PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 07

Preparação

Vídeos recomendados:
Antes de iniciar as atividades, recomenda-se assistir aos seguintes vídeos:

https://www.youtube.com/watch?v=LNu-0bzxpos&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=66
https://www.youtube.com/watch?v=ueg-IE8cZH4&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=67
https://www.youtube.com/watch?v=uYymG_oUPeY&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=68
https://www.youtube.com/watch?v=X6BcBhRCR8M&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=69
https://www.youtube.com/watch?v=FwW2T3jGvdg&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=70
https://www.youtube.com/watch?v=xN61MLUgkSg&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=72
https://www.youtube.com/watch?v=GDVPYnD-T_w&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=74
https://www.youtube.com/watch?v=4WlsKHHVda0&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=77
https://www.youtube.com/watch?v=jnotzdaKjOI&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=78

Tema: Introdução à programação III
Atividade: Arquivos em C

01.) Editar e salvar um esboço de programa em C, cujo nome será Exemplo0701.c, para guardar dados em arquivo.

```
/**
  writeInts - Gravar em arquivo texto certa quantidade de valores.
  @param fileName - nome do arquivo
  @param x - quantidade de valores
 */
void writeInts ( 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 );
   } // fim repetir

 // fechar arquivo (INDISPENSAVEL para gravacao)
   fclose ( arquivo );
} // fim writeInts ( )
```

```
/**
  Method01 - Mostrar certa quantidade de valores.
 */
void method01 ( )
{
 // identificar
   IO_id ( "EXEMPLO0702 - Method01 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method01 ( )
```

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.) Copiar a versão atual do programa para outra nova – Exemplo0702.c.

05.) Editar mudanças no nome do programa e versão.
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 );
   } // fim repetir

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

```
/**
  Method02.
 */
void method02 ( )
{
 // identificar
   IO_id ( "EXEMPLO0702 - Method02 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method02 ( )
```

OBS.:
Todo o conteúdo será lido como texto, sem distinções.

06.) 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.

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

08.) Copiar a versão atual do programa para outra nova – Exemplo0703.c.

09.) Editar mudanças no nome do programa e versão.
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_fprintf ( arquivo, "%d\n", x );
 // repetir para a quantidade de dados
   for ( y = 1; y <= x; y = y + 1 )
   {
    // gravar valor
      IO_fprintf ( arquivo, "%lf\n", (0.1*y) );
   } // fim repetir

 // fechar arquivo (INDISPENSAVEL para gravacao)
   fclose ( arquivo );
} // fim writeDoubles ( )
```

```
/**
  Method03.
 */
void method03 ( )
{
 // identificar
   IO_id ( "EXEMPLO0710 - Method03 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method03 ( )
```

OBS.:
Observar a necessidade de incluir a mudança de linha.

10.) 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.

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

12.) Copiar a versão atual do programa para outra nova – Exemplo0704.c.

13.) Editar mudanças no nome do programa e versão.
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;
   } // fim repetir
```

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

/**
   Method04.
 */
void method04 ( )
{
 // identificar
    IO_id ( "EXEMPLO0710 - Method04 - v0.0" );

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

 // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method04 ( )
```

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.) Copiar a versão atual do programa para outra nova – Exemplo0705.c.

17.) Editar mudanças no nome do programa e versão.
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 (INDISPENSAVEL para gravacao)
    fclose ( arquivo );
} // fim writeText ( )
```

```
/**
  Method05.
 */
void method05 ( )
{
 // identificar
   IO_id ( "EXEMPLO0710 - Method05 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method05 ( )
```

OBS.:
Observar a comparação de cadeias de caracteres.

18.) 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.

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

20.) Copiar a versão atual do programa para outra nova – Exemplo0706.c.

21.) Editar mudanças no nome do programa e versão.
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 ) );
   } // fim repetir

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

```
/**
  Method06.
 */
void method06 ( )
{
 // identificar
    IO_id ( "EXEMPLO0710 - Method06 - v0.0" );

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

 // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method06 ( )
```

22.) 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.

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

24.) Copiar a versão atual do programa para outra nova – Exemplo0707.c.

