https://doi.org/10.37817/ikraith-informatika.v11i1
P-ISSN : 2580-4316
E-ISSN : 2654-8054

Aplikasi Pembelajaran Ekspresi Aritmatika dengan Metode Regular Definition

[1.]Mika Yolanda Oktavia Manik, [2.] Swingly Purba , [3.] Melva E Pangaribuan
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Institut Sains dan Teknologi TD. Pardede, Medan
E-mail : [1.]Mikamanik0604@gmail.com,
[2.]Swingly.Purba61@gmail.com
[3.]melpangrib@gmail.com

ABSTRAK

Ekspresi aritmatika merupakan konstruksi penting dalam bahasa pemrograman karena memuat operand dan operator yang harus diproses sesuai aturan sintaks serta presedensi tertentu. Penelitian ini merancang aplikasi pembelajaran yang memvisualisasikan validasi struktur ekspresi menggunakan Regular Definition dan evaluasi ekspresi valid menggunakan notasi postfix serta struktur data stack. Regular Definition yang dirancang mendeskripsikan huruf, digit, variabel, dan konstanta integer. Operator yang didukung meliputi penjumlahan, pengurangan, perkalian, pembagian, dan pemangkatan. Alur aplikasi terdiri atas input ekspresi, pengenalan token, validasi, pemecahan subekspresi, konversi infix ke postfix, input nilai variabel, dan evaluasi postfix secara bertahap. Hasil perancangan menunjukkan bahwa aplikasi dapat menampilkan kategori token, daftar variabel dan konstanta, bentuk postfix, langkah operasi stack, serta hasil evaluasi. Aplikasi ini dapat digunakan sebagai media bantu visual untuk pembelajaran Teknik Kompilasi. Penelitian ini berfokus pada perancangan dan implementasi fungsional; pengukuran efektivitas pembelajaran dengan responden belum dilakukan.

Kata kunci: *Regular Definition; ekspresi aritmatika; postfix; stack; evaluator*

ABSTRACT

Arithmetic expressions are important programming-language constructs because they contain operands and operators that must be processed according to syntactic rules and operator precedence. This study designs a learning application that visualizes expression-structure validation using Regular Definition and evaluates valid expressions using postfix notation and a stack data structure. The proposed Regular Definition describes letters, digits, variables, and integer constants. The supported operators are addition, subtraction, multiplication, division, and exponentiation. The application flow consists of expression input, token recognition, validation, sub-expression decomposition, infix-to-postfix conversion, variable-value input, and step-by-step postfix evaluation. The design results indicate that the application can display token categories, variables and constants, postfix notation, stack-operation steps, and evaluation results. The application can therefore support visual learning in Compiler Techniques. This study focuses on functional design and implementation; learning effectiveness has not been measured with respondents.

Keyword: *Regular Definition; arithmetic expression; postfix; stack; evaluator*

1. PENDAHULUAN

Komputer menjalankan program melalui rangkaian instruksi yang ditulis menurut aturan bahasa pemrograman. Salah satu bentuk instruksi yang banyak digunakan adalah ekspresi aritmatika, yaitu kombinasi operand dan operator yang menghasilkan suatu nilai. Operand dapat berupa konstanta atau variabel, sedangkan operator menentukan operasi yang dilakukan. Agar ekspresi dapat diproses, setiap unsur harus dikenali dan disusun dalam struktur yang valid.

Dalam pembelajaran Teknik Kompilasi, Regular Expression dan Regular Definition digunakan untuk menggambarkan pola token, seperti identifier dan konstanta (Aho et al., 2006; Hopcroft et al., 2006). Ekspresi yang lolos validasi perlu diproses dengan memperhatikan presedensi operator. Notasi postfix menempatkan operator setelah operand sehingga urutan evaluasi menjadi eksplisit; bentuk ini dapat dievaluasi menggunakan stack dengan prinsip Last In First Out (Cormen et al., 2009).

