Métodos Computacionais

Arquivos

Arquivos

- Um arquivo representa um elemento de informação armazenado em memória secundária (disco)

- Características:
  - Informações são persistidas
  - Atribui-se nomes aos elementos de informação (arquivos e diretórios),em vez de endereços de memória
  - Acesso às informações são mais lentos

Persistência... pra quê?

- Não perder os dados no fim da execução de um programa
  - Memória temporária(volatil)
    - principal
    - Mais rápida e cara
  - Memória permanente
    - secundária
    - mais lenta e barata
- Para melhorar velocidade de acesso, a cada acesso, transfere-se trechos maiores do arquivo para espaços da memória (buffers)

Modos de Acesso a Arquivos em C

- Dois modos de acesso:
  - Texto e Binário

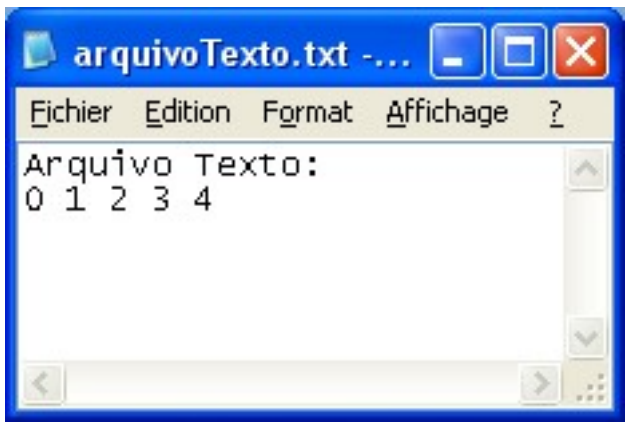
String

Inteiros

Informação persistida

```
Arquivo Texto:
0 1 2 3 4
```

```
Arquivo Binario:
0 1 2 3 4
```



Modo Texto

- É interpretado como uma seqüência de caracteres agrupadas em linhas
- Linhas são separadas por um caractere de nova linha

- Vantagens:
  - Pode ser lido facilmente por uma pessoa
  - Editado por editores de texto convencionais
- Desvantagens
  - Codificação dos caracteres pode variar (ASCII, UTF-8, ISSO-8859, etc)
  - Arquivos tendem a ser maiores (todas os dados são convertidos para caracteres)

Modo Binário

- Dados são armazenados da mesma forma que são armazenados na memória principal

- Vantagens:
  - Facilmente interpretados por programas
  - Maior velocidade de manipulação
  - Arquivos são, geralmente, mais compactos
- Desvantagens:
  - Difícil de serem entendidos por pessoas
  - Dependentes da máquina onde foram gerados

Operações em Arquivos

- Abertura
  - Sistema Operacional (SO) encontra arquivo pelo nome
  - Prepara buffer na memória
- Leitura
  - SO recupera trecho solicitado do arquivo
  - Parte ou todo trecho pode vir do buffer

Operações em Arquivos

- Escrita
  - SO altera conteúdo do arquivo
  - Alteração é feita primeiro no buffer para depois ser transferida para o disco
- Fechamento
  - Informação do buffer é atualizada no disco
  - Área do buffer utilizada na memória é liberada

Abertura de Arquivos em C

- Operações de manipulação de arquivos em C se encontram, geralmente, na stdio.h
- Função de Abertura

```
FILE* fopen(char* nome,char* modo);
```

  - **Nome**
    - Nome do arquivo
  - **FILE**
    - Tipo que representa uma abstração do arquivo
  - **modo**
    - **r** - Indica leitura
    - w – Indica escrita
    - **a** – Indica escrita ao final do existente
    - **t** – Indica modo texto
    - **b** – Indica modo binário

Abertura de Arquivos em C

- SO mantém um “cursor” que indica a posição de trabalho no arquivo
- Se modo w e arquivo já existe
  - Arquivo é destruído e um novo vazio é criado
- Se modo w ou a e arquivo não existe
  - Um novo arquivo é criado
- No modo r, o arquivo já deve existir
- Modos t e b podem ser combinados com os demais modos
- Se não for possível a abertura, a função retorna NULL

Exemplos de Abertura de Arquivos

```
FILE *fp;
fp = fopen("teste.txt", "rt");
```

Modo leitura e texto

```
fp = fopen("teste.bin", "wb");
```

Modo escrita e binário

```
fp = fopen("teste.txt", "r+");
```

Modo leitura e escrita em arquivo existente

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

Modo escrita e leitura em arquivo novo

Modos t e b podem ser omitidos

Fechamento de Arquivos

- Após leitura/escrita do arquivo, devemos fechá-lo
- Função de fechamento

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

  - Retorna 0 se o arquivo foi fechado com sucesso
  - Retorna constante **EOF** se houve algum erro

