ARQUIVOS: ENTRADA E SAÍDA DE DADOS

O uso de arquivos pode ser indicado em vários casos, por exemplo:

- quando é necessário armazenar os dados entre as rodadas do programa
- quando o volume de dados é maior do que a capacidade da RAM
- quando os dados de entrada par o programa são gerados pelo outro programa ou dispositivo.

Existem dois tipos de arquivos: binários ou texto. Em ambos os casos um arquivo é uma sequência de bytes.

A linguagem C/C++ permite realização de várias operações com arquivos de alto e baixo nível.

Abertura de arquivo

Para abrir um arquivo ou criar um arquivo novo é usada função **fopen().** Quando chamada, essa função inicializa um objeto do tipo **FILE**, que contem a informação necessária para controle de entrada/saída de dados.

O protótipo da função **fopen()**:

```
FILE *fopen( const char * filename, const char * mode );
```

onde:

filename – é uma **string** que contem o nome do arquivo
mode – é uma variável que define a forma de acesso (as operações permitidas)

Modo	Descrição
r	read – abre um arquivo texto existente para leitura
w	write – abre um arquivo texto para entrada de dados. Caso o arquivo não existe – ele é criado. O programa começa a escrever os dados no início do arquivo.
a	append – abre o arquivo para acrescentar os dados. Caso o arquivo não existe – ele é criado. O programa começa a escrever os dados no final do arquivo.
r+	abre o arquivo texto para entrada e saída de dados.
w+	abre o arquivo texto para entrada e saída de dados. Primeiro o tamanho da arquivo é zerado (caso o arquivo tinha tamanho diferente de zero), se o arquivo não existia ele é criado.
a+	abre o arquivo texto para entrada e saída de dados. A leitura começa do inicio do arquivo, mas os dados serão acrescentados no final do arquivo.

No caso dos arquivos binários o modo de acesso será modificado para:

```
rb, wb, ab, rb+, r+b, wb+, w+b, ab+, a+b
```

Fechamento de arquivo

O arquivo deve ser fechado usando a função **fclose().**
O protótipo dessa função tem o seguinte formato:

```
int fclose( FILE *fp );
```

Essa função retorna zero em caso de sucesso ou **EOF** se ocorrer um error na hora de fechar o arquivo.

A função passa todos os dados pendentes para o arquivo, fecha o arquivo e libera a memória.

EOF – é uma constante definida em **stdio.h.**

Escrita de dados para arquivo

Para escrever os caracteres individuais pode ser usada a função:

```
int fputc( int c, FILE *fp );
```

Essa função escreve o valor do **c** para o arquivo referenciado pelo **fp**.
Retorna o valor do caractere escrito em caso de sucesso ou **EOF** em caso de falha.

Para escrever uma sequência de caracteres (que termina com **null**) pode ser usada a função:

```
int fputs( const char *s, FILE *fp );
```

Essa função escreve o **string s** para o arquivo referenciado pelo **fp**.
Vai retornar o valor não nulo em caso de sucesso ou **EOF** em caso de falha.

A função **fprintf()** também pode ser usada para gravar uma sequência de caracteres para o arquivo.

```
int fprintf(FILE *fp, const char *format, ...)
```

Leitura de dados do arquivo

Para ler um caractere do arquivo é usada a função **fgetc():**

```
int fgetc( FILE * fp );
```

A função retorna o caractere lido ou **EOF** em caso de falha.

Para fazer a leitura de uma **string** de caracteres:

```
char *fgets( char *buf, int n, FILE *fp );
```

A função vai tentar fazer a leitura de **n-1** caracteres do arquivo referenciado pelo **fp**.

Ela copia a sequência lida para **buf** e acrescenta o caractere **null** para indicar o fim da sequência de caracteres.

Se a função encontrar o símbolo de nova linha **'\n'** ou o fim do arquivo **EOF** antes de processar **n-1** caracteres, ela vai copiar somente caracteres encontrados até esse momento.

Outra função que pode ser usada para leitura de dados é:

```
int fscanf(FILE *fp, const char *format, ...)
```

Essa função vai fazer a leitura de dados de forma similar, porém vai parar a leitura se encontrar um espaço em branco(' ').

