Ficha 8

Programação Imperativa

O programa apresentado ao lado calcula o número de palavras de um texto.
O objectivo das questões abaixo é modificar este programa de forma a calcular a palavra mais frequente de um texto.
Para isso vamos começar por calcular quantas vezes cada palavra existe num texto e de seguida calcularemos qual a que ocorre mais vezes.
Para isso vamos definir um tipo `Dicionario` para armazenar, para cada palavra que já apareceu no texto, o número de vezes que tal aconteceu.

```
typedef struct entrada{
char *palavra;
int ocorr;
struct entrada *prox;
} *Palavras;
```

```
#include <stdio.h>

int main (int argc, char *argv[]){
  FILE *input;
  int r=0, count=0;
  char word[100];
  if (argc==1) input=stdin;
  else input=fopen(argv[1],"r");
  if (input==NULL) r=1;
  else {
    while ((fscanf(input,"%s",word)==1))
      count ++;
    fclose (input);
    printf ("%d palavras\n", count);
  }
  return r;
}
```

Para cada uma das alternativas que vamos definir para o tipo `Dicionario` é necessário definir as funções

- `void initDic (Dicionario *d)` que inicializa o dicionário a vazio.
- `int acrescenta (Dicionario *d, char *pal)` que acrescenta uma ocorrência da palavra `pal` ao dicionário `*d`. A função deverá retornar o número de vezes que a palavra `pal` passou a ter após a inserção.
- `char *maisFreq (Dicionario d, int *c)` que calcula a palavra mais frequente de um dicionario (retornando ainda em `c` o número de ocorrências dessa palavra).

1. Usando as funções referidas, altere o programa de forma a escrever no ecran a palavra mais frequente (e o número de vezes que ocorre).
2. Defina essas funções para as seguintes formas alternativas de implementar o tipo `Dicionario`.
   (a) A forma mais simples de definir o tipo `Dicionario` é como uma lista ligada: `typedef Palavras Dicionario;`.
   A ordem pela qual as palavras são armazenadas pode ser:
   - Por ordem alfabética da palavra
   - Inserindo as novas palavras no fim (ordem cronológica)
   - Movendo para o início da lista cada nova actualização.

(b) Quando o número de palavras é muito grande, é costume partir essa lista em várias listas mais pequenas.
Em vez de usarmos uma lista para representar um dicionário usaremos um array de HSIZE listas. É necessário definir uma função que, dada uma palavra determine em qual dessas listas essa palavra está ou será armazenada. Como exemplo apresenta-se uma definição muito simples (da autoria de Dan Bernstein).

```
#define HSIZE 1000
typedef Palavras Dicionario[HSIZE];

unsigned int hash (char *pal, int s){
   // http://www.cse.yorku.ca/~oz/hash.html
   int r;
   for (r=5381; *pal != '\0'; pal++)
      r = r*33 + *pal;
   return (r%s);
}
```