Praktikum 9

Fuctions (Fungsi) di AWK

Tujuan Pembelajaran

Mahasiswa dapat memahami dan menggunakan fungsi dalam bahasa pemrograman awk.

Dasar Teori

Fungsi merupakan salah satu struktur dasar dalam program awk. awk menyediakan built-in function, yang dapat langsung dipanggil tanpa harus mendefinisikan terlebih dahulu program. Parameter fungsi dalam awk dibentuk oleh ekspresi yang diberikan melalui argumen. Argumen yang diberikan diolah sebelum dipakai sebagai parameter saat pemanggilan fungsi. Selain built-in function, user dapat membuat userdefined function yang dapat didefinisikan dalam program. Cara pemanggilannyasama seperti built-in function.

Percobaan 1: Built-in Function

Built-in function adalah fungsi yang telah didefinisikan dan dapat langsung dipanggin dalam program awk. Berikut ini contoh built-in function sqrt() sebagai fungsi akar kuadrat. Saat suatu fungsi dalam program awk dipanggil, ekspresi yang membentuk parameter fungsi diolah sebelum pemanggilan dilakukan. Variabel i ditambah (increment) menjadi 5 sebelum fungsi sqrt dipanggil. Maka argument saat fungsi sqrt tersebut dipanggil nilai argumennya adalah 5, bukan nilai awal i (4).

```
$ awk 'BEGIN {
      i=4
      j=sqrt(i++)
      printf("i = %d\nj = %d\n",i,j)
```

```
}'
```

Fungsi pada percobaan berikut ini memiliki lebih dari satu parameter. Urutan pengolahan ekspresi yang membentuk parameter fungsi belum terdefinisi. Berikut ini contoh fungsi **atan2().** Tampak pada hasil percobaan nilai i menjadi 12, hal ini menunjukkan bahwa urutan pengolahan paramaternya dari **kiri ke kanan**. Variabel i bernilai awal 5, yang kemudian ditambah menjadi 6, lalu menjadi 12. Jika urutannya dari kiri maka seharusnya nilai i adalah menjadi 10 lalu 11.

```
$ awk 'BEGIN {
      i=5
      j=atan2(i++, i*=2)
      printf("i = %d\nj = %d\n",i,j)
}'
```

Percobaan 2: Numeric Function

Percobaan berikut ini menggunakan built-in function **int()** yang merupakan fungsi yang mereturn nilai integer yang terdekat dari nilai argumennya.

```
$awk 'BEGIN {
      i0=int(3)
      i1=int(3.9)
      i2=int(-3.9)
      i3=int(-3)
      printf("i0=%d\n i1=%d\n i2=%d\n i3=%d\n", i0,i1,i2,i3)
}'
```

Percobaan berikut ini menggunakan built-in function **sqrt()** yang mengembalikan nilai akar dari suatu nilai. Jika argumen yang diberikan bernilai negatif maka akan muncul peringatan dan nilai return nya adalah "nan" (not a number).

```
$awk 'BEGIN {
      i0=sqrt(4)
      i1=sqrt(16.4)
      i2=sqrt(-25)
      printf("i0=%d\n i1=%d\n i2=%d\n", i0, i1, i2)
}'
```

Percobaan berikut ini menggunakan built-in function **exp()** yang mengembalikan nilai akar exponensial dari argumen, untuk x maka nilai return nya adalah (e^x). Jika argumen yang diberikan di luar batas maka akan muncul peringatan. Batas tersebut tergantung pada ukuran representasi nilai float mesin.

```
$awk 'BEGIN {
      i0=exp(10)
      i1=exp(15.4)
      i2=exp(-2)
      printf("i0=%d\n i1=%d\n i2=%d\n", i0, i1, i2)
}'
```

Percobaan berikut ini menggunakan built-in function **log()** yang mengembalikan nilai logaritma natura dari argumen. Jika argumen yang diberikan bernilai negatif maka akan muncul pesan error.

```
$ awk 'BEGIN {
      i0=log(10)
      i1=log(255.64)
      i2=log(-16)
      printf("i0=%d\n i1=%d\n i2=%d\n", i0, i1, i2)
}'
```

