Operações Fundamentais em Arquivos

Linguagem C

Profª Yorah Bosse

yorah.bosse@gmail.com
yorah.bosse@ufms.br

Uso da memória secundária

- Em muitos casos necessitamos da memória secundária (auxiliar), para armazenar informações
- Podemos utilizar discos como HD, pen-drives, etc.
- Utiliza-se **arquivos** para armazenar dados na memória secundária

Arquivos

- Entende-se por arquivo as estruturas de dados armazenadas na memória secundária (como discos) que posteriormente podem ser lidas e alteradas

- Não há perda de dados ao se desligar o computador e/ou programa.

- A biblioteca utilizada para manipulação de arquivos é <stdio.h>

- Uma vez que o arquivo está aberto, informações podem ser trocadas entre ele e o programa

Algumas funções para manipulação de arquivos

Nome	Função
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
putc()	Escreve um caractere em um arquivo
fputc()	O mesmo que putc()
getc()	Lê um caractere de um arquivo
fgetc()	O mesmo que getc()
fseek()	Posiciona o arquivo em um byte específico padrão
fprintf()	É para um arquivo o que o printf() é para o console
fscanf()	É para um arquivo o que o scanf() é para o console
fread()	Lê um registro do arquivo
fwrite()	Grava um registro no arquivo
feof()	Retorna verdadeiro se o fim do arquivo for atingido
ferror()	Retorna verdadeiro se ocorreu um erro
rewind()	Realoca o indicador de posição de arquivo no início do arquivo
remove()	Apaga um arquivo
fflush()	Descarrega um arquivo

Formato de um arquivo

- Arquivos podem ser classificados em dois tipos:

→ Arquivos de texto

→ Arquivos binários

Formato de um arquivo

FORMATO	VANTAGENS	DESVANTAGENS
Texto	• Legibilidade • Pode ser editado com editor de texto • Arquivo pode ser facilmente transferido para outra plataforma	• Representação numérica pode não ser precisa • Pode ocupar muito mais espaço
Binário	• Não existe erro de conversão de valores numéricos • Leitura ou escrita de arquivos é mais rápida • Pode ocupar menos espaço	• Transferência para outra plataforma pode ser problemático

Ponteiros para Arquivos

- Para declarar um ponteiro para arquivo é utilizado o tipo de dados FILE.

- Esse ponteiro serve para identificar um arquivo no disco, permitindo a gravação, alteração exclusão de dados de um arquivo.

- Declaração de um ponteiro para arquivo:

```
FILE *fp
```

Função fopen()

- Função para abrir o arquivo
- O ponteiro recebe o endereço de memória ocupado pelo arquivo, sendo que, se houver erro o ponteiro valerá nulo (NULL).


```
FILE * fp;
fp = fopen("arquivo.txt", "w");
if (fp == NULL ) {
        printf("Não foi possível abrir o arquivo.\n");
        exit(1);  // força o término da execução da rotina
}
```

Função fopen( )

```
FILE *fp;
fp = fopen("arquivo.txt", "w");
```

[onde guardar] *[nome]* *[tipo]*

- Para gerar um código de programa que abre um arquivo, o compilador precisa conhecer 3 coisas:

1- O nome do arquivo
   - Ex.: “arquivo.txt” ou “c:\\dados\\arquivo.txt”

2- O tipo de abertura
   - “w” para gravação

3- Onde guardar informações sobre o arquivo
   - Variável ponteiro para um arquivo (FILE)

Função fopen( )

- Dois modificadores podem ser usados junto ao tipo:
  - Letra “b” para modo binário
  - Sinal “+” quando o arquivo já existe e queremos atualizá-lo.

- Lista completa de opções de tipo para fopen():

Tipo	Significado
“r”	Abre um arquivo texto para leitura. O arquivo deve estar presente no disco.
“w”	Abre um arquivo texto para gravação. Se o arquivo estiver presente no disco ele será destruído e reinicializado. Se não existir, ele será criado.
“a”	Abre um arquivo texto para gravação. Os dados serão adicionados ao fim do arquivo existente, ou um novo arquivo será criado.

Tipo e Significados – fopen()

Tipo	Significado
“r+”	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser atualizado.
“w+”	Abre um arquivo texto para leitura e gravação. Se o arquivo existir ele será destruído e reinicializado. Se não existir, será criado.
“a+”	Abre um arquivo texto para atualizações e para adicionar dados ao fim do arquivo existente ou um novo arquivo será criado.
“rb”	Abre um arquivo binário para leitura. O arquivo deve estar presente no disco.
“wb”	Abre um arquivo binário para gravação. Se o arquivo estiver presente ele será destruído e reinicializado. Se não existir, ele será criado.

