SIGMA
Jurnal Teknologi Pelita Bangsa

Vol. 12 No. 1 Maret 2021 p-ISSN: 2407-3903 e-ISSN: 2407-3903

Diterima, 16 Februari 2021 | Direvisi, 08 Maret 2021 | Dipublikasikan, 28 Maret 2021

PERANCANGAN APLIKASI PEMBELAJARAN EKSPESI MATEMATIKA DENGAN METODE STACK MENGGUNAKAN DELPHI 2010

Candra Naya

Program Studi Teknik Informatika Fakultas Teknik Universitas Pelita Bangsa
candranaya@pelitabangsa.ac.id

Abstrak

Ekspresi matematika panjang mungkin akan menimbulkan kesulitan untuk mengalihkannya. Di dalam struktur data, tahapan-tahapan penyelesaian suatu ekspresi matematika dapat direpresentasikan dalam bentuk stack dan pohon binar. Dalam hal ini penulis akan menggunakan cara kerja stack untuk menyelesaikan suatu ekspresi matematika dan menggunakan Embarcadero Delphi 2010 sebagai bahasa pemrogramannya. Dalam kehidupan nyata, manusia lebih mengenal ekspresi matematika berupa notasi infix dibandingkan notasi-notasi lainnya. Berbeda dengan proses di dalam komputer, dalam menyelesaikan perhitungan tidak menggunakan bentuk notasi infix, tetapi menggunakan notasi prefix dan postfix. Di dalam notasi prefix dan postfix tidak akan dijumpai tanda kurung dan hierarki pada operator tidak berlaku, sehingga waktu pembacaan lebih cepat. Dunia pendidikan membutuhkan suatu alat pembelajaran yang dapat mempermudah dalam proses belajar mengajar, salah satunya adalah aplikasi-aplikasi perangkat ajar. Perancangan Aplikasi Pembelajaran Ekpresi Matematika Dengan Metode Stack dirancang untuk mempermudah dalam pembelajaran dan pengelolaan serta perhitungan ekspresi matematika. Dengan menggunakan sistem kerja stack, aplikasi ini mengkonversi ekspresi matematika yang biasa digunakan oleh manusia yaitu notasi infix ke dalam ekspresi matematika yang biasa digunakan oleh komputer yaitu notasi prefix dan postfix serta mengevaluasi hasilnya. Aplikasi ini tak terbatas hanya mengeksekusi operand dengan panjang operand satu karakter saja, tetapi juga mampu untuk mengeksekusi operand dengan panjang karakter lebih dari satu karakter. Model pendekatan perancangan perangkat lunak yang digunakan adalah model waterfall. Desain dan implementasi untuk Perancangan Aplikasi Pembelajaran Ekpresi Matematika ini difokuskan pada satu modul yaitu proses konversi dan perhitungan ekspresi matematika. Pada bagian akhir skripsi,dilakukan evaluasi terhadap proses dan produk pengembangan perangkat lunak.

Kata Kunci : Ekspresi Matematika,Metode *Stack, Infix, Prefix, Postfix.*

Abstract

Long mathematical expressions may make it difficult to divert them. In the data structure, the stages of completion of a mathematical expression can be represented in the form of a stack and a binary tree. In this case the author will use the stack work to complete a mathematical expression and use Embarcadero Delphi 2010 as the programming language. In real life, humans are more familiar with mathematical expressions of infix notation than other notations. Unlike the process inside the computer, in completing the calculation does not use the form of infix notation, but using prefix and postfix notation. In prefix and postfix notation there will be no brackets and the hierarchy on the operator is invalid, so the reading time is faster. The world of education requires a learning tool that can simplify the learning process, one of which is the application of teaching tools. Design of Learning Mathematical Ekpresi Application With Stack Method is designed to facilitate the learning and management and calculation of mathematical expressions. By using the stack work system, this application converts the mathematical expressions commonly used by human infix notation into the mathematical expressions commonly used by computers ie prefix and postfix notation and evaluate the results. This unlimited application only executes operands with a single character operand, but is also capable of executing operands with characters longer than one character. Model of software design approach used is waterfall model. Design and implementation for Application Design Learning Mathematical Expression is focused on one module that is the conversion process and calculation of mathematical expression. At the end of the thesis, evaluated the process and product of software development.