Kesulitan pembelajaran muncul ketika konsep hanya disajikan secara teoritis atau melalui hasil akhir. Skripsi MIKA merespons kebutuhan tersebut melalui Evaluator Ekspresi Matematika (EEM), yaitu aplikasi yang menampilkan proses pengecekan struktur berdasarkan Regular Definition, konversi infix ke postfix, serta evaluasi ekspresi menggunakan stack. Tujuan penelitian ini adalah merancang media pembelajaran yang dapat menentukan aturan variabel dan konstanta, membedakan

P-ISSN : 2580-4316
E-ISSN : 2654-8054
https://doi.org/10.37817/ikraith-informatika.v11i1

operand, menghasilkan postfix, dan memperlihatkan proses evaluasi.

2. LANDASAN TEORI

2.1 Regular Expression dan Regular Definition

Regular Expression merupakan notasi untuk menyatakan pola string dalam suatu alfabet. Operasi dasar yang digunakan meliputi alternation atau pilihan, concatenation atau penggabungan, Kleene closure yang menyatakan kemunculan nol kali atau lebih, serta positive closure yang menyatakan kemunculan satu kali atau lebih (Aho et al., 2006; Sipser, 2012). Regular Definition menyusun beberapa nama definisi secara berurutan sehingga definisi berikutnya dapat menggunakan definisi sebelumnya.

Pada EEM, aturan operand dirumuskan untuk membatasi bentuk ekspresi sesuai ruang lingkup penelitian. Variabel wajib dimulai dengan huruf, dapat diikuti huruf, digit, atau garis bawah, sedangkan konstanta hanya berupa bilangan integer dengan tanda minus opsional. Konstanta real belum termasuk dalam cakupan aplikasi.

Tabel 1. Regular Definition operand pada EEM

Nama	Definisi	Contoh valid	Contoh tidak valid
letter	A|B|…|Z|a|b|…|z	A, m	3, _
digit	0|1|…|9	0, 7	a
variabel	letter (letter|digit|_)*	a,e1,contoh_kasus	1d,_dr
konstanta	(ε|−)digit(digit)*	0, 25, −19	−0,15.5

2.2 Postfix dan Stack

Pada ekspresi infix, operator berada di antara operand, misalnya a + b. Pada postfix, operator ditempatkan setelah operand sehingga a b + tidak memerlukan tanda kurung untuk menyatakan urutan operasi. Konversi harus memperhatikan presedensi dan asosiasi operator. Stack digunakan untuk menyimpan operand dan hasil sementara; operasi biner mengambil dua nilai teratas, menghitungnya, kemudian mengembalikan hasil ke stack (Cormen et al., 2009; Wirth, 1976).

3. METODOLOGI

Penelitian ini menggunakan pendekatan perancangan dan implementasi perangkat lunak. Tahapan penelitian meliputi identifikasi kebutuhan, perumusan Regular Definition, perancangan algoritma validasi, perancangan transformasi postfix, perancangan evaluasi stack, perancangan antarmuka, dan pemeriksaan fungsional menggunakan contoh ekspresi. Pendekatan ini sesuai dengan alur pengembangan perangkat lunak yang menempatkan analisis kebutuhan, desain, implementasi, dan pengujian sebagai tahapan yang saling berurutan (Pressman dan Maxim, 2014; Sommerville, 2016).

Tabel 2. Tahapan penelitian dan keluaran

Tahap	Kegiatan	Keluaran
Identifikasi	Menetapkan token,operator,batas panjang,dan tipe operand.	Spesifikasi EEM
Perumusan aturan	Menentukan Regular Definition untuk variabel dan konstanta.	Aturan validasi operand
Perancangan algoritma	Menyusun validasi,subekspresi, postfix, dan evaluasi stack.	Algoritma pemrosesan
Implementasi	Membangun form dan komponen EEM dengan Visual Basic 6.0.	Aplikasi pembelajaran
Pemeriksaan fungsi	Menjalankan contoh ekspresi valid dan mengamati keluaran.	Bukti keluaran proses

3.1 Spesifikasi dan Batasan

Input EEM berupa string yang terdiri atas operator, konstanta, variabel, dan delimiter. Panjang ekspresi dibatasi maksimum 100 karakter. Variabel memiliki panjang maksimum 30 karakter dan minimum satu karakter. Nilai variabel dimasukkan sebagai integer, sedangkan hasil ditampilkan dalam format real sesuai spesifikasi skripsi. Operator utama yang digunakan adalah +, −, *, /, dan ^; tanda kurung digunakan sebagai delimiter.