Leitura (Modo Texto)

- A cada operação de leitura, os dados correspondentes são transferidos para a memória e o cursor avança e aponta para o próximo dado
- Existe em C a função **`fscanf`** para ler 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

Usando fscanf para Leitura (Modo Texto)

```
#include <stdio.h>
int main() {
   FILE *fp;
   char primeiraPalavra[121];
   fp = fopen("teste.txt","r");
   if  (fp == NULL) {
     printf("Impossível abrir arquivo");
     exit(1);
    }
    fscanf(fp, "%120s",primeiraPalavra);
    printf("A primeira palavra do arquivo teste.txt
       é: %s\n",primeiraPalavra);
       fclose(fp);
    return 0;
    }
```

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

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; char c;int nCaracteres = 0, nLinhas = 0;
  fp = fopen("teste.txt","r");
  if  (fp == NULL) {
      printf("Impossível abrir arquivo");
      exit(1);
  }
  while ((c = fgetc(fp)) != EOF) {
      if  (c == '\n') nLinhas++;
      else            nCaracteres++;
  }
  printf("Caracteres:%d,Linhas:%d\n",nCaracteres,nLinhas);
  fclose(fp);
  return 0;
  }
```

Usando fgets para a Leitura

```
int main() {
  FILE *fp;
  int nOcorrencias = 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) {
     if  (strstr(linha,palavra) != NULL)
           nOcorrencias++;
  }
  fclose(fp);
  printf("Ocorrências de %s = %d\n", palavra,nOcorrencias);
  return 0;
}
```

Programa conta ocorrências de uma palavra em um arquivo com linhas de no 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 escrever 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)

Leitura 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

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

```
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(strcat(nome_entrada,"_maiuscula"),"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

Leitura (Modo Binário)

```
int fread(void* p,int tam,int nelem,FILE* fp);
```

- p é o endereço de memória em que vai ser armazenado o que for lido
- tam é o tamanho em bytes de cada elemento lido
- nelem é o número de elementos de tamanho tam lidos
- Retorna a quantidade de bytes lidos com sucesso (tam * nelem)

Escrita (Modo Binário)

```
int fwrite(void* p,int tam,int nelem,FILE* fp);
```

- p é o endereço de memória cujo conteúdo deseja-se salvar em arquivo
- tam é o tamanho em bytes de cada elemento escrito
- nelem é o número de elementos de tamanho tam escritos
- Retorna a quantidade de bytes escritos com sucesso (tam * nelem)

Usando fwrite na Escrita

```
#include <stdio.h>
 typedef struct ponto {
     float x,y;
 } Ponto;

 void salva (char* arquivo, int n, Ponto* vet) {
     FILE* fp = fopen(arquivo, "wb");
     if (fp == NULL) {
        printf("Erro na abertura do arquivo.\n");
        exit(1);
     }
     fwrite(vet, sizeof(Ponto), n, fp);
     fclose(fp);
 }
```

Número de pontos do vetor

Função que salva um vetor de pontos em um arquivo binário

Usando fread na Leitura

```
void carrega (char* arquivo, int n, Ponto* vet) {
   FILE* fp = fopen (arquivo, “rb”);
   if (fp == NULL) {
      printf(“Erro na abetura do arquivo.\n”);
      exit(1);
   }
   fread (vet, sizeof(Ponto), n, fp);
   fclose(fp);
}
```

Função que recupera um vetor de pontos de um arquivo binário

Usando as Funções Definidas Anteriormente

```
int main() {
  Ponto *entrada, *saida; int nPontos, cont,pos ;
  FILE *arquivo;
  char nome_arquivo[121];
  printf("Digite o nome do arquivo:\n");
  scanf("%120s",nome_arquivo);
  printf("\nDigite o número de pontos:\n");
  scanf("%d",&nPontos);
  entrada = (Ponto *) malloc(nPontos*sizeof(Ponto));
  for (cont = 0; cont < nPontos;cont++) {
     printf("Digite coordenadas x,y:\n");
     scanf("%f%f",&entrada[cont].x,&entrada[cont].y);
  }
  /* continua */
```

Programa que salva e recupera um vetor de pontos em um arquivo binário

Usando as Funções Definidas Anteriormente

```
salva(nome_arquivo, nPontos, entrada);
do {
    printf("Digite agora a posição do ponto que
          deseja ver: \n");
    scanf("%d",&pos);
 } while (pos > nPontos || pos<=0 );
saida = (Ponto *) malloc (nPontos*sizeof(Ponto));
carrega(nome_arquivo, nPontos, saida);
printf("O ponto na posicao %d é  {%f,%f}\n", pos,
          saida[pos-1].x, saida[pos-1].y);
}
```

Gravando os pontos no arquivo

Lendo os pontos do arquivo

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

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;
}
```