Tipos e Significados - fopen()

Tipo	Significado
“ab”	Abre um arquivo binário para gravação. Os dados serão adicionados ao fim do arquivo existente, ou um novo arquivo será criado.
“rb+”	Abre um arquivo binário para leitura e gravação. O arquivo deve existir e pode ser atualizado.
“wb+”	Abre um arquivo binário para leitura e gravação. Se o arquivo existir ele será destruído e reinicializado. Se não existir, será criado.
“ab+”	Abre um arquivo binário para atualizações e para adicionar dados ao fim do arquivo existente ou um novo arquivo será criado.

Cuidado ao Abrir Arquivos

- Problemas:

  Gravação: não tem espaço em disco

  Leitura: o arquivo não existe

- Sempre verifique se o arquivo foi aberto com sucesso, antes de escrever ou ler.

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

if (fp==NULL)
   printf("\nHouve problemas, verifique.\n");
else
   printf("\nArquivo aberto corretamente.\n");
```

Função fclose()

- Fecha um arquivo, utilizando-se do ponteiro.
- A função retorna 0 (zero) se não houver problemas, caso contrário, retorna um erro. Exemplo:

```
int main()   {
    FILE *fp;
    int erro;
    fp= fopen("arquivo.dat", "w");
    if (fp == NULL)
        exit(1);
    erro= fclose(fp);
    if (erro==0)
        printf("\nArquivo fechado com sucesso.\n");
    else
       printf("\nErro no fechamento.\n");
    return 0;
}
```

Formas de Acessar um Arquivo

- Existem 4 diferentes formas de acessar arquivos:

1) Dados são lidos e escritos um caracter por vez.
   → Funções: fgetc() e fputc()
   → Funções: getc() e putc() (essas funções não serão tratadas neste material)

2) Dados são lidos e escritos como "strings".

3) Dados são lidos e escritos de modo formatado.

4) Dados são lidos e escritos em um formato chamado registro ou bloco.

Função fputc ( )

→ Grava **um caracter** no arquivo

- **`int fputc(int ch, FILE *fp)`**
- **`ch`** é o caracter a ser gravado
- **`fp`** é o ponteiro devolvido por fopen

Exemplo:

```
FILE *fp;
fp = fopen("arquivo.txt", "r+");
if (fp == NULL)
    exit(1);
fputc('a', fp);
```

Função fgetc ( )

→ Lê **um caracter** do arquivo

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

- **fp** é o ponteiro devolvido por fopen

Exemplo:

```
FILE *fp;
char ch;
fp = fopen("teste.txt", "r"));
if (fp == NULL) {
   exit(1);
}
ch = fgetc(fp);
while (ch != EOF) {
   ch = fgetc(fp);
}
```

Fim de Arquivo (EOF)

- Definida em **<stdio.h>**
- Retornada por **getc()** ou **fgetc()** quando tenta ler além do final de um arquivo
- Indica que o final do arquivo foi atingido
- Não pode ser utilizada com arquivos binários
- Pode ser utilizada com arquivos de texto

Função rewind()

→ Reinicia o arquivo, ou seja, movimenta o ponteiro do arquivo para seu início

Exemplo:

```
FILE *fp;
fp = fopen("arquivo.txt", "w"));
if (fp == NULL) exit(1);

while (ch != EOF)
        ch = getc(fp);

rewind(fp);
```

Exemplo

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

//dados são escritos caracter por caracter
int main ()
{
    FILE * pFile;
    char c;

    pFile=fopen("alfabeto.txt","w+");

    if (pFile == NULL) {
      printf("\nProblemas na abertura do arquivo!.\n");
      exit(1);
    }

    for (c = 'A' ; c <= 'Z' ; c++) {
       fputc (c , pFile);
    }

    rewind(pFile);
    c = fgetc(pFile);
    while (c != EOF){
        printf("%c\n",c);
        c = fgetc(pFile);
    }

    fclose (pFile);

    system ("pause");
    return 0;
}
```

Formas de Acessar um Arquivo

- Existem 4 diferentes formas de acessar arquivos:

1) Dados são lidos e escritos um caracter por vez.

2) Dados são lidos e escritos como "strings".
   → Funções: fgets() e fputs()

3) Dados são lidos e escritos de modo formatado.

