Programação de Computadores II

Arquivos

Slides da profa. Paula Rodrigues

Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)

Capítulo 15 – Arquivos

Tópicos Principais

- Motivação
- Abertura e fechamento de arquivos
- Leitura com *fscanf*
- Escrita com *fprintf*

Motivação

- Motivação: quando um programa precisa processar um volume de dados muito grande. Por exemplo:
  - Uma grande quantidade de dados de entrada se faz necessária e um erro de digitação de um dado obrigaria a entrada de todos os dados novamente, o que se torna impraticável.
  - analogamente, quando um programa exibe uma grande quantidade de dados de saída, é impraticável exibir todos os dados na tela.
- A estratégia adequada para se trabalhar com grande volume de dados é fazer uso de *arquivos de dados*.

Motivação

- Se um programa precisa de uma grande quantidade de dados de entrada, é mais adequado que o programa carregue (leia) estes dados de um arquivo.
- De forma análoga, se um programa precisa exibir uma grande quantidade de dados de saída, é mais adequado que o programa salve (escreva) estes dados em um arquivo.
- Para que programas possam ler e salvar dados em um arquivo, é necessário ter acesso a um conjunto de funções que permitam realizar essas operações.
- No nosso curso, usaremos um conjunto de funções presentes na biblioteca **stdio.h.**

Abertura de arquivo

- Para que possamos ler ou escrever em um arquivo é necessário que antes o arquivo esteja **aberto**.
- Cada arquivo é identificado pelo seu **nome**, que normalmente é o nome do arquivo mais a extensão: exemplo.txt, cap8.ppt, saida.txt
- Para abrir um arquivo é necessário definir se o mesmo será utilizado para leitura ou escrita:
  - Para abrirmos um arquivo para leitura é necessário que o arquivo já exista.
  - Quando abrimos um arquivo para escrita criamos um novo arquivo onde dados de saída serão escritos, ou sobrescrevemos um arquivo existente.

Abertura de arquivo

- Para abrir um arquivo, a biblioteca padrão oferece a função **fopen**.
  - Esta função serve tanto para abrir um arquivo para leitura como para escrita.
  - A função recebe dois parâmetros, o nome do arquivo que se deseja abrir e o modo de abertura:
    - Leitura: "r" (read)
    - Escrita: "w" (write)
  - Esta função retorna um **ponteiro** para o tipo **FILE** definido na biblioteca. Um ponteiro é um tipo de variável que serve para armazenar endereços de memória.

Abertura de arquivo

- Declaração de uma variável do tipo ponteiro para **FILE** (no exemplo, o nome dado à variável é **fp**)**:**
  - **`FILE *fp;`**

- Inicialização**:**
  - **Abrindo o arquivo para leitura de dados:**
    - **`fp = fopen(“entrada.txt”, “r”);`**
  - **Abrindo o arquivo para escrita de dados:**
    - **`fp = fopen(“saida.txt”, “w”);`**

Abertura de arquivo

- Se a abertura do arquivo não for bem sucedida, a função retorna o valor definido pela constante simbólica NULL:

```
   ...
fp = fopen(“entrada.txt”, “r”);
if (fp == NULL) {
  printf(“Erro na abertura do arquivo.\n”);
  exit(1); /* aborta programa */
}
...
```

O valor retornado pela função fopen deve ser passado como parâmetro para as demais funções para identificar em qual arquivo se deseja fazer a operação de entrada ou saída.

Fechamento de arquivo

- Após realizar as operações de entrada e saída no arquivo, este deve ser **fechado** com o uso da função **`fclose`**.

```
fp = fopen ("entrada.txt" , "r" ) ;

/* código de manipulação do arquivo */

...

fclose(fp);

...
```

Leitura: fscanf

- A principal função da biblioteca padrão para leitura de arquivos chama-se **fscanf**, e é análoga à função scanf para captura de dados via teclado.
  - Da mesma forma que a função fprintf, a diferença está no primeiro parâmetro que identifica o ponteiro para o arquivo.

```
int a;
float b;
FILE* fp = fopen("entrada.txt", "r");
. . .
fscanf(fp, "%d %f", &a, &b);
. . .
fclose(fp);
```

- A função fscanf tem como valor de retorno o número de parâmetros que foram lidos com sucesso (também vale para a função scanf).

