sesatmaxxing

algoritma dan pemrograman

the one and only

kafeyangasli

1/4/25

1. Diberikan potongan teks algoritma di bawah ini, pada akhir eksekusi berapa nilai akhir variable jum, jawaban anda disertai dengan proses step by stepnya untuk mendapatkan nilai akhir variable jum tersebut.

```
{Kamus}
   i, k, jum: integer
{Algoritma}
   i ← 1; j ← 0 {inisialisasi}
   while (i <= 4) do
         if ((i mod 2) = 0) then {i genap}
               k traversal [1..i]
                     jum ← jum+i+k
         else
               jum ← jum+i+1
         i ← i + 1
   {endWhile}
```

Pembahasan (asumsi memang soal begitu):

jum tidak akan bernilai apa-apa, karena tidak pernah diinisialisasi dalam algoritma, hanya dideklarasikan. Dan terdapat variabel *j* yang tidak pernah dideklarasikan, sehingga bisa diasumsikan algoritma tidak akan pernah berjalan.

Pembahasan (asumsi jika j pada line ke-2 mengacu pada jum):

Perulangan yang diberikan akan berhenti ketika i mencapai 5, dimana i sekarang bernilai 1. Sehingga, akan dilakukan perulangan sebanyak 4 iterasi.

- Iterasi Pertama
  i saat ini bernilai 1, dimana ia ganjil (i mod 2 ≠ 0). Maka, ia masuk ke kondisi kedua, yang berarti jum sekarang bernilai 2 (jum ← 0 + 1 + 1).
- Iterasi Kedua
  i saat ini bernilai 2, dimana ia genap (i mod 2 = 0). Maka, ia masuk ke kondisi pertama. Pada kondisi pertama, dilakukan iterasi sebanyak dua kali, yakni dari 1 sampai 2. Sehingga, jum akhir bernilai 9. (jum ← 2 + 2 + 1, jum ← 5 + 2 + 2).
- Iterasi Ketiga
  i saat ini bernilai 3, dimana ia ganjil (i mod 2 ≠ 0). Maka, ia masuk ke kondisi kedua, yang berarti jum sekarang bernilai 13 (jum ← 9 + 3 + 1).
- Iterasi Keempat
  i saat ini bernilai 4, dimana ia genap (i mod 2 = 0). Maka, ia masuk ke kondisi pertama. Pada kondisi pertama, dilakukan iterasi sebanyak empat kali, yakni dari 1 sampai 4. Sehingga, jum akhir bernilai 39. (jum ← 13 + 4 + 1, jum ← 18 + 4 + 2, jum ← 24 + 4 + 3, jum ← 31 + 4 + 4).

2. Diberikan sebuah tabel integer T dengan ukuran 1 s/d N. Tabel T berisi elemen mulai dari posisi ke 1 s/d Neff (dimana Neff <= N, dan elemen pada posisi yang belum terisi atau setelah Neff s/d N di set dengan nilai 0). Buatlah teks algoritma dalam bentuk program utama untuk melakukan pergeseran elemen-elemen tabel T dengan cara meletakan elemen posisi Neff pada posisi N, Neff-1 pada posisi N-1, Neff-2 pada posisi N-2, … dst. Kemudian nilai elemen pada posisi sisa dari hasil pergeseran di set menjadi 0.

Pembahasan:

Salah satu cara kita dapat mendekati proses algoritma dari soal adalah:

