Computação 2

Aula 8

Arquivos

Profª. Fabiany
fabianyl@utfpr.edu.br

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

E/S com Arquivos

- A linguagem C não possui nenhum comando de E/S.
- Todas as operações de E/S ocorrem mediante chamadas a funções de biblioteca C padrão.
- Essa abordagem faz o sistema de arquivos do C ser flexível.
- O sistema de E/S do C permitem que os dados possam ser transferidos na sua representação binária interna ou em formato de texto legível.

O ponteiro de Arquivo

- O ponteiro é o meio comum que une o sistema C ANSI de E/S.
- Um ponteiro de arquivo é um ponteiro para informações que definem várias coisas sobre o arquivo, incluindo seu nome, status e a posição atual do arquivo.
- O ponteiro de arquivo especifica um arquivo específico em disco e é usado para direcionar as operações das funções de E/S.
- Para obter uma variável ponteiro de arquivo, usa-se:

```
FILE *nome_ponteiro;
```

Funções mais comuns para sistemas de arquivos

Nome	Função
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
putc()	Escreve um caractere no arquivo
fputc()	O mesmo que putc()
getc()	Lê um caractere do arquivo
fgetc()	O mesmo que getc()
fseek()	Posiciona o arquivo em um byte específico
fprintf()	É para o arquivo o que printf() é para o console
fscanf()	É para o arquivo o que scanf() é para o console
feof()	Devolve verdadeiro se o fim do arquivo foi atingido
ferror()	Devolve verdadeiro se ocorreu um erro
rewind()	Recoloca o indicador de posição do arquivo no início do arquivo
remove()	Remove um aquivo

Abrindo um arquivo

- A função fopen() abre um arquivo e retorna um ponteiro de arquivo associado a esse arquivo.

FILE *fopen (const char* nome_arquivo, const char* modo);

- Exemplo:

```
FILE *fp;
fp = fopen ( "teste.txt", "w" );
   if ( fp == NULL)  // ou ((fp = fopen(“teste.txt”,”w”) ) == NULL)
   {
      printf ( "O arquivo não pode ser aberto. \n" );
      system ( "Pause" );
      exit (1);
   }
```

Modos de utilização de arquivos	
Modo	**Significado**
r	Abre um arquivo texto para leitura.
w	Cria um arquivo texto para escrita.
a	Anexa a um arquivo-texto.
rb	Abre um arquivo binário para leitura.
wb	Cria um arquivo binário para escrita.
ab	Anexa um arquivo binário.
r+	Abre um arquivo-texto para leitura/escrita.
w+	Cria um arquivo-texto para leitura/escrita.
a+	Anexa ou cria um arquivo-texto para leitura/escrita.
r+b	Abre um arquivo binário para leitura/escrita.
w+b	Cria um arquivo binário para leitura/escrita.
a+b	Anexa a um arquivo binário para leitura/escrita.

Funções de arquivos

- A função **fputc()** escreve caracteres em um arquivo que foi previamente aberto para escrita por meio da função **fopen().** O prototipo para essa função é:

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

onde fp é um ponteiro de arquivo devolvido por **fopen()** e ch é o caractere escrito. Se a operação **putc()** foi bem sucedida, ela devolverá o caractere escrito. Caso contrário, ela devolve **EOF**.

Funções de arquivos

- A função **fgetc()** lê caracteres de um arquivo aberto no modo leitura no **fopen()**. O protótipo é:

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

onde fp é um ponteiro de arquivo do tipo **FILE** devolvido por **fopen().** A função **getc()** devolve **EOF** quando o final do arquivo for alcançado.

