

Tema: Introdução à programação III

Atividade: Arquivos em C

01.) Editar e salvar um esboço de programa em C, cujo nome será Exemplo0700.c, para guardar dados em arquivo.

```
/**
 * writelnts - Gravar em arquivo texto certa quantidade de valores.
 * @param fileName - nome do arquivo
 * @param x - quantidade de valores
 */
void writelnts ( chars fileName, int x )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "wt" );
    int y = 0;

    // repetir para a quantidade de dados
    for ( y = 1; y <= x; y = y + 1 )
    {
        // gravar valor
        fprintf ( arquivo, "%d\n", y );
    } // end for

    // fechar arquivo (INDISPENSÁVEL para gravacao)
    fclose ( arquivo );
} // end writelnts ( )

/**
 * Method_01 - Mostrar certa quantidade de valores.
 */
void method_01 ( )
{
    // identificar
    IO_id ( "Method_01 - v0.0" );

    // executar o metodo auxiliar
    writelnts ( "DADOS1.TXT", 10 );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_01 ( )
```

02.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

Em caso de dúvidas, consultar a apostila, recorrer aos monitores ou apresentá-las ao professor.

03.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

- 04.) Acrescentar outro método para ler e mostrar os dados gravados.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
    readInts - Ler de arquivo texto certa quantidade de valores.
    @param fileName - nome do arquivo
    @param x - quantidade de valores
*/
void readInts ( chars fileName )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "rt" );
    int x = 0;

    // tentar ler o primeiro
    fscanf ( arquivo, "%d", &x );
    // repetir enquanto houver dados
    while ( ! feof ( arquivo ) )
    {
        // mostrar valor
        printf ( "%d\n", x );
        // tentar ler o proximo
        fscanf ( arquivo, "%d", &x );
    } // end while

    // fechar arquivo (RECOMENDAVEL para leitura)
    fclose ( arquivo );
} // end readInts ( )

/**
    Method_02.
*/
void method_02 ( )
{
    // identificar
    IO_id ( "Method_02 - v0.0" );

    // executar o metodo auxiliar
    readInts ( "DADOS1.TXT" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_02 ( )
```

OBS.:

Todo o conteúdo será lido como texto, sem distinções.

- 05.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 06.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

07.) Acrescentar outro método para gravar dados reais.

Na parte principal, incluir a chamada do método para testar o novo.

```
/**
    writeDoubles - Gravar em arquivo texto certa quantidade de valores.
    @param fileName - nome do arquivo
    @param x - quantidade de valores
*/
void writeDoubles ( chars fileName, int x )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "wt" );
    int y = 0;

    // gravar quantidade de valores
    IO_printf ( arquivo, "%d\n", x );
    // repetir para a quantidade de dados
    for ( y = 1; y <= x; y = y + 1 )
    {
        // gravar valor
        IO_printf ( arquivo, "%lf\n", (0.1*y) );
    } // end for

    // fechar arquivo (INDISPENSÁVEL para gravacao)
    fclose ( arquivo );
} // end writeDoubles ( )

/**
    Method_03.
*/
void method_03 ( )
{
    // identificar
    IO_id ( "Method_03 - v0.0" );

    // executar o metodo auxiliar
    writeDoubles ( "DADOS2.TXT", 10 );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_03 ( )
```

OBS.:

Observar a necessidade de incluir a mudança de linha.

08.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

09.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

- 10.) Acrescentar outro método para ler valores reais.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
    readDoubles - Ler de arquivo texto certa quantidade de valores.
    @param fileName - nome do arquivo
    @param x - quantidade de valores
*/
void readDoubles ( chars fileName )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "rt" );
    int    x =  0 ;
    int    y =  1 ;
    double z = 0.0;

    // tentar ler a quantidade de dados
    fscanf ( arquivo, "%d", &x );
    // repetir enquanto houver dados e
    // quantidade nao tiver sido alcancada
    while ( ! feof ( arquivo ) && y <= x )
    {
        // tentar ler
        fscanf ( arquivo, "%lf", &z );
        // mostrar valor
        printf ( "%2d: %lf\n", y, z );
        // passar ao proximo
        y = y + 1;
    } // end while

    // fechar arquivo (RECOMENDAVEL para leitura)
    fclose ( arquivo );
} // end readDoubles ( )

/**
    Method_04.
*/
void method_04 ( )
{
    // identificar
    IO_id ( "Method_04 - v0.0" );

    // executar o metodo auxiliar
    readDoubles ( "DADOS2.TXT" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_04 ( )
```

