



Algoritmos e Programação II

Prof.^a Noeli A. Pimentel Vaz
Prof. Joilson dos Reis Brito

Correção da APA3

Questão 1

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
    FILE *PontArq;
    char Aluno[40];
    float NotaVa1, NotaVa2, Media;
    setlocale(LC_ALL, "Portuguese");
    PontArq = fopen("Notas.txt", "r");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    printf("MAPA DE NOTAS\n");
    printf("Aluno\t1a Va\t2a Va\tMedia\tSituação\n");
    while(fscanf(PontArq, "%s %f %f", Aluno, &NotaVa1, &NotaVa2) != EOF)
    {
        Media = (NotaVa1*2 + NotaVa2*3)/5;
        printf("%s\t%.2f\t%.2f\t%.2f", Aluno, NotaVa1, NotaVa2, Media);
        if(Media >= 6)
            printf("\tAprovado\n");
        else
            printf("\tReprovado\n");
    }
    printf("\n");

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

Questão 2

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <locale.h>
int main()
{
    FILE *PonteiroArquivo, *PonteiroDestino;
    char Caracter, NomeArquivo[40], ArquivoCorrigido[40];
    int Flag = 0, TemPonto=0, TemEspacoPonto=0;
    setlocale(LC_ALL, "Portuguese");
    printf("Digite o nome do arquivo .txt sem a extensão: ");
    gets(NomeArquivo);
    strcpy(ArquivoCorrigido, NomeArquivo);
    strcat(NomeArquivo, ".txt");
    strcat(ArquivoCorrigido, "(Corrigido).txt");
    PonteiroArquivo = fopen(NomeArquivo, "r");
    PonteiroDestino = fopen(ArquivoCorrigido, "w");
    if(PonteiroArquivo == NULL || PonteiroDestino == NULL)
    {
        printf("Ocorreu um erro! O arquivo não foi aberto!\n");
        exit(1);
    }
    else
    {
        Caracter = fgetc(PonteiroArquivo);
        while(Caracter != EOF)//END OF FILE
        {
            if(Flag==0)
            {
                Caracter = toupper(Caracter);
                Flag=1;
            }
            if(TemPonto)
            {
                Caracter = toupper(Caracter);
                TemPonto=0;
            }

            if(Caracter=='.'||Caracter=='?'||Caracter=='!')
                TemPonto=1;

            fputc(Caracter, PonteiroDestino);
            Caracter = fgetc(PonteiroArquivo);
        }
        fclose(PonteiroDestino);
        printf("Arquivo corrigido com sucesso!\n ");
    }
    system("Pause");
    return 0;
}
```

Correção do Exercício 3

Lista9

Exercício 3

Faça um programa que cria um vetor de 7 posições do tipo `HistoricoSuperheroi`. Depois o programa deve ler os nomes e o ano de criação de 7 super-heróis. Por fim o programa deverá imprimir a relação dos superheróis classificados pelo ano de criação.

PLANO DE TESTE

ENTRADA

Super-herói: Quarteto Fantástico

Ano Criação: 1961

Super-herói: Lanterna Verde

Ano Criação: 1940

Super-herói: Batman

Ano Criação: 1938

Super-herói: Homem-Aranha

Ano Criação: 1962

Super-herói: Capitão América

Ano Criação: 1941

Super-herói: Superman

Ano Criação: 1938

Super-herói: Mulher-Maravilha

Ano Criação: 1941

SAÍDA

1938 Superman

1938 Batman

1940 Lanterna Verde

1941 Mulher-Maravilha

1941 Capitão América

1961 Quarteto Fantástico

1962 Homem-Aranha

```
struct HistoricoSuperHeroi
{
    char Nome[40];
    int AnoCriacao;
};
```

```
HistoricoSuperHeroi Menor;
```

Menor

Nome	AnoCriacao

```
HistoricoSuperHeroi SuperHeroes[7];
```

SuperHeroes

	Nome	AnoCriacao
0		
1		
2		
3		
4		
5		
6		

Após leitura:

SuperHeróis

	Nome	AnoCriação
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Ordenação:

