MINIAULA DE ALGORITMOS

ARQUIVOS

Prof. Ivanilton Polato
Departamento Acadêmico de Computação (DACOM–CM)
ipolato@utfpr.edu.br

UTFPR

Arquivos: o que são?

- Conjuntos de dados armazenados em memória secundária
  - *Disco rígido (HD, SSD, CD, DVD)*
- Necessidade de persistir os dados da memória principal
  - *Podem ser acessados em diferentes execuções de um programa*
  - *Podem ser acessados por diversos programas*
- Evolução: SGBDs

UTFPR

Arquivos: declaração

- Criamos um ponteiro para o arquivo:

```
FILE *arq;
```

- **arq** é uma variável (ponteiro) que armazena o endereço inicial de memória ocupado por um arquivo
  - *se o arquivo não puder ser aberto arq recebe NULL e a operação falha!*
- Erros: arquivo inexistente, permissões, espaço em disco

UTFPR

Arquivos: abertura

- Usamos a função fopen:

```
arq = fopen(nome_arquivo, modo_de_abertura);
```

- O nome do arquivo é o identificador do arquivo que se deseja abrir, podendo incluir o caminho para o arquivo

```
arq = fopen("teste.txt", "r");
arq = fopen("/pasta1/pasta2/teste.txt", "r");
```

- O modo de abertura determina como o arquivo poderá ser manipulado: leitura (**r**), escrita (**w** ou **a**), o tipo do arquivo (texto ou binário) e atualizações permitidas (**+**)

UTFPR

Arquivos: modo de abertura

Sigla	Modo	Efeito
r	READ: Abre o arquivo em formato texto no modo somente leitura	Se o arquivo não existir, ocorrerá um erro!
w	WRITE: Abre o arquivo em formato texto no modo escrita	Se o arquivo não existir o arquivo será criado; Se existir o arquivo será sobrescrito e o conteúdo original apagado!
a	APPEND: Abre o arquivo em formato texto no modo anexo	Se o arquivo não existir o arquivo será criado Se existir o arquivo será aberto e o conteúdo original preservado!
rb wb ab	BINARY: Modos similares, mas em formato binário	Utilizado para arquivos binários, em geral, gravar estruturas complexas (registros) através de um streaming de bytes.
r+/rb+ w+/wb+ a+/ab+	+: Modos similares, mas permite atualização de dados no arquivo (gravação sobrescrita)	O modificador + pode ser utilizado em situações onde são necessárias correções em dados gravados anteriormente nos arquivos.

UTFPR

Exemplo

```
#include <stdio.h>

int main (){
    FILE *arq;
    arq = fopen("arquivoTeste.txt","r");

    if(arq == NULL)
       printf("Erro: arquivo não pode ser aberto!");
    else{
      printf("Arquivo aberto com sucesso!\n");
      fclose(arq);
    }
}
```

Arquivo: fechamento

- É extremamente recomendável fechar um arquivo depois de utilizá-lo.
- Caso o arquivo não seja fechado corretamente podem ocorrer erros
  - *Perda de dados!*
  - *Perda do arquivo!*
- A função fclose() fecha um arquivo

```
fclose(arq);
```

Arquivos: gravando dados

- Para gravar **um caractere** no arquivo:
  - *Função fputc(char ch, FILE *arq);*

```
char c = '@';
fputc(c, arq);
```

- Para gravar **uma string** no arquivo:
  - *Função fputs(char *cadeia, FILE *arq);*

```
char str[]="Algoritmos";
fputc(str, arq);
```

UTFPR

Arquivos: lendo dados

- Para ler **um caractere** no arquivo:
  - *Função fgetc(FILE *arq);*

```
char c = fgetc(arq);
```

- Para ler **uma string** no arquivo:
  - *Função fgets(char *cadeia, int tamanho, FILE *arq);*

```
char str[100];
fgets(str, 100, arq);
```

  - *Essa função lê uma string do arquivo até encontrar uma quebra de linha (\n) ou até o tamanho máximo definido*
  - *Caso não seja possível, a função retorna NULL*

UTFPR

Exemplo de gravação (caracteres)

```
#include <stdio.h>
int main (){
    FILE *arq;
    char caracas = '!';
    arq = fopen("dados.txt","a");
    if (arq == NULL)
       printf("Erro na abertura!\n");
    else{
       while (caracas != '0'){
          printf("Digite um caractere, ou 0 para sair:\n");
          scanf(" %c", &caracas);
          fputc(caracas, arq);
          if (ferror(arq))
             printf("Erro ao tentar escrever no arquivo\n");
       }
    }
    fclose(arq);
}
```