Exemplo de Leitura

- Vamos considerar a existência de um arquivo que armazena notas que os alunos obtiveram em uma disciplina.
- Vamos considerar que a primeira linha contém um número inteiro que informa a quantidade de notas armazenadas a seguir, estando uma nota por linha.
- Arquivo notas.txt:

```
6
7.5
8.4
9.1
4.0
5.7
4.3
```

Exemplo de Leitura

- O nosso programa vai ler as notas do arquivo e escrever na tela a média dos alunos.

```
#include <stdio.h>

int main (void) {
  int i, n;
  float nota, soma = 0.0;
  FILE *fp ;

  /* abertura do arquivo para leitura */
  fp = fopen ("notas.txt", "r");
  /* teste para verificar se houve algum erro */
  if (fp == NULL) {
    printf("Erro na abertura do arquivo.\n");
    return 1; /* aborta programa ( retorna da função main) */
  }
```

Exemplo de Leitura

(continuação)

```
  /* leitura da quantidade de notas no arquivo */
  fscanf( fp , "%d", &n);
  /* laço para leitura de cada nota */
  for (i=0; i<n ; i++) {
    fscanf(fp, "%f ", &nota);
    soma = soma + nota ;
  }

  /* fechamento do arquivo */
  fclose(fp);

  /* cálculo da média e impressão na tela */
  printf("Media = %.2f \n", soma / n);
  return 0;
}
```

Exemplo de Leitura

- Quando escrevemos programas para processar dados de entrada de um arquivo, procuramos manter o formato dos dados presentes no arquivo o mais flexível possível.
- O formato do arquivo de notas do exemplo anterior não é muito flexível pois se formos adicionar uma nota no arquivo teremos também que atualizar a primeira linha do arquivo, que representa a quantidade de notas.
- Para tornar o formato mais flexível, podemos pensar em eliminar a informação da primeira linha, isto é, podemos listar apenas as notas, e o programa deve ser capaz de exibir a média de todas as notas.

```
7.5
8.4
9.1
4.0
5.7
4.3
```

Neste caso utilizaremos o comando **while** para processar as notas do arquivo enquanto o final do arquivo não é alcançado.
Para cada nota lida incrementamos o valor do contador.

```
#include <stdio.h>

int main (void) {
  int n;
  float nota, soma = 0.0;
  FILE *fp ;

  /* abertura do arquivo */
  fp = fopen ("notas.txt", "r");

  n = 0;

  /* laço para leitura de cada nota */
  while (fscanf(fp, "%f ", &nota) == 1) {
    soma = soma + nota;
    n++;
  }

  /* fechamento do arquivo */
  fclose(fp);

  /* cálculo da média e impressão na tela */
  if (n > 0) {
    printf("Media = %.2f \n", soma / n);
  }
  return 0;
}
```

Exemplo de Leitura

- Como exemplo adicional vamos pensar em um programa que calcula a maior nota:
  - Para determinar o valor máximo de um conjunto de valores podemos inicializar uma variável com o menor valor possível.
  - Como no nosso caso estamos avaliando notas, esse menor valor possível é zero.
  - Para cada nota encontrada testamos se o valor da nota lida é maior que o valor armazenado na nossa variável, se for a nossa variável passa a armazenar o valor da nota lida.