25.) Editar mudanças no nome do programa e versão.
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 );
      } // fim se

    // ler da origem
      strcpy ( linha, IO_freadln ( entrada ) );
   } // fim repetir

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

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

/**
  Method07.
 */
void method07 ( )
{
 // identificar
   IO_id ( "EXEMPLO0707 - Method07 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method07 ( )
```

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.) Copiar a versão atual do programa para outra nova – Exemplo0708.c.

29.) Editar mudanças no nome do programa e versão.
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_println ( "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 (INDISPENSAVEL para gravacao)
   fclose ( arquivo );
} // fim appendText ( )

/**
  Method08.
 */
void method08 ( )
{
 // identificar
   IO_id ( "EXEMPLO0710 - Method08 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method08 ( )
```

30.) 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.

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

32.) Copiar a versão atual do programa para outra nova – Exemplo0709.c.

33.) Editar mudanças no nome do programa e versão.
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 ) );
   } // fim repetir

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

/**
  Method09.
 */
void method09 ( )
{
 // identificar
   IO_id ( "EXEMPLO0710 - Method09 - v0.0" );

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

 // encerrar
   IO_pause ( "Apertar ENTER para continuar" );
} // fim method09 ( )
```

34.) 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.

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

36.) Copiar a versão atual do programa para outra nova – Exemplo0710.c.

37.) Editar mudanças no nome do programa e versão.
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 ) );
   } // fim repetir

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

 // retornar resultado
   return ( strcmp ( word, linha ) == 0 );
} // fim searchdWord ( )

/**
  Method10.
 */
void method10 ( )
{
 // identificar
   IO_id ( "EXEMPLO0710 - Method10 - 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" );
} // fim method10 ( )
```

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

38.) 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.

39.) 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.

01.) Incluir em um programa (Exemplo0711) um método para
ler um valor inteiro do teclado e
gravar essa quantidade em múltiplos de 3, pares, em ordem crescente, começando em 6.

02.) Incluir em um programa (Exemplo0712) um método para
ler um valor inteiro do teclado e
gravar essa quantidade em múltiplos de 3, ímpares, em ordem decrescente encerrando em 3.

03.) Incluir em um programa (Exemplo0713) um método para
ler um valor inteiro do teclado e
gravar essa quantidade em valores da sequência: 1 5 15 25 35 ...

04.) Incluir em um programa (Exemplo0714) um método para
ler um valor inteiro do teclado e
gravar essa quantidade em valores decrescentes da sequência: ... 1/343 1/49 1/7 1.

05.) Incluir em um programa (Exemplo0715) um método 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 $1/x^3$ $1/x^5$ ...
DICA: Usar **pow (x, y)** da biblioteca <math.h> para calcular a potência.

06.) Incluir em um programa (Exemplo0716) uma função para
ler um valor inteiro do teclado para representar certa quantidade de parcelas
a serem somadas dentre os primeiros valores gravados no exercício anterior.
Testar essa função para quantidades diferentes.
Gravar em outro arquivo ("RESULTADO06.TXT") cada quantidade e seu resultado.

07.) Incluir em um programa (Exemplo0717) uma função para
ler um valor inteiro do teclado para representar uma quantidade e
calcular a soma de tantos valores quanto os primeiros em valores gravados como
inversos das potências de 7 do exercício 04 acima.
Gravar em outro arquivo ("RESULTADO07.TXT") cada quantidade e seu resultado.

08.) Incluir em um programa (Exemplo0718) uma função para
ler um valor inteiro do teclado para representar uma quantidade e
gravar tantos valores quanto os correspondentes aos primeiros termos pares da série de Fibonacci.
Gravar em outro arquivo ("RESULTADO08.TXT") cada quantidade e seu resultado.

09.) Incluir em um programa (Exemplo0719) uma função para
para calcular a quantidade de minúsculas, menores que 'N', em cadeia de caracteres
de um arquivo texto, cujo nome será fornecido como parâmetro.
Gravar em outro arquivo ("RESULTADO09.TXT") cada cadeia de caracteres e seu resultado.

10.) Incluir em um programa (Exemplo0720) uma função para
para contar dígitos maiores que 5 em certa cadeia de caracteres passada como parâmetro.
Ler um arquivo texto que possa conter números e letras, cujo nome deverá ser fornecido e
gravar em outro arquivo ("RESULTADO10.TXT") cada cadeia de caracteres e o resultado
da aplicação da função.

Tarefas extras:

E1.) Incluir em um programa (Exemplo07E1) um método para
programa ler um valor inteiro do teclado, e
gravar em arquivo os seus divisores em ordem crescente.

E2.) Incluir em um programa (Exemplo07E2) uma função para
ler palavras de um arquivo, uma por linha, e
contar quantas começam com a letra 'a' (ou 'A') e que não terminem em 'a' (ou 'A').