***Keywords** : Mathematical Expression, Stack Method, Infix, Prefix, Postfix.*

1. Pendahuluan

Proses kegiatan belajar mengajar dalam suatu perguruan tinggi dengan jumlah mahasiswa yang banyak memerlukan sarana dan prasarana yang memadai untuk menunjang dan menciptakan sesuatu yang dapat mempermudah suatu pekerjaan.Begitu pula di dalam dunia pendidikan, juga membutuhkan suatu alat yang dapat membantu dalam proses belajar mengajar, salah satunya adalah aplikasi perangkat ajar. Hal inilah yang mendorong penulis untuk membuat perancangan aplikasi pembelajaran ekspresi matematika.

Ekspresi matematika yang sering kita gunakan dalam kehidupan sehari-hari dikategorikan sebagai notasiinfix yang mana operator berada diantara operand. Operator adalah lambing operasi matematika (penambahan, pengurangan,pembagian, perkalian, pangkat, dan sebagainya). Operand adalah nilai data atau variabel penampung nilai data. Tidak selamanya ekspresi matematika disimpan dalam bentuk infix di komputer. Ada yang menyimpan ekspresi tersebut secara postfix. Postfix adalah metode penulisan ekspresi matematika yang operatornya berada di belakang operandnya. Jika ekspresi matematika hanya sedikit, mungkin tidak menimbulkan masalah bagi kita untuk mengalihkan dari notasi infix ke notasi postfix.

Namun jika ekspresi matematika panjang mungkin akan menimbulkan kesulitan untuk mengalihkannya. Di dalam struktur data, tahapan-tahapan penyelesaian suatu ekspresi matematika dapat direpresentasikan dalam bentuk stack dan pohon binar. Dalam hal ini penulis akan menggunakan cara kerja stack untuk menyelesaikan suatu ekspresi matematika dan menggunakan Embarcadero Delphi 2010 sebagai bahasa pemrogramannya.

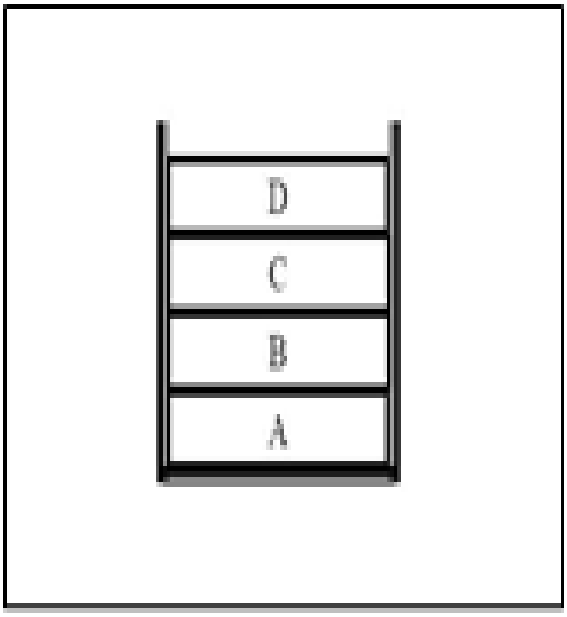
2. Landasan Pemikiran

2.1 Struktur Data

Struktur data memiliki peran yang penting dalam pembuatan sebuah program. Alokasi memori untuk menampung data yang akan diolah oleh program ditentukan melalui struktur data. Di dalam pemodelan ini struktur data yang digunakan adalah struktur data tumpukan (stack).

2.1.1 Tumpukan (Stack)

Menurut Bambang Wahyudi (2003: 57) Stack adalah metode atau tehnik dalam menyimpan atau mengambil data ke dan dari memori.Stack dapat diibaratkan sebuah tumpukan barang dalam sebuah tempat yang hanya memiliki satu pintu di atasnya (memasukkan dan mengambil barang hanya dapat dilakukan melalui pintu itu). Sehingga barang yang akan dikeluarkan pertama kali adalah barang yang terakhir kali dimasukkan. Dengan demikian, kaidah stack adalah First In Last Out atau Last In First Out.



