan perkakas dengan metode eksplorasi, diperoleh beberapa poin penting, diantaranya terdapat pada Tabel III.1.

1. Hasil Analisis dan Peluang Pemecahan Masalah

| **No.** | **Domain Persoalan** | **Perkakas OPT** | **Rekomendasi Perbaikan** | **Peluang Pengembangan** |
| --- | --- | --- | --- | --- |
| 1 | Aspek pertukaran informasi berupa JSON antara *client* dan *server* | Hanya mampu menampung maksimal 1000 iterasi | Memerlukan koneksi *Distributed Data Protocol* (DDP) yang dimiliki oleh pustaka *MeteorJS*[[6]](#footnote-6). | Peningkatan kapasitas fungsi komunikasi antar *client* dan *server* serta mendukung versi *mobile* dan *desktop* |
| 2 | Aspek Pola Desain (*Design Pattern*) | Masih tahap awal migrasi ke pemrograman *TypeScript* | Dapat menggunakan pola *Hierarchical Model View Controller* (HMVC) | Perkakas dapat dikembangkan dengan modularisasi |
| 3 | Aspek Simulasi dan Visualisasi | Hanya mampu menampilkan visual dengan tipe data primitif | Menggunakan *Matrix Framework* untuk mendukung visualisasi graf dengan dukungan *MeteorJS*. | Dapat mendukung visual struktur data lain dengan bahasa pemrograman yang lebih variatif |

### III.2.2 Analisis Optimasi Perkakas

## III.3 Perumusan Skema Graf

### III.3.1 Menggunakan Format Galur Eksekusi

### III.3.2 Pembaruan Format Galur Eksekusi Program

## III.4 Penunjang Pengembangan Perkakas

Untuk menunjang pengembangan perkakas diperlukan seperangkat kebutuhan yang harus dipenuhi, baik dari sisi perangkat lunak yang mutakhir maupun perangkat kerasnya. Subbab berikut akan memerinci spesifikasi kebutuhan sistem.

### III.4.1 Kebutuhan Perangkat Lunak

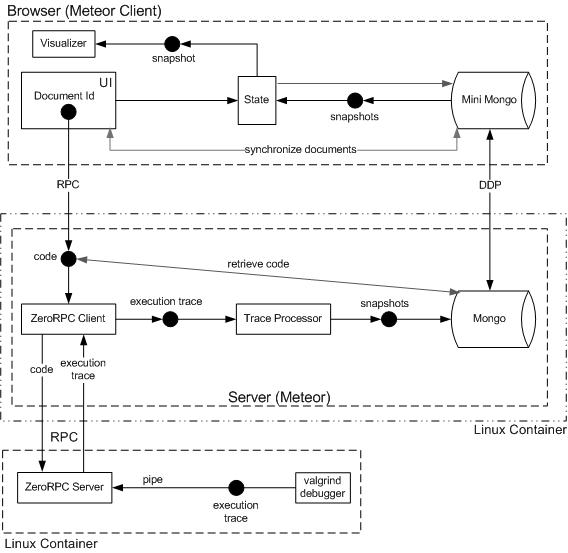
*Typescript* merupakan bahasa pemrograman yang cocok untuk pengembangan perkakas ini. Bahasa ini dibuat oleh Microsoft pada tahun 2012 untuk menunjang pengembangan aplikasi berbasis web JavaScript.

### III.4.2 Kebutuhan Perangkat Keras

# Bab IV Pengembangan dan Implementasi Perkakas

Bab ini menjelaskan proses pengembangan perkakas yang terdiri dari perbaikan arsitektur, pembaruan format galur eksekusi dan pola desain untuk pengembangan lebih lanjut. Salah satu tujuan pengembangan perkakas ini selain untuk dukungan visualisasi graf adalah dapat dikembangkan dengan modularisasi. Bersifat modular akan bermanfaat untuk penelitian dan penambahan fitur-fitur selanjutnya.

## IV.1 Perbaikan Arsitektur Perkakas



1. Perbaikan Arsitektur untuk Pengembangan Perkakas OPT

## IV.2 Format Galur Eksekusi Kode

## IV.3. Implementasi Perkakas

# Bab V Pengujian dan Evaluasi Perkakas

## V.1 Pengujian Perkakas

### V.1.1 Simulasi Mengajar / Studi Kasus

### V.1.2 Pengujian Secara Luring dan Daring

### V.1.3 Wawancara Mendalam

## V.2 Evaluasi Perkakas

### V.2.1 Tujuan Pengembangan Perkakas

### V.2.2 Indikator Keberhasilan Perkakas

### V.2.3 Perbandingan Perkakas Lama dan Baru

### V.2.4 Kelebihan dan Kekurangan Perkakas

# Bab VI Kesimpulan dan Saran

## VI.1 Kesimpulan

## VI.2 Saran

# DAFTAR PUSTAKA

Bonk, C.J. (2009): *The world is open: how Web technology is revolutionizing education*, 1st ed, San Francisco, Calif, Jossey-Bass.