- 11.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 12.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

- 13.) Acrescentar outro método para gravar texto em arquivo.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
    writeText - Gravar em arquivo texto certa quantidade de valores.
    @param fileName - nome do arquivo
    @param x - quantidade de valores
*/
void writeText ( chars fileName )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "wt" );
    chars linha   = IO_new_chars ( STR_SIZE );

    // repetir ate' desejar parar
    IO_println ( "Gravar linhas (para terminar, entrar com \"PARAR\"):\\n" );
    do
    {
        // ler do teclado
        strcpy ( linha, IO_readln ( "" ) );
        // gravar valor
        IO_fprintf ( arquivo, \"%s\\n\", linha );
    }
    while ( strcmp ( "PARAR", linha ) != 0 );

    // fechar arquivo (INDISPENSÁVEL para gravacao)
    fclose ( arquivo );
} // end writeText ( )

/**
    Method_05.
*/
void method_05 ( )
{
    // identificar
    IO_id ( "Method_05 - v0.0" );

    // executar o metodo auxiliar
    writeText ( "DADOS3.TXT" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_05 ( )
```

OBS.:
Observar a comparação de cadeias de caracteres.

- 14.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 15.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

- 16.) Acrescentar outro método para ler texto de arquivo.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
    readText - Ler de arquivo texto certa quantidade de valores.
    @param fileName - nome do arquivo
*/
void readText ( chars fileName )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "rt" );
    chars linha = IO_new_chars ( STR_SIZE );

    // tentar ler o primeiro
    strcpy ( linha, IO_freadln ( arquivo ) );
    // repetir enquanto houver dados
    while ( ! feof (arquivo) &&
            strcmp ( "PARAR", linha ) != 0 )
    {
        // mostrar valor
        printf ( "%s\n", linha );
        // tentar ler o proximo
        strcpy ( linha, IO_freadln ( arquivo ) );
    } // end while

    // fechar arquivo (RECOMENDAVEL para leitura)
    fclose ( arquivo );
} // end readText ( )

/**
    Method_06.
*/
void method_06 ( )
{
    // identificar
    IO_id ( "Method_06 - v0.0" );

    // executar o metodo auxiliar
    readText ( "DADOS3.TXT" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_06 ( )
```

- 17.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 18.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

- 19.) Acrescentar um método para copiar dados em arquivo.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
 copyText - Copiar arquivo texto.
 @param fileOut - nome do arquivo de saida (destino)
 @param fileIn - nome do arquivo de entrada (origem)
 */
void copyText ( chars fileOut, chars fileIn )
{
 // definir dados
 FILE* saida = fopen ( fileOut, "wt" );
 FILE* entrada = fopen ( fileIn , "rt" );
 chars linha = IO_new_chars ( STR_SIZE );
 int contador = 0;

 // ler da origem
 strcpy ( linha, IO_freadln ( entrada ) );
 // repetir enquanto houver dados
 while ( ! feof ( entrada ) )
 {
 // contar linha lida
 contador = contador + 1;

 // gravar no destino,
 // EXCEPCIONALMENTE sem a ultima linha, nesse caso
 if ( strcmp ( "PARAR", linha ) != 0 )
 {
 IO_fprintln ( saida, linha );
 } // end if

 // ler da origem
 strcpy ( linha, IO_freadln ( entrada ) );
 } // end while

 // informar total de linhas copiadas
 IO_printf ( "Lines read = %d\n", contador );

 // fechar arquivos
 fclose ( saida );
 fclose ( entrada );
} // end copyText ( )

/**
 Method_07.
 */
void method_07 ( )
{
 // identificar
 IO_id ( "Method_07 - v0.0" );

 // executar o metodo auxiliar
 copyText ( "DADOS4.TXT", "DADOS3.TXT" );

 // encerrar
 IO_pause ( "Apertar ENTER para continuar" );
} // end method_07 ( )
```

NOTA:

Deve-se observar a ordem dos parâmetros, pois o primeiro nome indicará o arquivo que será gravado. Se o arquivo já existir, isso resultará na perda de dados.

Para evitar problemas de perda de dados, recomenda-se verificar a existência do arquivo, antes de tentar abri-lo para gravação.

20.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

21.) Executar o programa.

Observar as saídas.

Registrar os dados e os resultados.

22.) Acrescentar um método para adicionar texto ao arquivo.

Na parte principal, incluir a chamada do método para testar o novo.