1. **Initial State (T_0)** → Diberikan input N dan Neff (Neff <= N)
2. **Final State (T_1)** → Tabel dengan elemen Neff ke N, Neff -1 ke N-1, dst.
3. **Proses** → Masukan N, Neff, dan T;
   Atur j = 0, lalu traversal dari i = Neff sampai 1;
   Elemen T_{N-j} diganti dengan T_i, tambah 1 ke j, lalu atur $T_i = 0$

```
{Kamus}
   N: integer
   Neff: integer
   i: integer
   j: integer
   T: array [1..N] of integer

{Algoritma}
   input(N, Neff)
   depend on N, Neff:
         N <= 0 OR Neff <= 0: output("Input harus positif!")
         Neff > N: output("Neff harus <= N!")
         else:
               i traversal [1..N] {Masukan T}
                     if i <= Neff then
                           input(Ti)
                     else
                           Ti = 0 {memastikan elemen setelah Neff diset 0}
               {endTraversal}
               j ← 0
               i traversal [Neff..1] {Proses pemindahan}
                     TN-j ← Ti
                     Ti ← 0
                     j ← j + 1
               {endTraversal}
               i traversal [1..N] {Output}
                     output(Ti)
```

3. Diberikan type `TabInt: array [1..100] of integer`. Buatlah teks algoritma dalam bentuk sub program untuk fungsi cek tabel simetris.

Pembahasan:

Kita diminta untuk menyusun algoritma berdasarkan spesifikasi yang diberikan:

```
Function IsTableSimetri(T1: TabInt, T2: Tabint) → boolean
{Mengirimkan TRUE jika nilai setiap elemen T1 sama dengan T2}
{Misal:}
{T1 = <1 3 4 5 7 8 9>, T2 = <1 3 4 5 7 8 9>, maka TRUE}
{T1 = <1 3 4 5 7 8 9>, T2 = <1 3 4 5 6 8 9>, maka FALSE}
```

Terdapat dua jenis perulangan yang bisa digunakan, antara `traversal` atau `while do`.

Jika menggunakan `traversal` maka akan terlihat seperti berikut:

```
Function IsTableSimetri(T1: TabInt, T2: TabInt) → boolean

{Kamus Lokal}
   i: integer {counter}
   beda: integer {jumlah elemen yang beda}

{Algoritma}
   beda ← 0
   i traversal [1..100]
         if (T1i != T2i) then
               beda ← beda + 1
   {endTraversal}
   → ( beda = 0 )
```

Jika menggunakan `while do` maka akan terlihat seperti berikut:

```
Function IsTableSimetri(T1: TabInt, T2: TabInt) → boolean

{Kamus Lokal}
   i: integer {counter}
   beda: integer {jumlah elemen yang beda}

{Algoritma}
   i ← 1; beda ← 0;
   while (beda = 0 AND i <= 100) do
         if (T1i != T2i) then
               beda ← 1
         else
               i ← i + 1
   {endWhile}
   → ( beda = 0 )
```

Keduanya akan sama-sama mengecek kesimetrisan dua tabel masukan, namun ketika kita menggunakan `while do`, perulangan akan berhenti ketika ditemukan satu elemen yang berbeda, sedangkan `traversal` akan memeriksa sampai elemen terakhir, sehingga ‘kurang efisien’.

4. Buatlah teks algoritma dalam bentuk program utama untuk menghitung banyaknya kata yang diakhiri dengan pasangan karakter ‘LE’ dari sebuah pita karakter. Definisi kata yang digunakan pada persoalan ini adalah jika ketemu spasi, koma, dan titik. Anda dapat langsung menggunakan primitif Mesin Karakter (START(), ADV(), dan EOP()) tanpa harus melakukan realisasi.

   Contoh: "**Sale** pisang lebih enak dibandingkan ikan **lele**, punya **Sule**."
   Dari pita karakter di atas akan menghasilkan 3 kata yang diakhiri dengan pasangan "LE"

   Pembahasan:

   Sebelumnya disclaimer saya kurang paham mengenai soal ini, jadi mohon maaf jika kurang jelas, but I will do the best of my ability to explain!

