

Arquivos – Acesso direto

Roberto Rocha

Arquivos acesso direto em C

O acesso aos registros pode ser feito fora da ordem, por meio da indicação de um índice. O índice natural de um registro é o seu número de ordem.

Abrindo arquivos

Para se abrir um arquivo, declaramos uma variável do tipo **FILE** (essa é, na verdade, um ponteiro para **FILE** – que está definido em **stdio.h** e não é do tipo primitivo):

```
FILE *fp; // ponteiro para FILE
```

e então utilizamos a função **fopen()**:

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

```
/* nome do arquivo e modo de abertura */
```

Modos de abertura

“**r+b**” – abre um arquivo binário para leitura/escrita. Se o arquivo não existir dará erro;

“**w+b**” – abre um arquivo binário para escrita. Se um arquivo com o mesmo nome existir, será sobrescrito;

“**a+b**” – abre um arquivo binário para anexação. Se o arquivo não existir, será criado.

Escrevendo no arquivo com a função fwrite()

A função `fwrite()` pode escrever qualquer tipo de dado em arquivos.

Sintaxe:

```
int fwrite(void * mem, size_t qtd_bytes, size_t cont, FILE *arq);
```

Onde:

mem representa a variável que armazena o conteúdo a ser gravado no arquivo;

qtd_bytes representa o total de bytes que será escrito no arquivo

cont representa o número de blocos de tamanho *qtd_bytes* que serão escritos no arquivo;

arq é a referência para o arquivo onde as informações serão escritas.

Quando a função for bem-sucedida, gerará como retorno um valor igual ao número de gravações realizadas (igual ao parâmetro *cont* informado). Se ocorrer algum erro, o valor retornado será menor que *cont*.

Lendo no arquivo com a função fread()

A função `fread()` pode ler qualquer tipo de dado em arquivos.

Sintaxe:

```
int fread(void * mem, size_t qtd_bytes, size_t cont, FILE *arq);
```

Onde:

mem representa a variável que armazena o conteúdo a ser lido do arquivo;

qtd_bytes representa o total de bytes que será lido do arquivo

cont representa o número de blocos de tamanho *qtd_bytes* que serão lidos do arquivo;

arq é a referência para o arquivo que será lido.

Quando a função for bem-sucedida, gerará como retorno um valor igual ao número de leituras realizadas (igual ao parâmetro *cont* informado). Se ocorrer algum erro, o valor retornado será menor que *cont*.

Toda vez que é realizada uma leitura com `fread` é posicionado o arquivo na próxima posição.

Buscando determinada posição do arquivo e apontando para o seu início



Podemos apontar para um local específico dentro de um arquivo pelo emprego de **fseek()**:

Sintaxe:

fseek (FILE ***arq**, long **qtd_bytes**, int **posição**);

Onde:

arq representa o arquivo que será percorrido pela função **fseek**;

qtd_bytes representa a quantidade de bytes que o cursor será movimentado a partir de posição;

posição representa o ponto a partir do qual a movimentação será executada, podendo assumir valores:

SEEK_SET – permite a movimentação de **qtd_bytes** a partir da posição **inicial do arquivo**;

SEEK_CUR – permite a movimentação de **qtd_bytes** no arquivo a partir do **ponto atual do cursor**;

SEEK_END – permite a movimentação de **qtd_bytes** a partir da posição **final do arquivo**.

EOF(*arq) **retorna -1** quando uma função atinge o final do arquivo.

Exemplo

Desenvolva um sistema para controlar um conjunto de mercadorias.

Os dados referentes as mercadorias serão:

- código
- descrição
- preço

O sistema deverá possuir um menu com as seguintes opções:

- incluir uma mercadoria
- alterar dados de uma mercadoria
- listar todas as mercadorias

Sistema mercadoria

Para armazenar os dados referentes as mercadorias criaremos uma estrutura:

- código
- descrição
- preço

```
struct Tmercadoria
{
    int codigo;
    char descricao[30];
    float preco;
};
typedef struct Tmercadoria mercadoria;
```

Sistema mercadoria

Programa principal

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <conio.h>
#include <locale.h>
int main()
{
    FILE *f;
    char op;
    setlocale(LC_ALL,"portuguese");
    if ((f = fopen("teste.dat", "r+b"))==NULL) // arquivo existe
    { printf("Arquivo não existia ... criando arquivo!");
      if((f = fopen("teste.dat", "w+b"))==NULL) //arq não existe
      { printf("Erro na criação do arquivo!!");
        exit(1);
      }
    }
    system("pause");
}
```