Gambar 1. Ilustrasi *Stack*

Pada Gambar 2.1 terlihat bahwa kotak B diletakkan di atas kotak A dan kotak C diletakkan di atas kotak B dan seterusnya. Menambah atau mengambil sebuah kotak dapat dilakukan lewat ujung bagian atas. Dengan ilustrasi ini dapat dilihat bahwa tumpukan merupakan kumpulan data yang sifatnya dinamis, artinya elemen tumpukan dapat ditambahkan dan dapat dikurangi (diambil).

2.1.2 Operasi Dasar pada Tumpukan (Stack)

Stack umumnya digunakan untuk menyimpan dan mengambil kembali data matematika, stack juga dapat memeriksa apakah ekspresi matematika yang dimasukkan sudah sesuai dengan kaidah penulisan ekspresi matematikanya atau belum, misalnya kekurangan tanda satu tanda kurung, kelebihan tanda operasi matematika, dan sebagainya. Operasi yang sering diterapkan pada stack adalah push dan pop. Operasi-operasi dasar yang dapat diterapkan adalah sebagai berikut:

1.CREATESTACK(S) Membuat tumpukan baru S, dengan jumlah elemen kosong.

2.MAKENULL(S) Mengosongkan tumpukan S, jika ada elemen maka semua elemen dihapus.

3.EMPTY Menguji apakah tumpukan kosong.

4.PUSH(x,S) Memasukkan elemen baru x ke dalam tumpukan S.

5.POP(S) Mengeluarkan elemen posisi atas pada tumpukan S. Berikut ini ilustrasi operasi push dan pop terhadap stack.

Tabel 1. Ilustrasi push dan pop

No	Operasi	Isi Tumpukan	Nilai Top
1	CREATESTACK(S)	: <Kosong>	0
2	PUSH ('a',S)	: a	1
3	PUSH ('b',S)	: a b	2
4	PUSH ('c',S)	: a b c	3
5	POP (S)	: a b	2
6	PUSH ('d',S)	: a b d	3
7	PUSH ('e',S)	: a b d e	4

Apa yang terjadi jika dilakukan POP(S) sebanyak 2 kali lagi? yang terjadi adalah underflow, artinya tumpukan kosong tidak ada elemen yang dapat diambil. Apa yang terjadi jika dilakukan PUSH(x,S) sebanyak sepuluh kali, jika kapasitas tumpukan adalah 5 lagi? yang terjadi adalah overflow, artinya tumpukan penuh tidak ada elemen yang dapat dimasukkan ke dalam tumpukan.

Pada proses PUSH, tumpukan harus diperiksa apakah jumlah elemen sudah mencapai maksimum atau tidak. Jika sudah mencapai maksimum maka overflow. Sedangkan pada proses POP, tumpukan harus diperiksa apakah ada elemen yang hendak dikeluarkan atau tidak. Jika tidak ada maka underflow.

1.Infix

Suatu notasi disebut infix jika operator berada diantara operandnya. Notasi ini merupakan notasi yang sering kita gunakan sehari – hari.

Contoh : A+B (operand, operator, operand).

2.Prefix

Notasi ini juga disebut Polish Notation (PN), yang mana simbol operatornya diletakkan sebelum dua operand.

Contoh : +AB (operator, operand, operand).

3.Postfix

Notasi ini disebut juga suffix atau disebut juga Reverse Polish Notation (RPN), yang mana simbol operatornya diletakan sesudah dua operand.

Contoh : AB+ (operand, operand, operator).

Untuk lebih jelasnya, berikut contoh konversi dari satu notasi ke notasi lainnya dengan cara biasa.