   Jadi, untuk mengiterasi suatu string / *array of characters* (`array [1..N] of character`), kita dapat menggunakan yang namanya **mesin karakter**. Mesin karakter ini akan memproses setiap karakter dari string yang dimasukkan. Dalam permasalahan algoritma seperti ini, kita tidak perlu menuliskan input(string) apabila merujuk kepada diktat pemrograman prosedural.
   *(Mohon koreksinya)

   Mesin karakter ini terdiri atas:
   1. Pita karakter atau string
   2. Tombol **START** dan **ADV**
      a. **START**
         START berguna sebagai penanda bahwa mesin akan diinisialisasikan dan siap membaca karakter pertama dari pita karakter.
      b. **ADV**
         ADV berguna sebagai petunjuk bagi mesin untuk memajukan pita tepat satu karakter (incremental)
   3. Lampu **EOP**
      EOP berguna sebagai penanda akhir dari sebuah pita. Dalam kasus ini, anggap saja akhiran dari semua masukan pita adalah titik.
   4. (unofficial) Hidden Variable **CC**
      Berdasarkan diktat prosedural milik ITB yang saya baca, terdapat juga variabel tak terlihat **CC** sebagai penanda mesin sedang berada di karakter mana. Anggap saja sebagai **indeks**.
      *(Mohon koreksinya)

   Lalu, bagaimana penggunaannya?
   Ketika kita ingin mulai membaca string, maka kita menulis **START** dalam algoritma, yang berarti CC sekarang menunjuk pada character pertama dari string. Jika diperlukan, kita juga dapat menginisialisasi indeks dengan nama bebas (misal *i*), sebagai penanda sudah berapa banyak karakter

yang dibaca. Ketika kita selesai mengecek karakter yang pertama, maka kita akan memanggil **ADV** untuk memberitahu mesin untuk melanjutkan ke karakter yang selanjutnya. Untuk memberitahu mesin bahwa kita sudah mencapai akhir dari string, kita dapat menggunakan **EOP**. Ketika **EOP** bernilai true, maka selesai proses mengiterasi karakter dalam string.

Misal, kita ingin menghitung panjang dari suatu string dengan mengecualikan karakter spasi. Algoritmanya akan tampak seperti berikut:

```
Program hitungPanjangTanpaSpasi
{Menghitung panjang dari suatu string}

Kamus
   panjang: integer {panjang string}
   Function EOP: (ch: character) → boolean
   {cek apakah sudah berakhir}
   {I.S: diberikan sebuah karakter}
   {F.S: TRUE bila ch = '.', FALSE bila ch != '.'}

Algoritma
   panjang ← 0
   START
   while (NOT EOP(CC)) do
         if CC != ' ' then {jika bukan karakter spasi}
               panjang ← panjang + 1
         ADV
   output(panjang)
```

Bagaimana? Sudah cukup bingung?
Jika masih, silahkan bertanya di Discord maupun WA(jika ada). 😊

