Estruturas de Dados e Algoritmos 1

Disciplina 193704

Prof. Mateus Mendelson
mendelson@unb.br

Universidade de Brasília
Faculdade do Gama
Engenharia de Software

Arquivos

1. Arquivos

- Os comandos de entrada e saída que usamos até o momento foram:

  - ✓ printf – mostra dados formatados no vídeo; e

  - ✓ scanf – lê dados formatados digitados do teclado.

- Todavia, dados podem ser lidos e gravados em arquivos.

- A cada arquivo está associado um nome, pelo qual o mesmo é conhecido externamente, isto é, o nome que consta no diretório do disco.

- Uma vez que um arquivo é uma sequência de *bytes* temos o marcador do final desse arquivo que é: EOF (EndOfFile)

1. Arquivos

- Vamos tratar de dois tipos de arquivos:

  ✓ **TEXTO**, onde são gravados caracteres e pode ser editado por um editor de texto.

  ✓ **BINÁRIO**, onde são gravados dados como estão na memória. Por exemplo, uma variável inteira é gravada com 4 *bytes* com o conteúdo exato que está na memória.

- Para tratar de arquivos, a linguagem C fornece um nível de abstração entre o programador e o dispositivo que está sendo acessado para gravação e leitura:

```
FILE *fp;
```

- Ou seja, um ponteiro *fp*, do tipo FILE.

1. Arquivos

• Para realizar a abertura de um arquivo para leitura ou gravação, temos a função ***fopen*** (file Open), que possui dois parâmetros:

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

- ✓ **filename**: o nome do arquivo (string); e
- ✓ **mode:** modo de abertura do arquivo.

1. Arquivos

- Modos possíveis:

  ✓ **r**: abre um arquivo **TEXTO** para **leitura.**
  ✓ **w**: cria um arquivo **TEXTO** para **gravação**, ou se o arquivo já existe, elimina seu conteúdo e recomeça a gravação a partir do seu início.
  ✓ **a**: abre um arquivo **TEXTO** já existente para gravação, a partir de seu final.

  ✓ **rb**: abre um arquivo **BINÁRIO** para **leitura.**
  ✓ **wb**: cria um arquivo **BINÁRIO** para **gravação**, ou se o arquivo já existe, elimina seu conteúdo e recomeça a gravação a partir do seu início.
  ✓ **ab**: abre um arquivo **BINÁRIO** já existente para gravação, a partir de seu final.

1. Arquivos

- Modos possíveis:

  ✓ **r+**: abre um arquivo **TEXTO** para **leitura e gravação**. O arquivo deve existir e pode ser modificado.
  ✓ **w+**: cria um arquivo **TEXTO** para **leitura e gravação**. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
  ✓ **a+**: abre um arquivo **TEXTO** para **leitura e gravação**. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso do arquivo não existir.

1. Arquivos

- Modos possíveis:

  ✓ **r+b**: abre um arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "r+" acima, só que o arquivo é binário.
  ✓ **w+b**: cria um arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "w+" acima, só que o arquivo é binário.
  ✓ **a+b**: acrescenta dados ou cria uma arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "a+" acima, só que o arquivo é binário.

1. Arquivos

- Exemplo:

```
FILE *fp;
char nomeArquivo[] = "c:\arquivo.txt";

fp = fopen(nomeArquivo, "w");
```

- Caso a função ***fopen*** não encontre o arquivo indicado, ela retorna NULL.

```
FILE *fp;

fp = fopen("teste.txt","w");
if (fp==NULL) {
    printf("Falha.\n");
    exit(1);
}
```

1. Arquivos

• Da mesma forma que abrimos um arquivo utilizando a função *fopen*, devemos fechá-lo quando não formos mais utilizá-lo, pois assim realmente garantimos que o arquivo será salvo em disco, e não ficará simplesmente no buffer (região de memória).