Exemplo fputc()

fopen (“nome**.**txt**", "**w**");**

fputc (caracter**,** arquivo**);**

Exemplo 1

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fp;
     char ch;

    fp = fopen ( "teste.txt", "w" );

    if ( NULL == fp )
    {
       printf ( "O arquivo não pode ser aberto. \n" );
       system ( "Pause" );
       exit (1);
    }

    printf ( " Programa para gravar caracteres em um arquivo chamado teste.txt." );
    printf ( " Digite $ para fechar o arquivo." );

    printf ( "Digite os caracteres: \n" );

    do
    {
       ch = getchar();
       fputc ( ch, fp );
    } while ( ch != '$' );

    fclose ( fp );

    system ("Pause");
    return 0;
}
```

Exemplo fgetc()

fopen (“nome**.**txt**", "**r**");**

fgetc (arquivo **);**

Exemplo 2

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   FILE *fp;
   char ch;
   char* nome;

   printf ( "Informe o nome do arquivo a ser lido: \n“ );
   gets ( nome );

   // nome = strcat(nome,".txt");
   fp = fopen ( nome, "r" );

   if (  NULL == fp)
   {
      printf ( "O arquivo não pode ser aberto. \n" );
      system ( "Pause" );
      exit (1);
   }

   ch = fgetc ( fp );    //a função getc() devolve EOF quando o final do arquivo for alcançado

   while ( ch != EOF )
   {
       putchar( ch );  /* Imprime na tela */
       ch = fgetc ( fp );
   }

   system ("Pause");

   return 0;
}
```

Exemplo 3

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   FILE *fp;
   char ch;

   fp = fopen ( "c:/documentos/arquivos/teste.txt", "w" );

   if ( fp == NULL )
   {
      printf ( "O arquivo não pode ser aberto. \n" );
      system ( "Pause" );
      exit (1);
   }

   printf ( "Programa para gravar caracteres em um arquivo chamado teste.txt." );
   printf ( "\n \n" );

   printf ( "Digite os caracteres: \n" );
   do
   {
      ch = getchar();
      fputc ( ch, fp );
   } while ( ch != '\n' );

   fclose ( fp );

   system ( "Pause" );
   return 0;

}
```

Exemplo 4

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   FILE *fp;
   char ch;
   char *nome;

   printf ( "Qual é o nome do arquivo! \n" );
   gets ( nome );

   fp = fopen ( nome, "w" );

   if ( fp == NULL )
   {
      printf ( "O arquivo não pode ser aberto. \n" );
      system ("Pause");
      exit (1);
   }

   printf ("Digite os caracteres: \n" );
   do
   {
      ch = getchar();
      fputc ( ch, fp );
   } while ( ch != '\n' );

   fclose ( fp );

   system ( "Pause" );
   return 0;
}
```

