Introdução a Programação

Manipulando Arquivos em Modo Texto

Tópicos da Aula

- Hoje aprenderemos funções de manipulação de arquivos texto
  - Funções de leitura em modo texto
  - Funções de escrita em modo texto
  - Considerações sobre a utilização de operações de leitura/escrita em modo texto e binário
- Aprenderemos também algumas funções que servem para posicionar o cursor no arquivo
  - Exemplos de uso

Leitura (Modo Texto)

- A cada operação de leitura, os dados correspondentes são lidos e o cursor avança e aponta para o próximo dado
- Existe em C a função **`fscanf`** para leitura formatada de dados de um arquivo modo texto
  - Similar a **`scanf`**
  - Lê de um arquivo passado como parâmetro em vez de somente da entrada padrão (teclado)
  - Pode ler também da entrada padrão (arquivo stdin)

Leitura com `fscanf`

```
int fscanf(FILE* fp, char* formato,end_variaveis)
```

- Retorna número de dados lidos com sucesso
- **`fp`** é o ponteiro para o arquivo
- **`formato`** equivale aos códigos de formatação (iguais ao do **`scanf`**)
- **`end_variaveis`** corresponde a lista de endereços de variáveis que armazenarão os dados lidos do arquivo

Leitura com `fscanf`

- A cada operação de leitura, os dados correspondentes são convertidos de caracteres para o tipo (formato) especificado em **`fscanf`**
- Portanto, arquivo deve ter sido previamente gravado em modo texto
  - Se gravado em modo binário, a função tenta mapear os dados lidos para caracteres para depois fazer a conversão para o tipo especificado

Usando `fscanf` para Leitura (Modo Texto)

Lê uma string do arquivo apontado por `fp` de até 120 caracteres e armazena em `primeiraPalavra`

```
#include <stdio.h>
int main() {
   FILE *fp;
   char primeiraPalavra[121];
   fp = fopen("teste.txt","r");
   if  (fp != NULL) {
       fscanf(fp, "%120s",primeiraPalavra);
       printf("Primeira palavra do arquivo é:
       %s\n",primeiraPalavra);
       fclose(fp);
    } else {
        printf("Impossível abrir arquivo");
    }
    return 0;
    }
```

Constante EOF

- A constante (simbólica) **EOF** indica o fim de arquivo
  - Definida no arquivo stdio.h
- Ela **NÃO** é um caracter
  - É um inteiro que indica fim de arquivo
  - Geralmente um valor negativo
- **NÃO** é um valor presente no arquivo
- É um valor retornado por funções de leitura indicando fim de arquivo ou erro de leitura

Outras Funções de Leitura (Modo Texto)

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

- Lê um caractere de um arquivo
- Retorna o código do caractere lido
- Retorna EOF se fim do arquivo for alcançado

```
char* fgets(char* s, int n, FILE* fp);
```

- s é a cadeia de caracteres que armazenará o que for lido
- n é o número máximo de caracteres a serem lidos
- Lê uma seqüencia de caracteres até que '\n' seja encontrado ou que n caracteres tenham sido lidos
- Retorna ponteiro para a cadeia s

Usando fgetc para a Leitura

Programa conta número de caracteres e conta o número de linhas de um arquivo

```
#include <stdio.h>
int main() {
  FILE *fp; int c;int nCaracteres = 0, nLinhas = 0;
  fp = fopen("teste.txt","r");
  if   (fp != NULL) {
    while ((c = fgetc(fp)) != EOF) {
          if   (c == '\n') nLinhas++;
          else       nCaracteres++;
    }
   printf("Caracteres:%d,Linhas:%d",nCaracteres,nLinhas);
   fclose(fp);
  }else {
    printf("Impossível abrir arquivo");
  }
  return 0;
  }
```

Lê caractere e ainda verifica se chegou no fim do arquivo

Usando fgets para a Leitura

```
int main() {
  FILE *fp;
  int achou = linhaOcorrencia = 0;
  char palavra[121], linha [121];
  printf ("Digite a palavra:\n");
  scanf(" %120[^\n]",palavra);
  fp = fopen("teste.txt","r");
  if  (fp == NULL) {
     printf("Impossível abrir arquivo");exit(1);
  }
  while (fgets(linha,121,fp) != NULL && !achou) {
     linhaOcorrencia++;
     if  (strstr(linha,palavra) != NULL)
           achou = 1;
  }
  fclose(fp);
  printf("Ocorrência de %s = %d\n", palavra,linhaOcorrencia);
}
```

Programa informa linha da ocorrência de uma palavra em um arquivo com linhas (maximo 120 caracteres)

Lê uma linha e armazena em `linha`

Utiliza função `strstr` de `string.h` para saber se `palavra` é substring de `linha`

Escrita (Modo Texto)

- A cada operação de escrita, os dados são gravados na memória e posteriormente no disco, e o cursor avança apontando para a proxima posição do arquivo:
- Existe em C a função **`fprintf`** para escrita formatada de dados em um arquivo modo texto
  - Similar a **`printf`**
  - Escreve em um arquivo passado como parâmetro em vez de somente na saída padrão (monitor)
  - Pode escrever também na saída padrão (arquivo stdout)

Escrita com `fprintf`

```
int fprintf(FILE* fp, char* formato,variaveis)
```

- Retorna número de dados escritos com sucesso
- **`fp`** é o ponteiro para o arquivo
- **`formato`** equivale aos códigos de formatação (iguais ao do **`printf`**)
- **`variaveis`** corresponde a lista de variáveis, cujos conteúdos serão escritos no arquivo

Escrita com `fprintf`

- A cada operação de escrita, os dados correspondentes são convertidos do tipo (formato) especificado em **`fprintf`** para caracteres