```
I = 0;  
Menor = SuperHerois[I];
```

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Quarteto Fantástico	1961

```
J = I + 1;  
Compara SuperHeroi[1].AnoCriacao comMenor.AnosCriacao
```

Ordenação:

```
I = 0;  
Menor = SuperHerois[I];
```

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Quarteto Fantástico	1961

Ordenação:

I = 0;

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Quarteto Fantástico	1961

J = I + 1; J = 1;

Compara SuperHeroi[1].AnoCriacao com Menor.AnnoCriacao.

É menor!

Ordenação:

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Lanterna Verde	1940

o Registro Menor recebe o registro armazenado na posição 1.

Aux = 1;

Ordenação:

J++; J = 2

Compara SuperHeroi[2].AnoCriacao com Menor.AnosCriacao.

É menor!

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Lanterna Verde	1940

o Registro Menor recebe o registro armazenado na posição 2.

Ordenação:

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

o Registro Menor recebe o registro armazenado na posição 2.

Aux = 2;

Ordenação:

J++; J = 3

Compara SuperHeroi[3].AnoCriacao com Menor.AnoCriacao.

Não é menor!

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

Ordenação:

J++; J = 4

Compara SuperHeroi[4].AnoCriacao com Menor.AnoCriacao.

Não é menor!

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

Ordenação:

J++; J = 5

Compara SuperHeroi[5].AnoCriacao com Menor.AnoCriacao.

Não é menor!

SuperHerois

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

Ordenação:

J++; J = 6

Compara SuperHeroi[6].AnoCriacao com Menor.AnosCriacao.

Não é menor!

SuperHeroes

	Nome	AnoCriacao
0	Quarteto Fantástico	1961
1	Lanterna Verde	1940
2	Batman	1938
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

Após a primeira passagem do segundo for, o programa identifica o primeiro menor ano de criação e faz a troca com o registro da posição 0;

Ordenação:

```
SuperHerois[2]= SuperHerois[0];  
SuperHerois[0] = Menor;
```

SuperHerois

	Nome	AnoCriacao
0	Batman	1938
1	Lanterna Verde	1940
2	Quarteto Fantástico	1961
3	Homem-Aranha	1962
4	Capitão América	1941
5	Superman	1938
6	Mulher-Maravilha	1941

Menor

Nome	AnoCriacao
Batman	1938

O processo se repete até que a variável *l* alcance o limite do vetor.

Exercício 3

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 5

struct HistoricoSuperheroi
{
    char Nome[40];
    int AnoCriacao;
};
```

```

int main()
{
    setlocale(LC_ALL,"Portuguese");
    int I, J, Aux;
    HistoricoSuperheroi SuperHerois[TAM];
    HistoricoSuperheroi Menor;
    for(I=0; I<TAM; I++)
    {
        printf("\nSuper Herói %d\n",I+1);
        printf("Nome: ");
        fflush(stdin);
        gets(SuperHerois[I].Nome);
        printf("Ano de criação: ");
        scanf("%d",&SuperHerois[I].AnoCriacao);
    }
    for(I=0;I<TAM - 1; I++)
    {
        Menor = SuperHerois[I];
        Aux = I;
        for(J=I+1; J < TAM; J++)
        {
            if(SuperHerois[J].AnoCriacao < Menor.AnoCriacao)
            {
                Menor = SuperHerois[J];
                Aux = J;
            }
        }
        SuperHerois[Aux]= SuperHerois[I];
        SuperHerois[I] = Menor;
    }
}

```

Exercício 3

Exercício 3

```
printf("Ano de Criação - Nome\n");
for(I=0; I<TAM; I++)
{
    printf("    %d    -  %s\n ",SuperHerois[I].AnoCriacao,SuperHerois[I].Nome);
}
system("Pause");
return 0;
}
```

Arquivo de Registros

Foram digitadas várias informações para testar esse programa dos Super Heróis...

