**Laporan Praktikum**

**STACK**



Disusun oleh:

Tomi Martino Affandi

2341720245

D-IV Teknik Informatika - 1E

Jl.Soekarno Hatta No.9, Jatimulyo, Kec.Lowokwaru, Kota Malang Jawa Timur 65141

Phone : (0341)404424, 404425

E-email : Polinema.ac.id

**1. Tujuan Praktikum**

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Membuat struktur data Stack

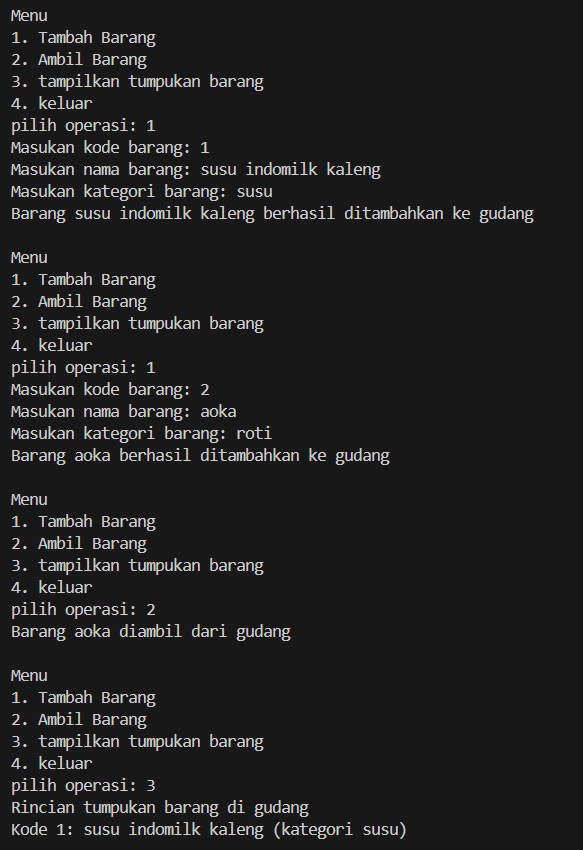
2. Menerapkan algoritma Stack ke dalam program

**2.1 Percobaan 1: Penyimpanan Tumpukan Barang dalam Gudang**

Code :

<https://github.com/TomiMartinoAffandi/Algoritma---Struktur-Data/commit/2dd413dec22c3173502acdb3e5144a5df366c8f0>

Output :



Pertanyaan

1. Tidak ada yang perlu di edit, semuanya sudah sesuai
2. 7 barang



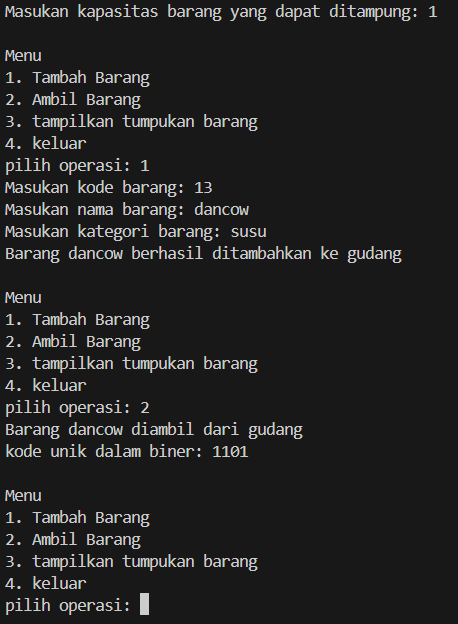
1. Kode !cekKosong() digunakan untuk cek kondisi apakah ada barang pada gudang, jika kode tersebut dihapus maka kode tidak akan mengeksekusi apa-apa
2. <https://github.com/TomiMartinoAffandi/Algoritma---Struktur-Data/commit/f571aa727a2220fd5089e6e9c2500841f1c96c2a>
3. Done

**2.2 Percobaan 2: Konversi Kode Barang ke Biner**

Code :

<https://github.com/TomiMartinoAffandi/Algoritma---Struktur-Data/commit/22009c5e5dd09882eabd3768bac5e66bdb605b8f>

Output :