Percobaan berikut ini menggunakan built-in function **sin(), cos(),** dan **atan2**. sin() adalah fungsi yang mengembalikan nilai sinus dari argumen. cos() adalah fungsi yang mengembalikan nilai cosinus dari argumen. atan2() adalah

fungsi yang mengembalikan nilai arctan dari y/x, untuk atan2(y,x). Semua argumen untuk fungsi tersebut adalah dalam satuan radian.

```
$ awk 'BEGIN {
      i0=sin(60)
      i1=cos(60)
      i2=atan2(120,2)
      printf("i0=%d\n i1=%d\n i2=%d\n", i0, i1, i2)
}'
```

Percobaan berikut ini menggunakan built-in function **rand()** yang mengembalikan nilai bilangan acak bulan positif. Pengali n yang diberikan akan menghasilkan bilangan acak yang kurang dari n yang berkisar atara 0 sampai n-1.

```
function randint(n){
      return int( n * rand() )
}
```

Percobaan berikut ini menggunakan built-in function **rand()** yang mengembalikan nilai bilangan acak bulan positif. Pengali n yang diberikan akan menghasilkan bilangan acak yang kurang dari n yang berkisar atara 0 sampai n-1. Dengan memberikan argumen 6, maka sebagai batas nilai random yang dihasilkan adalah antara 1 sampai 6.

```
#Function to roll a simulated dice
function roll(n){
      return 1 + int(rand() * n)
}

#Roll 3 six-sided dice and print total number of points
{
      printf("%d points\n", roll(6)+roll(6)+roll(6))
}
```

Percobaan 3: String-Manipulation Function

Percobaan berikut ini menggunakan built-in function **asort()** yang merupakan ekstensi gawk yang spesifik, mengembalikan nilai dari elemen dari array asal. Isi dari array asal diurutkan dengan aturan standar gawk. Nilai indeks dari array yang diurutkan tersebut diganti dengan urutan integer yang dimulai dari 1.

```
$awk 'BEGIN {
      a["last"]="de"
      a["first"]="sac"
      a["middle"]="cul"
      asort(a)
      for(i=1;i<4;i++)
            printf("\ni[%d] = %s",i,a[i])
      printf("\n")
}'
```

Percobaan berikut ini menggunakan built-in function **asorti()** yang bekerja seperti asort, namun yang diurutkan adalah indeks array, bukan nilai elemennya. Pengurutan indeks menggunakan aturan pembandingan string, karena indeks array selalu berupa string. Modifikasi program diatas seperti program dibawah ini:

```
$awk 'BEGIN {
      a["last"]="de"
      a["first"]="sac"
      a["middle"]="cul"
      asorti(a)
      for(i=1;i<4;i++)
            printf("\ni[%d] = %s",i,a[i])
      printf("\n")
}'
```

Percobaan berikut ini menggunakan built-in function **index()** yang mengembalikan index posisi dari substring yang dicari dalam string target. Misalnya string “an” dalam “pe**an**ut” berada pada indeks 3. Jika substring yang dicari tidak ada dalam string target maka nilai return fungsi ini adalah 0.

```
$ awk ‘BEGIN {
      print index(“peanut”,”an”)
}’
```

Percobaan berikut ini menggunakan built-in function **length()** yang mengembalikan jumlah karakter dari suatu string. Jika argumennya berupa angka, maka akan diperlakukan sebagai string. Untuk 15*35 hasilnya adalah 525 maka argumen yang digunakan ketika fungsi ini dijalankan adalah string “525”, sehingga nilai return nya adalah 3. Begitu juga dengan 15-35 adalah -20 dianggap sebagai string “-20”.

```
$awk ‘BEGIN {
      x[0]=length(“abcde”)
      x[1]=length(15*35)
      x[2]=length(15-35)

      for(i=0;i<3;i++)
            printf(“\nx[%d]=%s”, i, x[i])
      printf(“\n”)
}’
```

Percobaan berikut ini menggunakan built-in function **match()** yang mencari substring yang terpanjang dan paling kiri dalam substring yang cocok dengan regex. Return value nya adalah posisi karakter atau indeks dari hasil pencarian. Jika tidak ditemukan maka nilai return fungsi ini adalah nol.

