Aula 13:
Arquivos

***Disciplina*: Fundamentos de Programação**

Prof. Luiz Olmes

olmes@unifei.edu.br

Nas aulas anteriores...

- **O QUE JÁ ESTUDAMOS?**
  - Algoritmos.
  - Linguagem C.
  - Variáveis, operadores e tipos.
  - Estruturas de controle condicionais.
  - Estruturas de controle iterativas.
  - Vetores.
  - Matrizes.
  - Strings.
  - Estruturas.
  - Ponteiros e Alocação Dinâmica.
  - Funções.
- **OBJETIVOS:**
  - Definição
  - Arquivos de Texto
  - Arquivos Binários

Definição

- Um arquivo é uma coleção de bytes dispostos em um dispositivo de armazenamento secundário (disco rígido).

- Vantagens:
  - Persistência: os dados permanecem disponíveis para uso mesmo após a finalização do programa que os gerou.
  - Volume: arquivos permitem armazenar uma grande quantidade de informação.
  - Concorrência: mais de um programa é capaz de acessar os dados simultaneamente, ou seja, permite o acesso concorrente.
  - Forma de acesso: o acesso aos dados pode ser sequencial ou não.

Operações sobre arquivos

- A biblioteca padrão da linguagem C possui funções especialmente desenvolvidas para a manipulação de arquivos definidas no arquivo de cabeçalho `stdio.h`. As mais utilizadas são:

- `fopen`: utilizada para criar / abrir um arquivo.

- `fclose`: utilizada para fechar um arquivo.

- `fprintf`: utilizada para escrever dentro de um arquivo de texto.

- `fscanf`: utilizada para ler o conteúdo de um arquivo de texto.

- Além do objeto `FILE *p`, que permite referenciar um arquivo (`p`).

A função `fopen`

- A função `fopen` é utilizada para criar / abrir arquivos. Ela recebe dois parâmetros:
  - Nome do arquivo.
  - Modo de abertura.
- E retorna:
  - `NULL`, no caso de erro.
  - O ponteiro para o arquivo aberto.
- O modo de abertura informa como o arquivo será utilizado:
  - `r` (*read*): abre o arquivo (de texto) para leitura. O arquivo deve existir.
  - `w` (*write*): abre o arquivo (de texto) para escrita. Cria o arquivo, se ele não existir. Sobrescreve o arquivo (apaga seu conteúdo), caso ele já exista.

A função `fclose`

- Sempre que o programa terminar de utilizar um arquivo, este deve ser fechado.

- Somente quando um arquivo é fechado que as modificações realizadas sobre ele são efetivamente gravadas em disco.

- A função `fclose` recebe como único parâmetro o arquivo a ser fechado.

Exemplo: fopen / fclose

```
1. #include <stdio.h>
2. int main()
3. {
4.     FILE *arq;
5.     arq = fopen(“meuArquivo.txt”, “r”);
6.
7.     if (arq == NULL)
8.     {
9.         printf(“Arquivo naum pode ser aberto...\n”);
10.        exit(1); // encerra o programa...
11.    }
12.
13.    fclose(arq);
14.    return 0;
15.}
```

Exemplo: fopen / fclose

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      FILE *arq;
5.      arq = fopen("meuArquivo.txt", "r");
6.
7.      if (arq == NULL)
8.      {
9.          printf("Arquivo naum pode ser aberto...\n");
10.         exit(1); // encerra o programa...
11.     }
12.
13.     fclose(arq);
14.     return 0;
15. }
```

Abrindo o arquivo para leitura.
Tente trocar "r" por "w"...

Exemplo: fopen / fclose

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      FILE *arq;
5.      arq = fopen("meuArquivo.txt", "r");
6.
7.      if (arq == NULL)
8.      {
9.          printf("Arquivo naum pode ser aberto...\n");
10.         exit(1); // encerra o programa...
11.     }
12.
13.     fclose(arq);
14.     return 0;
15. }
```

Abrindo o arquivo para leitura. Tente trocar "r" por "w"...

Verifica se o arquivo foi aberto com sucesso. Se não foi, encerra.