Para executar meu programa vou precisar digitar todos os super heróis novamente?

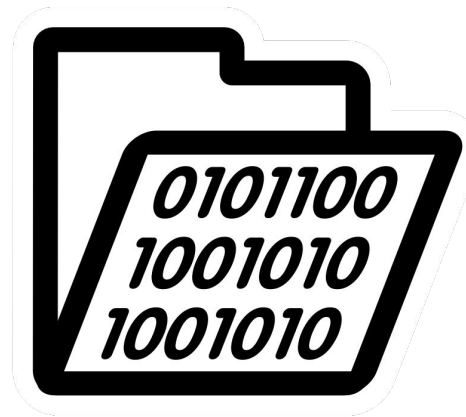
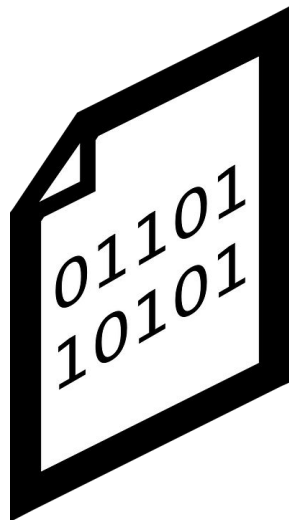
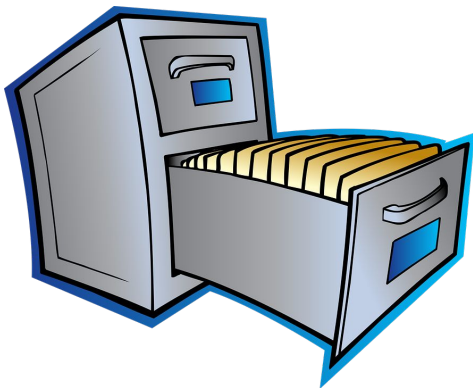


Isso acontece porque o vetor fica na memória RAM do computador.

Arquivo de Registros



ARQUIVOS BINÁRIOS



Arquivos Binários

Até agora, estudamos como processar arquivos de texto em nossos programas. Nos arquivos de texto, existe um formato interno que é identificado pela ocorrência do '\n' separando os caracteres encontrados no arquivo por linhas.

Arquivos binários são, em geral, arquivos de dados, executáveis, jogos, etc.

Em arquivos binários não faz sentido ler linha por linha, pelo fato de não estarem organizados dessa forma.

Arquivos Binários

As funções para tratamento de arquivos binários são diferentes das funções usadas para arquivos de texto.

As operações que vamos usar para processar arquivos binários são denominadas **Acesso Direto**, e só podem ser utilizadas para arquivos abertos em modo **binário**.

No acesso direto os dados são escritos em blocos da memória para o disco e lidos em blocos do disco para a memória.

Arquivos Binários

Arquivos Binários permitem escrever um vetor inteiro no disco de uma só vez, e o mesmo acontece para fazer a leitura do vetor também!



Modos de Abertura de Arquivo Binário

Por padrão, a abertura de um arquivo é realizada considerando que o arquivo é de texto.

Para abrir o arquivo em modo binário é necessário acrescentar um **b** aos modos de abertura que já estudamos:

rb, wb, ab, r+b, w+b, a+b.

Leitura e Escrita em Arquivos Binários

As funções que permitem leitura e escrita em arquivos binários são:

fread e fwrite



Escrita de Blocos em Arquivos Binários

A função `fwrite` faz parte do arquivo `stdio.h` e é responsável por escrever um bloco de bytes existente em memória para um arquivo aberto em modo binário.

