STRATEGI PEMBELAJARAN UNTUK MENINGKATKAN KETERAMPILAN PEMROGRAMAN DAN BERPIKIR KOMPUTASI: SEBUAH STUDI LITERATUR

Yeni Anistyasari[1)], Ekohariadi[2)], dan Munoto[3)]
[1, 2)]Jurusan Teknik Informatika, Universitas Negeri Surabaya
Surabaya, Indonesia
[3)]Jurusan Teknik Elektro, Universitas Negeri Surabaya
Surabaya, Indonesia
yenian@unesa.ac.id[1)], ekohariadi@unesa.ac.id[2)], munoto@unesa.ac.id[3)]

ABSTRAK

Berpikir komputasi dianggap sebagai kompetensi penting yang diperlukan untuk beradaptasi dengan teknologi masa depan. Oleh karena itu, berbagai penelitian tentang berpikir komputasi dilakukan oleh para peneliti. Namun, sedikit sekali penelitian yang mengulas tentang bagaimana strategi pembelajaran yang sesuai untuk diterapkan di mata kuliah pemrograman dasar guna meningkatkan pengetahuan dan berpikir komputasi. Pada artikel ini, dilakukan meta-review dari berbagai penelitian sebelumnya yang telah dipublikasikan di jurnal akademik pada tahun 2006-2019 tentang cara belajar-mengajar, media pembelajaran, dan bahasa pemrograman. Dari hasil studi literatur ditemukan bahwa berpikir komputasi telah diaplikasikan pada ilmu komputer dan bidang ilmu lain. Sebagian besar penelitian menggunakan Project-Based Learning, Problem-Based Learning, Cooperative Learning, dan Game-based Learning. Sebagian besar penelitan berfokus pada pelatihan keterampilan pemrograman dan komputasi matematis, sementara beberapa mengadopsi mode pengajaran lintas domain untuk memungkinkan siswa mengelola dan menganalisis materi berbagai domain dengan komputasi.

***Kata Kunci**: Berpikir komputasi, keterampilan pemrograman, dan strategi pembelajaran*

ABSTRACT

These instructions give you guidelines for preparing JVTE (Journal of Vocational and Technical Education) papers. Use this document as a template if you are using Microsoft Word 6.0 or later. The electronic file of your paper will be formatted further by JVTE editorial board. Paper titles should be written in uppercase. Avoid writing long formulas with subscripts in the title; short formulas that identify the elements are fine (e.g., "Nd–Fe–B"). Do not write "(Invited)" in the title. Full names of authors are preferred in the author field, but are not required. If you have to shorten the author name, leave first name and last name unshorten. Put a space between authors' initials. Do not cite references in the abstract. The length of abstract must between 200 – 250 words.

***Keywords**: Computational thinking, learning strategy, and programming skills*

I. PENDAHULUAN

Berpikir komputasi pertama kali dipresentasikan oleh Papert (1990), dan sejak itu definisi, pengajaran, dan evaluasi telah dibahas oleh berbagai peneliti (Grover & Pea, 2013). Wing (2006) menekankan bahwa berpikir komputasi adalah salah satu kecakapan hidup sehari-hari yang dibutuhkan setiap orang, dan bukan hanya keterampilan pemrograman yang hanya digunakan oleh para ilmuwan komputer. Wing (2010) lebih lanjut mendefinisikan pemikiran operasional sebagai proses penyelesaian masalah, sehingga agen pemrosesan pesan dapat dieksekusi secara efektif dan penyelesaian masalah. Komputer dapat membantu memecahkan masalah melalui dua langkah berikut: (1) mempertimbangkan langkah-langkah untuk menyelesaikan masalah, kemudian menggunakan keterampilan teknis untuk mengontrol komputer dalam menyelesaikan masalah. Misalnya, seseorang harus memahami rumus matematika dan menjelaskan masalahnya dan menggunakan metode atau rumus sederhana untuk menyelesaikan masalah melalui perhitungan komputer. Selain itu, saat membuat animasi, desainer harus merencanakan cerita dan cara pemotretan sebelum menggambar animasi komputer, dan menyelesaikan tugas menggunakan perangkat lunak dan perangkat keras komputer. Dalam dua contoh ini, berpikir komputasi adalah proses berpikir yang perlu dilakukan sebelum memulai operasi komputer dan mesin.