Arquivos: ferror()

- A função **ferror()** detecta se ocorreu algum erro durante uma operação com arquivos.
  - *A sintaxe correta é: ferror(FILE *arq);*
- A função ferror() retorna um número inteiro e deve ser chamada logo depois que qualquer outra função for invocada.
  - *Se o número retornado for diferente de zero, isto significa erro durante a última operação.*
  - *Se for zero não ocorreu erro.*

UTFPR

Exemplo de leitura (caracteres)

```
#include <stdio.h>
int main (){
   FILE *arq;
   char caracas = '!';
   arq = fopen("dados.txt","a");
   if (arq == NULL)
      printf("Erro na abertura!\n");
   else{
      while ( (caracas = fgetc(arq)) != EOF ){
         if (ferror(arq))
            printf("Erro na leitura do caractere\n");
         else
            printf("Caractere: %c\n", caracas);
      }
   }
   fclose(arq);
}
```

Arquivos: EOF

- É o caractere (marcador) que indica o fim de arquivo
- Usamos para verificar, em uma repetição, se chegamos ao final do arquivo e interromper a leitura no momento correto
- No exemplo, comparamos o resultado da operação fgetc(), que lê um caractere do arquivo e armazena em uma variável, com o EOF.
  - *Caso a operação de leitura retorne o EOF, a execução termina, e não são impressos caracteres indevidos!*

UTFPR

Arquivos: função `fprintf()`

- A função **fprintf()** envia texto formatado, assim como o **printf()**, para um arquivo.
- Funciona da mesma forma, mas agora temos que adicionar o ponteiro para o arquivo. Veja o exemplo:

```
#include <stdio.h>
int main(){
    FILE * arq;
    arq = fopen ("teste.txt", "a");
    fprintf(arq, "%s %d %c", "Estamos em", 2020, '!');
    fclose(arq);
}
```

Arquivos: função `fscanf()`

- A função **fscanf()** funciona de maneira similar ao scanf(), mas lendo dados de um arquivo e armazenando nas respectivas variáveis!
- Devem ser respeitadas as mesmas regras para o uso do & nas variáveis
- Podem ser feitas múltiplas leituras de uma vez só, se conhecermos a maneira como foram armazenados no arquivo!

UTFPR

Arquivos: função `fscanf()`

```c
#include <stdio.h>

int main(){
    char str[11];
    int ano;
    char caracas;
    FILE *arq;

    arq = fopen ("teste.txt", "r+");
    fscanf(arq, "%s %d %c", str, &ano, &caracas);

    printf("Dados lidos do arquivo!\n");
    printf("Frase: %s %d%c", str, ano, caracas);
    fclose(arq);
}
```

UTFPR

Arquivos binários: gravando bytes!

- Arquivos armazenam uma sequência de caracteres ou de bytes.
- Em alguns programas é mais útil e prático ler parte do conteúdo de um arquivo e gravar diretamente em uma variável simples, como ***int*** ou ***float***, ou ainda em uma variável de um tipo ***struct***.
  - *A função **fscanf()** permite fazer isso com tipos simples e strings, mas não com structs!*
- Quando isso for necessário, o programa deverá abrir com arquivos binários.
- Toda vez que uma operação de leitura ou de escrita for realizada, deverá ser informado o número de bytes que serão lidos ou gravados.
  - *Para isso, a função sizeof() será utilizada intensamente, uma vez que ela permite descobrir quantos bytes uma variável ou **struct** ocupa.*

UTFPR

Arquivos: função `fwrite()`

- A função **fwrite()** pode gravar qualquer tipo de dados, e não apenas caracteres ou strings! A forma geral é:

fwrite(void *dados, size_t qtBytes, size_t numItens, FILE *arq);

  - ***dados*** *representa a variável com o conteúdo a ser gravado no arquivo*
  - ***qtBytes*** *é o tamanho em bytes que será escrito no arquivo*
  - ***numItens*** *é o número de itens de tamanho qtBytes escritos no arquivo*
  - ***arq*** *é a referência para o arquivo onde as informações serão escritas*
- Quando a função fwrite() é bem sucedida, gera como retorno um valor igual ao número de gravações realizadas, igual ao parâmetro **numItens**!
  - *Se ocorrer algum erro, o valor retornado será menor que numItems.*

