Programação Imperativa 2022/2023 (CC1003), DCC/FCUP

Folha 11

11.1 Considere a seguinte descrição dum programa para implementar um dicionário usando programação modular. Primeiro, o programa lê palavras e as definições correspondentes da entrada padrão e as grava num dicionário. Um utilizador pode depois inserir uma palavra na entrada padrão e o programa procura tal palavra no dicionário. Caso esta palavra seja contida no dicionário, o programa retorna a definição da palavra na saída padrão. O programa continua a funcionar até que o utilizador decida sair.
O programa é constituído pelos seguintes ficheiros: `main.c`, `dict.c`, `dict.h`. O conteúdo destes ficheiros é descrito brevemente a seguir:

`main.c`

```
#include <stdio.h>
#include <stdlib.h>
#include "dict.h"

int main(){
...
}
```

`dict.c`

```
#include "dict.h"

/* estrutura de dados para o dicionário*/

char * dicionario[1000];

void carrega_dicionario(){
 ...
}

char * procura(char []){
 ...
}
```

dict.h

```
/* estrutura de dados para o dicionário*/
char * dicionario[1000];

/* protótipos das funções */
void carrega_dicionario();

char * procura(char []);
```

Responda às seguintes perguntas baseadas na estrutura de programa descrita.

1. Na implementação do programa quer aceder à variável `dicionario` a partir dos ficheiros `main.c` e `dict.c`. Doutro lado, sendo que o ficheiro *header* `dict.h` é incluído em ambos os ficheiros *source*, a variável é declarada em ambos os ficheiros, criando assim ambiguidade. Como pode resolver tal ambiguidade?

2. Assuma que já completou a escrita do programa. Qual é o comando necessário para compilar o programa e gerar o ficheiro executável `dicionario.o`?

11.2 Escreva um programa capaz de ler da entrada padrão um dado número de palavras (a introdução de palavras termina com uma palavra vazia). Deve utilizar o vetor de apontadores `palavras` para armazenar as palavras.
Para além disso, desenvolva também as seguintes funções:

1. Escreva uma função `void maior(char **palavras)` que imprime na saída padrão a palavra com mais carateres existente no ficheiro.

2. Escreva uma função `void rem_palavra(char **palavras, char *palavra)` que procura a palavra `palavra` `palavras` e a remove do vetor de apontadores.

3. Escreva uma função `void sem_repetidos(char **palavras)` que escreve na saída padrão todas as palavras que foram introduzidas pela entrada padrão sem repetições.

4. Escreva uma função `void palindromos(char **palavras)` que escreve na saída padrão todas as palavras em `palavra` que sejam palíndromos (ou seja, cuja leitura é igual começando do início para o fim ou em sentido inverso).

5. Escreva uma função `void cifrar(**palavras, int n)` que escreve na saída padrão uma versão cifrada de todas as palavras existentes em `palavras`, usando a cifra ROT13 descrita na aula 7. A cifra deve ser aplicada a letras minúsculas e maiúsculas. Eventuais outros tipos de carateres devem ficar inalterados.

6. Escreva uma função `void ordenar(**palavras)` que escreve na saída padrão as palavra em `palavras` por ordem alfabética.

11.3 Escreva um programa lê um dado número de linhas da entrada padrão e escreve tais linhas num ficheiro de texto. Exemplo:

```
Número dei linhas: 4
linha 1
linha 2
linha 3
linha 4
```

11.4 Escreva um programa que lê o ficheiro de texto gerado no exercício anterior e que guarda as linhas do ficheiro num vetor.

11.3 Escreva uma função `int numero_linhas(char *nome_ficheiro)` que retorna o número de linhas dum ficheiro.

11.5 Escreva um programa para contar palavras dum ficheiro. Considere que as palavras são sequências de "carateres normais" (e.g., letras, algarismo ou sinais de pontuação) separados por "carateres brancos" (espaços, tabulação ou mudança de linha, ou seja, `' '`, `'\t'`, ou `'\n'`).

11.6 Escreva um programa que lê o conteúdo de dois ficheiros de texto, "ficheiro1.txt" e "ficheiro2.txt", e escreve um novo ficheiro (de nome "uniao.txt") obtido unindo os conteúdos dos dois ficheiros originais.