Exemplo de Leitura

```
#include <stdio.h>

int main (void) {
  float vmax = 0.0;
  float nota;
  FILE *fp ;

  fp = fopen("notas.txt", "r");
  while (fscanf(fp , "%f", &nota) == 1) {
    if (nota > vmax) {
      vmax = nota;
    }
  }
  fclose(fp);
  printf(“Nota maxima = %f \n", vmax);
  return 0;
}
```

Escrita: fprintf

- A principal função de saída em arquivo da biblioteca padrão chama-se fprintf, sendo análoga à função printf para saída na tela.
  - A diferenca é que a função fprintf espera um primeiro parâmetro adicional que identica o arquivo.

```
int a ;
float b ;
FILE* fp = fopen("saida.txt", "w");
. . .
fprintf(fp, "Valores : %d %f \n", a, b);
. . .
fclose(fp);
```

  - Obviamente os valores de a e b precisam ser definidos antes da chamada à função fprintf.

Exemplo de Escrita

- Vamos considerar que numa disciplina cada aluno faz 3 provas, obtendo três notas que são consideradas para o cálculo da nota final.
- Considere que as notas obtidas para cada aluno estão armazenadas no arquivo “entrada.txt”

```
7.5   8.5   7.8
8.4   9.2   6.8
9.1  10.0   9.5
4.0   5.2   4.6
5.7   3.4   4.3
4.3   6.0   5.8
```

Exemplo de Escrita

- Vamos então desenvolver um programa que processe as notas do arquivo “entrada.txt”.
- Para cada aluno, o programa deve calcular a sua nota final (***nf = (p1+p2+p3)/3***) e verificar se o aluno foi aprovado ou reprovado.
- O programa deve gerar um arquivo de saída com o nome “saida.txt”. Neste arquivo de saída, para cada linha do arquivo de entrada, deve-se escrever a linha correspondente com a nota final do aluno e sua situação: A se o aluno foi aprovado (media >=5.0) ou R se o aluno foi reprovado.

Exemplo de Escrita

```
#include <stdio.h>

int main (void) {
  float p1, p2, p3, media;
  FILE *ent, *sai;

  ent = fopen("entrada.txt", "r");
  sai = fopen("saida.txt", "w");
  while (fscanf(ent, "%f %f %f", &p1, &p2, &p3) == 3) {
    media = (p1 + p2 + p3) / 3;
    fprintf(sai, “%.1f ”, media);
    if (media >= 5.0) {
      fprintf(sai, “A\n”);
    }
    else {
      fprintf(sai, “R\n”);
    }
  }
  fclose(ent);
  fclose(sai);
  return 0;
}
```

Exemplo de Escrita

- Como exemplo adicional, vamos considerar o movimento de um corpo com aceleração constante.
- Sabemos que, dados a posição inicial s0, a velocidade inicial v0, e a aceleração a, a posição e a velocidade do corpo no instante t são dadas, respectivamente, por:
  - $s = s0 + v0*t + a*t^2$
  - $v = v0 + a*t$

Exemplo de Escrita

- Nosso objetivo é criar um programa que leia do teclado os valores de entrada, s0, v0 e a, e salve num arquivo de nome "movimento.txt“ os valores das posições e velocidades ao longo do movimento, variando o tempo de 0.0 a 10.0 com incrementos de 1.0. Cada linha do arquivo deve ter os valores: tempo, posição e velocidade, conforme ilustrado a seguir:

0	s_0	v_0
1	s_1	v_1
2	s_2	v_2
3	s_3	v_3
4	s_4	v_4
5	s_5	v_5
6	s_6	v_6
7	s_7	v_7
8	s_8	v_8
9	s_9	v_9
10	s_{10}	v_{10}

Exemplo de Escrita

```
#include <stdio.h>
int main (void){
  float s0, v0, a;
  float t;
  float s, v;
  FILE *saida;

  printf("Entre com a posicao inicial: ");
  scanf("%f", &s0);
  printf("Entre com a velocidade inicial: ");
  scanf("%f", &v0);
  printf("Entre com a aceleracao: ");
  scanf("%f", &a);
  saida = fopen("movimento.txt","w");
  for (t = 0.0; t <= 10.0; t = t+1.0) {
    s = s0 + v0*t + a*t*t/2.0;
    v = v0 + a*t;
    fprintf(saida,"%f %f %f\n",t,s,v);
  }
  fclose(saida);
  return 0;
}
```