Berpikir komputasi menjelaskan proses dan metode yang digunakan untuk mengoperasikan sistem. Ini berfokus pada bagaimana orang memecahkan atau meneliti masalah menggunakan komputer (Wing, 2008), bukan pada

perangkat keras komputer atau meniru mode berpikir komputer. Selain itu, Wing (2008) menganggap bahwa berpikir komputasi tidak hanya pusat penyelesaian masalah, tetapi juga mengembangkan dan mengidentifikasi masalah. Ini juga mengingatkan pada inovasi awal dalam STEM dan mata pelajaran lain (Cheung, 2013), yang berarti bahwa itu tidak hanya memungkinkan komputer memahami cara untuk memecahkan masalah, tetapi juga membantu orang untuk memahami solusi dan masalah. Dengan kata lain, berpikir komputasi tidak selalu membutuhkan mesin, tetapi orang dapat menghasilkan proses berpikir komputasi dengan memanipulasi mesin (Wing, 2008). Oleh karena itu, Wing (2008) menganggap bahwa berpikir komputasi tidak lagi diperlukan untuk peserta didik di jurusan yang berkaitan dengan ilmu komputer, tetapi juga sangat diperlukan untuk peserta didik di domain lain. Para guru saat ini harus membuat dan mempromosikan fasilitas untuk belajar berpikir komputasi. Heintz dkk. (2016) menunjukkan bahwa berpikir komputasi yang telah dilakukan dalam pelajaran komputer atau pelajaran lain untuk melatih berpikir komputasi siswa di banyak negara yang berbeda. Kesulitan untuk membuat model atau menyalin metode pengembangan berpikir komputasi disebabkan perbedaan dalam sistem dan budaya pendidikan nasional. Oleh karena itu, banyak negara telah mulai menumbuhkan kemampuan berpikir komputasi dan kemampuan pemrograman.

Pemrograman digunakan pertama kali pada tahun 1960-an ketika pemrograman Logo pertama kali diperkenalkan sebagai kerangka kerja potensial untuk mengajar matematika (Feurzeig & Papert, 2011). Dalam Logo, siswa memindahkan kura-kura (panah) di layar dengan mengeluarkan perintah seperti FD 100 (meneruskan 100). Dalam buku seminalnya 'Mindstorms: Anak-anak, komputer dan ide-ide kuat', Papert (1980) menganjurkan penggunaan mode penemuan konstruksionis untuk mempelajari Logo. Namun demikian, Logo tidak populer di sekolah-sekolah umum pada 1980-an, mungkin karena ketidakcocokan antara pendekatan yang dapat ditemukan dan budaya sekolah behaviourist yang lebih konvensional saat itu (Agalianos, Noss, & Whitty, 2001). Papert (1980) mengklaim bahwa pengalaman pemrograman Logo dapat mengembangkan keterampilan berpikir intelektual yang kuat di antara anak-anak. Berlawanan dengan klaimnya, studi empiris pemrograman Logo tidak menemukan bukti konklusif untuk meningkatkan keterampilan berpikir anak-anak (Kurland, Pea, Clement, & Mawby, 1986; Pea, 1983). Setelah Logo, penggunaan pemrograman untuk mengajarkan keterampilan berpikir dalam tidak dilaporkan secara luas. Namun, dalam beberapa tahun terakhir, ada minat baru dalam memperkenalkan pemrograman untuk siswa (Grover & Pea, 2013; Kafai & Burke, 2013). Ini didorong oleh ketersediaan bahasa pemrograman visual yang mudah digunakan seperti Scratch (Burke, 2012; Lee, 2010), Toontalk (Kahn, Sendova, Sacristán, & Noss, 2011), Pencipta Kereta Api (Denner, Werner, & Ortiz, 2012) dan Alice (Graczyn´ ska, 2010). Banyak dari bahasa pemrograman baru ini seperti Scratch dan Alice telah dimodelkan setelah aspek Logo (Utting, Cooper, Kölling, Maloney, & Resnick, 2010).

Selama pemrograman, siswa dihadapkan pada berpikir komputasi, sebuah istilah yang dipopulerkan oleh Wing (2006). Ini melibatkan penggunaan konsep ilmu komputer seperti abstraksi, debugging, remixing dan iterasi untuk menyelesaikan masalah (Brennan & Resnick, 2012; Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011; Wing, 2008). Bentuk pemikiran ini dapat dianggap sebagai hal mendasar bagi siswa karena memerlukan "berpikir pada banyak abstraksi" (Wing, 2006, hal. 35). Lebih penting lagi, berpikir komputasi sejalan dengan banyak aspek kompetensi abad ke-21 seperti kreativitas, pemikiran kritis, dan pemecahan masalah (Ananiadou & Claro, 2009; Binkley et al., 2012). Dengan demikian, tidak mengherankan bahwa banyak guru menyatakan bahwa pemrograman penting bagi siswa di era ini (Kafai & Burke, 2013; Margolis, Goode, & Bernier, 2011; Resnick et al., 2009). Minat yang dihidupkan kembali dalam pemrograman untuk pengaturan menunjukkan kebutuhan untuk mempertimbangkan bagaimana hal itu dapat lebih baik terkait dengan jenis hasil pendidikan yang berpotensi dapat dipupuk. Beberapa hasil yang disarankan oleh para peneliti adalah kemampuan untuk berpikir lebih sistematis (Kafai & Burke, 2013) dan pengembangan keahlian matematika dan ilmiah (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). Namun, dalam literatur saat ini, ada kelangkaan makalah yang mengeksplorasi berpikir komputasi melalui pemrograman dalam konteks (Grover & Pea, 2013) karena studi pemrograman ini lebih sering diperiksa untuk siswa tersier yang mengambil kursus ilmu komputer (misalnya, Katai & Toth, 2010; Moreno, 2012). Karena itu, dalam makalah ini, kami mencoba untuk memeriksa studi empiris yang diterbitkan yang melibatkan siswa dalam konteks pendidikan tinggi dan sehingga memperoleh wawasan tentang berpikir komputasi melalui pemrograman untuk kurikulum.

