Universidade de Brasília – FGA
Orientador: Giovanni Almeida Santos
Alunos: André Mateus, Cleiton Gomes e Felipe Costa

ARQUIVOS

ARQUIVOS

Arquivo é uma unidade lógica que possui a função de armazenar dados em disco ou em qualquer outro dispositivo externo de armazenamento. Em outras palavras, arquivo é estrutura organizada de registros localizada em algum dispositivo de armazenamento de dados, que em geral é o disco rígido do próprio computador.

É extremamente importante o estudo de arquivos, pois a programação ensinada até aqui permiti-nos apenas manipular dados sem gravá-los, de tal modo que quando encerrada a execução do programa estes mesmos são “perdidos”. O conteúdo a seguir permitirá ao aluno armazenar dados em arquivos, assim como importar tais dados e manipulá-los.

FUNÇÕES BÁSICAS DE ARQUIVOS

fopen()	Abre um STREAM (arquivo)
fclose()	fecha um STREAM (arquivo)
putc()	Escreve um caracter
getc()	Lê um caracter
fseek()	Procura por um byte específico
fprintf()	Igual ao printf() do console
fscanf()	Igual ao scanf() do console
feof()	Retorna verdadeiro se encontrou fim arquivo
ferror()	Retorna verdadeiro se ocorreu erro
fread()	le um bloco de dados
fwrite	escreve um bloco de dados
rewind()	reposiciona o lacalizador no inicio do arquivo
remove()	apaga um arquivo

Na abertura de arquivos (fopen), temos o modo em que o arquivo é aberto. Na parte de importação e exportação de arquivos desse documento, esse assunto será entendido melhor.

"r"	abre um arquivo para leitura
"w"	cria uma arquivo para escrita
"a"	acrescenta dados a um arquivo já existente
"rb"	abre um arquivo binário para leitura
"wb"	cria um arquivo binário para escrita
"ab"	acrescenta dados a um arquivo binário existente
"r+"	abre um arquivo para leitura/escrita
"w+"	cria um arquivo para leitura/escrita
"a+"	acrescenta dados/cria um arquivo para leitura/escrita
"r+b"	abre um arquivo binário para leitura/escrita
“w+b"	cria um arquivo binário para leitura/escrita
"a+b"	acrescenta ou cria um arquivo binário para leitura/escrita
"rt"	abre um arquivo texto para leitura
"wt"	cria um arquivo texto para leitura

"at"	acrescenta dados a um arquivo texto
"r+t"	abre um arquivo texto para leitura/escrita
"w+t"	cria um arquivo texto para leitura/escrita
"a+t"	acrescenta ou cria arq. Texto para leit/escrita