```
/**
 * appendText - Gravar em arquivo texto certa quantidade de valores.
 * @param fileName - nome do arquivo
 * @param x - quantidade de valores
 */
void appendText ( chars fileName )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "at" );
    chars linha = IO_new_chars ( STR_SIZE );

    // repetir ate' desejar parar
    IO_printf ( "Gravar linhas (para terminar, entrar com \"PARAR\"): \n" );
    do
    {
        // ler do teclado
        strcpy ( linha, IO_readln ( "" ) );
        // gravar valor
        IO_fprintln ( arquivo, linha );
    }
    while ( strcmp ( "PARAR", linha ) != 0 );

    // fechar arquivo (INDISPENSÁVEL para gravacao)
    fclose ( arquivo );
} // end appendText ( )

/**
 * Method_08.
 */
void method_08 ( )
{
    // identificar
    IO_id ( "EXEMPLO0710 - Method_08 - v0.0" );

    // executar o metodo auxiliar
    appendText ( "DADOS4.TXT" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_08 ( )
```


- 23.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
- 24.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.
- 25.) Acrescentar um método para ler palavra em arquivo, uma por vez.
Na parte principal, incluir a chamada do método para testar o novo.

```
/**
  readWords - Ler palavras de arquivo.
  @param fileName - nome do arquivo
 */
void readWords ( chars fileName )
{
  // definir dados
  FILE* arquivo = fopen ( fileName, "rt" );
  chars linha = IO_new_chars ( STR_SIZE );

  // tentar ler a primeira
  strcpy ( linha, IO_fread ( arquivo ) );
  // repetir enquanto houver dados
  while ( ! feof (arquivo) &&
          strcmp ( "PARAR", linha ) != 0 )
  {
    // mostrar valor
    printf ( "%s\n", linha );
    // tentar ler o proximo
    strcpy ( linha, IO_fread ( arquivo ) );
  } // end while

  // fechar arquivo (RECOMENDAVEL para leitura)
  fclose ( arquivo );
} // end readWords ( )

/**
  Method_09.
 */
void method_09 ( )
{
  // identificar
  IO_id ( "Method_09 - v0.0" );

  // executar o metodo auxiliar
  readWords ( "DADOS4.TXT" );

  // encerrar
  IO_pause ( "Apertar ENTER para continuar" );
} // end method_09 ( )
```

- 26.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

27.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

28.) Acrescentar uma função para procurar palavra em arquivo, uma por vez.
Na parte principal, incluir a chamada do método para testar a função.

```
/**
    searchWord - Procurar palavra em arquivo.
    @return true, se encontrar; false, caso contrario
    @param fileName - nome do arquivo
    @param word - palavra a procurar
*/
bool searchWord ( chars fileName, chars word )
{
    // definir dados
    FILE* arquivo = fopen ( fileName, "rt" );
    chars linha = IO_new_chars ( STR_SIZE );

    // tentar ler a primeira
    strcpy ( linha, IO_fread ( arquivo ) );
    // repetir enquanto houver dados
    while ( ! feof (arquivo) &&
            strcmp ( word, linha ) != 0 )
    {
        // tentar ler o proximo
        strcpy ( linha, IO_fread ( arquivo ) );
    } // end while

    // fechar arquivo (RECOMENDAVEL para leitura)
    fclose ( arquivo );

    // retornar resultado
    return ( strcmp ( word, linha ) == 0 );
} // end ifarchdWord ( )

/**
    Method_10.
*/
void method_10 ( )
{
    // identificar
    IO_id ( "Method_10 - v0.0" );

