Nama : Ela Afitria Islami

NPM : G1F021028

Mata Kuliah : Algoritma dan Struktur Data

Tugas Stack (Infix, Prefix dan Postfix)

Soal

1. Tulis algoritma untuk mengonversi ekspresi infiks ke postfix dan infix ke prefix
2. Tulis algoritma untuk mengevaluasi ekspresi postfix dan prefix

Pembahasan

1. **a) Algoritma Infix to Postfix**

   Step 1 : Start
   Step 2 : Pindai ekspresi infix dari kiri ke kanan.
   Step 3 : Jika karakter yang dipindai adalah operand, keluarkan.
   Step 4 : Else,

   4.1 Jika prioritas dan asosiasi operator yang dipindai lebih besar dari prioritas dan asosiasi operator di tumpukan (atau tumpukan kosong atau tumpukan berisi '('), dorong.

   4.2 Operator '^' adalah asosiatif kanan dan operator lain seperti '+','-','*' dan '/' adalah asosiatif kiri. Periksa terutama untuk kondisi ketika kedua atas Stack operator dan operator yang dipindai adalah '^'. Dalam kondisi ini prioritas operator yang dipindai lebih tinggi karena asosiatifitasnya yang tepat. Jadi itu akan didorong di Stack operator. Dalam semua kasus lain ketika bagian atas Stack operator sama dengan operator yang dipindai, kami akan mengeluarkan operator dari Stack karena asosiasi kiri karena operator yang dipindai memiliki prioritas yang lebih rendah.

   4.3 Else ,Pop semua operator dari Stack yang lebih besar dari atau sama dengan prioritas dari operator yang dipindai. Setelah melakukan itu, Push operator yang dipindai ke Stack. (Jika Anda menemukan tanda kurung saat muncul, berhenti di situ dan tekan operator yang dipindai ke dalam Stack.)

   Step 5 : Jika karakter yang dipindai adalah '(', Push ke Stack.
   Step 6 : Jika karakter yang dipindai adalah ')', keluarkan tumpukan dan keluarkan hingga '(' ditemukan, dan buang kedua tanda kurung.
   Step 7 : Ulangi langkah 2-6 hingga ekspresi infiks dipindai.
   Step 8 : Print keluaran
   Step 9 : Pop dan keluarkan dari Stack hingga tidak empty

   Step 10 : End

b) Algoritma Infix to Prefix

Step 1 : Start

Step 2 : Reverse infix yang diberikan

Step 3 : Scan infix dari kiri ke kanan

Step 4 : Setiap ada operand maka print

4.1 Jika operator datang dan Stack ditemukan kosong, maka cukup Push operator ke dalam stack.

4.2 Jika operator yang masuk memiliki prioritas lebih tinggi daripada TOP dari Stack, Push operator yang masuk ke dalam Stack.

4.3 Jika operator yang masuk memiliki prioritas yang sama dengan TOP dari Stack, Push operator yang masuk ke dalam stack.

4.4 Jika operator yang masuk memiliki prioritas lebih rendah dari TOP stack, pop, dan print bagian atas stack. Uji operator yang masuk ke bagian atas stack lagi dan keluarkan operator dari stack sampai menemukan operator dengan prioritas yang lebih rendah atau prioritas yang sama.

4.5 Jika operator yang masuk memiliki prioritas yang sama dengan bagian atas stack dan operator yang masuk adalah ^, maka pop bagian atas stack sampai kondisi benar. Jika kondisinya tidak benar, tekan operator ^.

Step 5 : ketika sudah mencapai akhir dari ekspresi infix, Pop dan print semua operator dari atas Stack

Step 6 : Jika operatornya adalah ‘)’, maka Push ke dalam Stack

6.1 Jika operatornya adalah ‘(‘, maka print semua operator dari Stack hingga ditemukan tanda ‘)’ di Stack

6.2 Jika bagian atas Stack adalah ‘)’, Push operator pada Stack

Step 7 : Reverse output

Step 8 : End

2. a) Algoritma evaluasi Postfix

Step 1 : Start

Step 2 : Buat Stack untuk menyimpan operand (atau nilai).

Step 3 : Scan ekspresi yang diberikan dan lakukan hal tersebut untuk setiap elemen yang dipindai.

3.1 Jika elemennya adalah angka, Push ke dalam tumpukan

3.2 Jika elemennya adalah operator, Pop operand untuk operator dari stack. Evaluasi

operator dan Push hasilnya kembali ke Stack

Step 4 : Ketika ekspresi berakhir, nomor di Stack adalah jawaban terakhir

Step 5 : End

b) Algoritma evaluasi Prefix

Step 1 : Start dari elemen terakhir dari ekspresi.

Step 2 : periksa elemen saat ini.

2.1 Jika itu adalah operand, Push ke stack.

2.2 Jika itu adalah operator, Pop dua operand dari Stack. Lakukan operasi dan Push elemen kembali ke stack.

Step 3 : Lakukan ini sampai semua elemen ekspresi dilalui dan kembalikan stack teratas yang akan menjadi hasil operasi

Step 4 : End