O PONTEIRO DE ARQUIVO

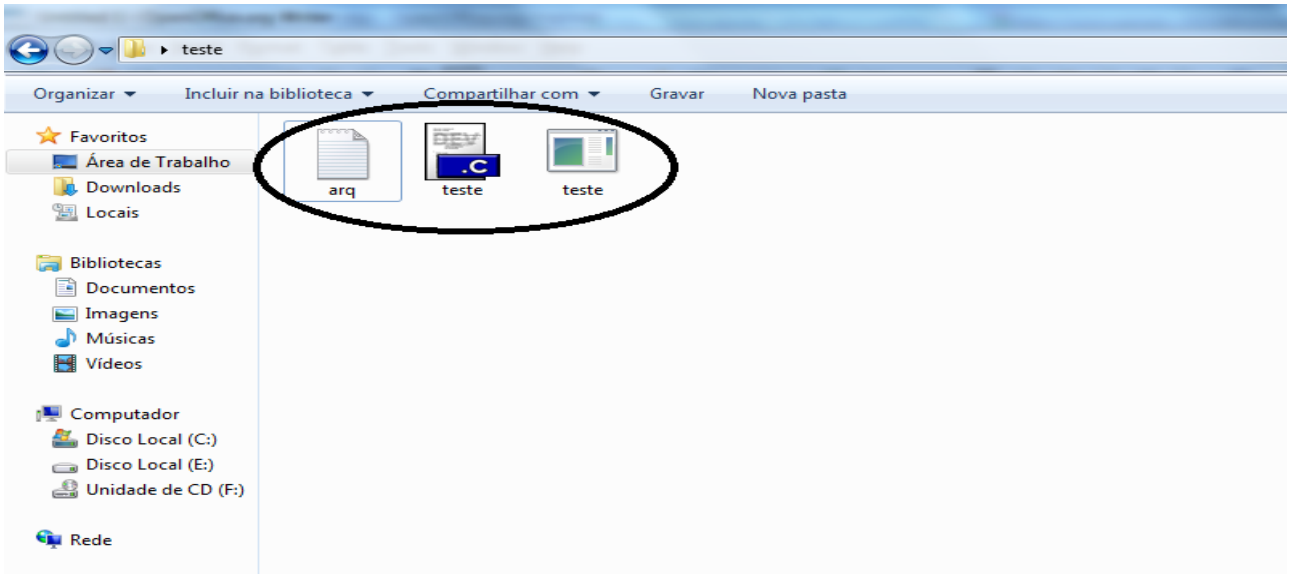
Um ponteiro de arquivo é uma variável ponteiro do tipo FILE. Essa variável é definida no arquivo de cabeçalho stdio.h, mas isso depende do compilador que o usuário estiver usando. Para ler ou escrever em arquivos seu programa precisa usar os ponteiros de arquivo. Para declarar uma variável como ponteiro de arquivo use a seguinte sintaxe:

```
FILE *arquivo;
```

IMPORTANTO ARQUIVOS

Se o arquivo ainda não existir na memória secundária, ele deve ser criado. Caso o arquivo já existir (pelo fato de ter sido criado em uma execução anterior do programa) ele pode ser aberto para que novos dados sejam acrescentados ou para que os dados guardados nele possam ser lidos. Os arquivos podem ser de dois tipos: tipo texto e tipo binário. A escolha se um arquivo é texto ou binário depende do tipo de sua aplicação. Se você for armazenar textos para serem lidos em outro lugar (como um editor de texto ou um relatório), você deve utilizar um arquivo do tipo texto. Se você for armazenar dados sobre alguma pessoa ou objeto (tipo registro ou estrutura), você deve utilizar um arquivo binário, pois, além de ocupar menos espaço no armazenamento das informações, ele as protege um pouco de outros programas bisbilhoteiros.

Para importar um arquivo é necessário primeiro que ele se encontre no mesmo diretório



(local) do código fonte. A figura 01 mostra o código fonte e o executável, denominado de teste e o txt chamado arq, que estão na mesma pasta. Também, de acordo com o exemplo da figura 01, deve-se lembrar que importar um arquivo é importar as informações contidas no documento arq.

Figura 01 : Arquivos no mesmo diretório.

Os principais comandos usados para manipulação de dados importados através de arquivos estão na biblioteca *stdio.h.* Para manipular os dados de um arquivo na linguagem c, você deve primeiro abri-lo. Para isso temos o comando fopen que abre um arquivo.

fopen("A.txt", "B"); → onde o primeiro parâmetro (A) é o nome do arquivo seguido da extensão e o segundo parâmetro (B) é a forma de abertura do arquivo, como por exemplo: "rt" para abri-lo e ler seu conteúdo ou "wt" para abri-lo e gravar dados. Esta função retorna um ponteiro para arquivo.