Syntax:

```
int fwrite(const void *ptr, int size, int n, FILE *arq)
```

Escrita de Blocos em Arquivos Binários

int fwrite(const void *ptr, int size, int n, FILE *arq)

Em que:

ptr - é um ponteiro que contém o endereço de memória daquilo que queremos guardar no arquivo (const indica que o parâmetro não será alterado)

size - indica o tamanho em bytes de cada um dos elementos que pretendemos escrever. O comando sizeof(tipo) devolve a quantidade de bytes do endereço para o qual ptr aponta.

n - Indica o número de elementos que queremos escrever.

arq - ponteiro para o arquivo no qual queremos escrever.

A função **fwrite** devolve, após a gravação, o número de itens que conseguiu escrever com sucesso no arquivo.

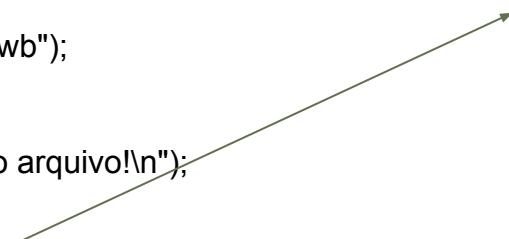
Exemplo 1 - Escrita de Blocos em Arquivos Binários

Programa que realiza leitura de um vetor com 10 idades a partir do teclado e as armazena do arquivo Dados.dat.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 10

int main()
{
    FILE *PontArq;
    int Idades[TAM], i;
    int Validacao;
    setlocale(LC_ALL, "Portuguese");
    printf("Digite as Idades:");
    for(i=0; i<TAM; i++)
    {
        printf("\nIdades[%d]: ", i);
        scanf("%d", &Idades[i]);
    }
    PontArq = fopen("Dados.dat", "wb");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    Validacao = fwrite(Idades, sizeof(int), TAM, PontArq);
    if(Validacao != TAM)
        printf("\nErro ao gravar todos os dados!\n");
    else
        printf("Dados Gravados com sucesso no arquivo!\n");
    fclose(PontArq);
    system("Pause");
    return 0;
}
```

A função `sizeof(int)` devolve o inteiro referente a quantidade de bytes que um `int` ocupa.



Leitura de Blocos em Arquivos Binários

A função `fread` também faz parte do arquivo `stdio.h` e é responsável por ler um bloco de bytes existente em memória para um arquivo aberto em modo binário.

Syntax:

```
int fread(const void *ptr, int size, int n, FILE *arq)
```

Leitura de Blocos em Arquivos Binários

int fread(const void *ptr, int size, int n, FILE *arq)

Em que:

ptr - é um ponteiro que contém o endereço de memória daquilo que queremos ler do arquivo (const indica que o parâmetro não será alterado)

size - indica o tamanho em bytes de cada um dos elementos que pretendemos ler.

n - Indica o número de elementos que queremos ler.

arq - ponteiro para o arquivo no qual queremos ler.

A função fread retorna o número de itens que conseguiu ler com sucesso no arquivo.

Exemplo 2 - Leitura de Blocos em Arquivos Binários

Programa que carrega um vetor com 10 idades lidas a partir do arquivo Dados.dat gravados na memória pelo programa anterior. Depois de ler os 10 inteiros, o programa deverá mostrar o vetor na tela.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 10

int main()
{
    FILE *PontArq;
    int Idades[TAM], l;
    int Validacao;
    setlocale(LC_ALL,"Portuguese");
    PontArq = fopen("Dados.dat","rb");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    Validacao = fread(Idades,sizeof(int),TAM, PontArq);
    if(Validacao != TAM)
        printf("\nErro ao ler todos os dados!\n");
    else
    {
        for(l=0; l<TAM; l++)
        {
            printf("Idades[%d]: %d\n",l, Idades[l]);
        }
    }
    fclose(PontArq);
    system("Pause");
    return 0;
}
```

Acesso Sequencial e Acesso Direto a Arquivos

O acesso à informação pode ser realizado de duas formas:

Acesso Sequencial: Como temos feito até agora, percorrendo o arquivo até localizarmos aquilo que pretendemos.

Acesso Direto: Colocando-nos na posição que queremos de um arquivo, sem ter que percorrer todos os bytes que se encontram antes dessa posição.