4) Dados são lidos e escritos em um formato chamado registro ou bloco.

Função fputs ( )

→ Grava **uma string** no arquivo

- `int fputs(int *s, FILE *fp)`

- **`s`** é a string a ser gravada
- **`fp`** é o ponteiro devolvido por fopen

→ Retorna um valor positivo quando a escrita é bem sucedida; caso contrário, retorna EOF

Função fputs ( )

```
#include <stdio.h>

int main ()  {
   FILE * pFile;
   char string [10];

   printf ("Escreva uma frase: ");
   gets (string);

   pFile = fopen ("frase.txt","w+");
   if (pFile == NULL) exit(1);

   fputs (string,pFile);

   fclose (pFile);
   return 0;

}
```

Função fgets ( )

→ Lê **uma linha** por vez do arquivo

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

- **s** é a string a ser lida
- **n** número máximo de caracteres que serão gravados
- no vetor **s**
- **fp** é o ponteiro devolvido por fopen

Função fgets ( )

- Lê caracteres até atingir um caractere `'\n'`, ou o final do arquivo ou o número máximo de caracteres especificado

- Escreve um caractere nulo `'\0'` após o último caractere armazenado no array

- Quando o final do arquivo é atingido antes de armazenar algum caractere no vetor, ela retorna **NULL**; caso contrário, retorna o argumento **`s`**

Função fgets ( )

```
#include <stdio.h>

int main()  {
   FILE *pFile;
   char string[80];
   char *presult;
   pFile = fopen ("frase.txt","r");
   if (pFile == NULL) exit(1);
   presult = fgets(string, 80, pFile);
   while(presult != NULL)   {
       printf("%s", string);
       presult = fgets(string, 80, pFile);
    }
   fclose (pFile);
   return 0;
}
```

Formas de Acessar um Arquivo

- Existem 4 diferentes formas de acessar arquivos:

1) Dados são lidos e escritos um caracter por vez.

2) Dados são lidos e escritos como "strings".

3) Dados são lidos e escritos de modo formatado.
   → Funções: fscanf() e fprintf()

4) Dados são lidos e escritos em um formato chamado registro ou bloco.

Função fprintf()

→ Grava um arquivo de modo **formatado**

```
fprintf(FILE *fp, char *s, … )
```

- **fp** é o ponteiro devolvido por fopen
- **s** string formatada
- **...** variáveis

Função fprintf()

```
#include <stdio.h>

int main()   {
   FILE * pFile;
   int n, idade;
   char nome [100];
   pFile = fopen ("formatado.txt","w"); // Testar arquivo
    if (pFile == NULL) exit(1);
   for (n=0 ; n<3 ; n++) {
     printf ("Digite seu nome: ");
     gets (nome);
     printf ("Digite sua idade:");
     scanf("%d", &idade);
     fprintf (pFile, "\nNome: %s, \nIdade: %d\n\n",nome,idade);
     fflush(stdin);
   }
   fclose (pFile);
}
```

Função fscanf()

→ Lê dados **formatados**

```
fscanf(FILE *fp, char *s, … )
```

- **fp** é o ponteiro devolvido por fopen
- **s** string formatada
- **...** endereço das variáveis

Função fscanf()

```
#include <stdio.h>

int main(){
   char str [80];
   float f;
   FILE * pFile;

   pFile = fopen ("arq.txt","w+");
   if (pFile == NULL) exit(1);

   fprintf (pFile, "%f = %s", 3.1416, "PI");
   rewind (pFile);
   fscanf (pFile, "%f", &f);
   fscanf (pFile, "%s", str);
   fclose (pFile);
   printf ("Leitura do arquivo: %f e %s \n",f,str);
}
```

Formas de Acessar um Arquivo

- Existem 4 diferentes formas de acessar arquivos:

1) Dados são lidos e escritos um caracter por vez.

2) Dados são lidos e escritos como “strings”.

3) Dados são lidos e escritos de modo formatado.

4) Dados são lidos e escritos em um formato chamado registro ou bloco.
   → Funções: fread() e fwrite()
   → Utilizado para arquivos binários

Função fwrite()

- Lê um array em memória e escreve no arquivo

```
int fwrite(void *mem, int n_bytes, int cont, FILE *fp);
```

→ fwrite() recebe 4 argumentos

1º : ponteiro para a localização da memória do dado a ser gravado.

2º : tamanho da representação binária a ser gravada.

3º : número inteiro que informa a fwrite() quantos itens do mesmo tipo serão gravados.