2.1.3 Ekspresi Matematika

Salah satu pemanfaatan tumpukan adalah untuk menulis ungkapan matematika dengan menggunakan notasi tertentu. Seperti kita ketahui, dalam penulisan ungkapan khususnya ungkapan numerik, kita selalu menggunakan tanda kurung untuk mengelompokkan bagian mana yang harus dikerjakan lebih dahulu. Ekspresi matematika seperti A+B*C yang sering kita gunakan dikategorikan sebagai notasi infix yang mana operator berada diantara operand. Operator adalah lambang operasi matematika (penambahan, pengurangan, pengalian, dan sebagainya). Operand adalah nilai data atau variabel penampung nilai data. Notasi.

Infix mudah dimengerti oleh manusia, hanya saja dalam notasi infix perludiperhatikan prioritas pengerjaan karena berhubungan dengan hierarki operator pada komputer. Prioritas pengerjaanya adalah:

a.Tanda kurung: (….)
b.Eksponensial atau pangkat: ^
c.Perkalian, Pembagian: *, /
d.Penjumlahan, Pengurangan: +, -.

Contoh: (A-B)*(C+D)

Prioritas pengerjaan soal di atas adalah sebagai berikut:
a.Dalam kurung paling kiri: (A-B)
b.Dalam kurung kedua: (C+D)
c.Perkalian hasil pengurangan dan hasil penjumlahan.

3. Metode Penelitian

1.Infix

Suatu notasi disebut infix jika operator berada diantara operandnya. Notasi ini merupakan notasi yang sering kita gunakan sehari – hari.

Contoh : A+B (operand, operator, operand).

2.Prefix

Notasi ini juga disebut Polish Notation (PN), yang mana simbol operatornya diletakkan sebelum dua operand.

Contoh : +AB (operator, operand, operand).

3.Postfix

Notasi ini disebut juga suffix atau disebut juga Reverse Polish Notation (RPN), yang mana simbol operatornya diletakan sesudah dua operand.

Contoh : AB+ (operand, operand, operator).

Untuk lebih jelasnya, berikut contoh konversi dari satu notasi ke notasi lainnya dengan cara biasa.

1.Infix

Contoh: (A+B) – (C*D) diubah dalam bentuk prefix dan postfix. Untuk mengubah dalam bentuk prefix :

a.Pengerjaan dalam kurung ke-1: (A+B), prefixnya adalah +AB.
b.Pengerjaan dalam kurung ke-2: (C*D), prefixnya adalah *CD.
c.Terakhir adalah operator -, +AB - *CD, prefixnya adalah –+AB*CD.

Untuk mengubah dalam bentuk postfix:

a. Pengerjaan dalam kurung ke-1: (A+B), postfixnya adalah AB+.
b. Pengerjaan dalam kurung ke-2: (C*D), postfixnya adalah CD*.
c. Terakhir adalah operator -, AB+ - CD*, postfixnya adalah AB+CD*-

2.Prefix

Contoh: + / *ABCD diubah dalam bentuk infix dan postfix. Dalam pengerjaan prefix,untuk mencari operator dimulai dari operand terkanan. Untuk mengubah dalam bentuk infix:

a.Cari operator ke-1: *, ambil dua operand sebelumnya yaitu A dan B,sehingga infixnya adalah (A*B).
b.Cari operator ke-2: / , ambil duaboperand sebelumnya yaitu (A*B) dan C,sehingga infixnya adalah ((A*B) / C).
c.Cari operator ke-3: +, ambil dua operand sebelumnya yaitu ((A*B) / C) dan D, sehingga infixnya adalah ((A*B) / C)+D.
Untuk mengubah dalam bentuk postfix:
a.Cari operator ke-1: *, ambil dua operand sebelumnya yaitu A dan B,sehingga postfixnya adalah AB*.
b.Cari operator ke-2: / , ambil dua operand sebelumnya yaitu AB* dan C,sehingga postfixnya adalah AB*C / .
c.Cari operator ke-3: +, ambil dua operand sebelumnya yaitu AB*C / dan D,sehingga postfixnya adalah AB*C / D+.

3.Postfix
Contoh: ABCD* / - diubah dalam bentuk infix dan prefix.
Dalam pengerjaan postfix, untuk mencari operator dimulai dari operand terkiri. Untuk mengubah dalam bentuk infix:

a. Cari operator ke-1: *, ambil dua operand sebelumnya yaitu C dan D,sehingga infixnya adalah (C*D).
b.Cari operator ke-2: / , ambil dua operand sebelumnya yaitu B dan (C*D),sehingga infixnya adalah (B / (C*D)).
c.Cari operator ke-3: -, ambil dua operand sebelumnya yaitu A dan(B /(C*D)), sehingga infixnya adalah A - (B / (C*D)).