Exemplo: fopen / fclose

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      FILE *arq;
5.      arq = fopen(“meuArquivo.txt”, “r”);
6.
7.      if (arq == NULL)
8.      {
9.          printf(“Arquivo naum pode ser aberto...\n”);
10.         exit(1); // encerra o programa...
11.     }
12.
13.     fclose(arq);
14.     return 0;
15. }
```

Abrindo o arquivo para leitura. Tente trocar “r” por “w”...

Verifica se o arquivo foi aberto com sucesso. Se não foi, encerra.

Fecha o arquivo.

A função `fprintf`

- A função `fprintf` é uma das funções que permitem escrever dentro de um arquivo de texto.

- Sintaxe: `fprintf(arq, "Expressão", Argumentos);`

- Ela recebe três parâmetros:
  - O arquivo em que se deseja trabalhar.
  - Expressão: idêntica à da função `printf`. Indica a mensagem ou o formato dos dados a serem gravados.
  - Argumentos: idênticos à função `printf`. Contém a lista de variáveis a serem armazenadas.

Exemplo: fprintf

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      FILE *arq;
5.      int x = 2;
6.      arq = fopen("meuArquivo.txt", "w");
7.
8.      if (arq == NULL)
9.      {
10.         printf("Arquivo naum pode ser aberto...\n");
11.         exit(1); // encerra o programa...
12.     }
13.
14.     fprintf(arq, "A\n");
15.     fprintf(arq, "%d\n", x);
16.
17.     fclose(arq);
18.     return 0;
19. }
```

Abrindo o arquivo para escrita.

Grava um caractere: letra A.

Grava uma variável inteira: x

A função `fscanf`

- A função `fscanf` é uma das funções que permitem recuperar dados de dentro de um arquivo de texto.

- Sintaxe: `fscanf(arq, "Expressão", Argumentos);`

- Ela recebe três parâmetros:
  - O arquivo em que se deseja trabalhar.
  - Expressão: idêntica à da função `scanf`. Indica o formato dos dados a serem lidos.
  - Argumentos: idênticos à função `scanf`. Contém a lista de variáveis a serem armazenadas (não esquecer do `&` nas variáveis!).

Exemplo: fscanf

```
1.  #include <stdio.h>
2.  int main() {
3.      FILE *arq;
4.      int num;
5.      char letra;
6.      arq = fopen("meuArquivo.txt", "r");
7.
8.      if (arq == NULL) {
9.          printf("Arquivo naum pode ser aberto...\n");
10.         exit(1); // encerra o programa...
11.     }
12.
13.     fscanf(arq, "%c", &letra);
14.     fscanf(arq, "%d", &num);
15.
16.     fclose(arq);
17.     printf("Letra = %c\nInteiro = %d\n", letra, num);
18.     return 0;
19. }
```

Abrindo o arquivo para leitura.

Lê um caractere.

Lê um inteiro.

Imprime os valores lidos.

Exemplo completo

- Dois programas que simulam uma aplicação bancária:

- Primeiro (escrita): faz a leitura do número da conta, nome e saldo dos clientes e armazena as informações em um arquivo de texto.
  - Número de conta = 0 encerra a leitura.

- Segundo (leitura): lê o arquivo de texto criado pelo programa anterior e exibe os dados armazenados.

Exemplo: escrita

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int conta;
6.      char cliente[30];
7.      float saldo;
8.
9.      FILE *arq;
10.
11.     arq = fopen(“banco.txt”, “w”);
12.
13.     if(!arq)
14.     {
15.         printf(“Falha ao manipular arquivo...\n”);
16.         exit(1);
17.     }
18.
```

Exemplo: escrita

```
19.     printf("Digite nro da conta (0 = encerrar), nome e saldo:\n");
20.     scanf("%d %s %f", &conta, cliente, &saldo);
21.
22.     while(conta)
23.     {
24.         fprintf(arq, "%d\t%s\t%f\n", conta, cliente, saldo);
25.
26.         printf("Digite nro da conta (0 = encerrar), nome e saldo:\n");
27.         scanf("%d %s %f", &conta, cliente, &saldo);
28.     }
29.
30.     fflush(arq); // força a gravação dos dados neste momento.
31.     fclose(arq);
32.     printf("Dados salvos com sucesso!\n");
33.     return 0;
34. }
```

Exemplo: leitura

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int conta;
6.      char cliente[30];
7.      float saldo;
8.
9.      FILE *arq;
10.
11.     arq = fopen(“banco.txt”, “r”);
12.
13.     if(!arq)
14.     {
15.         printf(“Falha ao manipular arquivo...\n”);
16.         exit(1);
17.     }
18.
```

Exemplo: leitura

```
19.     printf("Conta\tNome\tSaldo:\n");
20.     fscanf(arq, "%d %s %f", &conta, cliente, &saldo);
21.
22.     while(!feof(arq)) // Enqto não chegar ao fim do arquivo 'arq'
23.     {
24.         printf("%d\t%s\t%f\n", conta, cliente, saldo);
25.
26.         fscanf(arq, "%d %s %f", &conta, cliente, &saldo);
27.     }
28.
29.     fclose(arq);
30.
31.     return 0;
32. }
```

Exemplo completo

- O que acontece se o programa que realiza a escrita for novamente executado?

