Matheus Augusto de Souza
118164136
mataugusto1999@gmail.com

O programa count-digits recebe um arquivo, ou até mesmo algo escrito por você, como entrada e retorna o número de vezes que os dígitos de 1 á 9 aparecem cada, além dos espaços em branco e a soma de todos outros caracteres não especificados.

Vamos ver o seu código concomitantemente com as explicações para ficar mais compreensível de se ler a semântica do próprio. A parte dos códigos que nos interessa, é justamente a parte da criação e compilação do count-digits, então é pelo código fonte que iremos iniciar nossa análise.

Vamos começar pela segunda função que temos no código, para que seja mais fácil a compreensão.

“
```
  while ((c = getchar()) != EOF) {
    if (c >= '0' && c <= '9')
      ++ndigit[c - '0'];
    else if (c == ' ' || c == '\n' || c == '\t')
      ++nwhite;
    else
      ++nother;
  }
```
“

Começamos aqui com um laço while, que irá ler nossa entrada e terminará de executar a sua condição quando chegar ao fim do arquivo (End-Of-File = EOF), a sua condição será dada pelo valor atribuído em “getchar”. A função getchar vai ser a responsável por incluirmos um argumento na função count-digits, já que quando ela é executada, o programa espera que o usuário digite alguma coisa. Por conta dessa função podemos passar um arquivo, ou nós mesmos podemos escrever algo, para que o count-digits seja executado usando esse argumento como entrada. Apesar de não estar no início da linha de códigos, podemos entender essa função como a primeira a ser executada.

Com o if do código, vemos que a condição dele funcionar é o valor de c ter sido atribuído aos índices de 0 a 9, temos que “uma expressão usando **&&** é verdadeira somente se ambos os operadores forem verdadeiros (não zero)”, essa linha nos retornará o número de vezes que cada índice aparece, já que estamos dentro de um laço while, e adicionar o valor à variável “ndigit”.

O else if, que segue o comando anterior, vai contabilizar o número de vezes que espaços em branco, seja por apertar barra de espaço, tab ou enter para pular de linha, aparecem e adicionar à variável “nwhite”.

E, por fim, o else final irá contabilizar o número de outros caracteres, sejam letras ou símbolos, e adicionar à variável “nother”.

Agora podemos ver o inicio do código para compreender melhor.

“

```
#include <stdio.h>

#define IN 1
#define OUT 0

int main(void) {
  int c, nwhite, nother;
  int ndigit[10]; /* 0, 1, 2, 3, 4, ..., 9 */

  nwhite = nother = 0;
  for (int i = 0; i < 10; ++i)
    ndigit[i] = 0;
```

“

Aqui o código começa atribuindo valores a duas variáveis, são elas: IN (dentro) = 1 e OUT (fora) = 0. Após isso, vemos que os valores que “c”, “nwhite”, “nother” e “ndigit” assumirem serão todos valores inteiros e que os valores de “ndigit” (número de dígitos) será de 0 a 9, ou seja, 10 índices.

Para cada valor de i (0 a 9) a condição “for” vai contabilizar o número de vezes que ele aparecer e terminada a contabilização deste número ele adicionará +1 no valor de “i” e esse novo valor será o valor a ser consultado e contabilizado na entrada, no fim retornará um número inteiro, como vimos anteriormente, com o número de vezes que o valor aparece.

Agora podemos ver e compreender melhor como os valores de “ndigit”, “nwhite” e “nother” serão retornados para nós.

“

```
  printf("digits =");
  for (int i = 0; i < 10; ++i)
    printf(" %d", ndigit[i]);
  printf(", white space = %d, other = %d\n", nwhite, nother);
}
```

“

A função printf tem como objetivo retornar para nós o valor de qualquer tipo de dado, no nosso caso será os valores das variáveis. “ndigit’, “nwhite” e “nother”.

A variável “digits” irá fazer uma lista, com o uso do “for”, com todos os números de vezes que cada um dos dígitos de “ndigit” aparece, ou seja, uma lista com 10 valores, onde o primeiro equivale ao número de vezes que o dígito 0 aparece e o último equivale ao número de vezes que o dígito 9 aparece.

O último printf que temos, vai nos retornar o número de vezes que os espaços em branco aparece e o número da soma de todos os caracteres restantes, ou seja, os não incluídos como espaços em branco e como dígitos.

Agora vamos ver a parte de compilação do arquivo count-digits.c.

A compilação do count-digits no arquivo Makefile se dá através do seguinte código:

“

```
count-digits: count-digits.o
	gcc -o count-digits count-digits.o

count-digits.o: count-digits.c
	gcc $(CFLAGS) -c count-digits.c
```

”

Esse código, automatiza o processo de compilação que costumamos chamar manualmente no shell que usamos (nesse caso, o Sublime), por essa razão vemos os seguintes comandos “gcc $(CFLAGS) -c count-digits.c” que transforma o código C que escrevemos em um código que pode ser compilado por qualquer máquina no futuro e “gcc -o count-digits count-digits.o” que compila o código anterior em um código executável pelo sistema em questão.

A variável CFLAGS define as opções que devem ser passadas ao compilador, para este código temos CFLAGS = -Wall -x c -g -std=c99 -pedantic-errors.