II. SUMBER DATA

Sumber data yang digunakan dalam penelitian ini adalah SCOPUS. Pertama, kata kunci yang digunakan adalah computational thinking untuk mencari topik makalah, abstrak, dan kata kunci dalam database. Ditemukan total 2.230 artikel. Kedua, ditetapkan periode pencarian dari 1 Januari 2016 hingga Februari, 2019. Setelah menetapkan periode waktu yang dipublikasikan, hasil pencarian menunjukkan bahwa ada 212 artikel berpikir komputasi antara periode ini. Kemudian, diatur jenis artikel untuk terbitan makalah jurnal akademik, makalah jurnal akademik (dalam pers), dan buku-buku, yang memberikan total 162 makalah atau buku jurnal. Akhirnya, pengecualian terhadap artikel jurnal non-SCI dan non-SSCI, memberikan 120 artikel untuk analisis tindak lanjut. Informasi yang diperoleh kemudian dikelompokkan.

III. HASIL STUDI LITERATUR

A. Definisi Berpikir Komputasi

Istilah berpikir komputasi dipopulerkan oleh Wing (2006). Dalam artikelnya tentang berpikir komputasi, ia berpendapat bahwa berpikir komputasi:

> *"represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use"*

Terjemahannya adalah "mewakili sikap dan keterampilan yang berlaku universal yang ditetapkan setiap orang, tidak hanya ilmuwan komputer, akan bersemangat untuk belajar dan menggunakannya". Sejak itu, berpikir komputasi telah mendapatkan daya tarik di AS. Namun, definisi berpikir komputasi masih tetap diperdebatkan karena tidak ada wacana dominan yang berkuasa (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013). Sebagai contoh, International Society for Technology in Education (ISTE) memandang berpikir komputasi sebagai pemikiran algoritmik dengan alat otomatisasi dan representasi data dengan penggunaan simulasi. Di sisi lain, National Research Council (NRC) merekomendasikan matematika dan berpikir komputasi untuk menjadi salah satu dari delapan praktik penting untuk dimensi ilmiah dan teknik yang diuraikan dalam "Framework for K-12 Science Education" (NRC, 2012). Dalam kerangka ini, matematika dan berpikir komputasi melibatkan penggunaan komputer untuk mewakili variabel fisik dan hubungan di antara mereka.

Berpikir komputasi adalah keterampilan universal dalam hidup; itu tidak lagi hanya kesan stereotip dari keterampilan yang dibutuhkan oleh ilmuwan komputer. Setiap orang harus memiliki sikap positif dalam memahami dan menggunakan keterampilan ini dalam kehidupan sehari-hari (Wing, 2006). Kemampuan dan keterbatasan berpikir komputasi didasarkan pada pemrosesan komputasi, terlepas dari apakah pikiran orang atau komputer yang digunakan untuk memproses masalah. Pada tahap pembelajaran awal, anak-anak seharusnya tidak hanya dilatih dalam kemampuan calistung (membaca, menulis, dan berhitung), tetapi juga harus diajarkan bagaimana menerapkan berpikir komputasi ke dalam praktik dan bagaimana melakukan analisis logis (Wing, 2006). Ada empat keterampilan operasional berpikir komputasi, yaitu menyederhanakan, menanamkan, mengubah, dan simulasi. Untuk mengubah masalah menjadi masalah yang mudah dipahami (Wing, 2006), berpikir komputasi menggunakan konsep dasar ilmu komputer untuk memecahkan masalah, merancang sistem, dan mengubahnya menjadi mode berpikir yang dapat dipahami oleh manusia (Wing, 2006)). Secara bersamaan, berpikir komputasi memungkinkan untuk mengadopsi mode berpikir yang mirip dengan ilmuwan komputer ketika menghadapi masalah (Grover & Pea, 2013).