Exemplo completo

- O que acontece se o programa que realiza a escrita for novamente executado?
- Dados anteriores serão perdidos...
- A ideia seria incluir novas informações no arquivo, mas preservar aquelas já existentes.

Exemplo completo

- O que acontece se o programa que realiza a escrita for novamente executado?
- Dados anteriores serão perdidos...
- A ideia seria incluir novas informações no arquivo, mas preservar aquelas já existentes.
- Modo de abertura do arquivo: “a” (append)
  - Realiza a abertura ou criação do arquivo, com escritas sendo realizadas no final do arquivo.

Prática - especificação do problema

- Com base nos programas que realizam a leitura e escrita dos dados bancários, escreva um único programa para listar os dados dos clientes, inserir novos clientes e apagar todos os clientes. Os dados devem ser mantidos em arquivo.
- Considere os seguintes protótipos:
- `int menu()`: mostra as possíveis opções ao usuário e devolve a opção escolhida (listar, inserir, apagar, sair), para o `main()` tratar.
- `void listar()`: realiza a leitura do arquivo, imprimindo os dados na tela.
- `void inserir()`: solicita os dados ao usuário e grava no final do arquivo.
- `void apagar()`: apaga todos os registros do arquivo.

Prática - solução

```
1. #include <stdio.h>
2.
3. void apagar();
4. void inserir();
5. void listar();
6. int menu();
7.
8. int main()
9. {
10.     int opcao = 9;
11.
12.     while (opcao != 0)
13.     {
14.         opcao = menu();
15.
16.         switch (opcao)
17.         {
```

Prática - solução

```
18.             case 0:
19.                 printf(“Bye!\n”);
20.                 break;
21.             case 1:
22.                 listar();
23.                 break;
24.             case 2:
25.                 inserir();
26.                 break;
27.             case 3:
28.                 apagar();
29.                 break;
```

Prática - solução

```
30.             default:
31.                 printf(“Opcao invalida!!!\n\n”);
32.                 break;
33.         } // fim switch
34.     } // fim while
35.
36.     return 0;
37. } // fim main()
38.
```

Prática - solução

```
39. void apagar()
40. {
41.     FILE *f = fopen(“arq3.txt”, “w”);
42.
43.     if (!f)
44.     {
45.         printf(“Falha ao abrir arquivo...\n”);
46.         exit(1);
47.     }
48.
49.     fflush(f);
50.     fclose(f);
51. } // fim apagar()
52.
```

Prática - solução

```
53. void inserir()
54. {
55.     int conta;
56.     char nome[30];
57.     double saldo;
58.     FILE *f = fopen(“arq3.txt”, “a”);
59.
60.     if (!f)
61.     {
62.         printf(“Falha ao abrir arquivo...\n”);
63.         exit(1);
64.     }
65.
```

Prática - solução

```
66.     printf(“Digite o nro da conta: ”);
67.     scanf(“%d”, &conta);
68.     printf(“Digite o nome: ”);
69.     scanf(“ %s”, nome);
70.     printf(“Digite o saldo: ”);
71.     scanf(“%lf”, &saldo);
72.
73.     fprintf(f, “%d %s %lf\n”, conta, nome, saldo);
74.
75.     fflush(f);
76.     fclose(f);
77. } // fim inserir()
78.
```

Prática - solução

```
79. void listar()
80. {
81.     int conta;
82.     char nome[30];
83.     double saldo;
84.     FILE *f = fopen(“arq3.txt”, “r”);
85.
86.     if (!f)
87.     {
88.         printf(“Falha ao abrir arquivo...\n”);
89.         exit(1);
90.     }
91.
```

Prática - solução

```
92.     fscanf(f, “%d %s %lf”, &conta, nome, &saldo);
93.
94.     printf(“Nro\tNome\tSaldo\n”);
95.     while (!feof(f))
96.     {
97.         printf(“%d\t%s\t%lf\n”, conta, nome, saldo);
98.         fscanf(f, “%d%s%lf”, &conta, nome, &saldo);
99.     }
100.
101.     fclose(f);
102.} // fim listar()
103.
```

Prática - solução

```
104.int menu()
105.{
106.    int opcao;
107.
108.    printf(“Selecione a opcao:\n”);
109.    printf(“\t1 = Listar.\n”);
110.    printf(“\t2 = Inserir.\n”);
111.    printf(“\t3 = Apagar.\n”);
112.    printf(“\t0 = Sair.\n”);
113.    scanf(“%d”, &opcao);
114.
115.    return opcao;
116.} // fim menu()
117.
```

Tipos de arquivos: texto / binário

- Basicamente, a linguagem C trabalha com dois tipos de arquivos:

- Arquivos de texto: armazenam caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto simples, como o Bloco de Notas.

