Nomenklatura

Preliminares

Temos hoje três problemas clássicos com nomes de pessoas: coligir nomes próprios, contar as ocorrências de nomes próprios, e listar nomes por ordem de apelido.

Trata-se de exercícios sobre cadeias de caracteres, sobre arrays de cadeias de caracteres e sobre arrays de estruturas. Muitas das funções estudadas nas duas primeiras aulas teóricas têm utilidade aqui.

Há de reparar que em todas as funções estudadas os arrays de cadeias de caracteres são `const`, ou, melhor, são declaradas, por exemplo, `const char **a`, nos argumentos de funções, e `const char *a[]`, nas variáveis. Isso é uma regra nossa, não é uma regra do C. Na nossa cadeira fazemos uma "promessa" de só usar arrays de cadeias de caracteres desta forma. Habitue-se.

Num array de cadeias `const char **`, as cadeias são *imutáveis*. Quer dizer, os caracteres que as compõem não podem mudar. Isso não significa que os arrays não possam mudar. Podem, com certeza! Podemos acrescentar cadeias ao array e podemos substituir cadeias no array por outras. Mas cada cadeia individualmente não pode ser modificada.

Isto parece ser uma grande restrição, mas até não é. Como dizia o outro, "primeiro estranha-se, mas depois entranha-se".

Nomes próprios

SAMUEL VIEGAS GONÇALVES
TOMÁS DOS SANTOS ARVANAS
VASCO FILIPE FARIA DA PAZ
GABRIEL ALEXANDRE CAVACO MANGAS
MIGUEL BERNARDO GUERREIRO
QUÉVIN ANDRE PROTSENKO
RODRIGO MIGUEL LEAL CARDOSO
CARLOS DANIEL FERREIRA
DANIEL FERROS FERNANDES
DIOGO AFONSO NOBRE ZACARIAS
FILIPE MIGUEL DOMINGOS DUARTE
GUILHERME JACQUES PACHECO CRUZ DA GLÓRIA

Neste primeiro exercício, queremos uma função que, dado um array de cadeias de caracteres onde cada cadeia representa o nome completo de uma pessoa, ordenado por ordem alfabética, crie outro array com os primeiros nomes próprios presentes nos nomes do primeiro array.

A função deve ter o seguinte protótipo:

```
int unique_first_names(const char **a, int n, const char **b)
```

Os nomes completos estão usam apenas letras maiúsculas sem sinais diacríticos.

No array de saída, os primeiros nomes próprios estarão também por ordem alfabética e não deve haver nomes repetidos.

O primeiro nome próprio é a primeira palavra do nome. Não há espaços antes da primeira palavra do nome nem depois da última.

A função de teste lerá um ficheiro com os nomes completos, linha a linha, da consola para o array, fará as contas (por meio da função `unique_first_names`, e escreverá também na consola o array de nomes próprios calculado, um nome em cada linha.

Submeta no problema A.

Indicações

Não deixe de reler os textos das aulas e de estudar os programas fornecidos.

Ranking dos nomes próprios

Queremos agora contar as ocorrências dos nomes próprios e listar os nomes próprios, por ordem decrescente do respetivo número de ocorrências, desempatando por ordem alfabética. Estamos interessados apenas no *primeiro* nome próprio, que é a primeira palavra do nome.

A função de teste lerá um ficheiro com os nomes completos, linha a linha, da consola para o array, fará as contas (mediante uma função apropriada), e escreverá também na consola o array com a contagem. Em cada linha do output virá um nome seguido do respetivo número de ocorrências, pela ordem indicada acima.

Submeta no problema B.

Indicações

Como precisamos de saber quantas vezes cada nome ocorre, convém associar essas duas informações — nome e número — numa estrutura. Sugiro a seguinte:

```
typedef struct {
  const char *string;
```

```
  int number;
} StringInt;
```

O tipo `StringInt` representa um *par*, em que o primeiro elemento é uma cadeia de caracteres e o segundo é um número `int`.

Programe o construtor, que, de acordo com as nossas regras de estilo, deve ter o seguinte protótipo:

```
StringInt string_int(const char* string, int number);
```

O resultado da contagem deve ser um array de `StringInt`. Esse array será escrito na consola, no seio da função de teste. Programe uma função para isso, com o seguinte protótipo:

```
void string_ints_print(const StringInt *a, int n);
```

A função escreverá um nome e um número em cada linha, separados por um espaço.

A função de contagem deve calcular um array de `StringInt` a partir de um array de cadeias de caracteres. O protótipo será o seguinte:

```
int strings_tally(const char **a, int n, StringInt *b);
```

O resultado deve estar ordenado, da maneira que indicámos. Logo, precisamos de uma função para ordenar um array de `StringInt`. Neste exercício use o *insertionsort*. O protótipo da função será o seguinte:

```
void string_ints_isort(StringInt *a, int n);
```

A ordenação deve ser decrescente pelo número, desempatando pela cadeia.