Wing (2008) lebih lanjut mendefinisikan berpikir komputasi sebagai (1) sebuah konseptualisasi daripada proses pengembangan bahasa pemrograman. Oleh karena itu, para siswa diminta untuk menerapkan berbagai lapisan berpikir abstrak. Berpikir komputasi tidak terbatas untuk menggunakan komputer untuk belajar (Wing, 2008); (2) proses logis lebih dipilih daripada perilaku berulang dari operasi mekanik. Oleh karena itu, orang dapat lebih fleksibel dalam menggunakan keahlian mereka sendiri melalui berpikir komputasi; (3) cara berpikir manusia, bukan mode perhitungan komputer. Dengan kata lain, berpikir komputasi adalah cara untuk memecahkan masalah manusia, tidak hanya menyalin mode berpikir komputer, karena manusia lebih pintar dan lebih imajinatif daripada komputer (Wing, 2008); (4) kombinasi pemikiran matematika dan pemikiran teknik untuk memperluas fondasi matematika; (5) produk pemikiran yang selesai, yang membantu memecahkan masalah dalam hidup, mengelola perilaku kehidupan sehari-hari dan keterampilan komunikasi dan interaksi dengan orang lain; dan (6) keterampilan

dasar dalam kehidupan sehari-hari, bukan filsafat abstrak.

Untuk ISTE dan NRC, siswa dapat dianggap menunjukkan berpikir komputasi meskipun mereka tidak menciptakan dengan alat teknologi. Sebaliknya, pemrograman melibatkan siswa yang menunjukkan berpikir komputasi melalui konstruksi artefak (Kafai & Burke, 2013; Resnick et al., 2009). Dengan demikian, definisi umum tentang berpikir komputasi yang disarankan oleh ISTE dan NRC mungkin tidak cocok untuk pemrograman. Oleh karena itu, dalam ulasan ini tentang berpikir komputasi melalui pemrograman, digunakan kerangka kerja yang diusulkan untuk Scratch oleh Brennan dan Resnick (2012). Scratch adalah bahasa pemrograman populer yang digunakan dalam pengaturan K-12 (mis., Baytak & Land, 2011; Kafai, Fields, & Burke, 2010; Tangney, Oldham, Conneely, Barrett, & Lawlor, 2010; Theodorou & Kordaki, 2010). Sehubungan dengan Scratch, Brennan dan Resnick (2012) mengusulkan tiga dimensi berpikir komputasi: konsep komputasi, praktik komputasi, dan perspektif komputasi. Tabel 1 merangkum ide-ide utama pada tiga dimensi ini. Dimensi ini sesuai untuk memahami bagaimana siswa K-12 mendekati pemrograman karena mereka juga sejalan dengan pengetahuan bahasa pemrograman Logo yang diusulkan oleh Mayer (1992). Ini termasuk sintaksis, semantik, pengetahuan skematik (konsep komputasi) dan pengetahuan strategis (praktik komputasi). Selain itu, Scratch berbagi fitur serupa dengan bahasa pemrograman visual kontemporer untuk siswa (mis., Alice). Bahasa-bahasa ini mudah dimengerti karena memberikan umpan balik visual dari program dalam bentuk objek animasi dan memungkinkan siswa untuk membuat media interaktif (mis. animasi dan game). Oleh karena itu, kerangka kerja ini sesuai untuk mempertimbangkan berpikir komputasi dalam konteks pemrograman.

TABEL I
PENJELASAN TIGA DIMENSI BERPIKIR KOMPUTASI

Dimensi	PENJELASAN	Contoh
Konsep berpikir komputasi	Konsep yang digunakan programmer	Variabel
Praktek praktek berpikir komputasi	Praktek pemecahan masalah yang ter-jadi pada proses pemrograman	Loops, iterasi
Prespektif berpikir kom-putasi	Pemahaman siswa tentang dirinya, hubungannya dengan orang lain, dan teknologi di sekelilingnya	Testing and debugging

B. Taksonomi Berpikir Komputasi

Menurut definisi Wing (2006), berpikir komputasi dapat diklasifikasikan ke dalam beberapa proses berpikir, termasuk abstraksi, desain algoritma, dekomposisi, pengenalan pola, dan perwakilan data. Dalam 10 tahun terakhir, berpikir komputasi telah diterapkan dalam berbagai mata pelajaran. Para ahli telah mencoba berbagai strategi pembelajaran untuk membantu siswa belajar. Seperti yang ditunjukkan Tabel II, penelitian ini mencantumkan strategi pembelajaran yang telah digunakan dalam studi sebelumnya, termasuk pembelajaran berbasis masalah, pembelajaran kolaboratif, pembelajaran berbasis proyek, pembelajaran berbasis game, perancah, mendongeng, teori pembelajaran komputasi, pengalaman estetika, konsep pembelajaran berbasis, pembelajaran berbasis perwujudan, pengajaran interaksi manusia-komputer, dan desain universal untuk pembelajaran (Tabel III).