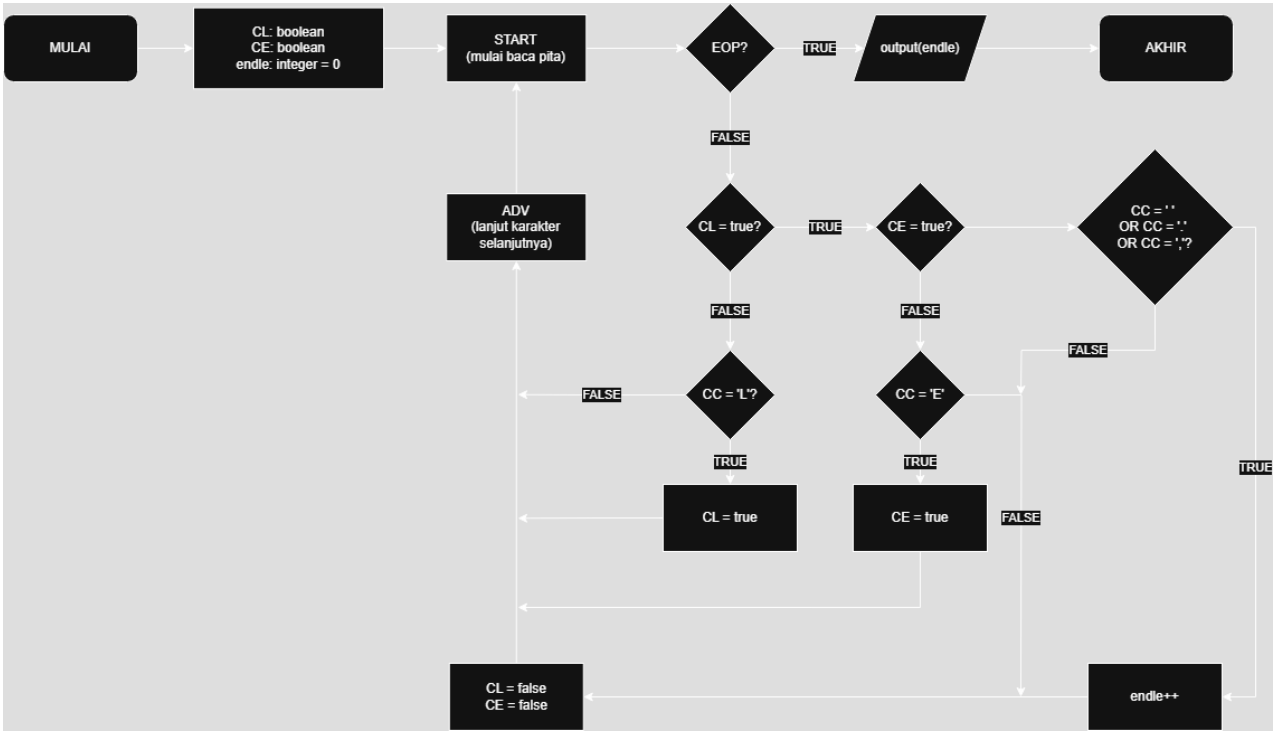
Oke, sekarang kita fokus ke soal.

Kita diminta untuk menghitung banyaknya kata yang diakhiri dengan pasangan karakter ‘LE’ dari sebuah pita karakter, yang dimana definisi kata adalah karakter selanjutnya merupakan spasi, titik, atau koma.

Pendekatan yang bisa kita ambil adalah dengan mengecek CC sama dengan L atau bukan? Jika iya, maka akan membuat suatu variabel bernilai true, yang dimana jika variabel tersebut bernilai true, akan mengecek jika karakter CC setelah L adalah E. Jika iya, maka akan membuat suatu variabel lainnya bernilai true, yang dimana jika variabel tersebut bernilai true, akan mengecek jika karakter CC setelah

E adalah spasi, titik, atau koma. Jika true, maka itulah kata yang berakhiran dengan LE sehingga ditambah ke variabel jumlah kata berakhiran LE!

Bingung memvisualisasikannya? Amati flowchart berikut:



Sehingga, didapatkan teks algoritmiknya:

```
Program hitungKataAkhiranLE
{Menghitung kata dengan akhiran le.}

Kamus
   CL: boolean
   CE: boolean
   endle: integer {jumlah kata berakhiran LE}
   Function EOP: (ch: character) → boolean
   {cek apakah sudah berakhir}
   {I.S: diberikan sebuah karakter}
   {F.S: TRUE bila ch = '.', FALSE bila ch != '.'}

   Procedure RESETC: (input/output: CL,CE: boolean)
   {I.S: CL, CE bernilai sembarang}
   {F.S: CL dan CE bernilai FALSE}
```

```
Algoritma
   endle ← 0
   CL ← false
   CE ← false
   START
   while (NOT EOP(CC)) do
         depend on CL, CE, CC:
               CE = true: {jika CE = true, maka CL pasti true}
                     if (CC = '.' OR CC = ' ' OR CC = ',') then
                           endle ← endle + 1
                     else
                           RESET(CL, CE)
               CL = true: {baru hanya CL saja)
                     if (CC = 'E') then
                           CE ← true
                     else
                           RESET(CL, CE)
               else: {kondisi dimana belum ditemukan karakter L dan E}
                     if (CC = 'L') then
                           CE ← true
         ADV
   {akhirWhile}
   output(endle)
```