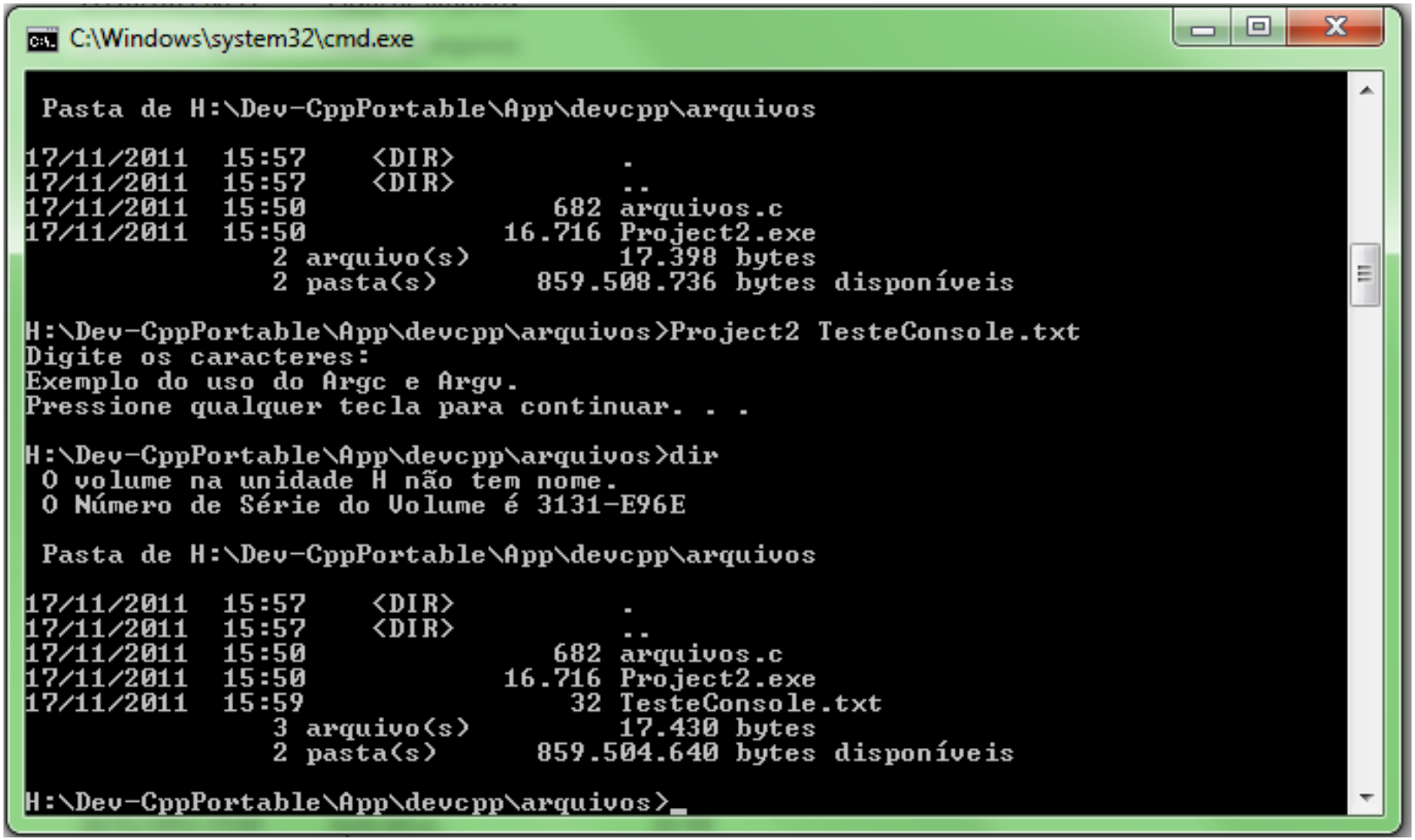
Arquivos

Utilizando *argc* e *argv*.

Exemplo 5

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
   FILE *fp;
   char ch;

   if ( argc != 2 )  {
      printf ( "Você esqueceu de digitar o nome do arquivo! \n" );
      system ("Pause");
      exit (1);
   }

   fp = fopen ( argv[1], "w" );

   if ( fp == NULL )  {
      printf ( "O arquivo não pode ser aberto. \n" );
      system ( "Pause" );
      exit (1);
   }

   printf ( "Digite os caracteres: \n" );
   do
   {
      ch = getchar();
      fputc ( ch, fp );
   } while (ch != '\n');

   fclose ( fp );

   system ("Pause");
   return 0;
}
```

Exemplo 5


C:\Windows\system32\cmd.exe
Pasta de H:\Dev-CppPortable\App\devcpp\arquivos
17/11/2011 15:57 <DIR> .
17/11/2011 15:57 <DIR> ..
17/11/2011 15:50 682 arquivos.c
17/11/2011 15:50 16.716 Project2.exe
2 arquivo(s) 17.398 bytes
2 pasta(s) 859.508.736 bytes disponíveis
H:\Dev-CppPortable\App\devcpp\arquivos>Project2 TesteConsole.txt
Digite os caracteres:
Exemplo do uso do Argc e Argv.
Pressione qualquer tecla para continuar. . .
H:\Dev-CppPortable\App\devcpp\arquivos>dir
O volume na unidade H não tem nome.
O Número de Série do Volume é 3131-E96E
Pasta de H:\Dev-CppPortable\App\devcpp\arquivos
17/11/2011 15:57 <DIR> .
17/11/2011 15:57 <DIR> ..
17/11/2011 15:50 682 arquivos.c
17/11/2011 15:50 16.716 Project2.exe
17/11/2011 15:59 32 TesteConsole.txt
3 arquivo(s) 17.430 bytes
2 pasta(s) 859.504.640 bytes disponíveis
H:\Dev-CppPortable\App\devcpp\arquivos>

Funções de arquivos

- A função fputs() opera como puts(), mas escreve a string no arquivo especificado;

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

- A função fgets() lê uma string do arquivo até que um caractere de nova linha seja lido ou que length-1 caracteres sejam lidos.

