Laporan UTS Praktikum SDA

Kelompok 6
Nama:Abdullah Asy-Syifawi
- M.Anis Fathin

Program ini menggunakan stack untuk memproses ekspresi aritmatika dalam bentuk infix, postfix, dan prefix.

Penjelasan fungsi

1. main()

Fungsi utama ini menjalankan program dan memberikan antarmuka kepada pengguna untuk memilih jenis konversi ekspresi (infix, postfix, atau prefix). Berikut adalah langkah-langkah yang dilakukan oleh main():

- Menampilkan menu konversi ekspresi kepada pengguna (infix ke postfix, postfix ke infix, dll).
- Membaca pilihan konversi dari pengguna.
- Membaca ekspresi yang akan dikonversi, tergantung pada jenis konversi yang dipilih.
- Menggunakan perintah switch-case untuk memanggil fungsi konversi yang sesuai, tergantung pada pilihan pengguna.
- Menampilkan hasil konversi setelah selesai.

2. precedence(char c)

Fungsi ini mengembalikan tingkat prioritas (precedence) operator yang diberikan. Fungsi ini digunakan untuk menentukan apakah operator yang ada di dalam ekspresi memiliki prioritas lebih tinggi atau lebih rendah dibandingkan operator lain.

- ^ memiliki tingkat prioritas tertinggi (3).
- *** dan /** memiliki prioritas lebih rendah dari ^ (2).
- **+ dan -** memiliki prioritas lebih rendah dari * dan / (1).
- Jika karakter bukan operator, fungsi ini mengembalikan 0.

3. strrev(char *str)

Fungsi ini membalikkan (reverse) string yang diberikan sebagai argumen. Fungsi ini digunakan dalam beberapa konversi, misalnya dari infix ke prefix atau dari prefix ke infix, untuk membalikkan urutan ekspresi.

- Fungsi ini bekerja dengan menukar karakter pertama dan terakhir dalam string, kemudian bergerak menuju ke tengah string, membalikkan urutan karakter sampai mencapai pertengahan.

4. infixToPostfix(char *infix, char *postfix)

Fungsi ini mengkonversi ekspresi **infix** (notasi yang umum digunakan) menjadi ekspresi **postfix** (notasi pasca-fiks).

- Menggunakan **stack** untuk menyimpan operator sementara.
- Membaca ekspresi infix dari kiri ke kanan. Jika menemui operand (huruf atau angka), langsung ditambahkan ke ekspresi postfix.
- Jika menemui operator, fungsi ini memeriksa operator di stack dan memastikan operator yang lebih kuat berada di depan dalam ekspresi postfix.
- Fungsi ini juga menangani tanda kurung dengan memindahkan operator ke postfix hingga menemukan tanda kurung yang cocok.
- Setelah selesai membaca seluruh ekspresi, operator yang tersisa di stack dipindahkan ke ekspresi postfix.

5. postfixToInfix(char *postfix, char *infix)

Fungsi ini mengkonversi ekspresi **postfix** menjadi ekspresi **infix**.

- Membaca ekspresi postfix dari kiri ke kanan.
- Jika menemui operand, simpan dalam stack.
- Jika menemui operator, ambil dua operand teratas dari stack dan buat ekspresi infix dengan operator di antara keduanya, kemudian masukkan kembali ke dalam stack.
- Setelah seluruh ekspresi diproses, hasil infix akhir akan berada di stack, dan pop hasilnya untuk menjadi ekspresi infix.

6. infixToPrefix(char *infix, char *prefix)

Fungsi ini mengkonversi ekspresi **infix** menjadi ekspresi **prefix** (notasi Polandia terbalik).

- Langkah pertama adalah membalikkan ekspresi infix.
- Setiap tanda kurung dalam ekspresi dibalik (ubah (menjadi) dan sebaliknya).
- Setelah itu, ekspresi infix yang sudah dimodifikasi dikonversi menjadi ekspresi postfix.
- Setelah konversi postfix selesai, hasil postfix dibalik untuk mendapatkan hasil dalam bentuk prefix.

7. prefixToInfix(char *prefix, char *infix)

Fungsi ini mengkonversi ekspresi **prefix** menjadi ekspresi **infix**.

- Pertama, ekspresi prefix dibalik.
- Ekspresi yang sudah dibalik diproses dengan cara yang mirip dengan proses konversi infix ke postfix. Dalam hal ini, operasi dimulai dari kiri ke kanan.
- Operand dipush ke dalam stack, dan ketika operator ditemukan, dua operand teratas diambil dari stack untuk membentuk ekspresi infix.
- Setelah selesai, ekspresi infix akhir ada di stack dan dipop untuk menjadi hasil infix.

8. prefixToPostfix(char *prefix, char *postfix)

Fungsi ini mengkonversi ekspresi **prefix** menjadi ekspresi **postfix**.

- Langkah pertama adalah mengkonversi ekspresi prefix menjadi ekspresi infix menggunakan fungsi prefixToInfix.
- Setelah mendapatkan ekspresi infix, ekspresi infix tersebut kemudian dikonversi menjadi postfix menggunakan fungsi infixToPostfix.

9. postfixToPrefix(char *postfix, char *prefix)

Fungsi ini mengkonversi ekspresi **postfix** menjadi ekspresi **prefix**.

- Langkah pertama adalah mengkonversi ekspresi postfix menjadi ekspresi infix menggunakan fungsi postfixToInfix.
- Setelah mendapatkan ekspresi infix, ekspresi infix tersebut kemudian dikonversi menjadi prefix menggunakan fungsi infixToPrefix.