Bostock, M., Ogievetsky, V. and Heer, J. (2011): D$^3$ data-driven documents, *IEEE Trans. Vis. Comput. Graph.*, **17**, 2301–2309.

Boulay, B.D. (1986): Some Difficulties of Learning to Program, *J. Educ. Comput. Res.*, **2**, 57–73.

Cetin, I. (2016): Preservice Teachers’ Introduction to Computing: Exploring Utilization of Scratch, *J. Educ. Comput. Res.*, 735633116642774.

Cetin, I. and Andrews-Larson, C. (2016): Learning sorting algorithms through visualization construction, *Comput. Sci. Educ.*, **26**, 27–43.

Clancy, M., Titterton, N., Ryan, C., Slotta, J. and Linn, M. (2003): New roles for students, instructors, and computers in a lab-based introductory programming course, *ACM SIGCSE Bulletin*, ACM, 132–136, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=611951 (accessed 21 July 2016).

Clark, R.C. and Mayer, R.E. (2011): *E-learning and the science of instruction : proven guidelines for consumers and designers of multimedia learning - 3rd ed.*, San Francisco, CA, Pfeiffer.

Gračanin, D., Matković, K. dan Eltoweissy, M. (2005): Software visualization, Innov. Syst. Softw. Eng., 1, 221–230.

Guo, P.J. (2013): Online python tutor: embeddable web-based program visualization for cs education, Proceeding of the 44th ACM technical symposium on Computer science education, ACM, 579–584.

Harary, F. (1969): *Graph theory*, Canada, USA, Addison-Wesley Publishing Company, Inc.

Helminen, J. dan Malmi, L. (2010): Jype-a program visualization and programming exercise tool for Python, Proceedings of the 5th international symposium on Software visualization, ACM, 153–162.

Inggriani Liem (2008): *Diktat Struktur Data*, Edisi 2008, Bandung, Teknik Informatika, Institut Teknologi Bandung.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. and Tenhunen, P. (2004): *Matrix - a framework for interactive software visualization*, Research Report TKO-B 154/04, Department of Computer Science and Engineering, Helsinki University of Technology, 26–35.

Margulieux, L.E., Catrambone, R. and Guzdial, M. (2016): Employing subgoals in computer programming education, *Comput. Sci. Educ.*, **26**, 44–67.

Mayer, R.E. (2014): *The Cambridge handbook of multimedia learning - Second Edition*, New York, USA, Cambridge University Press.

Pathania, U. dan Singh, A. (2014): Visualization Tool for Tree and Graph Algorithms with Audio Comments, Int. J. Softw. Web Sci. IJSWS, 14, 51–58.

Piteira, M. and Costa, C. (2013): Learning computer programming: study of difficulties in learning programming, *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ACM, 75–80, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=2503871 (accessed 8 September 2016).

Rinaldi Munir (2010): *Matematika Diskrit Edisi 3*, Bandung, Informatika Bandung.

Sorva, J. (2012): *Visual program simulation in introductory programming education*, Aalto University publication series Doctoral dissertations, Espoo, Aalto Univ. School of Science.

Sorva, J. dan Sirkiä, T. (2010): UUhistle: a software tool for visual program simulation, Proceedings of the 10th Koli Calling International Conference on Computing Education Research, ACM, 49–54.

Sorva, J., Karavirta, V. dan Malmi, L. (2013): A review of generic program visualization systems for introductory programming education, ACM Trans. Comput. Educ. TOCE, 13, 15.

Valiente, G. (2002): *Algorithms on Trees and Graphs*, Berlin, Heidelberg, Springer Berlin Heidelberg, diperoleh melalui situs internet: http://link.springer.com/10.1007/978-3-662-04921-1 (accessed 15 September 2016).

Wang, P.S. (2015): *From computing to computational thinking*, CRC Press, Florida, USA.

Ware, C. (2004): *Information visualization: perception for design, 2nd edition*, 2nd edition, San Francisco, Kanada, Elsevier Inc.

1. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-1)
2. https://kominfo.go.id/index.php/content/detail/3415/Kominfo+%3A+Pengguna+Internet+di+In donesia+63+Juta+Orang/0/berita\_satker [↑](#footnote-ref-2)
3. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-3)
4. https://togetherjs.com/ [↑](#footnote-ref-4)
5. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-5)
6. https://www.meteor.com/ [↑](#footnote-ref-6)