Exemplo de uso do fwrite()

```c
#include <stdio.h>

struct DADOS{
   int codigo;
   char nome[20];
}cliente;

int main (){
   FILE *arq;
   arq = fopen("dados.dat", "ab+");
   if (arq == NULL) printf("Erro na abertura do arquivo.\n");
   else{
      printf("Digite o código do cliente: ");
      scanf(" %d", &cliente.codigo);
      printf("Digite o nome: ");
      scanf("%[^\n]", cliente.nome);
      fwrite(&cliente, sizeof(cliente), 1, arq);
      if (ferror(arq))
         printf("Erro na gravação de dados.\n");
      else
         printf("Dados gravados com sucesso!\n");
      fclose(arq);
   }
}
```

UTFPR

Arquivos: função `sizeof()`

- A função sizeof() calcula o tamanho em bytes de um tipo de dados
- Seu uso é importante quando gravando bytes em arquivo, pois as funções fwrite() e fread() precisam saber quantos bytes serão gravados no arquivo!
- Podemos aplicar aos tipos simples também!

```
int x;
printf("Tamanho do int: %d bytes!", sizeof(x));
```

  - *Nesse caso, teríamos:* ***Tamanho do int: 4 bytes!***

UTFPR

Arquivos: função `fread()`

- A função **fread()** funciona como a fwrite(), mas lê qualquer tipo de dados do arquivo! A forma geral é:

fwrite(void *dados, size_t qtBytes, size_t numItens, FILE *arq);

  - ***dados*** *representa a variável que vai receber o conteúdo do arquivo*
  - ***qtBytes*** *é o tamanho em bytes que será lido do arquivo*
  - ***numItens*** *é o número de itens de tamanho* ***qtBytes*** *lidos do arquivo*
  - ***arq*** *é a referência para o arquivo de onde as informações serão lidas*
- Quando a função fread() é bem sucedida, gera como retorno um valor igual ao número de leituras realizadas, igual ao parâmetro **numItens**!
  - *Se ocorrer algum erro, o valor retornado será menor que numItems.*

Exemplo de uso do fread()

```c
#include <stdio.h>

struct DADOS{
   int codigo;
   char nome[20];
}cliente;

int main (){
   FILE *arq;
   arq = fopen("dados.dat", "ab+");
   if (arq == NULL) printf("Erro na abertura do arquivo.\n");
   else{
      while((fread(&cliente, sizeof(cliente), 1, arq)) != EOF){
         if(ferror(arq))
            printf("Erro na leitura de dados do arquivo.\n");
         else{
            printf("Código do cliente: %d\n", cliente.codigo);
            printf("Nome do cliente: %s\n", cliente.nome);
         }
      }
      fclose(arq);
   }
}
```

UTFPR

Funções adicionais: `rewind()`

- Cursor é um ponteiro que indica a partir de que posição, dentro do arquivo, uma operação será executada.
- Por exemplo, quando um arquivo acaba de ser aberto, seu cursor está apontando para a posição zero.
- Caso seja feita uma leitura com o comando fread(), o cursor se movimentará quantos bytes forem lidos.
- A função **rewind()** reposiciona o cursor de volta ao início do arquivo. Sua sintaxe é:

```
rewind(FILE *arq);
```

UTFPR

Funções adicionais: `fseek()`

- A função **fseek()** é utilizada para mudar a posição do cursor sem que haja necessidade de leituras ou escritas no arquivo. Sua sintaxe é:

```
fseek(FILE *arq, long qtBytes, int pos);
```

  - *arq – representa o arquivo*
  - *qtd_bytes – representa a quantidade de bytes que o cursor será movimentado a partir de pos*
  - *pos – representa o ponto a partir do qual a movimentação será executada*

UTFPR

Funções adicionais: `fseek()`

- O parâmetro pos da função fseek() pode assumir três valores:
  - *SEEK_SET – movimenta qtBytes a partir da posição inicial do arquivo*
  - *SEEK_CUR – movimenta qtBytes (positivos para frente, negativos para retroceder) a partir do ponto atual do cursor;*
  - *SEEK_END – movimenta qtBytes (negativos, para retorceder) a partir da posição final do arquivo.*
- Para o SEEK_SET e SEEK_END, se utilizarmos 0 bytes, o ponteiro vai ficar no início e no fim do arquivo, respectivamente!

UTFPR