Usando `fprintf` para Escrita (Modo Texto)

Programa escreve uma palavra de no máximo 120 caracteres em um arquivo

```
#include <stdio.h>
int main() {
    FILE *fp;
    char  palavra[121];
    fp = fopen("teste.txt","w");
    if  (fp == NULL) {
        printf("Impossível abrir arquivo");
        exit(1);
    }
   printf ("Digite a palavra:\n");
   scanf(" %120[^\n]",palavra);
   fprintf(fp,"%s\n",palavra);
   fclose(fp);
   return 0;
   }
```

Outras Funções de Escrita (Modo Texto)

```
int fputc(FILE* fp, char c);
```

- Escreve um caractere em um arquivo
- Retorna o código do caractere escrito

```
char* fputs(char* s,FILE* fp);
```

- s é a cadeia de caracteres que será escrita
- Retorna ponteiro para a cadeia s

Usando fputs e fputc para Escrita

```
int main() {
  FILE *e; FILE *s;
  int caractere; char nome_entrada[121];
  printf ("Digite o nome do arquivo de entrada:\n");
  scanf("%120s", nome_entrada);
  e = fopen(nome_entrada,"r");
  if  (e== NULL) {
      printf("Impossível abrir arquivo de entrada");exit(1);
  }
  s = fopen("arquivo_maiusculo","w");
  if  (s== NULL) {
      printf("Impossível abrir arquivo de saida");exit(1);
  }
  while ((caractere = fgetc(e)) != EOF)
      fputc(toupper(caractere),s);
  fputs("\nArquivo Convertido!",s);
  fclose(e); fclose(s);
  return 0;}
```

Programa lê um arquivo e gera outro com todas as letras convertidas para maiusculas

Usuário fornece o nome do arquivo

Converte caractere a caractere e escreve no arquivo

Operações de Leitura/Escrita (Modo Texto x Modo Binário)

- As operações de leitura/escrita em modo texto permitem uma visualização melhor do resultado
- Para operações que envolvem estruturas ou vetores, operações em modo binário são menos trabalhosas
- Considere a estrurura abaixo:

```
struct aluno {
      char nome [60];
      int mat;
      char sexo;
};
```

Usando Operações de Leitura/Escrita (Modo Texto)

Programa escreve e lê uma estrutura em/de um arquivo

```
int main() {
  FILE *arq;
  struct aluno aluno1, copia;
  strcpy(aluno1.nome,"Jose");
  aluno1.mat = 1;
  aluno1.sexo = 'M';
  arq = fopen("arquivoAluno1.txt","w+");
  if  (arq== NULL) {
     printf("Impossível abrir arquivo de entrada");exit(1);
  }
 fprintf(arq,"%s\n%d\n%c",aluno1.nome,aluno1.mat,aluno1.sexo);

 rewind(arq);

 fscanf(arq,"%s %d %c", copia.nome,&copia.mat,&copia.sexo);
 return 0;
 }
```

Escreve membro a membro da estrutura

Volta para inicio do arquivo

Lê membro a membro da estrutura

Usando Operações de Leitura/Escrita (Modo Binário)

Programa escreve e lê uma estrutura em/de um arquivo

```
int main() {
  FILE *arq;
  struct aluno aluno1, copia;
  strcpy(aluno1.nome,"Jose");
  aluno1.mat = 1;
  aluno1.sexo = 'M';
  arq = fopen("arquivoAluno1.bin","wb+");
  if  (arq== NULL) {
     printf("Impossível abrir arquivo de entrada");
     exit(1);
  }
  fwrite(&aluno1,sizeof(Aluno),1,arq);
  rewind(arq);
  fread(&copia,sizeof(Aluno),1,arq);
  return 0;
  }
```

Escreve toda a estrutura

Lê toda a estrutura

Funções Utilitárias

```
long ftell(FILE* fp);
```

- Retorna a posição atual do cursor do arquivo
- Corresponde a distancia em bytes em relação ao começo do arquivo

- Devem ser utilizadas com cautela em arquivos no modo texto, pois nem sempre o posicionamento do cursor vai ser o desejado
  - Certas plataformas podem colocar caracteres de formatação não visíveis que podem alterar o tamanho do arquivo (número de bytes)

Funções Utilitárias

```
long fseek(FILE* fp,long dist,int origem);
```

- Utilizada para posicionamento do cursor em um arquivo
- dist é o número de bytes em relação a origem
- origem é uma posição do cursor do arquivo em bytes (SEEK_CUR – posição corrente; SEEK_SET – início do arquivo; SEEK_END – final do arquivo)
- Retorna a nova posição do cursor

```
void rewind(FILE* fp);
```

- Utilizada para posicionamento do cursor no início do arquivo

Funções Utilitárias

Função que recupera o i-ésimo ponto armazenado em um arquivo

```
Ponto le_ponto (FILE* fp, int i) {
    Ponto p;
    fseek (fp, i*sizeof(Ponto), SEEK_SET);
    fread(&p, sizeof(Ponto), 1, fp);
    return p;
}
```

Função que retorna o tamanho do arquivo em bytes

```
int tamanho_arquivo (FILE *fp) {
     int tamanho;
     fseek (fp, 0, SEEK_END);
     tamanho =ftell (fp);
     return tamanho;
}
```

Resumindo ...

- Funções de leitura em modo texto
  - fscanf
  - fgets
  - fgetc
- Funções de escrita em modo texto
  - fprintf
  - fputs
  - fputc
- Quando utilizar operações de leitura/escrita em modo texto e binário
- Posicionamento dos cursores em arquivos