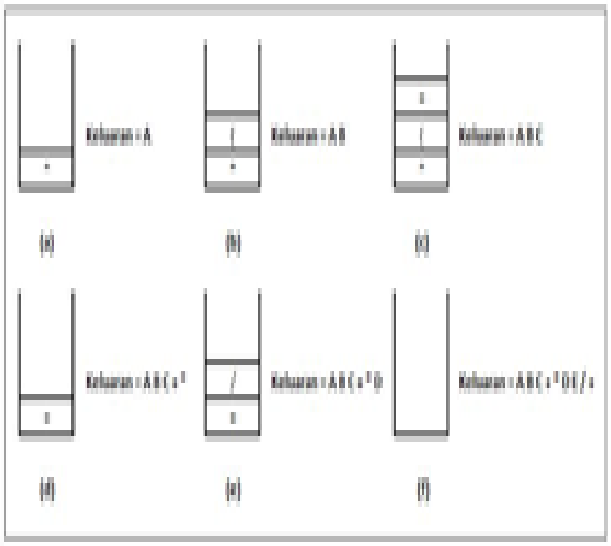
Untuk mengubah dalam bentuk prefix:

a. Cari operator ke-1: *, ambil dua operand sebelumnya yaituC dan D,sehingga prefixnya adalah *CD.
b. Cari operator ke-2: / , ambil dua operand sebelumnya yaitu B dan*CD,sehingga prefixnya adalah / B*CD.
c.Cari operator ke-3: -, ambil dua operand sebelumnya yaitu A dan / B*CD,sehingga prefixnya adalah – A / B*CD.