IV. KESIMPULAN

Dalam studi ini, makalah berpikir komputasi yang diterbitkan dalam jurnal akademik antara 2006 dan 2018 ditinjau dan dianalisis dan dibahas oleh klasifikasi data. Ditemukan bahwa jumlah makalah berpikir komputasi telah melihat peningkatan substansial dalam beberapa tahun terakhir, dan berpikir komputasi telah menerima komentar positif dari para peneliti dari sejumlah negara (Balanskat & Engelhardt, 2014; Falkner et al., 2014; Heintz et al., 2016; Sysło & Kwiatkowska, 2015), menyiratkan arti pentingnya untuk mencapai tujuan pendidikan di masa depan. Dari hasil analisis terungkap bahwa kegiatan berpikir komputasi sebagian besar diperkenalkan untuk desain program, ilmu komputer, biologi, dan kursus desain robot. Dapat disimpulkan bahwa, sejauh ini, sejumlah kegiatan berpikir komputasi telah diintegrasikan ke dalam mata pelajaran yang berbeda dalam cara berbasis kehidupan, mengaplikasikan konsep berpikir komputasi yang diusulkan oleh Wing (2006). Wing menganggap berpikir komputasi sebagai keterampilan yang dapat diterapkan secara luas di lingkungan hidup daripada secara eksklusif dipekerjakan oleh ilmuwan komputer; sebaliknya, itu adalah keterampilan yang pantas mendapatkan sikap positif dalam kehidupan sehari-hari dan harus diketahui dan dilibatkan. Oleh karena itu, berpikir komputasi adalah topik yang layak dipelajari secara mendalam di masa depan, dan dampak berpikir komputasi terhadap kinerja akademik anak juga topik diskusi yang layak.

Aplikasi pembelajaran berpikir komputasi dan strategi pembelajaran juga dibahas; ditemukan bahwa sebagian besar penelitian berpusat pada Pembelajaran Berorientasi Proyek, Pembelajaran Berorientasi Masalah, Pembelajaran Kooperatif, dan Pembelajaran Berbasis Game. Selama dekade terakhir, sejumlah sarjana penelitian telah menyebutkan manfaat berpikir komputasi untuk pembelajaran anak-anak; karenanya, penelitian di masa depan harus berusaha untuk memperkenalkan strategi pembelajaran yang berbeda, termasuk Strategi Pembelajaran Perancah, Belajar Mendongeng, dan Pengalaman Estetika, sehingga dapat membantu peserta didik dalam berbagai cara dalam hal pengembangan mata pelajaran atau pelatihan kemampuan tingkat tinggi, pelatihan dalam pemikiran kritis dan kemampuan memecahkan masalah. Sedangkan media pembelajaran yang digunakan yaitu LOGO, LEGO, ViMAP, MATLAB, Alice, Turtle Art, Scratch, Scratch4SL, Code.org, AgentCubes, Scalable Game Design, Java, C, C++, dan Python.

TABEL II
STRATEGI PEMBELAJARAN UNTUK MENINGKATKAN KETERAMPILAN PEMROGRAMAN DAN BERPIKIR KOMPUTASI

Strategi	PENJELASAN
problem-based learning	Definisi pembelajaran berbasis masalah adalah membantu siswa untuk menetapkan tujuan belajar mereka sendiri melalui adegan masalah. Siswa akan mengeksplorasi solusi pembelajaran sendiri, dan melaporkan kesimpulan pembelajaran mereka sendiri dan umpan balik kepada tim. Pembelajaran berbasis masalah tidak hanya digunakan untuk memecahkan masalah, tetapi juga untuk meningkatkan pemahaman siswa tentang pengetahuan baru melalui pertanyaan yang sesuai (Wood, 2003).
collaborative learning (teamwork)	Pembelajaran kelompok dibagi menjadi: pembelajaran kolaboratif dan pembelajaran kooperatif. Dalam pembelajaran kooperatif, mitra membagi pekerjaan, menyelesaikan subtugas secara terpisah, dan kemudian mengumpulkan hasil parsial menjadi hasil akhir. Dalam pembelajaran kolaboratif, anggota kelompok diminta untuk menyelesaikan tugas bersama, bernegosiasi, dan berbagi makna yang relevan dengan tugas pemecahan masalah (Dillenbourg, 1999; Roschelle & Teasley, 1995).
project-based learning	Pembelajaran berbasis proyek (PBL) adalah model yang mengatur pembelajaran di sekitar proyek. Proyek adalah tugas yang kompleks, berdasarkan pertanyaan atau masalah yang menantang, yang melibatkan siswa dalam desain, penyelesaian masalah, pengambilan keputusan, atau kegiatan investigasi; PBL memberi siswa kesempatan untuk bekerja secara relatif mandiri selama periode waktu yang panjang, dan berpuncak pada produk atau presentasi yang realistis (Jones, Rasmussen, & Moffitt, 1997).
game-based learning	Game Based Learning (GBL) mirip dengan Problem Based Learning (PBL), di mana skenario masalah tertentu ditempatkan dalam kerangka bermain (Barrows & Tamblyn, 1980). GBL dapat menyediakan pendekatan Student-Centered e-Learning (SCeL) (Motschnig-Pitrik & Holzinger, 2002). Selain itu, game mencakup banyak karakteristik penyelesaian masalah, mis. hasil yang tidak diketahui, banyak jalur ke tujuan, konstruksi konteks masalah, kolaborasi dalam kasus beberapa pemain, dan mereka menambahkan elemen kompetisi dan peluang.
scaffolding	Scaffolding memberikan kerangka belajar untuk membantu siswa mempelajari pengetahuan baru di awal. Tujuan perancah adalah untuk melatih siswa untuk memecahkan masalah secara mandiri.
problem solving system	Untuk menemukan solusi untuk masalah melalui metode logis atau khusus, dan untuk memahami tujuan masalah dan menerapkan kemampuan dan metode yang tepat untuk menyelesaikan masalah.
storytelling	Pesola (1991, hal. 340) mengemukakan bahwa mendongeng adalah "salah satu alat paling ampuh untuk mengelilingi pelajar muda dengan bahasa. Menurut Isbell (2002), banyak cerita yang bekerja dengan baik pada anak-anak termasuk frasa berulang, kata-kata unik, dan deskripsi yang menarik. Karakteristik ini mendorong siswa untuk bergabung secara aktif untuk mengulang, menyanyikan, bernyanyi, atau bahkan menceritakan kembali cerita. Banyak dari bahasa yang dipelajari anak-anak mencerminkan bahasa dan perilaku model dewasa yang berinteraksi dengan dan didengarkan (Strickland & Morrow, 1989). Mendengarkan cerita menarik perhatian pada suara bahasa dan membantu anak-anak mengembangkan kepekaan terhadap cara kerja bahasa (Isbell, 2002).