    // procurar palavra
    IO_printf ( "Procurar ("%s") = %d\n", "pqr", searchWord ( "DADOS4.TXT", "pqr" ) );
    IO_printf ( "Procurar ("%s") = %d\n", "pqs", searchWord ( "DADOS4.TXT", "pqs" ) );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // end method_10 ( )
```

OBS.:

Observar a necessidade de verificar a existência de dado antes de testá-lo.

29.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

30.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

Exercícios:

DICAS GERAIS: Consultar o Anexo C 02 na apostila para outros exemplos.

Prever, realizar e registrar todos os testes efetuados.

Integrar as chamadas de todos os programas em um só.

Restrições:

- As repetições sobre cadeias de caracteres deverão usar o tamanho.
- Testes deverão usar expressões lógicas e usar funções próprias em lugar das disponíveis em bibliotecas nativas.

- 01.) Incluir um método (0711) para
ler um valor inteiro do teclado e
gravar essa quantidade em múltiplos de 3, pares, em ordem crescente, começando em 3.

Exemplo: $n = 5 \Rightarrow \{ 3, 9, 15, 21, 27 \}$

- 02.) Incluir um método (0712) para
ler um valor inteiro do teclado e
gravar essa quantidade em múltiplos de 12, pares, em ordem decrescente encerrando em 24.

Exemplo: $n = 5 \Rightarrow \{ 72, 60, 48, 36, 24 \}$

- 03.) Incluir um método (0713) para
ler um valor inteiro do teclado e
gravar essa quantidade em valores da sequência: 1 4 16 64 256 ...

Exemplo: $n = 5 \Rightarrow \{ 1, 4, 16, 64, 256 \}$

- 04.) Incluir um método (0714) para
ler um valor inteiro do teclado e
gravar essa quantidade em valores decrescentes da sequência: ... 1/64 1/16 1/8 1/4 1.

Exemplo: $n = 5 \Rightarrow \{ 1/64, 1/16, 1/8, 1/4, 1 \}$

- 05.) Incluir um método (0715) para
ler um valor inteiro do teclado (n) e outro valor real (x),
gravar essa quantidade (n) em valores reais da sequência: 1 $1/x^2$ $1/x^4$ $1/x^6$...
DICA: Usar **pow (x, y)** da biblioteca <math.h> para calcular a potência.

Exemplo: $n = 5 \Rightarrow \{ 1, 1/x^2, 1/x^4, 1/x^6, 1/x^8 \}$

- 06.) Incluir um método e uma função (0716) para
ler um valor inteiro do teclado para representar certa quantidade de valores
a serem somados dentre os primeiros gravados no exercício anterior (0715).
Testar essa função para quantidades diferentes.
Gravar em outro arquivo ("RESULTADO06.TXT") cada quantidade e seu resultado.

07.) Incluir um método e uma função (Exemplo0717) para ler um valor inteiro do teclado, e até essa quantidade, um valor por vez, calcular a soma dos inversos das potências do exercício (0714). Gravar em outro arquivo ("RESULTADO07.TXT") cada quantidade e seu resultado.

Exemplo: $n = 5 \Rightarrow \{ 1/64 + 1/16 + 1/8 + 1/4 + 1 \}$

08.) Incluir um método e uma função (0718) para ler um valor inteiro do teclado, e até atingir essa quantidade, um valor por vez, gravar o valor correspondente aos primeiros termos pares da série de Fibonacci. Gravar em outro arquivo ("RESULTADO08.TXT") cada quantidade e seu resultado. DICA: Separar o gerador de Fibonacci do mapeamento de pares.

Exemplo: $n = 5 \Rightarrow \{ 2, 8, 34, 144, 610 \}$

09.) Incluir um método e uma função (0719) para para calcular a quantidade de minúsculas em cadeia de caracteres de um arquivo texto. Gravar em outro arquivo ("RESULTADO09.TXT") cada cadeia de caracteres e seus resultados. Testar essa função com cadeias de tamanhos diferentes.

Exemplo: PaReDe de TiJoLoS AmaREIOs

10.) Incluir um método e uma função (0720) para para contar dígitos maiores ou iguais a 4 em uma cadeia de caracteres. Gravar em outro arquivo ("RESULTADO10.TXT") cada cadeia de caracteres e seu resultado. Testar essa função para cadeias de tamanhos diferentes.

Exemplo: P4R3D3 de T1J0L05 4maR3105

Tarefas extras:

E1.) Incluir um método (07E1) para programa ler um valor inteiro do teclado, e gravar em arquivo apenas seus divisores ímpares em ordem decrescente.

E2.) Incluir um método e uma função (07E2) para ler palavras de um arquivo, uma por linha, e contar quantas começam com a letra 'f' (ou 'F') e tenham, no máximo, 15 caracteres.