- Arquivos binários: um arquivo binário armazena uma sequência de bits que está sujeita às convenções dos programas que o gerou. São exemplos de arquivos binários os arquivos executáveis, arquivos compactados, arquivos de registros, etc.

Operações sobre arquivos binários

- A biblioteca padrão da linguagem C possui funções especialmente desenvolvidas para a manipulação de arquivos binários definidas no arquivo de cabeçalho `stdio.h`. As mais utilizadas são:

- `fopen`: utilizada para criar / abrir um arquivo.

- `fclose`: utilizada para fechar um arquivo.

- `fwrite`: utilizada para escrever dentro de um arquivo binário.

- `fread`: utilizada para ler o conteúdo de um arquivo binário.

- `fseek`: posiciona o ponteiro do arquivo na posição especificada.

A função `fopen`

- A função `fopen` é utilizada para criar / abrir arquivos. Ela recebe dois parâmetros:
  - Nome do arquivo.
  - Modo de abertura.

- E retorna:
  - `NULL`, no caso de erro.
  - O ponteiro para o arquivo aberto.

- No caso de arquivos binários, o modo de abertura do arquivo varia, como mostrado na tabela a seguir:
  - (Próximo slide)

A função `fopen`: modos de abertura

Modo	Função
`rb`	Leitura. Arquivo deve existir.
`wb`	Escrita. Cria arquivo se não existir. Apaga o anterior se existir.
`ab`	Escrita. Cria arquivo se não existir. Dados adicionados no fim do arquivo ("*append*"). Operações de reposicionamento do ponteiro (`fseek`, `rewind`, ...) são ignoradas.
`r+b`	Leitura/escrita. O arquivo deve existir e pode ser modificado.
`w+b`	Leitura/escrita. Cria arquivo se não existir. Apaga o anterior se existir.
`a+b`	Abre arquivo para atualização (leitura e escrita) com operações de escritas no fim do arquivo. Operações de reposicionamento do ponteiro (`fseek`, `rewind`, ...) são permitidas. Cria arquivo se não existir.

A função `fclose`

- Se comporta de modo análogo ao empregado nos arquivos de texto.
- Sempre que o programa terminar de utilizar um arquivo, este deve ser fechado.
- Somente quando um arquivo é fechado que as modificações realizadas sobre ele são efetivamente gravadas em disco.
- A função `fclose` recebe como único parâmetro o arquivo a ser fechado.

As funções `fwrite` e `fread`

- As funções `fwrite` e `fread` são utilizadas, respectivamente, para escrever e ler blocos de bytes em arquivos binários.

- Podem manipular quaisquer tipos de dados:
  - Tipos primitivos: `int`, `float`, `double`, `char`, `bool`.
  - Tipos compostos homogêneos: vetores.
  - Tipos compostos heterogêneos: `structs`.

- Manipulam a informação como ela está disposta na memória, isto é, sem a necessidade de conversões de dados.

A função `fwrite`

- A função `fwrite` é uma das funções que permitem escrever dentro de um arquivo binário.

- Sintaxe: `fwrite(*ptr, tam, qtd, *arq);`

- Ela recebe quatro parâmetros:
  - `*ptr`: ponteiro para o bloco de dados a ser escrito.
  - `tam`: tamanho em bytes de cada unidade de dado a ser gravada.
  - `qtd`: quantos elementos serão escritos, cada um de tamanho `tam`.
  - `*arq`: ponteiro para o arquivo em que se deseja trabalhar.

Exemplo: fwrite

```
1. #include <stdio.h>
2. #include <string.h>
3.
4. typedef struct aluno
5. {
6.     int matricula;
7.     char nome[30];
8. }aluno;
9.
10. int main()
11. {
12.     FILE *arq;
13.     float pi = 3.14;
14.     int vet[5] = {2, 4, 6, 8, 10};
15.     aluno a;
16.     a.matricula = 1234;
17.     strcpy(a.nome, “Ana Maria”);
18.
```

Exemplo: fwrite

```
19.     arq = fopen(“meuArquivo.dat”, “wb”);
20.
21.     if (arq == NULL)
22.     {
23.         printf(“Arquivo naum pode ser aberto...\n”);
24.         exit(1); // encerra o programa...
25.     }
26.
27.     fwrite(&pi, sizeof(float), 1, arq);
28.
29.     fwrite(vet, sizeof(int), 5, arq);
30.
31.     fwrite(&a, sizeof(aluno), 1, arq);
32.
33.     fclose(arq);
34.
35.     return 0;
36. }
```

