Laporan Praktikum Algoritma dan Struktur Data

Praktikum 04 : Stack

Oleh:

Muhammad Dimas Ardiansyah / 5223600019

Program Studi Teknologi Game

Departemen Teknik Multimedia Kreatif

Politeknik Elektronika Negeri Surabaya

2024

Tahapan Praktikum:

1. **Implementasikan Stack menggunakkan Linked List**

   Stack digunakan untuk menyimpan dan mengakses data dengan prinsip Last In First Out (LIFO). Program ini memiliki dua kelas utama: `Node` untuk merepresentasikan elemen dalam stack, dan `ListStack` untuk implementasi stack menggunakan linked list. Elemen-elemen ditambahkan ke stack menggunakan metode `push`, dihapus dari stack menggunakan metode `pop`, dan nilai teratas dari stack dapat dilihat menggunakan metode `peek`.

```
Output

/tmp/OdHjgx57oT.o
Top element: 3
Size of stack: 3
Top element after pop: 2


=== Code Execution Successful ===
```

2. **Memeriksa Keseimbangan Tanda Kurung**

   Menggunakan stack, ia memeriksa apakah setiap tanda buka memiliki tanda tutup yang sesuai. Jika semua tanda kurung seimbang, program mengembalikan `true`, jika tidak, mengembalikan `false`. Dalam contoh ini, ekspresi pertama "{()}" seimbang, sedangkan yang kedua "{())}" tidak seimbang.

```
Output

/tmp/5EA4qPiKNk.o
Expression: {()} is balanced
Expression: {())} is not balanced


=== Code Execution Successful ===
```

3. **Konversi Infix ke Postfix**

   Program ini mengonversi ekspresi matematika dari notasi infix menjadi postfix menggunakan algoritma Shunting-yard. Itu menggunakan stack untuk menyimpan

operator sementara. Selama iterasi melalui karakter-karakter dalam ekspresi infix, program memproses operand dan operator sesuai dengan aturan operator dan prioritasnya. Hasilnya, ekspresi postfix yang setara diperoleh dan dicetak. Dalam contoh ini, ekspresi infix "a+b*(c^d-e)^(f+g*h)-i" dikonversi menjadi ekspresi postfix "abcd^e-fgh*+^*+i-".

```
Output

/tmp/MqwDfyFX6w.o
Infix expression: a+b*(c^d-e)^(f+g*h)-i
Postfix expression: abc(*+de-^fgh*+i-^


=== Code Execution Successful ===
```

4. **Konversi Infix ke Prefix**

Program ini mengonversi ekspresi matematika dari notasi infix menjadi prefix menggunakan algoritma serupa dengan Shunting-yard. Pertama, ekspresi infix dibalik dan tanda kurung dibalik juga. Selama iterasi, operand langsung dimasukkan ke ekspresi prefix, sementara operator-operator dimasukkan ke dalam stack dengan mempertimbangkan prioritasnya. Ketika menemukan tanda kurung tutup, operator-operator di stack dikeluarkan hingga menemukan tanda kurung buka yang sesuai. Operator `^` memiliki prioritas tertinggi. Setelah semua karakter diproses, operator-operator yang tersisa di stack dimasukkan ke dalam ekspresi prefix. Hasilnya, ekspresi prefix yang setara diperoleh dan dibalik kembali untuk mendapatkan ekspresi yang benar. Dalam contoh ini, ekspresi infix "a+b*(c^d-e)^(f+g*h)-i" dikonversi menjadi ekspresi prefix "-+a*b^-cde+*fgh".

```
Output

/tmp/cKzgldvM9R.o
Infix expression: a+b*(c^d-e)^(f+g*h)-i
Prefix expression: -^)+a*bcd+e^f*gh-i


=== Code Execution Successful ===
```

5. **Evaluasi Ekspresi Postfix**

Program tersebut adalah sebuah program C++ yang mengevaluasi ekspresi matematika dalam bentuk postfix. Ekspresi postfix adalah notasi matematika di mana operator ditempatkan setelah operand-operand yang sesuai. Program menggunakan stack untuk menangani operand-operand dan operator dalam ekspresi postfix. Setiap kali program menemukan operand, itu dimasukkan ke dalam stack. Ketika program menemukan operator, ia mengambil dua operand teratas dari stack, melaksanakan operasi yang sesuai, dan hasilnya dimasukkan kembali ke dalam stack. Setelah iterasi selesai, hasil akhir evaluasi diperoleh dari nilai teratas stack. Dalam contoh ini, ekspresi postfix yang diberikan adalah "83+72*-" yang setelah dievaluasi menghasilkan nilai -25.

```
Output

/tmp/QdhMZPYuat.o
Postfix expression: 83+72*-
Result: -3


=== Code Execution Successful ===
```

6. **Palindrome String menggunakkan Stack**

Program tersebut adalah program C++ untuk memeriksa apakah sebuah string adalah palindrom atau tidak. Ini dilakukan dengan menggunakan sebuah stack untuk membandingkan setengah karakter pertama dengan setengah karakter terakhir dari string. Program akan menghapus spasi dan mengubah huruf-huruf menjadi huruf kecil sebelum melakukan pengecekan.

```
Output

/tmp/aDiWaTmrSv.o
Masukkan sebuah string: Hannah
"Hannah" is a palindrome


=== Code Execution Successful ===
```