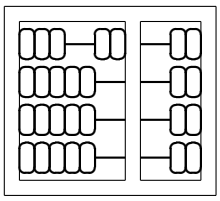

MC102 – ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES
Instituto de Computação — UNICAMP
1º Semestre de 2005

Aula 23 - Arquivos - parte 2

1 Objetivos

Apresentar e exemplificar as funções de manipulação de arquivos existentes na biblioteca padrão `stdio.h`.

2 Motivação

Fazer programas que manipulem estruturas de dados em arquivos, usando registros, por exemplo..

3 Arquivos binários

A representação binária de tipos numéricos simples como int e float costumam ser mais compactas do que sua representação como texto. Por exemplo, o número 12345678 é representado com 8 algarismos (8 bytes) como texto, mas com somente 4 bytes de forma binária. Por isso para gravar grandes quantidades de números em arquivos é preferível usar o formato binário.

Os comandos `fread` e `fwrite` são usados para leitura e escrita, de sequências de bytes. A função de leitura recebe quatro parâmetros: o endereço da variável que será recuperada do arquivo, o tamanho da representação binária desta variável, o número de elementos que serão lidos (uso para vetores) e finalmente a variável que controla o arquivo aberto para leitura. A função de escrita recebe os mesmos quatro parâmetros, mas faz a operação inversa: grava a representação binária da variável no arquivo. A função `sizeof` pode ser usada para determinar o tamanho binário de uma variável ou de um tipo em C.

Para abertura de um arquivo em formato binário, a letra "b" deve ser anexada a um dos modos de abertura de arquivo. A função abaixo ilusta a gravação de um vetor para um arquivo binário.

```
void grava_vetor (char nome_arq[], int vet[], int tam)
{
  FILE *arq;

  /* Abre o arquivo com o nome pedido no modo de escrita binária */
  arq = fopen (nome_arq, "wb");

  /* Grava o numero de elementos do vetor, que é uma variável inteira */
  fwrite (&tam, sizeof(int), 1, arq);

  /* Grava o vetor vet, que tem tam elementos int, no arquivo arq */
  fwrite (vet, sizeof(int), tam, arq);

  /* Fecha o arquivo */
  fclose(arq);
}
```

Uma vantagem do formato binário é que o tamanho ocupado por cada elemento é conhecido, isso significa que é possível calcular a posição de cada um no arquivo. Uma vez sabida a posição de um elemento que se deseja acessar, o arquivo não precisa ser lido por completo. A função `fseek` pode ser usada para acessar

diretamente qualquer ponto do arquivo. Seus argumentos são: a variável que controla o arquivo aberto, a posição relativa do arquivo que se deseja acessar e, por fim, uma marcação que indica se a posição que se deseja é relativa ao início do arquivo (`SEEK_SET`), à posição atual (`SEEK_CUR`) ou ao fim do arquivo (`SEEK_END`).

A função a seguir lê do arquivo criado pela `grava_vetor` o valor correspondente ao índice solicitado. Depois este valor é incrementado e gravado na mesma posição do arquivo.

```
void incrementa_elemento (char nome_arq[], int indice)
{
  int tam, elem;
  long pos;
  FILE *arq;

  /* Abre o arquivo com o nome pedido no modo de leitura/escrita binária */
  arq = fopen (nome_arq, "r+b");

  /* Le o numero de elementos do vetor */
  fread (&tam, sizeof(int), 1, arq);

  /* Se existir o elemento com indice solicitado */
  if (indice < tam) {

    /* Calcula a posição do elemento com indice solicitado, lembrando
       que o primeiro número é o tamanho do vetor */
    pos = sizeof(int) + indice * sizeof(int);

    /* Posiciona o arquivo para ler desta posição */
    fseek (arq, pos, SEEK_SET);

    /* Le um elemento de tipo int */
    fread (&elem, sizeof(int), 1, arq);

    /* Incrementa o elemento */
    elem++;

    /* A posição avançou para o próximo elemento, volta ela para a
       posição anterior */
    fseek (arq, pos, SEEK_SET);

    /* Grava um elemento de tipo int */
    fwrite (&elem, sizeof(int), 1, arq);

  }
  else printf ("O vetor em arquivo não possui o indice solicitado.\n");

  /* Fecha o arquivo */
  fclose(arq);
}
```

4 Lendo arquivos de tamanho desconhecido

Existem situações em que não sabemos o número de elementos gravados no arquivo. Por exemplo, ao gravarmos somente os dados do vetor sem gravar no início o número de elementos como no exemplo anterior. Nestes casos, podemos usar uma função auxiliar também presente na biblioteca `stdio.h`. A função `feof`, que recebe como único argumento a variável que controla o arquivo, testa se a última operação de leitura alcançou o fim do arquivo, ou seja, se não existem mais dados a serem lidos.

O programa abaixo coleta todos os números presentes em um arquivo e os coloca em um vetor, para depois imprimí-los na tela. No máximo uma quantidade MAX será lida, o resto é descartado. É importante notar que a convenção do arquivo mudou, o primeiro número agora também faz parte do vetor.

```
#include <stdio.h>

#define MAX 1000

int main()
{
  int i, n, vet[MAX];
  char nome_arquivo[] = "vetor.txt";
  FILE *arq;

  /* abrir um arquivo binário para leitura */
  arq = fopen (nome_arquivo, "r");

  /* ler os dados do vetor do arquivo */
  /* obs: pára no fim do arquivo ou quando tiver lido MAX elementos */
  i = 0;
  while (! feof(arq) && i < MAX) {
    fscanf (arq, "%d", &vet[i]);
    i++;
  }

  /* fechar o arquivo */
  fclose (arq);

  /* gravar em n quantos elementos foram lidos */
  n = i;

  /* mostrar o vetor na tela */
  printf("Foi lido um vetor de tamanho %d com os elementos: ", n);
  for (i = 0; i < n; i++) printf("%d ", vet[i]);
}
```

5 Exercícios

Repita os exercícios anteriores feitos para arquivos texto agora usando arquivos no formato binário.

6 Referências

Estas aulas foram baseadas nas notas de aula do **prof. Alexandre Falcão**

(http://www.dcc.unicamp.br/ afalcao/mc102/notas-aula.pdf)

na apostila do **prof. Flávio Keidi Miyazawa** e no material de aula do **prof. André Shiguemoto**.