Fundamentos de Programação

Filipe Francisco

24 de maio de 2018

Sobre o trabalho

- Tipo de jogo (PxP, PxC):
  - não é necessário implementar pessoa x computador
  - basta implementar um jogo entre duas pessoas!
- Leitura da posição para efetuar a jogada:
  - por exemplo, pode-se ler um valor de 0 a 8, de 1 a 9, ou linha e coluna
  - considere que sempre estamos lendo números
  - portanto, você não precisa tratar casos em que inserimos uma letra em vez de um número
- Qual jogador jogará com cada símbolo (X ou O)?
  - você não precisa perguntar qual símbolo cada pessoa quer usar
  - você pode simplesmente definir um símbolo fixo
  - fica a seu critério!
- locale.h e setlocale(LC_ALL,"Portuguese"):
  - se você utilizar a locale.h para português, os caracteres são alterados e os caracteres especiais do grid não existirão mais!
  - não precisa utilizar, simplesmente driblem os acentos e 'ç'

Registros e arquivos

- Até aqui, trabalhamos com alguns tipos de dados
- Tipos básicos de dados:
  - inteiro
  - caractere
  - real
- E a partir destes, trabalhamos com algumas estruturas:
  - estruturas unidimensionais: vetores (e strings)
  - estruturas multidimensionais: matrizes
- Porém, note que estas estruturas só trabalham com um único tipo básico
  - por exemplo, não podemos criar um vetor em que parte dos seus elementos sejam inteiros e parte sejam caracteres
- Pergunta: como unir dois tipos básicos diferentes em uma variável?

Registros e arquivos

- Resposta: Structs!
- Registros (ou estruturas)
- Um registro é uma estrutura composta por um conjunto de variáveis que possivelmente têm tipos diferentes
  - cada uma destas variáveis é chamada de campo do registro
- Por exemplo, uma pessoa pode ter nome, CPF, identidade e data de nascimento
- Podemos criar um registro para esta pessoa, contendo cada um destes dados!
  - nome: string com até 100 caracteres
  - CPF: inteiro (sem hífen)
  - identidade: inteiro
  - data de nascimento: string com 10 caracteres (dd/mm/aaaa)

Registros e arquivos

- Como declarar um registro deste tipo em C?
- Maneira 1: criar o registro e criar uma instância dele

```
struct dados{
    int cpf, ident;
    char nome[101], nasc[11];
};
struct dados p1;
```

  - a última linha cria uma instância deste registro

- Maneira 2: definir um novo tipo a partir de um registro

```
typedef struct dados{
    int cpf, ident;
    char nome[101], nasc[11];
}pessoa;
pessoa p1;
```

  - criamos um novo tipo "pessoa" a partir do registro apresentado
  - a última linha cria uma instância do tipo "pessoa"

Registros e arquivos

- Comentários:
  - podemos criar um registro com vários campos, de vários tipos
    - podemos ter campos de tipos básicos
    - podemos ter campos de ponteiros
    - podemos até ter campos de outras estruturas!
  - utilizando **typedef**, "economizamos" um pouco na criação de variáveis
  - definindo um registro, podemos criar estruturas de registros
    - podemos criar registros com campos de registros
    - podemos criar registros com campos de ponteiros para registros
    - podemos criar vetores (ou matrizes) de registros
- Uma das principais aplicações de registro é facilitar o trabalho com muitas informações
- Exemplo:
  - suponha que você tem uma lista de pacientes, cada qual cadastrada com seu nome, cpf, identidade, data de nascimento e prioridade
  - seu objetivo é ordenar esta lista de pacientes de acordo com a prioridade de atendimento de cada pessoa

Registros e arquivos

- Solução ruim: criar um vetor para cada uma das informações
  - vetores de inteiros para armazenar cpf, identidade e prioridade
  - matrizes de caracteres (ou vetores de strings) para armazenar nome e data de nascimento
  - ordenamos o vetor de prioridades e a cada troca, trocamos também os elementos de todos os outros vetores
- Solução boa: utilizar um vetor de registros
  - desta forma, só precisamos ordenar o vetor de registros em função do campo prioridade
  - assim, cada troca feita na ordenação será apenas uma troca de posições entre registros no vetor