File : data

```
FIND ru+n
My program runs
But not very quickly
FIND Melvin
JF+KM
This line is property of Reality Engineering Co.
Melvin was here.
```

File : match.awk

```
{
      if($1==”FIND”)
            regex = $2
      else {
            where = match($0, regex)
            if(where != 0)
            print “Match of”, regex, “found at”, where, “in”, $0
      }
}
```

Jalankan perintah dibawah ini :

```
$ awk –f match.awk data
```

Fungsi match() memberikan nilai built-in variable RSTART ke indeks, dan RLENGTH untuk panjang karater dari substring yang cocok. Jika tidak ada yang cocok, maka RSTART di-set menjadi 0, dan RLENGTH menjadi -1. Percobaan berikut ini menggunakan built-in function **split()** yang berfungsi memecah string menjadi bagian yang dipisahkan oleh separator. Pada percobaan ini separator yang diberikan adalah “-“, sehingga string “cul-de-sac” akan pecah menjadi 3 field, “cul” “de” “sac” yang masing-masing tersimpan sebagai elemen array target a.

```
$ awk ‘BEGIN {
      split(“cul-de-sac”, a, “-“)
```

```
      for(i=1;i<4;i++)
            printf("\na[%d]=%s", i, a[i])
      printf("\n")
}'
```

Percobaan berikut ini menggunakan built-in function **sprintf()** yang berfungsi yang mengembalikan nilai yang sama dengan hasil output dari fungsi printf. Pada percobaan ini, hasil dari sprintf disimpan dalam variabel "pival". Ketika nilai pival ditampilkan maka sama seperti menampilkan argumen sprintf dengan menggunakan fungsi printf.

```
$awk 'BEGIN {
      pival = sprintf("pi = %.2f (approx)", 22/7)
      print pival
}'
```

Percobaan ini akan menghasilkan nilai numerik dari field pertama yang dibaca. Input dengan awalan '0x' maka akan dianggap sebagai nilai heksa.

```
$ echo 0x11 | gawk '{
      printf "\n%d\n", strtonum($1)
}'
```

Percobaan berikut ini menggunakan built-in function **sub()** yang berfungsi mengganti substring terpanjang paling kiri pada target string yang cocok dengan pattern menjadi string yang diberikan. Pada percobaan ini pattern /at/ ditemukan pada "water, water, every where", sehingga string tersebut menjadi "wither, water, every where". "water" yang kedua tidak berganti karena sub() hanya berlaku sekali (pada substring terpanjang paling kiri pada target string yang cocok).

```
$ awk 'BEGIN {
      str = "water, water, every where"
      temp = str
      sub(/at/, "ith", str)
      printf("\nstr = %s\ntemp = %s\n", str,temp)
}'
```

Percobaan berikut ini menggunakan built-in function **gsub()** yang berfungsi mengganti semua substring target string yang cocok dengan pattern menjadi string yang diberikan, sama seperti sub namun gsub berlaku global. Pada percobaan ini pattern /water/ ditemukan pada "water, water, every where", sehingga string tersebut menjadi "fire, fire, every where".

```
$awk 'BEGIN {
      str = "water, water, every where"
      temp = str
      gsub(/water/, "fire", str)
      printf("\nstr = %s\ntemp = %s\n", str,temp)
}'
```

Percobaan di atas menggunakan built-in function **substr()** berfungsi untuk mengambil sebagian string (substring) dari string target. substr("washington", 5, 3) akan mengambil subtring dari indeks 5 dengan panjang 3 karakter, sehingga hasilnya "ing". Jika panjang substring tidak diberikan, maka akan diambil sampai akhir string. substr("washington", 5) akan mengambil subtring dari indeks 5 hingga akhir string, hasilnya "ington".

```
$awk 'BEGIN {
      print substr("washington", 5, 3)
      print substr("washington", 5)
}'
```