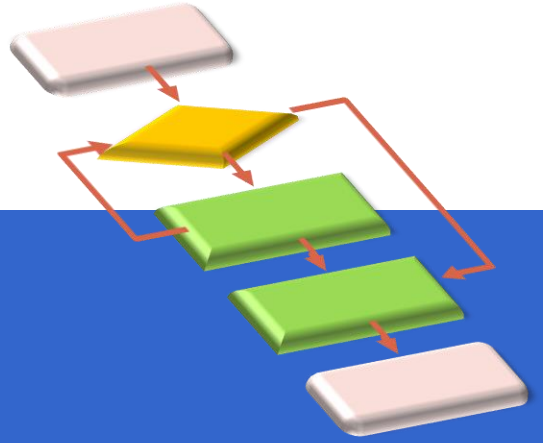
INSTITUTO DE COMPUTAÇÃO



Algoritmos II

ARQUIVOS
Linguagem C – Arquivos Binários

Prof.ª Vanessa de Oliveira Campos

Arquivos binários

- Os **arquivos binários** armazenam unidades de informação em binário puro.
- Se o conteúdo de um arquivo binário é listado em editor de texto, ele geralmente e incompreensível.
- Em **arquivos binários**, pode-se armazenar tipos de informações diferentes num mesmo arquivo.

Funções de acesso - uso no acesso sequencial

- No processamento de arquivos binários, são utilizadas as funções:
  - **fwrite**
  - **fread**
  - **fflush**
  - **feof**

Escrita

- **fwrite** : Grava blocos de bytes em um arquivo a partir do elemento corrente.

  fwrite (const void *varbuffer, unsigned int numbytes , unsigned int quant , FILE *f)

  - **numbytes**: tamanho do tipo de dados que será gravado.
  - **quant**: quantidade de vezes que o tipo de dado será gravado.
  - **f**: variável arquivo que já deve ter sido aberto.

Exemplo:

```
fwrite (&valor, sizeof(float), 1, arq);
```

```
#include <stdio.h>
void main()
{
   FILE *f;
   float x;
   f = fopen("exemplo.dat", "wb");
   printf("\nEntre com valores reais positivos.");
   printf("\nNumero real (negativo para parar): ");
   scanf("%f", &x);
   while(x >= 0)
   {
      fwrite(&x, sizeof(float), 1, f);
      printf("\nNumero real (negativo para parar): ");
      scanf("%f",&x);
   }
   fclose(f);
   printf("fim do programa");
}
```

```
void RotinaInsere(FILE *arq2, Tatleta buffer, int max)
{
   printf("\nNome (com no máximo %d caracteres): ", max);
   scanf(" %[^\n]s", buffer.nome);
   printf("\nIdade: ");
   scanf("%d",&buffer.idade);
   printf("\nAltura: ");
   scanf("%f",&buffer.altura);
   fwrite(&buffer, sizeof(struct Tatleta), 1, arq2);
}
```

Leitura

- A função fread lê blocos de bytes de um arquivo a partir do elemento corrente.

  fread (cons t void *varbuffer, unsigned int numbytes, unsigned int quant , FILE *f)

  numbytes, **quant** e **f** têm o mesmo significado que no fwrite.

- O parâmetro **varbuffer** é usado para armazenar os dados que são lidos do arquivo e deve ter espaço de memória alocada suficiente para isso.

- A função retorna a quantidade de bytes lidos.

Exemplo:

```
fread(&valor, sizeof(integer), 1, arq);
```

```
#include <stdio.h>
void main()
{
   FILE *f;
   float x;
   f = fopen("exemplo.dat", "rb");

   if ( f != NULL )
      while( fread(&x, sizeof(float), 1, f) !=0)
         printf("\n%6.2f ",x);

   fclose(f);
   printf("fim do programa");
}
```

```
typedef struct Tatleta {
    char nome[TAM];
    int idade;
    float altura;
} Tatleta;

void main()
{
    FILE *f;
    Tatleta a;
    f = fopen("BDAtletas.dat", "rb");
    printf("----- Dados dos Atletas -----\n");
    while(!(feof(f)))
        if (fread(&a, sizeof(Tatleta), 1, f) == 1)
        {
            printf("Nome: %s\n", a.nome);
            printf("Idade: %d\n", a.idade);
            printf("Altura: %.2f\n\n", a.altura);
        }
    fclose(f);
}
```

função fflush

- A função fflush descarrega o conteúdo dos buffers para o arquivo, sem fechar o fluxo de dados.
- Seu protótipo é

```
fflush ( FILE *f );
```