Posicionamento ao Longo do Arquivo

Sempre que se abre um arquivo através da função **fopen** é criada em memória uma estrutura do tipo FILE que contém informações sobre o arquivo que se está processando.

Nessa informação existe a posição em que estamos no arquivo, isso porque as funções de leitura e escrita avançam automaticamente no arquivo à medida que vão lendo ou escrevendo informações.

Posicionamento ao Longo do Arquivo

A linguagem C não vê o arquivo de forma formatada, por isso a posição corrente do arquivo é dada em termos de bytes.

Sempre que um arquivo é aberto (exceto para acrescentar “a”), ele é posicionado antes do primeiro byte do arquivo, isto é, na posição 0 (zero).

Se o arquivo for aberto para acrescentar, ele é posicionado após o último byte do arquivo.

Posicionamento ao Longo do Arquivo



O posicionamento para arquivos só faz sentido para **Arquivos Binários**.

Posicionamento ao Longo do Arquivo

Para saber a posição corrente em um arquivo, existe a função **ftell**.

Sintaxe:

```
long ftell(FILE *Arq)
```



NOVIDADE

Observe que a função retorna um long e não um int, isso porque a dimensão dos arquivos pode ultrapassar o valor máximo representado por um inteiro.

Exemplo 3 - Posição do Arquivo

Programa que carrega de um vetor com 10 idades lidas a partir do arquivo Dados.dat gravados na memória pelo programa anterior. Depois de ler os 10 inteiros, o programa deverá mostrar o vetor na tela e acessar o posicionamento final do arquivo mostrando seu valor.

```
#define TAM 10
int main()
{
    FILE *PontArq;
    int Idades[TAM], I;
    int Validacao;
    setlocale(LC_ALL, "Portuguese");
    PontArq = fopen("Dados.dat", "rb");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    printf("Posição inicial do arquivo: %d\n", ftell(PontArq));
    Validacao = fread(Idades, sizeof(int), TAM, PontArq);
    if(Validacao != TAM)
        printf("\nErro ao ler todos os dados!\n");
    else
    {
        for(I=0; I<TAM; I++)
        {
            printf("Idades[%d]: %d\n", I, Idades[I]);
        }
    }
    printf("O tamanho d arquivo é de %d Bytes.\n", ftell(PontArq));
    fclose(PontArq);
    system("Pause");
    return 0;
}
```

Posicionamento ao Longo do Arquivo

Independente do local onde nos encontramos no arquivo, é sempre possível voltar a colocar o ponteiro para apontar para o início do arquivo através do **rewind**, evitando fechar o arquivo e abri-lo novamente simplesmente para voltar ao início do arquivo.



A função mais importante de posicionamento num arquivo é a função **fseek**, pois nos permite apontar o ponteiro para qualquer posição do arquivo.

Posicionamento ao Longo do Arquivo

Sintaxe:

int fseek(FILE *Arq, long Salto, int Origem)

Onde:

Arq - Arquivo sobre o qual podemos operar.

Salto - Indica o número de bytes que pretendemos andar. Um valor positivo indica andar para frente, um valor negativo indica que pretendemos andar para trás.

Origem - Indica o local a partir do qual queremos realizar o salto no arquivo. São admitidos apenas três valores, que são definidos como constantes.

Constante	Valor	Significado
SEEK_SET	0	O salto é realizado a partir da origem do arquivo.
SEEK_CUR	1	O salto é realizado a partir da posição corrente do arquivo.
SEEK_END	2	O salto é realizado a partir do final do arquivo.

A função devolve 0 se o movimento foi realizado com sucesso dentro do arquivo, ou um valor diferente de 0, caso contrário.

