Arquivos

A linguagem C utiliza o conceito de *streams* e arquivos para manipular entradas e saídas de dados. Um *stream* é um dispositivo lógico representando um arquivo e pode ser de dois tipos: de texto ou binário.

- Um stream de texto é uma sequencia de caracteres dividida em linhas (terminadas em \n).
- Um stream binario é uma sequencia de valores inteiros, reais, ou complexos usando suas representações de memória.

1. Criação de Arquivos

A biblioteca *stdio.h* contém as funções de manipulação de arquivo, e portanto deve ser incluída no início do programa.

A função *fopen* pode ser utilizada para abrir um arquivo em C com dois argumentos, sendo o primeiro argumento o nome do arquivo e o segundo a forma de abertura:

```
"r":  abre um arquivo para leitura
"w":  cria uma arquivo para escrita
"a":  acrescenta dados a um arquivo já existente
```

t ou *b* representam arquivo texto ou binário:

```
"rt": abre um arquivo texto para leitura
"rb": abre um arquivo binário para leitura
…
```

Exemplo:

```
FILE *arq;

arq = fopen("nomeArquivo.txt", "rt");

if (arq == NULL){
    printf("O arquivo não pode ser criado\n");
    return;
}
...
```

2. Funções para Manipulação de Arquivos

Funções para abrir e fechar arquivos:

```
FILE* fopen (char* nome_arquivo, char* modo);
int fclose (FILE* fp);
```

Funções para arquivo em modo texto:

```
int fscanf (FILE* fp, char* formato, ...);
int fgetc (FILE* fp);
char* fgets (char* s, int n, FILE* fp);
int fprintf(FILE* fp, char* formato, ...);
int fputc (int c, FILE* fp);
```

Funções para arquivos em modo binário:

```
int fwrite (void* p, int tam, int nelem, FILE* fp);
int fread (void* p, int tam, int nelem, FILE* fp);
int fseek (FILE* fp, long offset, int origem);
```

3. Exercícios

1) Escreva um programa que, dado um arquivo de entrada cujo nome foi fornecido pelo usuario, converta todas as letras minúsculas para maiúsculas. Use as funções abaixo:

```
int fgetc (FILE* pt_in);
int fputc (int ch, FILE* pt_out);
char toupper (char ch);  (ctype.h)
```

Exercícios da Prova → Inclusão da gravação dos resultados em disco.

2) Faça uma função que recebe 2 vetores A e B com n valores inteiros ordenados em ordem crescente, por parâmetro. O procedimento deve gerar um vetor C que contém os elementos de A e B em ordem crescente. A área para o vetor C dever ser alocada dinamicamente dentro da função. A função retorna o vetor C se a função foi executada com sucesso ou NULL caso contrário:

```
int* combina(int* A, int* B, int n);
```

3) Escreva um programa que dados dois vetores A e B de n valores inteiros, cujos valores foram dados pelo usuário, combine os elementos de A e B de tal forma a gerar um terceiro vetor cujos elementos estão ordenados em ordem crescente, usando a função do item 1. O valor para n deve ser dado por linha de comando. Imprima o vetor C no monitor e grave o vetor em um arquivo cujo nome é dado por linha de comando.

4) Faça uma função que dadas duas strings str1 e str2, retorna uma terceira string str3 que é a concatenação de str1 e str2, sem modificar str1 ou str2. A área de str3 deve ser alocada dinamicamente dentro da função. A função retornar a string str3 se a função foi executada com sucesso ou NULL caso contrário. A função também retorna o tamanho da str3 por parâmetro.

```
char* concat(char* str1, char* str2, int* tam_str3);
```

5) Escreva um programa que leia duas strings via interação com o usuário e as concatene usando a função do item anterior. Imprima a string concatenada e grave a string em um arquivo cujo nome é dado via interação como usuário. Lembre-se que a string pode conter caracteres brancos.