• Para isso utilizamos o comando ***fclose***, da seguinte forma:

```
fclose(fp);
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ Para fazermos a leitura e gravação, podemos utilizar as funções ***fscanf*** e ***fprintf***, respectivamente.

- Por exemplo, se queremos ler números inteiros de um arquivo:

```
int numero;
fp = fopen("teste.txt", "r");
fscanf(fp, "%d", &numero);
```

- Para gravar no arquivo:

```
int numero = 10;
fp = fopen("teste.txt", "w");
fprintf(fp, "%d", numero);
```

1. Arquivos

- Exemplo gravação TEXTO:

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

int main (int argc, char *argv[])
{
    FILE *fp;
    int numero;

    if (argc!=2) {
        printf("You forgot to enter the filename.\n");
        exit(1);
    }

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

    if(fp==NULL) {
        printf("Cannot open file.\n");
        exit(2);
    }
```

1. Arquivos

- Exemplo gravação TEXTO:

```
    do {
        printf("Digite um numero inteiro positivo:");
        scanf("%d", &numero);
        if(numero != -1) fprintf(fp, "%d\n", numero);
    } while (numero != -1);

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

1. Arquivos

- Exemplo gravação TEXTO:

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

int main (int argc, char *argv[])
{
   FILE * pFile;
   int n;
   char name [100];

   pFile = fopen ("myfile.txt","w");
   for (n=0 ; n<3 ; n++)
   {
     puts (“Please, enter a name: ");
     gets (name);
     fprintf (pFile, "Name %d [%-10.2s]\n",n,name);
   }
   fclose (pFile);

   return 0;
}
```

1. Arquivos

- Exemplo leitura TEXTO:

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

int main (int argc, char *argv[])
{
    FILE *fp;
    int numero;

    if (argc!=2) {
        printf("You forgot to enter the filename.\n");
        exit(1);
    }

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

    if(fp==NULL) {
        printf("Cannot open file.\n");
        exit(2);
    }
```

1. Arquivos

- Exemplo leitura TEXTO:

```
    fscanf (fp, "%d", &numero);

    while (!feof(fp)) {
        printf("%d\n", numero); /* print on screen */
        fscanf(fp, "%d", &numero);
    }

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

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ putc( )

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

int main(int argc, char *argv[])
{
    FILE *fp;
    char ch;

    if(argc!=2) {
       printf("Faltou o nome do arquivo. \n");
       exit(1);
    }
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ putc( )

```
if((fp=fopen(argv[1], "w"))==NULL) {
    printf("Cannot open file.\n");
    exit(1);
}

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

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

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ getc( )

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

int main(int argc, char *argv[])
{
    FILE *fp;
    char ch;

    if(argc!=2) {
      printf("Faltou o nome do arquivo. \n");
      exit(1);
    }
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ getc( )

```
if((fp=fopen(argv[1], "r"))==NULL) {
    printf("Cannot open file.\n");
    exit(1);
}
ch = getc(fp); /* read one character */
while (!feof(fp)) {
    putchar(ch); /* print on screen */
    ch = getc(fp);
}
fclose(fp);
return 0;
}
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ fputs( )

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

int main(int argc, char *argv[])
{
    char str[80];
    FILE *fp;

    if((fp = fopen(argv[1], "w"))==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ fputs( )

```
do {
    printf("Digite string (ENTER => fim):\n");
        gets(str);
        strcat(str, "\n"); /* add a newline */
        fputs(str, fp);
    } while(*str!='\n');

    return 0;
}
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ fgets( ) e rewind()

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

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

if((fp = fopen(argv[1], "w+"))==NULL) {
    printf("Cannot open file.\n");
    exit(1);
}
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:

  ✓ fgets( ) e rewind()

```
do {
    printf("Enter a string (CR to quit):\n");
    gets(str);
    strcat(str, "\n");
    fputs(str, fp);
} while(*str!='\n');

rewind(fp);

while(!feof(fp)) {
    fgets(str, 80, fp);
    printf("%s", str);
}
```

1. Arquivos

- Trabalhando com arquivos de TEXTO:
  - ✓ fgets( ) e rewind()

```
fclose(fp);

return 0;

}
```

1. Arquivos

- Trabalhando com arquivos BINÁRIOS:

  ✓ fwrite()

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

int main ()
{
  FILE * pFile;
  int buffer[] = {1,2,3};

  pFile = fopen ("myfile.bin" , "wb");
  fwrite (buffer , sizeof(int), 3, pFile );

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

1. Arquivos

- Trabalhando com arquivos BINÁRIOS:

  ✓ fread()

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

int main(void)
{
     FILE *fp;
     double d = 12.23;
     int i = 101;

if((fp=fopen("test", "wb+"))==NULL) {
     printf("Cannot open file.\n");
     exit(1);
}

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

1. Arquivos

- Trabalhando com arquivos BINÁRIOS:

  ✓ fread()

```
rewind(fp);

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

printf("%.2f %d \n", d, i);

fclose(fp);

return 0;

}
```