Exemplo 4 - Posição do Arquivo

Programa que abre o arquivo Dados.dat e mostra quantos bytes ele tem, obtendo esse valor através de Acesso Direto, isto é, sem ter que percorrer todo o arquivo.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 10

int main()
{
    FILE *PontArq;
    int Idades[TAM], i;
    int Validacao;
    setlocale(LC_ALL,"Portuguese");
    PontArq = fopen("Dados.dat","rb");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    fseek(PontArq,0,SEEK_END); //OU fseek(PontArq,0,2);
    printf("O tamanho do arquivo é de %d Bytes.\n",ftell(PontArq));
    fclose(PontArq);
    system("Pause");
    return 0;
}
```

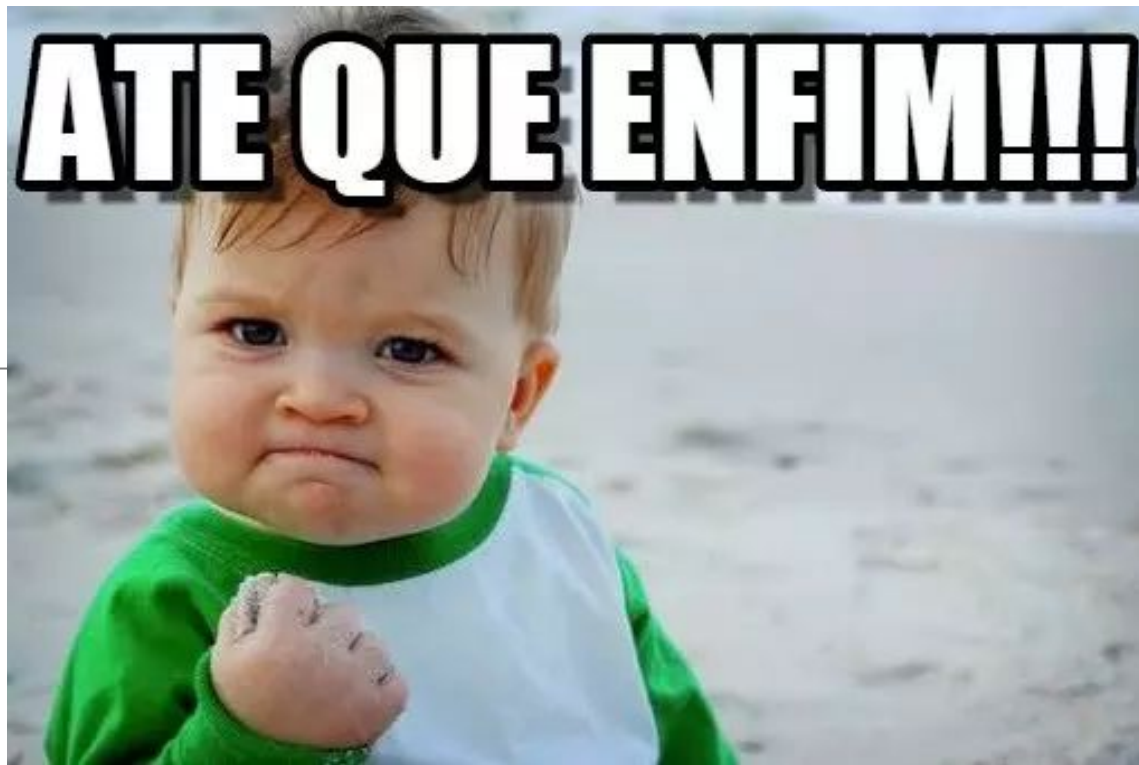
Exemplo 5 - Posição do Arquivo

Programa que abre o arquivo Dados.dat e solicita ao usuário qual a idade que ele deseja pesquisar no arquivo. Mostra também a primeira e última idade do arquivo.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 5

int main()
{
    FILE *PontArq;
    int Idades[TAM], l;
    int OrdemIdade, IdadePesquisada;
    setlocale(LC_ALL,"Portuguese");
    PontArq = fopen("Dados.dat","rb");
    if(PontArq == NULL)
    {
        printf("Impossível Abrir o arquivo!\n");
        exit(1);
    }
    printf("Qual a ordem da idade que deseja visualizar?\n");
    scanf("%d",&OrdemIdade);
    fseek(PontArq,(OrdemIdade-1)*sizeof(int),SEEK_SET);
    fread(&IdadePesquisada,sizeof(int),1,PontArq);
    printf("\nA %dª Idade do arquivo é: %d\n",OrdemIdade,IdadePesquisada);
    rewind(PontArq);
    fread(&IdadePesquisada,sizeof(int),1,PontArq);
    printf("\nA 1ª Idade do arquivo é: %d\n",IdadePesquisada);
    fseek(PontArq,- sizeof(int),SEEK_END);
    fread(&IdadePesquisada,sizeof(int),1,PontArq);
    printf("\nA 10ª Idade do arquivo é: %d\n",IdadePesquisada);
    fclose(PontArq);
    system("Pause");
    return 0;
}
```