Para leitura de arquivos texto pode-se usar a função *fgets* ou *fscanf*. A função fgets lê uma linha inteira de uma vez. A função fcanf funciona como a função scanf, porém, ao invés de ler os dados de teclado, estes dados são lidos de arquivo.

fgets(char X, int num , FILE* Y)* → o primeiro parâmetro (X) é um ponteiro para uma string na qual será gravada a informação disponível no arquivo Y. O segundo parâmetro delimita o número de caracteres que serão lidos do arquivo. O terceiro parâmetro (Y) é um ponteiro para um arquivo. Caso nenhum dado seja gravado na string, ou seja, não haja dados no arquivo, a função retornará NULL (nulo). O parâmetro do meio (100) é o tamanho da string. *O* 'fgets' lê até 99 caracteres ou até o '\n'.

fscanf(FILE* Z, %k, tipo *W) → o primeiro parâmetro (Z) é um ponteiro para o arquivo que será lido. O segundo parâmetro (%k) é o formato da informação (exemplos: %d → int, %f → float,%s → char). O terceiro parâmetro (W) é um ponteiro que aponte para onde se deve armazenar a informação lida do arquivo. Esta função retorna a constante EOF caso nada seja lido.

Segue abaixo a Figura 02 que ilustra um exemplo de importação de dados de um arquivo.

```c
//Bibliotecas
#include <stdio.h>
#include <stdlib.h>

main()
{
    char teste[90]; // Variável local
    FILE *arquivo; // Ponteiro do  arquivo

    // instrução para abrir o arquivo
    arquivo=fopen("arq.txt","rt");

    //Condição se o arquivo tiver nulo
    if(arquivo == NULL){
      printf("\nNao foi possivel abrir o arquivo\n");
      printf("\nPressione qualquer tecla para encerrar o programa\n");
      getch(); exit(0);
    }

    //recebendo o arquivo
    fgets(teste,90,arquivo);
    puts(teste); // Impressão da variável teste
    getchar(); // pausa do sistema
}
```

Figura 02: Exemplo de código de importação de dados de um arquivo.

EXPORTANDO ARQUIVOS

Exportar arquivos nada mais é que guardar dados no arquivo para que esses dados possam ou não ser utilizados no futuro. Isso é muito interessante, pois o usuário do programa pode guardar algum dado, desligar o computador e quando for utilizar novamente o programa, os dados armazenados no disco rígido através do arquivo continuam lá. Para gravação de arquivos texto usam-se as funções fputs e fprintf.

fputs(mensagem, arquivo) → O primeiro parâmetro são os caracteres que podem ser digitados pelo usuário no executável e o segundo parâmetro indica que os caracteres serão guardados no arquivo.

fprintf(arquivo,"Linha %d\n",i) → O primeiro parâmetro é o ponteiro do arquivo e o segundo parâmetro é o índice em que esta sendo gravado o caráter. Já que temos um índice, deve-se imaginar que tal instrução deva ficar dentro de um laço.

Para ficar mais claro como funciona a exportação de arquivos, a figura 03 mostra a exportação utilizando fprintf e a figura 04 mostra a exportação utilizando fputs e fprintf.

```c
//Bibliotecas
#include <stdio.h>
#include <conio.h>
#define LACOS 10

int main()
{
  FILE *arquivo;
  int i;
  int resultado;

  // Cria um arquivo texto para gravação
  arquivo = fopen("arq.txt", "wt");

  //Laço para percorrer cada caracter
  for (i = 0; i< LACOS;i++) {
      resultado = fprintf(arquivo,"Linha %d\n",i);
  }
  fclose(arquivo); //fecha o arquivo
  getch(); // pausar o sistema
}
```

Figura 03 – gravando arquivos usando a função fprintf.

```c
//Bibliotecas
#include<stdio.h>
#include <string.h>
//struct armazena
struct{
    char nome[100];
    int idade;
    float peso;
    char sexo[10];
}armazena;

main(){
    FILE *arquivo; //Ponteiro do arquivo

    //Abrindo o arquivo para gravação
    arquivo = fopen("arq.txt" , "wt");

    //Função de cópia: destino < origem
    strcpy(armazena.nome, "Débora");
    strcpy(armazena.sexo, "\nF");
    //Solicitando e armazenando dados para outras variáveis
    printf("\nDebora, digite sua idade e seu peso\n");
    scanf("%d %f" , &armazena.idade, &armazena.peso);

    //Instruções para guardar as informações no arquivo
    fprintf(arquivo, "%d\n", armazena.idade);
    fprintf(arquivo, "%f\n", armazena.peso);
    fputs(armazena.nome,arquivo);
    fputs(armazena.sexo,arquivo);

    fclose(arquivo); //Instrução para fechar o arquivo

    getch(); //Pausar o sistema
}
```

Figura 04 – gravando arquivos usando a função fputs.