3.2 Algoritma Utama

Validasi membaca ekspresi dari kiri ke kanan melalui fungsi FGetTypeToken. Karakter huruf, digit, dan garis bawah digabungkan menjadi kandidat operand. Tanda minus pada posisi awal atau setelah operator ditandai sebagai unary minus

https://doi.org/10.37817/ikraith-informatika.v11i1
P-ISSN : 2580-4316
E-ISSN : 2654-8054

dengan simbol internal ~. Ekspresi ditolak apabila operator muncul di awal, operator muncul berturut-turut, operand terpisah tanpa aturan yang benar, atau terdapat ketidakseimbangan delimiter.

```
FGetTypeToken(token):
  jika token kosong, kembalikan KOSONG
  jika token = _ , kembalikan SPESIAL_KARAKTER
  jika token huruf, kembalikan HURUF
  jika token digit, kembalikan DIGIT
  jika token ∈ {^, *, /, +, −}, kembalikan OPERATOR
  jika token ∈ {(, )}, kembalikan DELIMITER
  selain itu, kembalikan ERROR
```

Pemecahan subekspresi dilakukan dengan memproses pasangan kurung terdalam terlebih dahulu. Setelah itu, operator diproses berdasarkan urutan presedensi: unary minus, pangkat, perkalian/pembagian, lalu penjumlahan/pengurangan. Untuk operator dengan tingkat yang sama, proses dilakukan dari kiri ke kanan sebagaimana rancangan pada skripsi.

4. HASIL DAN PEMBAHASAN

4.1 Rancangan Fungsional Aplikasi

EEM dirancang menggunakan Microsoft Visual Basic 6.0 dengan MDI Form sebagai form utama dan beberapa child form. Komponen yang digunakan meliputi TextBox untuk input, Command Button untuk eksekusi, Option Button untuk pilihan kecepatan animasi, MSFlexGrid untuk tabel, List Box untuk keluaran, Picture Box dan Image untuk tampilan, serta Timer untuk mengatur interval langkah. Struktur form diringkas pada Tabel 3.

Tabel 3. Form dan fungsi pembelajaran EEM

Form	Fungsi	Keluaran utama
Main	Navigasi aplikasi dan menu file.	Menu teori, input, dan bantuan.
Teori	Menampilkan materi pendukung.	Ekspresi, Regular Definition, postfix, stack.
Input Ekspresi	Menerima string ekspresi.	Textbox dan tombol operator.
Checking Struktur	Menampilkan validasi token.	Kategori token, aturan, variabel, konstanta.
Evaluasi EM	Menampilkan hasil transformasi.	Postfix dan input nilai variabel.
Hitung Nilai EM	Menampilkan evaluasi bertahap.	Stack, akumulator, dan hasil.
Cara Pemakaian	Memberi panduan penggunaan.	Langkah penggunaan aplikasi.

4 .2 Contoh Konversi Infix ke Postfix

Contoh pertama adalah a + b * c − d / e. Perkalian dan pembagian dikerjakan lebih dahulu sehingga bentuk postfix yang dihasilkan adalah a b c * + d e / −. Contoh kedua adalah 3 * (2 + 4) − 25 / 5. Subekspresi dalam kurung menghasilkan 2 4 +, kemudian seluruh ekspresi berubah menjadi 3 2 4 + * 25 5 / −.

Tabel 4. Contoh ekspresi dan bentuk postfix

Ekspresi infix	Bentuk postfix	Keterangan
a + b * c − d / e	a b c * + d e / −	* dan / mendahului + dan −.
3 * (2 + 4) − 25 / 5	3 2 4 + * 25 5 / −	Kurung diproses lebih dahulu.
−a + b	a ~ b +	~ merepresentasikan unary minus.

4.3 Evaluasi dengan Stack

Untuk postfix 3 2 4 + * 25 5 / −, proses evaluasi berjalan dari kiri ke kanan. Operand 3, 2, dan 4 dimasukkan ke stack. Operator + mengambil 2 dan 4 sehingga menghasilkan 6. Operator * kemudian menghitung 3 × 6 = 18. Setelah 25 dan 5 dimasukkan, operator / menghasilkan 5. Operator − terakhir menghitung 18 − 5 sehingga hasil akhir adalah 13.