ARQUIVOS DE REGISTROS (struct)



Arquivos de Registros (struct)

O processamento de structs armazenadas em arquivos é sempre realizado recorrendo a arquivos abertos em modo binário.

As structs são, em geral, lidas e escritas através de funções `fread` e `fwrite`.

Exemplo 6 - Programa dos Super Heróis

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#define TAM 3

struct HistoricoSuperheroi
{
    char Nome[40];
    int AnoCriacao;
};

int main()
{
    setlocale(LC_ALL, "Portuguese");
    int I, J, Aux;
    HistoricoSuperheroi SuperHerois[TAM];
    HistoricoSuperheroi Menor;
    FILE *PontArq;
    PontArq = fopen("ArquivoSuperHerois.dat", "w+b");
    if(PontArq == NULL)
    {
        printf("Impossível abrir o arquivo.\n");
        exit(1);
    }
```

Exemplo 6 - Programa dos Super Heróis

```
for(l=0; l<TAM; l++)
{
    printf("\nSuper Herói %d\n",l+1);
    printf("Nome: ");
    fflush(stdin);
    gets(SuperHerois[l].Nome);
    printf("Ano de criação: ");
    scanf("%d",&SuperHerois[l].AnoCriacao);
}
fwrite(SuperHerois,sizeof(HistoricoSuperheroi),TAM,PontArq);
```

Exemplo 6 - Programa dos Super Heróis

```
fclose(PontArq);
PontArq = fopen("ArquivoSuperHerois.dat","rb");
if(PontArq == NULL)
{
    printf("Impossível abrir o arquivo.\n");
    exit(1);
}
fread(SuperHerois,sizeof(HistoricoSuperheroi),TAM,PontArq);
fclose(PontArq);
for(l=0;l<TAM - 1; l++)
{
    Menor = SuperHerois[l];
    Aux = l;
    for(J=l+1; J < TAM; J++)
    {
        if(SuperHerois[J].AnoCriacao < Menor.AnoCriacao)
        {
            Menor = SuperHerois[J];
            Aux = J;
        }
    }
    SuperHerois[Aux]= SuperHerois[l];
    SuperHerois[l] = Menor;
}
fclose(PontArq);
```

Exemplo 6 - Programa dos Super Heróis

```
PontArq = fopen("ArquivoSuperHerois.dat","wb");
if(PontArq == NULL)
{
    printf("Impossível abrir o arquivo.\n");
    exit(1);
}
fwrite(SuperHerois,sizeof(HistoricoSuperheroi),TAM,PontArq);
PontArq = fopen("ArquivoSuperHerois.dat","r+b");
if(PontArq == NULL)
{
    printf("Impossível abrir o arquivo.\n");
    exit(1);
}
fread(SuperHerois,sizeof(HistoricoSuperheroi),TAM,PontArq);
printf("Ano de Criação - Nome\n");
for(l=0; l<TAM; l++)
{
    printf("    %d    -  %s\n ",SuperHerois[l].AnoCriacao,SuperHerois[l].Nome);
}
fclose(PontArq);
system("Pause");
return 0;
}
```