Exercícios
Arquivos Organizados em Campos e Registros

É comum organizar arquivos de dados utilizando campos e registros. Embora se utilize com frequência uma estrutura com campos e registros de tamanho fixo, isso não é obrigatório. Conforme discutido em aula, existem diversas formas de organizar arquivos de campos e registros.

A aula de hoje consiste em fazer alguns programas que sejam capazes de ler arquivos organizados em campos e registros. Os arquivos necessários para a aula se encontram no Moodle. Existem três arquivos:

- **fixo.dad** é um arquivo com campos e registros de tamanho fixo;
- **campo_var_reg_fixo.dad** é um arquivo com campos de tamanho variável e registros de tamanho fixo;
- **campo_var_reg_var.dad** é um arquivo com campos e registros de tamanho variável.

Todos os arquivos possuem a mesma informação: dez registros com os seguintes campos:

1. Um nome com no máximo 15 caracteres incluindo o caractere terminador de string;
2. Um sobrenome com no máximo 15 caracteres incluindo o caractere terminador de string;
3. Um nome de rua com no máximo 26 caracteres incluindo o caractere terminador de string;
4. Um número (da casa) que é um inteiro com 4 bytes (**int**).

Todos os arquivos foram gravados como arquivo binário. Os dados dos arquivos referentes as Tarefas 2 e 3 foram gravados com uso de bufferização.

O objetivo é fazer um programa para cada organização de arquivo descrita acima. Esse programa deve inicialmente imprimir os registros contidos no arquivo e logo em seguida pedir ao usuário um número entre 1 e 10 e imprimir o registro correspondente.

Tarefa 1: Arquivo fixo.dad

O arquivo **fixo.dad** possui 10 registros gravados em um formato de campos e registros de tamanho fixo. Essa organização é uma das mais simples. Para a leitura de todos os dados tem-se duas opções principais:

1. Pode-se fazer uma chamada a **fread** para cada registro;
2. Pode-se definir um vetor de registros e ler todos os 10 registros com uma única chamada a **fread**. Para isso, ajuste o terceiro parâmetro do **fread**.

Em ambas as opções, utilize **sizeof** para calcular o tamanho em bytes ocupado por um registro. Por exemplo, **sizeof(reg)** retorna o número de bytes ocupados pelo registro **reg**.

Para a leitura de um registro pedido pelo usuário deve-se utilizar a função **fseek**. Para tanto, deve-se calcular o número de bytes existentes entre o início do arquivo e o registro de interesse. Novamente, **sizeof** é útil para descobrir quantos bytes ocupa cada registro.

Tarefa 2: Arquivo campo_var_reg_fixo.dad

O arquivo **campo_var_reg_fixo.dad** possui 10 registros gravados em um formato de campos de tamanho variável e registros de tamanho fixo. Foi utilizado o caractere '|' como separador de campos. Cada registro possui um tamanho fixo igual à soma dos tamanhos máximos de cada um dos campos (da mesma maneira que na Tarefa 1). Um exemplo de registro nesta organização é:

```
João|Moreira|XV de Novembro|110|<== espaço não utilizado ==>Eduardo|...
```

Para a leitura de todos os registros, pode-se ler blocos de dados cujo tamanho é igual a um registro. Entretanto, é necessário criar uma sub-rotina capaz de identificar cada um dos campos separados pelo caractere '|'.

Para a leitura de um registro especificado pelo usuário, pode-se calcular o número de bytes existentes entre o início do arquivo e o registro de interesse, da mesma maneira que na Tarefa 1. Mas a sub-rotina para identificar os campos também é necessária.

Tarefa 3: Arquivo campo_var_reg_var.dad

O arquivo **campo_var_reg_var.dad** possui 10 registros gravados em um formato de campos e registros de tamanho variável. Foi utilizado o caractere '|' como separador de campos. Cada registro possui um tamanho variável igual à soma dos tamanhos de cada um dos campos (diferente do utilizado nas Tarefas 1 e 2). Para auxiliar na leitura dos registros, cada registro inicia com um inteiro de 1 byte (**char**) que indica o tamanho do registro. Um exemplo de registro nesta organização é:

```
28João|Moreira|XV de Novembro|110|33Eduardo|...
```

No qual 28 e 33 são os tamanhos do primeiro e segundo registros, respectivamente. Não esqueça que esses dois valores estão armazenados em um único byte.

Para a leitura de todos os registros, pode-se ler um único byte e logo em seguida um bloco de dados cujo tamanho é igual ao número indicado no byte. Logo em seguida deve-se utilizar uma sub-rotina idêntica à da Tarefa 2 para identificar os campos.

Para a leitura de um registro especificado pelo usuário, é necessário ler os bytes indicativos de tamanho e logo em seguida usar a função **fseek** para saltar para o próximo registro. Isso deve ser repetido até localizar o registro de interesse.

Obs.: nesse último caso faça uma versão alternativa usando [fgetc]+[looping com EOF], abrindo o arquivo como texto (r+t), e veja o que acontece...