4º : ponteiro que controla o arquivo aberto para escrita.

Exemplo

```
int main()  {
      struct pessoa pes;
      FILE *fp;
      int erro, op;

     fp= fopen("teste.dat","wb");

     if (fp==NULL)
           exit(1);

      printf("\nNome: "); gets(pes.nome);
      printf("\nIdade:" ); scanf("%d",&pes.idade);
      printf("\nPeso:" ); scanf("%f",&pes.peso);
      fwrite(&pes, sizeof(struct pessoa), 1, fp);
      erro=fclose(fp);
      return 0;
}
```

```
struct pessoa  {
    char nome[30];
    int idade;
    float peso;
};
```

Função fread()

- Lê dados do arquivo

int fread(void *mem, int n_bytes, int cont, FILE *fp);

→ fread() possui 4 argumentos

1º : ponteiro para a localização da memória onde serão armazenados os dados lidos.

2º : indica a quantidade de bytes do tipo de dados a ser lido.

3º : quantidade de itens a serem lidos a cada chamada

4º : ponteiro que controla o arquivo aberto para escrita.

Função fread()

→Retorno da função fread():

- Número de itens lidos
- Deve ser o mesmo valor do terceiro argumento
- Quando é 0, a interpretação é ambígua

  - Pode indicar que o final do arquivo foi atingido

  - Pode indicar que ocorreu algum erro antes da leitura de algum elemento

Exemplo

```
int main()  {
      struct spessoa pes;
      FILE *fp;
      int erro, op;
      fp= fopen("teste.dat","rb");
      if (fp==NULL)
         exit(1);
      while(fread(&pes,sizeof(spessoa),1,fp)==1) {
         printf("\nNome: %s", pes.nome);
         printf("\nIdade: %d", pes.idade);
         printf("\nPeso: %0.2f\n", pes.peso);
      }
      erro=fclose(fp);
      return 0;
}
```

```
struct spessoa {
   char nome[30];
   int idade;
   float peso;
};
```

Exercícios

1) Elabore um programa em C que leia os seguintes campos para o cadastro de um fornecedor:

- Nome
- Endereço
- Telefone
- E-mail

Grave estes dados no arquivo “forn.txt”.

Em seguida, apresente na tela todos os fornecedores armazenados no arquivo “forn.txt”.

Exemplo

- Exemplo EXTRA com programa completo que manipula um arquivo binário de cliente, podendo ser executadas as seguintes funções: cadastrar, listar, pesquisar, alterar, excluir registro (exclusão física e não lógica) e excluir todos registros.

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <unistd.h>

struct data{
     int dia, mes, ano;
};

typedef struct sCliente{
    char nome[20], fone[10];
    struct data nasc;
}sCli;

FILE *fp;
sCli cli;
```

Exemplo

```
void abre_arquivo(){
    //fp é um ponteiro para o arquivo
    //cliente.dat é o nome do arquivo
    //rb+ é o tipo de abertura para leitura e gravação em arq.binário
    //com rb+ o arquivo deve existir
    fp = fopen("cliente.dat", "rb+");
    if (fp == NULL)
        //se não conseguiu abrir, tenta novamente com w+
        //wb+ cria e abre para leitura e gravação em arq.binário
        fp = fopen("cliente.dat","wb+");
        if (fp == NULL){
            printf("Não foi possível abrir o arquivo.\n");
            exit(1);  // força o término da execução da rotina
    }
}
```

Exemplo

```
void fecha_arquivo(){
    //fclose fecha o arquivo e retorna 0 se fechou com sucesso
   if (fclose(fp) != 0)
       printf("ERRO! Fechamento de arquivo!\n");
}

char resp(char *perg){
    char r;
    printf("\n\n%s ",perg);
    do{
      r = toupper(getch());
    }while(r != 'S' && r != 'N');
    return r;
}
```

Exemplo

```
void maiusculo(char *n){
    int letra;
    for (letra=0; letra < strlen(n); letra++)
      n[letra] = toupper(n[letra]);
}

void leituraDados(){
    printf("\nNome : ");
    fflush(stdin);
    scanf("%20[^\n]s",cli.nome);
    maiusculo(cli.nome);
    printf("Telefone de contato : (");
    int cont = 0;
    do{
     do{
        cli.fone[cont] = getch();
     }while(cli.fone[cont] < '0' || cli.fone[cont] > '9');
```

Exemplo