4. Pembahasan

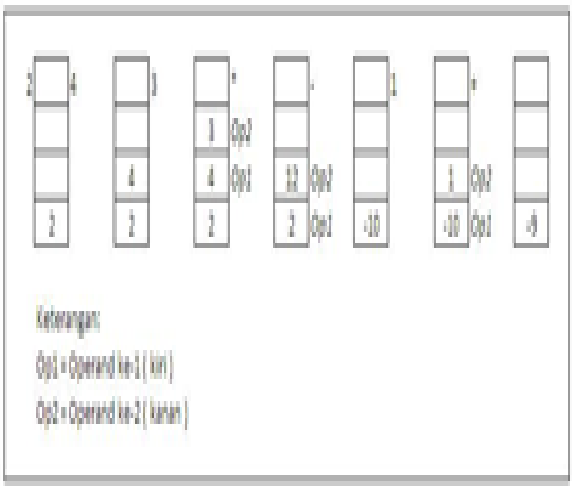
Proses konversi yang akan dibahas selanjutnya adalah bagaimana menerima persamaan infix sebagai masukan, kemudian menghasilkan persamaan postfix sebagai keluarannya. Ide umum yang dapat digunakan adalah menggunakan tumpukan untuk menyimpan operand yang dijumpai dan mengeluarkan operand mengikuti apa yang menjadi pendahulunya. Secara spesifik, ekspresi infix kita telusuri dari kiri ke kanan dan memprosesnya mengikuti aturan berikut ini:
1.Kapan pun operand dijumpai, ia dilewatkan sebagai keluaran.
2.Setiap kali operator dibaca, operator itu dimasukkan ke tumpukan dan operand-operand dilewatkan sebagai keluaran, hingga operator yang dicapai memiliki nilai pendahulu (sebutlah sebagai derajad / bobot) yang lebih rendah daripada operator yang terakhir dibaca.Operator yang dibaca terakhir kali kemudian dimasukkan ke tumpukan.
3.Saat akhir dari persamaan infix dijumpai, semua simbol yang ada pada tumpukan dikeluarkan satu persatu dan dilewatkan sebagai keluaran.
4.Karena tanda kurung (kurung buka dan kurung tutup) dapat digunakan untuk merubah urutan pengerjaan persamaan pada ekspresi infix maka kita harus mempertimbangkannya pada proses konversi. Ini dapat diselesaikan dengan mengangap tanda kurung adalah memiliki nilai pendahulu yang paling tinggi dibandingkan operator-operator yang lain. Sebagai tambahan, kita tidak mengijinkan tanda kurung tutup dimasukkan ke tumpukan, dan kita hanya mengijinkan tanda kurung buka dimasukkan ke tumpukan setelah tanda kurung tutup telah dibaca. Perhatikan tanda-tanda kurung tidak perlu dikeluarkan saat mereka dikeluarkan dari tumpukan karena mereka tidak dibutuhkan kehadirannya pada ekspresi postfix.
Untuk memperlihatkan proses konversi ini, perhatikan contoh ekspresi infix berikut A* (B+C)+D / E. Simbol yang pertama kali dibaca adalah A. Karena ia merupakan operand, maka ia langsung dituliskan sebagai keluaran. Kemudian operator '*' dibaca. Karena tumpukan masih kosong, tidak ada operator yang dikeluarkan, maka '*' dimasukkan ke tumpukan sehingga kita mendapatkan gambaran seperti yang ditunjukan pada Gambar 2.2 (a). Kemudian, operator '(' dibaca dan karena semua operator memiliki nilai pendahulu yang lebih rendah dari tanda kurung buka, ia langsung dimasukkan ke tumpukan. Kemudian 'B'dibaca dan dilewatkan sebagai keluaran, hasilnya seperti yang ditunjukkan pada Gambar 2.2 (b). Simbol berikutnya adalah '+', meskipun '(' memiliki nilai pendahulu paling tinggi (juga lebih tinggi dari operator '+'), ia tidak dapat dikeluarkan dari tumpukan hingga tanda kurung tutup ')' dijumpai. Maka, tidak ada operator yang dikeluarkan dari tumpukan sehingga '+' langsung dimasukkan ke tumpukan. Kemudian yang berikutnya 'C' dibaca dan dilewatkan sebagai keluaran sehingga gambarannya seperti yang ditunjukkan pada Gambar 2.2 (c). Yang dibaca berikutnya adalah tanda kurung tutup ')', karena '+' memiliki pendahulu yang lebih rendah dari operator ini, tidak ada yang dikeluarkan daritumpukan. Tanda ')' dibuang, ketika situasi ini terjadi, kita sekarang akan bisa mengeluarkan tanda kurung buka '(' karena tanda kurung tutup yang berhubungan dengannya telah dijumpai. Simbol berikutnya yang dibaca adalah '+'. Pembacaan simbol ini akan menyebabkan semua yang dimasukkan ketumpukan akan dikeluarkan sebab semua operator dalam tumpukan memiliki nilai pendahulu yang lebih tinggi dari simbol '+', tetapi hanya operator '+' dan '*' yang dilewatkan sebagai keluaran. Tanda '+' yang dibaca kemudian dimasukkan ke tumpukan sehingga kita memiliki tumpukan seperti yang ditunjukkan pada Gambar 2.2 (d). Berikutnya 'D' dibaca dan dikirimkan sebagai keluaran, diikuti pula simbol ' / '. Karena penambahan (+) memiliki pendahulu yang lebih rendah dari pembagian (/) , maka operator ' / ' dimasukkan ke tumpukan seperti ditunjukan pada Gambar 2.2 (e). Simbol terakhir yang dibaca adalah 'E'merupakan keluaran maka operator-

operator yang tersisa dalam tumpukan dikeluarkan sehingga hasil akhir seperti yang ditunjukkan pada Gambar 2. (f).