TABEL III
BAHASA PEMROGRAMAN YANG DIGUNAKAN UNTUK MENGAJARKAN BERPIKIR KOMPUTASI

Bahasa Pemrograman	Penjelasan
LOGO	LOGO adalah bahasa pemrograman komputer yang mudah dipelajari dan digunakan. Siswa dapat menggunakannya untuk menggambar pola, menghitung dan memancarkan suara, dan itu juga merupakan cara baru bagi siswa sekolah dasar untuk belajar bahasa pemrograman komputer.
LEGO	Lego adalah pemrograman kotak perintah yang menggabungkan bangunan dengan batu bata LEGO yang sudah dikenal, menggunakan perangkat lunak pengkodean yang mudah digunakan, membuat pengkodean menjadi menyenangkan dan relevan bagi siswa sekolah dasar dan menengah.
ViMAP	Bahasa pemrograman ViMAP adalah bahasa pemrograman open-source dan lingkungan pemodelan yang dirancang untuk kelas sains K12. ViMAP juga memungkinkan anak-anak membuat perintah pemrograman sendiri.
MATLAB	MATLAB (laboratorium matriks) adalah lingkungan komputasi numerik multi-paradigma. Ini memungkinkan manipulasi matriks, memplot fungsi dan data, implementasi algoritma, pembuatan antarmuka pengguna, dan berinteraksi dengan program yang ditulis dalam bahasa lain (C, C ++, C #, Java, dan sebagainya.)
Alice	Alice adalah bahasa pemrograman pendidikan berbasis objek terbuka dengan lingkungan pengembangan terintegrasi (IDE). Ini menggunakan fungsi drag and drop untuk membuat animasi komputer dengan model 3D.
Turtle Art	Turtle Art adalah perangkat lunak dengan grafis "Logo" yang terinspirasi Logo yang menggabungkan dengan elemen pemrograman visual snap-like snap-together dan seni berwarna-warni.
Scratch	Scratch adalah bahasa pemrograman visual online yang dikembangkan oleh MIT Media Lab. Pengguna dapat membuat proyek online dan menjadikannya apa saja dengan mengkode dengan blok sederhana.
Scratch4SL	S4SL didasarkan pada Scratch dan merupakan cara mudah baru untuk menambahkan perilaku dan interaktivitas ke objek Anda di Second Life. Ini menggunakan bahasa pemrograman grafis untuk membuat proyek dengan menyeret blok grafis.
Code.org	Code.org adalah situs web yang mencakup pelajaran pengkodean gratis dan yang juga berupaya mendorong guru untuk memasukkan lebih banyak kelas ilmu komputer dalam kurikulum. Pengguna menggunakan Blockly untuk menulis kode. Ini adalah bahasa pemrograman komputer virtual yang menarik, seperti bahasa markup.
AgentCubes	AgentCubes adalah bahasa pemrograman pendidikan untuk anak-anak untuk membuat game dan simulasi online 3D dan 2D. Ini adalah alat berpikir komputasi untuk mengajar anak-anak berpikir komputasi melalui game dan desain simulasi berdasarkan kurikulum Desain Game Scalable.
Scalable Game Design	Scalable Game Design adalah kurikulum untuk belajar tentang konsep komputasi pada tingkat berpikir komputasi yang relevan dengan desain game serta ilmu komputasi.
Java	Java adalah bahasa pemrograman komputer open source. Kepercayaan utama Java adalah dapat berjalan di semua platform yang mendukung Java tanpa perlu dikompilasi ulang.
C	C adalah bahasa pemrograman komputer yang penting dan instruksi mesin yang khas, yang menyediakan jembatan untuk menanamkan dan mengoperasikan sistem dengan berbagai jenis perangkat lunak aplikasi.
C++	C ++ adalah bahasa yang dikompilasi, dengan implementasi tersedia di banyak platform. Efisiensi dan fleksibilitas C ++ juga bermanfaat dalam banyak konteks lainnya
Python	Python adalah bahasa pemrograman tujuan umum tingkat tinggi yang ditafsirkan. Diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991, filosofi desain Python menekankan keterbacaan kode dengan penggunaan spasi spasi yang signifikan. Konstruk bahasanya dan pendekatan berorientasi objek bertujuan untuk membantu programmer menulis kode yang jelas dan logis untuk proyek skala kecil dan besar.