```
char *fgets(chat *str, int length, FILE *fp);
```

Exemplo fputs() e fgets()

fputs (caracter**,** arquivo**);**

fgets (str**,** 179**,** arquivo**);**

Exemplo 7

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char str [180];
   FILE *fp;

   if ( ( fp = fopen ("TEST2.TXT", "w+")  ) == NULL ) {
      printf ( "O arquivo nao pode ser aberto. \n" );
      system ( "Pause" );
      exit (1);
   }

   // Gravar arquivo
   printf ( "Digite uma ou mais linhas (linha em branco para sair) \n" );
   do {
      gets (str);
      strcat ( str, "\n");
      fputs ( str, fp );
   } while ( *str != '\n' );

   // Ler arquivo
   rewind ( fp );

   while ( !feof ( fp ) )  {
         fgets ( str, 179, fp );
         puts ( str );
   }
   fclose ( fp );
   system ( "Pause" );
   return 0;
}
```

Exemplo 8

```c
/* copia um arquivo */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   FILE *entrada, *saida;

   char ch;
   char nomearq1[300], nomearq2[300];

   printf ( "Programa para copiar arquivos. \n" );

   // Arquivo fonte
   printf ("Digite o nome do arquivo fonte: \n");
   gets (nomearq1);

   entrada = fopen (nomearq1, "w+b");

   if ( entrada == NULL )
   {
      printf ( "O arquivo nao pode ser aberto (1).\n" );
      system ("Pause");
      exit (1);
   }

   printf( "Digite o conteudo do arquivo-fonte. \n" );
   do
   {
      ch = getchar();
      fputc ( ch, entrada );
   }while (ch != '\n' );

   // Reposiciona o indicador de posição no inicio no arquivo.
   rewind ( entrada );

   // Arquivo destino
   printf ("Digite o nome do arquivo destino: \n");
   gets ( nomearq2 );

   saida = fopen ( nomearq2, "w+b");
   if ( saida == NULL )
   {
      printf ("O arquivo-destino nao pode ser aberto.");
      system("Pause");
      exit(1);
   }

   // Esse codigo copia de fato o arquivo.

   while ( !feof ( entrada ) )
   {
         ch = fgetc (entrada) ;
         if ( !feof ( entrada ) )
         {
            fputc (ch, saida);
         }
   }

   fclose (entrada);
   fclose (saida);

return 0;
}
```

Funções para escrever mais de um byte em um arquivo

- Para ler e escrever tipos de dados maiores que um byte, o sistema de arquivo C ANSI fornece duas funções: fread() e fwrite().
- Essas funções permitem a leitura e a escrita de blocos de qualquer tipo de dados.
- Seus protótipos:

size_t **fread**(void *buffer, size_t num_bytes, size_t count, FILE *fp);

size_t **fwrite**(const void *buffer, size_t num_bytes, size_t count, FILE *fp);

Funções para escrever mais de um byte em um arquivo

- Para **fread(),** *buffer* é um ponteiro para uma região de memória que receberá os dados do arquivo.
- Para **fwrite(),** *buffer* é um ponteiro para as informações quer serão escritas no arquivo.
- O número de bytes a ler ou escrever é especificado por *num_bytes*. O argumento count determina quantos itens (cada um de comprimento *num_bytes*) serão lidos ou escritos.
- A função **fread()** devolve o número de itens lidos. Esse valor poderá ser menor que *count* se o final do arquivo for atingido ou ocorrer um erro. A função **fwrite()** devolve o número de itens escritos. Esse valor será igual a *count* a menos que ocorra um erro.