```
  printf("%c",cli.fone[cont]);
  if (cont == 1)
    printf(") ");
  else
    if (cont == 5)
      printf("-");
  cont++;
}while (cont < 10);

do{
  printf("\nData de Nascimento\nDia : ");
  fflush(stdin);
  scanf("%d",&cli.nasc.dia);
}while(cli.nasc.dia<=0 || cli.nasc.dia>31);
```

Exemplo

```
    do{
      printf("Mes : ");
      fflush(stdin);
      scanf("%d",&cli.nasc.mes);
    }while(cli.nasc.mes<=0 || cli.nasc.mes>12);

    do{
      printf("Ano : ");
      fflush(stdin);
      scanf("%d",&cli.nasc.ano);
    }while(cli.nasc.ano<=1900 || cli.nasc.ano>2011);
}
```

Exemplo

```
void cadastrar(){
    abre_arquivo();
    do{
      system("cls");
      printf("****   C A D A S T R A R   ****\n\n");
      leituraDados();
      //posiciona no final do arquivo
      fseek(fp, 0, SEEK_END);
      //grava os dados de cli no arquivo
      fwrite(&cli,sizeof(sCli),1,fp);
    }while(resp("Deseja realizar novo cadastro?(S/N): ") == 'S');
    fecha_arquivo();
}
```

Exemplo

```
void mostra_dados(){
    int cont;
    printf("Nome : %s\n\n", cli.nome);
    printf("Data de nascimento : %d/%d/%d\n",cli.nasc.dia,cli.nasc.mes,cli.nasc.ano);
    printf("Telefone : (");
    cont = 0;
    do{
        printf("%c",cli.fone[cont]);
        if (cont == 1)
          printf(") ");
        else
          if (cont == 5)
              printf("-");
        cont++;
    }while (cont < 10);
}
```

Exemplo

```
void listar(){
  abre_arquivo();
  //o ponteiro retorna para o início do arquivo
  rewind(fp);
  //fread lê um dado do arquivo e retorna 1 caso tenha lido com sucesso.
  //O dado lido é gravado em cli
  int x = fread(&cli,sizeof(sCli),1,fp);
  if (x==1)
    while (x==1){
        system("cls");
        printf("****   L I S T A R   ****\n\n");
        mostra_dados();
        printf("\n\nAperte qualquer tecla para vizualizar o proximo\n\n");
        getch();
        x = fread(&cli,sizeof(sCli),1,fp);
    }
```

Exemplo

```
  else{
    system("cls");
    printf("\n\nListagem...Arquivo vazio!\n\n");
    system("pause");
  }
  fecha_arquivo();
}

int achou(char *n){
   int x;
   //retorna o ponteiro para o início do arquivo
   rewind(fp);
   //x armazena o retorno da leitura que será 1 caso lido sem erro
   x = fread(&cli,sizeof(sCli),1,fp);
```

Exemplo

```
   //repete a leitura caso tenha conseguido ler um dado e
   //o nome procurado seja diferente do nome lido no arquivo
   while (x==1 && strcmp(cli.nome,n)!=0)
     x = fread(&cli,sizeof(sCli),1,fp);
   if (strcmp(cli.nome,n)==0)//achou
     return 1;
   else  //não achou
     return 0;
}

void pesquisar(){
    abre_arquivo();
    char n[20];
    do{
        system("cls");
        printf("****   P E S Q U I S A R   ****\n\n");
```

Exemplo

```
        printf("Digite o nome do cliente a ser pesquisado : ");
        fflush(stdin);
        scanf("%30[^\n]s",n);
        maiusculo(n);
        //se achou retornar 1, 1 é verdadeiro, logo ele entra no if
        //se achou retornar 0, 0 é falso, logo ele entra no else
        if(achou(n)){
           mostra_dados();
        }else
           printf("\nCliente não existente!\n\n");
    }while(resp("Deseja realizar nova pesquisa? (S/N) : ") == 'S');
    fecha_arquivo();
}
```

Exemplo

```
void alterar(){
    abre_arquivo();
    char n[20];
    do{
        system("cls");
        printf("****   A L T E R A R   ****\n\n");
        printf("Digite o nome do cliente a ser alterado : ");
        fflush(stdin);
        scanf("%30[^\n]s",n);
        maiusculo(n);
        if(achou(n)){
            mostra_dados();
            if(resp("\n\nDeseja alterar seus dados?(S/N) : ") == 'S'){
                leituraDados();
                //o ponteiro do arquivo está apontando para o dado a ser
                //modificado,deve-se então retonar 1(tamanho da struct)
                fseek(fp,sizeof(sCli)*-1,SEEK_CUR);
```

Exemplo