Pertanyaan

1. Hasil sama saja, namun ketika kode kurang dari 0 maka kode biner yang dihasilkan juga mines
2. Cara kerja:

-terdapat parameter kode bertipe integer

-pembuatan objek stack dari class StackKonversi26

-terdapat perulangan dengan kondisi != 0

-variable sisa untuk menyimpan hasil bagi dari kode%2

-memanggil method push dari class StackKonversi26 dengan parameter variable sisa, variable sisa akan disimpan dalam sebuah array

-kode dibagi 2

-iterasi akan berhenti jika kode = 0

-membuat variable biner

-membuat perulangan dengan kondisi !stack.isEmpty dimana method isEmpty mereturn top -1 jika kondisi true(top = -1)

-variable biner akan ditambah dengan stack.pop(), dalam method pop merupakan pemindahan value variable sisa dari sebuah array

-iterasi akan terus dilakukan selama belum memenuhi kondisi true dari method isEmpty

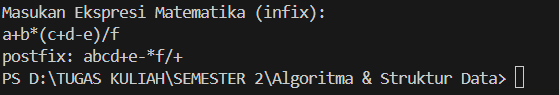
-return value dari biner

**2.3 Percobaan 3: Konversi Notasi Infix ke Postfix**

Code :

<https://github.com/TomiMartinoAffandi/Algoritma---Struktur-Data/commit/d226b3ee7be228c47f48e61a5ebc6ee3c072f038>

Output :



Pertanyaan

1. Karena skala prioritas setipa operator ada yang berbeda dan ada yang sama, contoh:

(\*, /, %) skala prioritas nya sama maka return value nya juga sama, jika return value dari ketiga operator tersebut diubah maka akan mempengaruhi urutan evaluasi dalam ekspresi postfix yang dihasilkan

1. Alur kerja:

* deklarasi variabel n, top, dan stack. Variabel n menampung panjang dari string yang akan dikonversi, top adalah indeks teratas dari stack, dan stack merupakan array karakter untuk menyimpan operator.
* deklarasi konstruktor postfix26 yang menerima satu parameter total yang akan digunakan untuk menginisialisasi variabel n, top, dan stack, serta melakukan push karakter '(' ke dalam stack.
* metode push(char c) untuk menambahkan elemen ke dalam stack. dilakukan dengan menambahkan nilai top dan memasukkan karakter c ke dalam indeks top pada array stack.
* metode pop() untuk mengeluarkan elemen teratas dari stack. dilakukan dengan mengambil nilai pada indeks top dari array stack dan kemudian mengurangi nilai top.
* metode isOperand(char c) yang mengembalikan nilai true jika karakter c merupakan operand (huruf besar, huruf kecil, angka, spasi, atau titik), dan false jika tidak.
* metode isOperator(char c) yang mengembalikan nilai true jika karakter c merupakan operator (^, %, /, \*, -, atau +), dan false jika tidak.
* metode derajat(char c) yang mengembalikan prioritas dari operator c. Prioritas operator ditentukan secara tetap dengan operator seperti ^ memiliki prioritas tertinggi (3), diikuti oleh %, /, dan \* (prioritas 2), dan terakhir + dan - (prioritas 1).
* metode konversi(String Q) yang melakukan konversi dari notasi infix ke postfix. menerima string Q sebagai input dan mengembalikan string P yang merupakan hasil konversi postfix. Pada setiap iterasi, karakter dari string input Q dicek:
* Jika karakter adalah operand, maka karakter tersebut ditambahkan ke string P.
* Jika karakter adalah '(', maka karakter tersebut dimasukkan ke dalam stack.
* Jika karakter adalah ')', maka karakter tersebut di-pop dari stack sampai ditemukan '('.
* Jika karakter adalah operator, maka dilakukan pengecekan prioritas operator pada stack dengan operator saat ini. Operator-operator dengan prioritas yang lebih tinggi di-stack akan dikeluarkan dan dimasukkan ke dalam string P sampai prioritas operator pada stack lebih rendah daripada operator saat ini. Kemudian, operator saat ini dimasukkan ke dalam stack.
  + Proses ini berlanjut sampai semua karakter pada string input Q telah diproses. String P yang merupakan hasil konversi postfix akan dikembalikan.

1. Untuk mengambil karakter dari string q pada indek ke-i. dan akan disimpan ke dalam variable c

**2.4 Latihan Praktikum**