Registros e arquivos

- Um registro é uma coleção de variáveis utilizando um único nome
- Um registro agrupa várias informações relacionadas
  - um livro tem título, ISBN, autores, quantidade de páginas, ...
  - um carro tem marca, modelo, chassi, placa, potência, ...
- E uma das melhores aplicações de registro é trabalhar com arquivos!
- Uma pergunta que você sabe responder: o que é um arquivo?
- E uma pergunta que talvez não saiba: por que usar um arquivo?

Registros e arquivos

- O que é um arquivo?
  - um arquivo é um lugar no disco onde alguma informação é armazenada
- Por que usar um arquivo?
  - tudo o que fizemos até agora utiliza a memória
  - problema 1: a memória tem um armazenamento volátil!
    - se desligarmos a fonte de energia, tudo o que está armazenado na memória se perde
  - problema 2: para colocarmos coisas na memória, tínhamos de digitar cada uma das entradas
    - imagine o quão chato deve ser digitar uma matriz 10×10
  - problema 3: se desejarmos passar dados de um computador para outro, devemos salvar o que está na memória em forma de um armazenamento persistente

Registros e arquivos

- E como o arquivo pode ajudar nestes pontos?
  - problema 1: a memória tem um armazenamento volátil!
    - o arquivo é gravado diretamente no disco (HD ou SSD)
    - como estes são armazenamentos persistentes, não há perda de dados se desligarmos o computador!
  - problema 2: para colocarmos coisas na memória, tínhamos de digitar cada uma das entradas
    - podemos montar um loop para ler os dados diretamente de um arquivo, linha a linha
    - você pode digitar uma matriz 10×10 em um arquivo de texto e montar um algoritmo que lê esta matriz
  - problema 3: se desejarmos passar dados de um computador para outro, devemos salvar o que está na memória em forma de um armazenamento persistente
    - ou seja, passamos para um arquivo!
- No início da disciplina, comentamos que podíamos escrever diretamente no disco, e agora vamos ver como fazer isto!

Registros e arquivos

- Podemos trabalhar com dois tipos de arquivos
- Arquivo de texto:
  - arquivos .txt
  - podem ser criados utilizando basicamente qualquer programa de edição de texto (como notepad ou gedit)
  - o conteúdo do arquivo é armazenado como texto propriamente dito, ou seja, você pode alterar diretamente este conteúdo
  - ocupa um maior espaço no disco
- Arquivo binário:
  - arquivos .bin
  - o conteúdo do arquivo é armazenado como vários 0's e 1's
  - por este motivo, a sua leitura é mais complicada
  - porém, ele oferece mais segurança do que arquivos de texto
  - ocupa um menor espaço no disco
- Para facilitar, vamos trabalhar com arquivos de texto!

Registros e arquivos

- Relembrando, quando abrimos um arquivo, este é movido para a memória
- Portanto, para lermos este arquivo, precisamos saber onde ele está na memória
- Ou seja, utilizaremos um ponteiro para o arquivo!
- Para isto, utilizaremos um tipo especial de variável, existente na biblioteca "stdio.h": FILE
- Deste modo, podemos dizer que para declararmos um arquivo, basta declararmos um ponteiro para um tipo FILE, da seguinte forma:

```
FILE *arq;
```

- Este ponteiro é essencial para a comunicação entre programa e arquivo, e o utilizaremos para trabalhar com o arquivo de diversas maneiras!