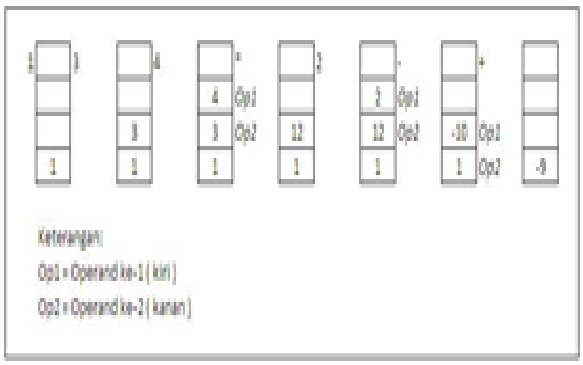
Gambar 2. Proses konversi infix ke postfix

Untuk mengubah notasi infix ke dalam notasi prefix pada umumnya langkah-langkanya hampir sama dengan mengubah notasi infix ke notasi postfix,yang membedakannya adalah pada konversi notasi infix ke notasi prefix proses penelusuran notasi infix dilakukan dari kanan ke kiri dan proses penulisan keluaran dimulai dari kanan ke kiri juga.
Notasi infix akan lebih mudah dihitung hasil evaluasinya jika sudah diubah dalam bentuk prefix dan postfix seperti contoh berikut ini, Misalkan perhitungan aritmatika dalam notasi infix: 2-4*3+1 diubah dalam bentuk postfix menjadi 243*-1+, evaluasinya sebagai berikut:



Gambar 3. Evaluasi Nilai Postfix

Algoritma evaluasi nilai notasi postfix:
1. Buat tumpukan kosong.
2. Ulangi langkah berikut sampai elemen habis.
a. Ambil elemen satu persatu dari aritmatika dimulai dari kiri ke kanan.
b. Jika elemen itu adalah operand, masukkan ke dalam tumpukan dan jika operator maka keluarkan dua nilai teratas dari tumpukan (Op1dan Op2), lalu hitung dengan operator yang bersangkutan. Hasilnya dimasukkan ke tumpukan (jika tidak ada dua operand dalam tumpukan maka ada kesalahan dalam notasi tersebut).
c. Bila elemen dari notasi itu habis, maka nilai yang tertinggal (teratas dari tumpukan) adalah hasilnya.
Apabila contoh di atas diubah dalam bentuk prefix menjadi +-2*431, maka evaluasinya sebagai berikut:



Gambar 4. Evaluasi Nilai Prefix

1. Buat tumpukan kosong.
2. Ulangi langkah berikut sampai elemen habis.
   a. Ambil elemen satu persatu dari aritmatika dimulai dari kanan ke kiri.
   b. Jika elemen itu adalah operand, masukkan kedalam tumpukan dan jika operator maka keluarkan dua nilai teratas dari tumpukan (Op1 dan Op2, kebalikan dari postfix), lalu hitung dengan operator yang bersangkutan. Hasilnya dimasukkan ke tumpukan (jika tidak ada dua operand dalam tumpukan maka ada kesalahan dalam notasi tersebut).
3. Bila elemen dari notasi itu habis, maka nilai yang tertinggal (teratas dari tumpukan) adalah hasilnya.

5. Penutup

Berdasrkan uraian yang ada maka perancangan model aplikasi ini sangat bermanfaat buat pembelajaran anak sd dan smp yang akan menstimulus kerja otak kita untuk lebih aplikatif belaar dengan media komputer

Daftar Pustaka

[1] Wahyudi,Bagus. Delphi 2010 dan Firebird membuat aplikasi mini market Yogyakarta,Penerbit Gava Media,2010.

[2] Wahana Komputer. Membuat Program Kreatif dan Profesional dengan Delphi

[3] Darmayuda, Ketut. Pemrograman Aplikasi Client Server Bandung: Informatika Bandung,2007.

[4] Analisis dan Perancangan Sistem informasi dengan Metodologi Berorientasi

Objek (Edisi Revisi) Bandung: Informatika Bandung, 2005.

[5] Sukamto, Rossa A dan M. Shalahuddin. Modul Pembelajaran Rekayasa

Perangkat Lunak (Terstruktur dan Berorientasi Objek) Bandung: Modula, 2011.

[6] Wahyudi, Bambang. Pengantar Struktur Data dan Algoritma Yogyakarta: Andi,2004.

[7] Widodo,Prabowo Pudjo dan Herlawati.Menggunakan UML Bandung: Informatika Bandung, 2011.