Exemplo: fwrite

```
19.     arq = fopen("meuArquivo.dat", "wb");
20.
21.     if (arq == NULL)
22.     {
23.         printf("Arquivo naum pode ser aberto...\n");
24.         exit(1); // encerra o programa...
25.     }
26.
27.     fwrite(&pi, sizeof(float), 1, arq);
28.
29.     fwrite(vet, sizeof(int), 5, arq);
30.
31.     fwrite(&a, sizeof(aluno), 1, arq);
32.
33.     fclose(arq);
34.
35.     return 0;
36. }
```

Abrindo arquivo binário para escrita.

Grava uma variável `float`

Grava um array de `int`

Grava uma variável do tipo da `struct`.

A função `fread`

- A função `fread` é uma das funções que permitem recuperar dados de dentro de um arquivo binário.

- Sintaxe: `fread(*ptr, tam, qtd, *arq);`

- Ela recebe quatro parâmetros:
  - `*ptr`: ponteiro para um bloco de memória com ao menos (`tam` * `qtd`) bytes: onde os dados recuperados serão gravados.
  - `tam`: tamanho em bytes de cada unidade de dado a ser lida.
  - `qtd`: quantos elementos serão lidos, cada um de tamanho `tam`.
  - `*arq`: ponteiro para o arquivo em que se deseja trabalhar.

Exemplo: fread

```
1.  #include <stdio.h>
2.  #include <string.h>
3.
4.  typedef struct aluno
5.  {
6.      int matricula;
7.      char nome[30];
8.  }aluno;
9.
10. int main()
11. {
12.     FILE *arq;
13.     float pi;
14.     int vet[5], i;
15.     aluno a;
16.
```

Exemplo: fread

```
17.     arq = fopen("meuArquivo.dat", "rb");
18.
19.     fread(&pi, sizeof(float), 1, arq);
20.
21.     fread(vet, sizeof(int), 5, arq);
22.
23.     fread(&a, sizeof(aluno), 1, arq);
24.
25.     fclose(arq);
26.
27.     printf("Float: %f\nVetor: ", pi);
28.     for(i = 0; i < 5; i++) printf("%d\n", vet[i]);
29.     printf("Aluno: %s (%d)\n", a.nome, a.matricula);
30.
31.     return 0;
32. }
```

Abrindo arquivo binário para leitura.

Lê uma variável `float`

Lê um array de `int`

Lê uma variável do tipo da `struct`.

A função `fseek`

- A função `fseek` é utilizada para mover o ponteiro do arquivo para a posição especificada.

- Sintaxe: `fseek(*arq, num, ref);`

- Ela recebe três parâmetros:
  - `*arq`: ponteiro para o arquivo em que se deseja trabalhar.
  - `num`: número de bytes a partir da origem.
  - `ref`: posição usada como referência para o deslocamento de `num` bytes:
    - `SEEK_SET`: início.
    - `SEEK_CUR`: posição atual do ponteiro.

Exemplo: fseek

```
1.  #include <stdio.h>
2.  #include <string.h>
3.
4.  typedef struct aluno
5.  {
6.      int matricula;
7.      char nome[30];
8.  }aluno;
9.
10. int main()
11. {
12.     FILE *arq = fopen("alunos.bin", "wb");
13.     aluno a;
14.     int n;
15.
16.     a.matricula = 1234;
17.     strcpy(a.nome, "Ana");
18.     fwrite(&a, sizeof(aluno), 1, arq);
19.
20.     a.matricula = 2345;
21.     strcpy(a.nome, "Lia");
22.     fwrite(&a, sizeof(aluno), 1, arq);
23.
24.     a.matricula = 3456;
25.     strcpy(a.nome, "Zeca");
26.     fwrite(&a, sizeof(aluno), 1, arq);
27.
28.     fclose(arq);
29.
```

Exemplo: fseek

```
30.     arq = fopen("alunos.bin", "rb");
31.
32.     // Recuperando o segundo registro
33.     n = 2;
34.
35.     fseek(arq, (n - 1) * sizeof(aluno), SEEK_SET);
36.
37.     fread(&a, sizeof(aluno), 1, arq);
38.
39.     printf("Aluno: %s (%d)\n", a.nome, a.matricula);
40.
41.     fclose(arq);
42.
43.     return 0;
44. }
```

Posicionando o ponteiro no local desejado.

Dúvidas?

Aula 13:
Arquivos

***Disciplina*: Fundamentos de Programação**

Prof. Luiz Olmes

olmes@unifei.edu.br