A função de teste lerá um ficheiro de nomes, a partir da consola, fará as contas (por meio da função `strings_tally`), e escreverá o array na consola.

Listagem de nomes por ordem de apelido

Antigamente havia uma coisa chamada *lista telefónica*, que fazia parte do equipamento de todos os lares. A lista telefónica era um volumoso livro que continha os números de telefone de todos os assinantes de uma certa zona e era distribuída gratuitamente pela companhia dos telefones. Nessa lista, os nomes dos vinham ordenados pelo último apelido. Cada ano havia uma lista nova. As listas velhas eram depois usadas pelos vendedores de castanhas assadas, para embrulhar as castanhas.

Agora ainda há listas telefónicas, na Web, mas não dão para embrulhar castanhas.

O terceiro exercício de hoje é inspirado nas listas telefónicas. Mais precisamente, queremos programar uma função que, dado um array com nomes, o ordene por ordem alfabética de apelido, desempatando por ordem alfabética de nome completo.

A função de teste lerá o array a partir da consola, fará as contas por meio de uma função apropriada e escreverá a o array ordenado da forma indicada, também na consola.

Submeta no problema C.

Indicações

Para efeitos do exercício, o apelido é o último nome.

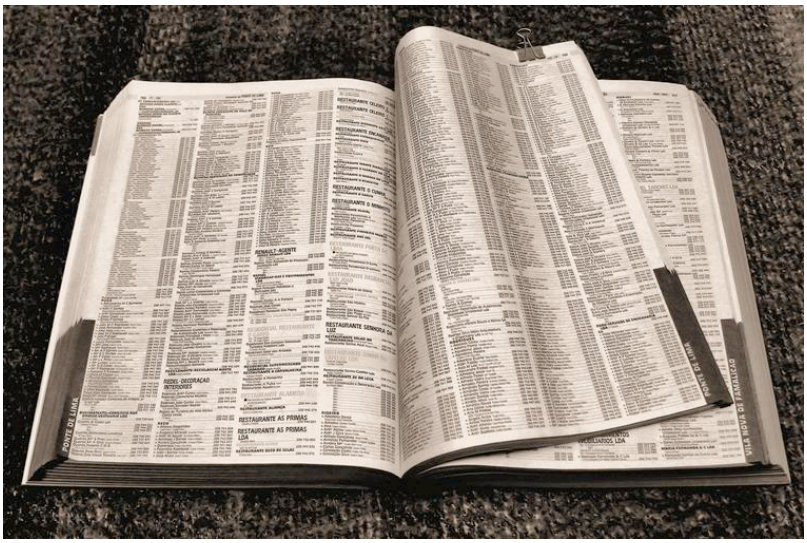
Comecemos então por calcular o apelido. À força bruta, basta calcular o array com todas as palavras do nome (o que já sabemos fazer) e escolher a última.

No entanto, uma vez que a cadeia com o nome não tem caracteres espaço espúrios, à direita da última palavra, basta retornar o endereço do primeiro caractere dessa última palavra. Em esquema, será assim, tendo a variável `x` o número de caracteres à esquerda da última palavra:

```
const char *last_word(const char *s)
{
  // ...
  return s+x;
}
```

Dispondo da função `last_word`, agora, para ordenar o array dos nomes, basta programar a função de comparação que compara primeiro pelo último nome e depois pelo nome completo.

Listagem de nomes por ordem de apelido, versão alternativa



Ordenar o array dos nomes, usando a tal função de comparação que compara primeiro pelo último nome e depois pelo nome completo é interessante, mas

pouco eficiente. Repare, de cada vez que se faz uma comparação é preciso calcular a última palavra de cada uma das duas cadeias em causa. Ora, cada cadeia entrará em várias comparações ao longo da ordenação. Portanto, o cálculo da última palavra faz-se várias vezes, para cada cadeia, dando sempre o mesmo resultado. Isso constitui um desperdício de tempo de cálculo, ainda que, nos nossos exemplos, com poucos nomes, os cálculos sejam instantâneos.

Em todo o caso, programemos uma versão alternativa, que consiste em calcular à partida o último apelido de cada um dos nomes completos dados, e depois usar nas comparações esses apelidos já pré-calculados.

Para associar o último apelido ao nome completo usamos uma estrutura para pares de cadeias:

```
typedef struct {
  const char *_1;
  const char *_2;
} StringString;
```

Esta estrutura tem dois membros, de nomes `_1` e `_2`. O primeiro membro representará o nome completo e o segundo membro representará o último apelido. No entanto, a estrutura servirá, em geral, para representar pares de cadeias de caracteres e voltaremos a usá-la noutros problemas.

Programe as funções do costume para este `typedef`: construtor, escrita, comparação e escrita arrays de `StringString`. Programe também o *insertionsort* para arrays destes.

A função de teste é análoga à do problema anterior: lê o array a partir da consola, faz as contas, mediante uma função apropriada, e escreve o array ordenado da forma indicada.

Submeta no problema D.