Funções de entrada/saída de dados para arquivos binários

Para fazer a entrada/saída de dados para arquivos binários são usadas funções:

```
size_t fread(void *ptr, size_t size_of_elements, size_t number_of_elements, FILE *a_file);
```

```
size_t fwrite(const void *ptr, size_t size_of_elements, size_t number_of_elements, FILE *a_file);
```

Exemplo 1: Criação do arquivo **output.txt**

```c
#include<stdio.h>
//criação do arquivo output.txt

int main()
{
    FILE *filePtr;
    int i;

    // criar um arquivo
    filePtr =fopen("output.txt", "w");

    if (filePtr == NULL)
     {
       printf("\n Erro! Não foi possivel criar arquivo! \n ");
       return 1; // código de eero
     }

    // gravar dados para arquivo
    for (i = 1; i <= 10; i++)
       fprintf(filePtr,"%d\n", i);

    // fechar o arquivo
    printf("\n O programa gravou números de [1,10] para arquivo output.txt \n ");
    fclose(filePtr);

    return  0;
}
```

O programa gravou números de [1,10] para arquivo output.txt

Exemplo 2: Leitura de dados do arquivo **output.txt**

```c
#include<stdio.h>
// leitura de dados do arquivo output.txt
// o arquivo output.txt deve estar presente na pasta atual
int main()
{
    FILE *in_fPtr;
    int num;

    // abertura do arquivo para leitura
    in_fPtr =fopen("output.txt", "r");

    if (in_fPtr == NULL)
     {
       printf("\n Erro! Não foi possivel abrir o arquivo! \n ");
       return 1; // código de eero
     }

    // leitura de dados do arquivo
    printf("\n Leitura de dados do arquivo output.txt \n");
    while( fscanf(in_fPtr, "%d", &num ) == 1 )
       printf("Número: %d\n", num);

    // fechar o arquivo
    fclose(in_fPtr);

    return  0;
}
```

```
 Leitura de dados do arquivo output.txt
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
Número: 6
Número: 7
Número: 8
Número: 9
Número: 10
```

Exemplo 3: Criação do arquivo **alunos.txt**

```c
//criação do arquivo alunos.txt
#include<stdio.h>

int main()
{
    FILE *filePtr;
    int id, i = 1, ch;
    float n;
    char name[100];

    // criar um arquivo
    filePtr =fopen("alunos.txt", "w");

    if (filePtr == NULL)
     {
       printf("\n Erro! Não foi possivel criar arquivo! \n ");
       return 1; // código de eero
     }

    // ler dados e gravar para arquivo
    printf("\n Programa grava dados para arquivo alunos.txt \n ");

    while(i != 0)
    {
      printf("\n Digite matricula do aluno: ");
      scanf("%i",&id);

     //  fflush(stdin);
      printf("\n Digite nome do aluno: ");
      scanf(" %[^\t\n]s",name);

      printf("\n Digite a nota do aluno: ");
      scanf("%f",&n);

      printf("\n Voce digitou: ");
      printf("\n Matricula: %i", id);
      printf("\n Nome: %s", name);
      printf("\n Nota: %.2f", n);

      printf("\n Gravara para arquivo? 1- Sim; 0 - Não : ");
      scanf("%i",&ch);
```

```
42
43          if( ch == 1)
44          {  // gravar dados para arquivo
45            fprintf(filePtr, "%d\n", id);
46            fprintf(filePtr, "%s\n", name);
47            fprintf(filePtr, "%.2f\n", n);
48          }
49
50          printf("\n Deseja continuar? 1 - Sim; 0 - Não : ");
51          scanf("%i",&i);
52       }
53
54       fclose(filePtr);
55       return  0;
56    }
```

Exemplo 4: Leitura de dados do arquivo **alunos.txt**

```c
//leitura do arquivo alunos.txt
#include<stdio.h>

int main()
{
    FILE *filePtr;
    int id;
    float n;
    char name[100];

    // criar um arquivo
    filePtr =fopen("alunos.txt", "r");

    if (filePtr == NULL)
     {
       printf("\n Erro! Não foi possivel abrir arquivo! \n ");
       return 1; // código de eero
     }

    // ler dados e gravar para arquivo
    printf("\n Programa faz a leitura de dados do arquivo alunos.txt \n ");

    while( fscanf(filePtr, "%i", &id) != EOF )
    {
      fscanf(filePtr, "%i", &id);
      fgets(name, 100, filePtr );
      fscanf(filePtr, "%f", &n);

      printf("\n Leitura de dados: ");
      printf("\n Matricula: %i", id);
      printf("\n Nome: %s", name);
      printf(" Nota: %.2f \n\n", n);
    }

   fclose(filePtr);
   return  0;
}
```

Exercício:

a) Criar um vetor **a** com a quantidade de elementos definida pelo usuário usando alocação didinâmica de memória.

Fazer a leitura de dados e preencher o vetor **a**.

Selecionar somente os elementos pares desse vetor e gravar esses elementos em um arquivo de texto chamado **out.txt**

b) Ler os elementos do vetor do arquivo **out.txt** (criado no item **a**) e calcular o valor médio desses elementos.