Tabel 5. Jejak evaluasi postfix menggunakan stack

Langkah	Token	Isi stack operand	Operasi
1	3	3	Push
2	2	3, 2	Push
3	4	3, 2, 4	Push
4	+	3, 6	2 + 4 = 6
5	*	18	3 × 6 = 18
6	25	18, 25	Push
7	5	18,25, 5	Push
8	/	18, 5	25 / 5 = 5
9	−	13	18 − 5 = 13

4.4 Pembahasan

Hasil perancangan memperlihatkan pemisahan yang jelas antara validasi dan evaluasi. Ekspresi yang tidak memenuhi

struktur dihentikan pada tahap validasi sehingga kesalahan dapat ditampilkan sebelum proses transformasi. Pada ekspresi valid, postfix menjadi bentuk perantara yang menjadikan urutan operasi eksplisit. Stack kemudian menyediakan mekanisme sederhana untuk menampilkan pengambilan operand, perhitungan, dan penyimpanan hasil sementara.

Fitur visualisasi merupakan kontribusi utama EEM. Pengguna dapat mengamati karakter yang sedang diperiksa, aturan yang digunakan, daftar operand, postfix, dan perpindahan nilai pada stack. Pilihan interval animasi 250, 500, 1000, 1500, 2000, dan 3000 milisekon memberi fleksibilitas tempo pembelajaran. Namun, skripsi sumber belum melaporkan pengujian kuantitatif terkait akurasi, usability, waktu eksekusi, atau peningkatan hasil belajar. Karena itu, hasil artikel ini dibatasi pada ketercapaian fungsi yang dirancang.

5. KESIMPULAN

Aplikasi pembelajaran ekspresi aritmatika berhasil dirancang dengan memadukan Regular Definition untuk validasi operand, presedensi operator untuk menentukan urutan proses, postfix sebagai bentuk perantara, dan stack untuk evaluasi. Aturan yang dirancang dapat membedakan variabel dan konstanta integer sesuai batasan sistem. Aplikasi juga dirancang untuk menampilkan proses validasi, daftar operand, bentuk postfix, langkah evaluasi, dan hasil akhir secara visual. Dengan demikian, EEM dapat digunakan sebagai media bantu pembelajaran Teknik Kompilasi, sedangkan pengujian efektivitas pembelajaran dan perluasan operator menjadi agenda pengembangan selanjutnya.

6. UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih kepada Institut Sains dan Teknologi TD. Pardede, dosen pembimbing, dosen Program Studi Teknik Informatika, serta keluarga yang telah memberikan dukungan selama penyusunan skripsi dan artikel ini.

DAFTAR PUSTAKA

Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, techniques, and tools* (2nd ed.). Pearson.

Appel, A. W. (1998). *Modern compiler implementation in Java*. Cambridge University Press.

Cooper, K. D., & Torczon, L. (2011). *Engineering a compiler* (2nd ed.). Morgan Kaufmann.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Grune, D., van Reeuwijk, K., Bal, H. E., Jacobs, C. J. H., & Langhem, K. (2008). *Modern compiler design*. Springer.

Hadi, & Rahadian. (n.d.). *Pemrograman Windows API dengan Microsoft Visual Basic*. PT Elex Media Komputindo.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to automata theory, languages, and computation* (3rd ed.). Pearson.

Louden, K. C. (1997). *Compiler construction: Principles and practice*. PWS Publishing.

Muchnick, S. S. (1997). *Advanced compiler design and implementation*. Morgan Kaufmann.

Pressman, R. S., & Maxim, B. R. (2014). *Software engineering: A practitioner's approach* (8th ed.). McGraw-Hill.

Sethi, R. (1996). *Programming languages: Concepts and constructs* (2nd ed.). Addison-Wesley.

Sipser, M. (2012). *Introduction to the theory of computation* (3rd ed.). Cengage Learning.

Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.

Utdirartatmo, F. (n.d.). *Teknik kompilasi*. J&J Learning.

Wirth, N. (1976). *Algorithms + data structures = programs*. Prentice-Hall.