```
                        //SEEK_CUR = posição atual
                        //SEEK_SET = inicio do arquivo
                        //SEEK_END = fim do arquivo
            //após retornar 1(tamanho da struct, executa-se o fwrite
            //para gravar no arquivo. Antes de gravar o ponteiro
            //pula uma posição automaticamente pra frente, ficando
            //então, o ponteiro parado sobre o dado a ser substituído
            //aí sobre ele é gravado o novo dado
            fwrite(&cli,sizeof(sCli),1,fp);
            printf("\n\nDados alterados com sucesso!");
         }
         else
            printf("\n\nDados não foram alterados");
      }else
         printf("\nCliente não existente!\n\n");
   }while(resp("Deseja realizar nova alteracao? (S/N) : ") == 'S');
   fecha_arquivo();
}
```

Exemplo

```
void excluir(){  //aqui está sendo feita a exclusão física de um registro
    abre_arquivo();
    int tamanho;
    char n[20];
    do{
        fseek(fp,0,SEEK_END);    //posiciona no final do arquivo
        tamanho = ftell(fp);     //tamanho recebe o tamanho do arquivo em bytes
        system("cls");
        printf("****   E X C L U I R   ****\n\n");
        printf("Digite o nome do cliente a ser excluido : ");
        fflush(stdin);
        scanf("%30[^\n]s",n);
        maiusculo(n);
        if(achou(n)){
           mostra_dados();
           if(resp("\n\nDeseja realmente exclui-lo?(S/N) : ") == 'S'){
              //exclusao física
```

Exclusão física
X
lusão lógica

Exemplo

```
    //pula uma posicao e lê o dado do arquivo e grava em cli
    while(fread(&cli,sizeof(sCli),1,fp)==1){
        //volta duas posições
        fseek(fp,sizeof(sCli)*-2,SEEK_CUR);
        //pula um posição pra frente e grava no arquivo
        fwrite(&cli,sizeof(sCli),1,fp);
        //pula uma posição pra frente
        fseek(fp,sizeof(cli),SEEK_CUR);
    }
    //o ponteiro volta uma posição
    fseek(fp,sizeof(sCli)*-1,SEEK_CUR);
    //apaga os dados da posição seguinte ao ponteiro
    //até o final
    ftruncate(fileno(fp),tamanho-sizeof(sCli));
    printf("\n\nCliente excluido com sucesso!");
}
else
```

Exemplo

```
            printf("\n\nDados não foram alterados");
        }else
            printf("\nCliente não existente!\n\n");
    }while(resp("Deseja realizar nova exclusao? (S/N) : ") == 'S');
    fecha_arquivo();
}

void apagarTudo(){
    system("cls");
    printf("****   E X C L U I R    T U D O   ****\n\n");
    if(resp("Deseja excluir todos cliente?(S/N) : ") == 'S'){
       //cria e abre um novo arquivo vazio sobre o existe
       fp = fopen("cliente.dat","wb+");
       if (fp == NULL){
          printf("\n\nOcorreu um erro.\n\n");
          exit(1);  // força o término da execução da rotina
```

Exemplo

```
        }else{
            printf("\n\nDados apagados com sucesso!\n");
            fecha_arquivo();
        }
    }else
        printf("\n\nNenhum dado foi deletado\n\n");
    system("pause");
}

void menu(){
    char op;
    do{
        system("cls");
        printf("\n*****   M E N U   *****\n\n");
        printf("[ C ] Cadastrar cliente\n");
        printf("[ L ] Listar todos clientes\n");
```

Exemplo

```
printf("[ P ] Pesquisar cliente\n");
printf("[ A ] Alterar dados cliente\n");
printf("[ E ] Excluir cliente\n");
printf("[ X ] Apagar todos Clientes\n");
printf("[ S ] Sair do sistema\n\n");
printf("Opcao :  ");
do{
  op = toupper(getch());
}while (op != 'C' && op != 'L' && op != 'P' && op != 'A' &&
      op != 'E' && op != 'X' && op != 'S');
switch (op){
      case 'C' :
          cadastrar();
          break;
      case 'L' :
          listar();
          break;
```

Exemplo

```
            case 'P' :
                pesquisar();
                break;
            case 'A' :
                alterar();
                break;
            case 'E' :
                excluir();
                break;
            case 'X' :
                apagarTudo();
                break;
        }
    }while (op != 'S');
}
```

```
int main(){
    menu();
    return 0;
}
```