```
do
{ printf("Escolha:\n");
  printf("a-incluir arquivo\n");
  printf("b-alterar arquivo\n");
  printf("c-listar arquivo\n");
  printf("d-sair do sistema\n");
  op=getch();
  switch (op)
  {
    case 'a':
      printf("preenchendo o arquivo...\n");
      inclui_mercadoria(f);
      break;
    case 'b':
      printf("alterando o arquivo...\n");
      altera_arquivo(f);
      break;
    case 'c':
      printf("imprimindo o arquivo...\n");
      imprime_arquivo(f);
      system("pause");
      break;
  }
}
while (op!='d');
fclose(f);
return 0;
}
```


Sistema mercadoria

Para a função de localizar um código no arquivo – se não existir devolvera -1

```
int localiza_mercadoria(FILE *f,int codigo)
{ int posicao=-1,achou=0;
  mercadoria m;
  fseek(f,0,SEEK_SET);
  fread(&m, sizeof(m),1, f);
  while (!feof(f) && !achou)
  { posicao++; // semelhante a posicao = posicao +1;
    if (m.codigo==codigo)
    { achou=1;
    }
    fread(&m, sizeof(m),1, f);
  }
  if (achou)
  { return posicao;
  }
  else
  { return -1;
  }
}
```

Sistema mercadoria

Para a função

- incluir uma mercadoria

```
void inclui_mercadoria(FILE *f)
{ mercadoria m;
  int posicao;
  //lendo os dados do teclado
  printf("Digite o código da mercadoria...:");
  fflush(stdin);
  scanf("%d",&m.codigo);
  posicao=localiza_mercadoria(f,m.codigo);
  if (posicao==-1) //não tinha codigo no arquivo
  { printf("Digite a descrição da mercadoria...:");
    fflush(stdin);
    gets(m.descricao);
    printf("Digite o preço da mercadoria...:");
    fflush(stdin);
    scanf("%f",&m.preco);
    fseek(f,0,SEEK_END); // posicionado o arquivo no final
    fwrite(&m, sizeof(m),1,f); //gravando os dados
    fflush(f);
  }
  else
  { printf("Código %d já existe no arquivo. Inclusão não realizada!\n");
  }
}
```

Sistema mercadoria

Para a função
- alterar uma mercadoria

```
void altera_arquivo(FILE *f)
{ int codigo,posicao;
  mercadoria m;
  printf("Diga qual o codigo da mercadoria para alterar:");
  scanf("%d",&codigo);
  posicao=localiza_mercadoria(f,codigo);
  if (posicao!=-1) // localizou a mercadoria
  { fseek(f,sizeof(m)*(posicao),SEEK_SET);
    fread(&m, sizeof(m),1, f);
    printf("Codigo atual:%d - %s preço atual atual:%.2f\n",m.codigo,m.descricao,m.preco);
    printf("Nova descrição:");
    fflush(stdin);
    gets(m.descricao);
    printf("Novo preço....:");
    scanf("%f",&m.preco);
    fseek(f,sizeof(m)*(posicao ),SEEK_SET);
    fwrite(&m, sizeof(m),1, f);
    fflush(f);
  }
}
```

Sistema mercadoria

Para a função
Imprimir as mercadorias

```
void imprime_arquivo(FILE *f)
{
    mercadoria m;
    fseek(f,0,SEEK_SET);
    fread(&m, sizeof(m),1, f);
    while (!feof(f))
    {
        printf("Código....:%d \n",m.codigo);
        printf("Descrição.:%s \n",m.descricao);
        printf("Preço.....:%.2f\n",m.preco);
        fread(&m, sizeof(m),1, f);
    }
}
```

Exercício

Faça um sistema de cadastro de Alunos.

O cadastro deve possuir pelo menos os seguintes campos:

Nome,matricula,nota1,nota2,nota3

Deverá ter as seguintes opções:

- 1 – incluir
- 2 – alterar
- 3 – excluir (logicamente) obs. Crie na estrutura um campo para dizer se o registro está excluído ou não!
- 4 – listar todos os alunos e suas respectivas notas
- 5 – Listar todas os alunos cuja média das notas seja acima de 6.

Aluno		
Nome: Ana	Matricula: 1234	
Nota1: 9.5	Nota2: 8.0	Nota3: 10.0



PUC Minas
Virtual