Exemplos

fwrite() & fread()

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  FILE*   fp;
  double  d = 12.23;
  int     i = 101;
  long    l = 123023;

  fp = fopen ( "TesteBin", "wb+");

  if ( fp == NULL )  {
     printf ( "O arquivo nao pode ser aberto. \n" );
     system ( "Pause" );  exit (1);
  }

  fwrite ( &d, sizeof ( double ), 1, fp) ;
  fwrite ( &i,  sizeof ( int ),        1, fp) ;
  fwrite ( &l,  sizeof ( long ),     1, fp) ;

  rewind ( fp );

  fread ( &d, sizeof ( double ), 1, fp);
  fread ( &i,  sizeof ( int ),        1, fp);
  fread ( &l,  sizeof ( long ),     1, fp);

  printf ("%f %d %ld \n", d, i, l);
  fclose ( fp );


  return 0;
}
```

Exemplo

Lista Postal

```c
/* Um programa de lista postal muito simples */

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

#define TAM 2

struct Elemento
{

      char nome [100];
      char rua [100];
      char cidade [100];
      char estado [2];
      char cep [10];

};

struct Elemento  Lista [TAM];

char menu ();

void inicia_lista ();

void cadastra ();

void mostra ();

void salva ();

void carrega ();
```

```c
int main()
{
   char escolha;
   inicia_lista();

   for ( ;; )
   {
      escolha = menu();
      switch (escolha)
      {
         case 'c':
         case 'C':   { cadastra();  }  break;

         case 'm':
         case 'M':   { mostra();    } break;

         case 's':
         case 'S':   { salva();       } break;

         case 'a':
         case 'A':  { carrega();    } break;

         case 't':
         case 'T':  { exit (0 );       } break;

         default : { printf ( "Opcao invalida. \n" );  }

      printf ( "\n \n \n" );
   }
   system ( "Pause" );
   return 0;
}
```

```c
/* Um programa de lista postal muito simples */

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

#define TAM 2

struct Elemento
{

      char nome [100];
      char rua [100];
      char cidade [100];
      char estado [2];
      char cep [10];

} Lista [TAM];

char menu ();

void inicia_lista ();

void cadastra ();

void mostra ();

void salva ();

void carrega ();
```

```c
int main()
{
   char escolha;
   inicia_lista();

   for ( ;; )
   {
      escolha = menu();
      switch (escolha)
      {
         case 'c':
         case 'C':   { cadastra();  }  break;

         case 'm':
         case 'M':   { mostra();    } break;

         case 's':
         case 'S':   { salva();       } break;

         case 'a':
         case 'A':  { carrega();    } break;

         case 't':
         case 'T':  { exit (0 );       } break;

         default : { printf ( "Opcao invalida. \n" );  }

      printf ( "\n \n \n" );
   }
   system ( "Pause" );
   return 0;
}
```

```
char menu()
{
    printf ("\n \n \n");
    char opcao;

    printf ( " (C)adastrar.    \n"  );
    printf ( " (M)ostrar.      \n"  );
    printf ( " C(A)arregar.    \n"  );
    printf ( " (S)alvar.       \n"   );
    printf ( " (T)erminar.     \n"  );

    fflush ( stdin );
    scanf ( "%c", &opcao );

    return opcao;
}
```

```
void mostra()
{
    printf ("\n \n \n");

    register int t;

    for( t = 0; t < TAM; t++ )
    {
        if (*Lista[t].nome )
        {
            printf ( "%s \n", Lista[t].nome);
            printf ( "%s \n", Lista[t].rua);
            printf ( "%s \n", Lista[t].cidade);
            printf ( "%s \n", Lista[t].estado);
            printf ( "%s \n", Lista[t].cep);
        }
        printf ("\n");
    }
}
```

```
void inicia_lista()
{
    register int t;
    for (t = 0; t < TAM; t++)
    {
        *Lista[t].nome = '\0';      // Lista[t].nome[0] = '\0';
    }
}
```