DAFTAR PUSTAKA

[1] Armoni, M. (2012). Teaching CS in kindergarten: How early can the pipeline begin? ACM Inroads, 3(4), 18–19.
[2] Balanskat, A., & Engelhardt, K. (2014). Computing our future: Computer programming and coding-priorities, school curricula and initiatives across Europe. European Schoolnet.
[3] Barrows, H. S., & Tamblyn, R. M. (1980). Problem-based learning: An approach to medical education. Springer Publishing Company.

[4] Berland, M., & Lee, V. R. (2012). Collaborative strategic board games as a site for distributed computational thinking. Developments in Current Game-Based Learning Design and Deployment, 285.
[5] Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. Computers & Education, 72, 145–157.
[6] Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. ACM Transactions on Computing Education (TOCE), 14(2), 9.
[7] Chang, C. K. (2014). Effects of using Alice and Scratch in an introductory programming course for corrective instruction. Journal of Educational Computing Research, 51(2), 185–204.
[8] Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. Computers & Education, 109, 162–175.
[9] Cheung, R. H. P. (2013). Exploring the use of the pedagogical framework for creative practice in preschool settings: A phenomenological approach. Thinking Skills and Creativity, 10, 133–142.
[10] Choi, J., Lee, Y., & Lee, E. (2016). Puzzle based algorithm learning for cultivating computational thinking. Wireless Personal Communications, 1–15.
[11] Denning, P. J. (2017). Remaining trouble spots with computational thinking. Communications of the ACM, 60(6), 33–39.
[12] Dillenbourg, P. (1999). Collaborative learning: Cognitive and computational approaches. Advances in learning and instruction series. PO Box 945, Madison Square Station, New York, NY 10160-0757: Elsevier Science, Inc.
[13] Dodig-Crnkovic, G. (2011). Significance of models of computation, from Turing model to natural computation. Minds and Machines, 21(2), 301–322.
[14] Evia, C., Sharp, M. R., & Pérez-Quiñones, M. A. (2015). Teaching structured authoring and DITA through rhetorical and computational thinking. IEEE Transactions on Professional Communication, 58(3), 328–343.
[15] Falkner, K., Vivian, R., & Falkner, N. (2014, January). The Australian digital technologies curriculum: Challenge and opportunity. Proceedings of the sixteenth Australasian computing education conference: 148, (pp. 3–12). Australian Computer Society, Inc.
[16] Farris, A. V., & Sengupta, P. (2016). Democratizing Children's Computation: Learning computational science as aesthetic experience. Educational Theory, 66(1–2), 279–296.
[17] de Freitas, E. (2016). Number sense and the calculating child: Measure, multiplicity and mathematical monsters. Discourse: Studies in the Cultural Politics of Education, 37(5), 650–661.
[18] Grover, S., & Pea, R. (2013). Computational thinking in K–12 a review of the state of the field. Educational Researcher, 42(1), 38–43.
[19] Gynnild, A. (2014). Journalism innovation leads to innovation journalism: The impact of computational exploration on changing mindsets. Journalism, 15(6), 713–730.
[20] Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. Frontiers in education conference (FIE), 2016 IEEE(pp. 1–9). IEEE.
[21] Hitchcock, C., Meyer, A., Rose, D., & Jackson, R. (2002). Providing new access to the general curriculum: Universal design for learning. Teaching Exceptional Children, 35(2), 8–17.
[22] Hwang, G. J., & Tsai, C. C. (2011). Research trends in mobile and ubiquitous learning: A review of publications in selected journals from 2001 to 2010. British Journal of Educational Technology, 42(4), E65–E70.
[23] Hwang, G. J., & Wu, P. H. (2014). Applications, impacts and trends of mobile technology-enhanced learning: A review of 2008-2012 publications in selected SSCI journals. International Journal of Mobile Learning and Organisation, 8(2), 83–95. http://dx.doi.org/10.1504/IJMLO.2014.062346.
[24] Isbell, R. (2002). Telling and retelling stories – learning language and literacy. Young Children, 57(2), 26–30.
[25] ISTE, C. (2011). Computational thinking in K–12 education leadership toolkit.
[26] Jones, B. F., Rasmussen, C. M., & Moffitt, M. C. (1997). Real-life problem solving: A collaborative approach to interdisciplinary learning. American Psychological Association.
[27] , E. J., & Carnine, D. W. (1998). Effective teaching strategies that accommodate diverse learners. Order Processing, PO Box 11071, Des Moines, IA 50336–1071: Prentice-Hall Inc.
[28] Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Understanding computational thinking before Programming: Developing guidelines for the design. Developments in Current Game-Based Learning Design and Deployment.
[29] Libeskind-Hadas, R., & Bush, E. (2013). A first course in computing with applications to biology. Briefings in Bioinformatics, 14(5), 610–617.