Exemplo:

```
FILE *f;
Tatleta a[3];
f = fopen("BDAtletas.dat", "ab");
. . .
fwrite(a, sizeof(Tatleta),3, f);
fflush(f);
. . .
```

Acesso aleatório em um arquivo binário

- É possível, atingir diretamente um determinado registro ou posição do arquivo binário, sem ter de passar por cada elemento.

- Na linguagem C, esse conceito é implementado através da função fseek:

```
int fseek ( FILE *f , long int deslocamento , int origem ) ;
```

  - **deslocamento**: quantidade de bytes deslocada a partir da posição definida no parâmetro origem.
  - **origem**: pode ser:
    - SEEK_SET : início do arquivo;
    - SEEK_CUR : posição atual do cursor;
    - SEEK_END : fim do arquivo.

```
void main()
{
   FILE *f;
   int pos;
   Tatleta a;
   f = fopen("BDAtletas.dat", "rb");
   printf("Qual posição de registro gostaria de verificar? ");
   scanf("%d", &pos);
   fseek(f, pos*sizeof(Tatleta),SEEK_SET);
   if (fread(&a, sizeof(Tatleta), 1, f) == 1)
   {
      printf("Nome: %s\n", a.nome);
      printf("Idade: %d\n", a.idade);
      printf("Altura: %.2f\n\n", a.altura);
   }
   fclose(f);
}
```

Apagando um arquivo

- A função remove apaga um arquivo. A sintaxe é:

```
remove(char *nome_arq);
```

onde:

nome_arq: nome físico do arquivo que será removido, podendo ser incluído o caminho.

- Se a função remove for executada com êxito, será devolvido o número zero.
- É importante fechar o arquivo antes de removê-lo.

Renomeando um arquivo

- A função rename troca o nome de um arquivo. Sintaxe:

```
rename(char *nome_atual, char *nome_novo);
```

onde

nome_atual: nome físico atual do arquivo, podendo ser incluído o caminho;

nome_novo: nome físico que se pretende dar ao arquivo, podendo ser incluído o caminho.

Renomeando um arquivo

Exemplos:

```
rename(“c:\\exemplo\\clientes.dat”, “c:\\exemplo\\dados.cli”);

rename(“c:\\ exemplo\\clientes.dat”, “c:\\atividades\\dados.cli”);
```

Exercício de Fixação

1) Escreva os programas a seguir, que propõem realizar o controle de bolsistas de uma instituição:

a) Gere, de forma sequencial, o arquivo binário bolsista1.cad, com no mínimo 5 registros. Cada bolsista terá seus dados armazenados em uma estrutura com os seguintes campos: codigo (inteiro entre 1 e 25); nome (string de 30 caracteres); tipo_bolsa (inteiro, com os valores válidos: 1 – trabalho; 2 – iniciação; 3 – pesquisa); e e_mail (string de 30 caracteres). Realize a verificação de validade nos campos codigo e tipo_bolsa e só aceite valores corretos.

b) Liste, sequencialmente, o arquivo “bolsista1.cad”.

Exercício de Fixação

2) Gere um arquivo funcionário, em que cada registro armazene o código do funcionário (inteiro entre 1 e 140), seu nome (35 caracteres), endereço (45 caracteres), salário (real) e a função (30 caracteres), com no mínimo 10 funcionários, tendo pelo menos 3 com a função de gerente. O acesso aos dados dos funcionários será feito com base no seu código, utilizando acesso direto. Os dados de um funcionário estarão na posição do seu código.

Exercício de Fixação

3) A partir dos dados armazenados no arquivo funcionário (exercício anterior), gere um relatório que liste o nome e o salário dos funcionários cuja função é “gerente”.

Exercício de Fixação

4) Faça um programa que atualize o salário de um funcionário no arquivo dos exercícios anteriores. O código do funcionário e o novo salário devem ser lidos do teclado.