```
void cadastra ()
{
    printf ("\n \n \n");

    for ( int i = 0; i < TAM; i++ )
    {
        printf ( "Nome: \n" );
        fflush ( stdin );
        gets ( Lista[i].nome );

        printf ( " Rua: \n" );
        fflush ( stdin );
        gets ( Lista[i].rua );

        printf ( "Cidade: \n" );
        fflush (stdin);
        gets ( Lista[i].cidade );

        printf ( "Estado: \n" );
        fflush ( stdin );
        gets ( Lista[i].estado );

        printf ( "CEP: \n" );
        fflush ( stdin );
        gets ( Lista[i].cep );
    }
}
```

```
void salva ()
{

    printf ("\n \n \n");

    FILE *fp;
    int result;

    fp = fopen ("cadastro", "wb");

    if ( fp == NULL )
    {

      printf ( "O arquivo nao pode ser aberto. \n" );
      return;

    }

    for ( int i = 0; i < TAM; i++ )
    {

       if ( *Lista[i].nome )
       {

         result = fwrite ( &Lista[i], sizeof ( struct Elemento ), 1, fp );

          if ( result != 1 )
          {
              printf ( "Erro de escrita no arquivo. \n" );
          }

       }

    }

    fclose (fp);

}
```

```
void carrega ()
{

    printf ("\n \n \n");

    FILE  *fp;
    int resp;

    fp = fopen ( "cadastro", "rb“ );

    if ( fp == NULL )
    {

      printf ( "O arquivo nao pode ser aberto. \n" );
      return;

    }

    inicia_lista ();
    for ( int i = 0; i < TAM; i++ )
    {

       resp = fread ( &Lista[i], sizeof (struct Elemento), 1, fp );

        if ( resp != 1 )
        {

          if ( feof ( fp ) )
          {
            break;
          }
          printf ( " Erro de leitura no arquivo. \n" );
        }

    }
    fclose ( fp );

}
```

Funções fprintf() e fscanf()

- Essas funções comportam-se exatamento como **printf()** e **scanf()** exceto por operarem em arquivos
- Seus protótipos:

int **fprintf**(FILE *fp, const char * control_ string,...);

int **fscanf**(FILE *fp, const char * control_ string,...);

onde fp é um ponteiro de arquivo devolvido por uma chamada **fopen()**. **fprintf()** e **fscanf()** direcionam suas operações de E/S para o arquivo apontado por *fp.*

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  FILE*   fp;
  char  s[80];
  int    t;

  if  ((fp = fopen ( “teste”, “w“)) == NULL)  {
     printf ( "O arquivo nao pode ser aberto. \n" );
     exit (1);
  }

 printf (“ Digite uma string e um numero :  ”);
 fscanf(stdin, “%s%d”, s, &t);   //lê do teclado

 fprintf(fp, “%s %d”, s, t);   //escreve no arquivo
 fclose ( fp );

 if  ((fp = fopen ( “teste”, “r”)) == NULL)  {
     printf ( "O arquivo nao pode ser aberto. \n" );
     exit (1);
  }

 fscanf(fp, “%s%d”, s, &t);   //lê do arquivo
 fprintf(stdout, “%s %d”, s, t);   //escreve na tela


  return 0;
}
```

Exercícios

1) Faça uma agenda de telefones que permita cadastrar as seguintes informações: nome, endereço, aniversário, email e telefone.
   - ✓ Na agenda é permitido cadastrar uma quantidade determinada de pessoas e fazer qualquer alteração necessária.
   - ✓ As opções devem ser feitas através de um menu: Cadastrar, Mostrar dados, Alterar Dados, Salvar no arquivo e Recuperar do arquivo.

Referências Bibliográficas

- Baseado nos slides do Professor Jean Simão disponível em: http://www.pessoal.utfpr.edu.br/jeansimao/Fundamentos1/Fundamentos1.htm

- **SHILDT**, H. C, Completo e Total, 3a edição, rev. e atual. Ed. Makron. São Paulo, c1997.