MANIPULANDO ARQUIVOS COM STRUCT

A manipulação de arquivos utilizando struct permite que o programador deixe o código mais eficiente e mais fácil de entender. Sendo assim é extremamente viável utilizar arquivos e struct para elaborar pequenos e grandes projetos utilizando a linguagem c. Da mesma forma que os arquivos recebem e fornecem informação do executável da maneira convencional, com struct não é diferente e muito mais fácil de ser feito.

A função *fseek*(FILE *arquivo, numbytes, int origm) . Para se fazer procuras e acessos randômicos em arquivos usa-se a função **fseek()**. Esta move a posição corrente de leitura ou escrita no arquivo de um valor especificado, a partir de um ponto especificado. O parâmetro *origem* determina a partir de onde os *numbytes* de movimentação serão contados. Para o parâmetro origem existem na linguagem c as constantes SEEK_SET(começo do SEEK_CUR(um ponto corrente no arquivo) e SEEK_END(no final do arquivo).

A última figura desse documento mostra o uso da função fseek() junto com a manipulação de struct.

```c
//Bibliotecas
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TAM_MAX 100
//struct tipo pessoa
typedef struct
{
    char nome[TAM_MAX];
    float idade;
    float peso;
} PESSOA;


void gravaEmArquivo(PESSOA candidato){ // função que escreve dados em arquivo
    FILE *arquivo; // ponteiro para arquivo

    arquivo = fopen("arq.txt","r+b"); //abrindo arquivo
    if(arquivo){ // Caso fopen retorne 0 , o arquivo não pode ser aberto
        fseek(arquivo,0,SEEK_END); // posicionando ponteiro no final do arquivo
        fwrite(&candidato,sizeof(candidato),1,arquivo); //escrevendo toda a struct
        printf("Dados armazenados com sucesso!");]
        getch(); // para execução, esperando que seja digitado um caractere
        fclose(arquivo); // fechando o arquivo
    }
    else
        printf("Erro ao abrir arquivo!!!\n");
}
void procuraEmArquivo(){ // função que busca dados do arquivo
    char candidatoProcurado[TAM_MAX];
    PESSOA candidato;

   arquivo = fopen("arq.txt","r+b");
   if(arquivo){
       printf("Digite o nome do candidato que deseja procurar:\n");
       gets(candidatoProcurado);

       fseek(arquivo,0,SEEK_SET); //posicionando ponteiro no começo do arquivo

       while(!feof(arquivo)){ // a função feof retorna 0 quando o ponteiro atinge o final do arquivo
           fread(&candidato,sizeof(candidato),1,arquivo); // Lê toda struct do arquivo.
           // A função strstr retorna NULL quando as strings candidatoProcurado e candidato.nome são diferentes
           if(strstr(candidatoProcurado,candidato.nome)!=NULL){
               printf("Nome ->");
               puts(candidato.nome);
               printf("Idade ->%.2f\n",candidato.idade);
               printf("Peso ->%.2f\n",candidato.peso);
               getch();
           }
       }
       fclose(arquivo);
   }
   else
       printf("Erro ao abrir arquivo!!!\n");

}
int  main()
{
    PESSOA candidato;
    int aux;
    printf("Digite :\n");
    printf("1 - > Inserir candidato;\n");
    printf("2 - > procurar candidato;\n");
    scanf("%d",&aux);
    fflush(stdin);
    switch(aux){
        case 1:
            printf("Digite o nome:\n");
            gets(candidato.nome);
            printf("Digite a idade\n");
            scanf("%f",&candidato.idade);
            printf("Digite o peso:\n");
            scanf("%f",&candidato.peso);
            gravaEmArquivo(candidato); break;
        case 2:
                procuraEmArquivo();break;
        default :
            printf("Opcao invalida!!!");
    }
    return 0;
}
```