Registros e arquivos

Como abrir um arquivo?

fopen(arquivo,modo);

- exemplo: arq=fopen("testando.txt","a");
- recebe o endereço (string) do arquivo a ser aberto e o modo de abertura, e retorna um ponteiro para o arquivo
- este modo indica o que queremos fazer com o arquivo
- modos principais para se abrir um arquivo:
  - 'r': abrir para leitura (read)
    - obs: o modo 'r' não cria arquivos, só lê se já existir
  - 'w': abrir para escrita (write)
  - 'a': abrir para escrita no final (append)
    - obs1: se o arquivo não existe, os modos 'w' e 'a' o criam em branco
    - obs2: se o arquivo já está criado, o modo 'w' o apaga e cria um novo
    - obs3: se o arquivo já está criado, o modo 'a' mantém o que está no arquivo, e adiciona linhas no final

Registros e arquivos

Como abrir um arquivo? (continuação)

fopen(arquivo,modo);

- obs: em alguns casos, não é possível abrir o arquivo
  - exemplo: arquivo não existe/endereço do arquivo incorreto
  - exemplo: você não tem permissão para alterar o arquivo
  - exemplo: tentar abrir um arquivo em um modo que não permite (como tentar alterar um arquivo somente leitura)
- nestes casos, o método fopen retorna um ponteiro null (NULL)

Como fechar um arquivo?

fclose(arquivo);

- exemplo: fclose(arq);
- recebe o ponteiro para o arquivo a ser fechado

Registros e arquivos

Como escrever informações em um arquivo?

fprintf(arquivo,formato,...);

- exemplo: fprintf(arq,"%d %c %f\n",num,carac,real);
- recebe um ponteiro para o arquivo, e escreve uma expressão, podendo incluir o valor de uma variável na escrita
  - podemos escrever apenas uma mensagem (string), ou optar por inserir valores de variáveis
  - o formato utilizado no fprintf é semelhante ao do printf!

Como ler informações de um arquivo?

fscanf(arquivo,formato,...);

- exemplo: fscanf(arq,"%d %c %f",&num,&carac,&real);
- recebe um ponteiro para o arquivo, lê um certo tipo de informação no arquivo, e armazena o valor lido em uma variável
  - o formato utilizado no fscanf é semelhante ao do scanf!

Registros e arquivos

Problema: ler string

- já vimos que ler uma string utilizando "%s" é ruim

Como ler uma string em um arquivo?

fgets(string,tamanho,arquivo);

- exemplo: fgets(str,**sizeof**(str),arq);
- recebe o vetor de caracteres (string) onde armazenaremos o que for lido, a quantidade de caracteres máximo a ser lido, e um ponteiro para o arquivo
  - a quantidade inclui o caractere nulo!
- lê uma linha do arquivo até encontrar um '\n' ou o máximo de caracteres for lido

Como ler um caractere do arquivo por vez?

carac=fgetc(arquivo);

- recebe o ponteiro para o arquivo, e retorna o caractere lido

Registros e arquivos

Supondo que estamos lendo um arquivo linha a linha (utilizando fgets), devemos parar de ler no final do arquivo
Problema: como saber que chegamos ao final do arquivo?

- uma opção seria utilizar a função "feof(arq)"
  - recebe um ponteiro para o arquivo, e retorna se já encontramos o final do arquivo
- problema: para saber que estamos no final do arquivo, precisamos tentar ler algo mais
  - no momento em que não conseguirmos ler mais nada, sabemos que chegamos no final do arquivo
- portanto, um loop verificando "feof(arq)" não é a melhor opção
  - fazendo isto, tentaríamos ler uma linha a mais, o que não seria possível, e imprimiríamos a última linha duas vezes!

Registros e arquivos

Problema: como saber que chegamos ao final do arquivo? (continuação)

- para resolver este problema, podemos utilizar o retorno da função fgets!
  - esta função retorna um ponteiro para a string, se algo foi lido, e retorna NULL se nenhum caractere foi lido (ou seja, se alcançamos o fim do arquivo)
- portanto, podemos fazer da seguinte forma:

```
while(fgets(str,sizeof(str),arq)!=NULL) {
    printf("%s",str);
}
```

  - no momento que o fgets não conseguir ler, sabemos que estamos no fim do arquivo, portanto saímos do loop