[30] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41, 51–61.
[31] Manson, J. R., & Olsen, R. J. (2010). Diagnostics and rubrics for assessing learning across the computational science curriculum. Journal of Computational Science, 1(1), 55–61.
[32] Navlakha, S., & Bar-Joseph, Z. (2011). Algorithms in nature: The convergence of systems biology and computational thinking. Molecular Systems Biology, 7(1), 546.
[33] Ngan, S.-C., & Law, K. M. (2015). Exploratory Network Analysis of Learning Motivation Factors in e-Learning Facilitated Computer Programming Courses. The Asia- Pacific Education Researcher, 24(4), 705–717.
[34] Orvalho, J. (2017, July). Computational thinking for teacher education. Scratch2017BDX: Opening, inspiring, connecting (pp. 6). .
[35] Papert, S. (1990). A critique of technocentrism in thinking about the school of the future. Epistemology and learning memo #2. September 1990. Cambridge MA: MIT.Retrieved 29 August 2015 from www.papert.org/articles/ACritiqueofTechnocentrism.html.
[36] Pellas, N., & Peroutseas, E. (2017). Leveraging Scratch4SL and second life to motivate high school students' participation in introductory programming courses: Findings from a case study. New Review of Hypermedia and Multimedia, 23(1), 51–79.
[37] Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. Computer supported collaborative learning (pp. 69–97). Berlin, Heidelberg: Springer.
[38] Rubinstein, A., & Chor, B. (2014). Computational thinking in life science education. PLoS Computational Biology, 10(11), e1003897.
[39] Shell, D. F., & Soh, L. K. (2013). Profiles of motivated self-regulation in college computer science courses: Differences in major versus required non-major courses. Journal of Science Education and Technology, 22(6), 899–913.
[40] Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. Computers & Education, 100, 1–17.
[41] Stefan, M. I., Gutlerner, J. L., Born, R. T., & Springer, M. (2015). The quantitative methods boot camp: Teaching quantitative thinking and computing skills to graduate students in the life sciences. PLoS Computational Biology, 11(4), e1004208.
[42] Strickland, D. S., & Morrow, L. M. (1989). Emerging literacy: Young children learn to read and write. 800 Barksdale Rd., PO Box 8139, Newark, DE 19714–8139: International Reading Association.
[43] Sysło, M. M., & Kwiatkowska, A. B. (2015, September). Introducing a new computer science curriculum for all school levels in Poland. International conference on informatics in Schools: Situation, evolution, and perspectives (pp. 141–154). Cham: Springer.
[44] Wang, H. Y., Liu, G. Z., & Hwang, G. J. (2017). Integrating socio-cultural contexts and location-based systems for ubiquitous language learning in museums: A state of the art review of 2009–2014. British Journal of Educational Technology, 48(2), 653–671.
[45] Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127–147.
[46] Wilkerson-Jerde, M. H. (2014). Construction, categorization, and consensus: Student generated computational artifacts as a context for disciplinary reflection. Educational Technology Research and Development, 62(1), 99–121.
[47] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.
[48] Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical transactions of the royal society of London a: Mathematical, Physical and Engineering Sciences, 366(1881), 3717–3725.
[49] Wing, J. M. (2010). Computational thinking: What and why? Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.
[50] Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. ACM Transactions on Computing Education (TOCE), 11(2), 9.
[51] Wood, D. F. (2003). ABC of learning and teaching in medicine: Problem based learning. BMJ: British Medical Journal, 326(7384), 328.
[52] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. ACM Transactions on Computing Education (TOCE), 14(1), 5.
[53] Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. Journal of Educational Computing Research, 53(4), 562–590.