Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

30/09/2024

Aula anterior

Aula anterior

Alocação Dinâmica de Memória

```
int x;

x = 7;
```

```
int* x;
x = malloc( sizeof(int) );

*x = 7;
```

Aula anterior

Alocação Dinâmica de Memória

```
int x;

x = 7;
```

```
int A;
int* x;
x = &A ;

*x = 7;
A = 7 ;
```

Alocando memória de forma dinâmica

```
int main() {
      int *x;
      printf("\nO ponteiro x foi declarado na posição %p", &x);
      x = malloc( sizeof(int) );
       printf("\nOcupando %i bytes", sizeof(int));
      printf("\nE aponta para %p posição", x);

      *x = 7;
      printf("\nE guarda o valor %i", *x);
      return 0;
}
```

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29

Experimente

```
int main() {
	int A = 7;
	int *p;

	p = &A;

	// Escreva a posição de A
	// Escreva a posição do ponteiro p
	// Escreva para onde aponta o ponteiro p
	// Escreva o valor de A
	// Escreva o valor guardado no local para onde aponta p

	return 0;
}
```

Exercício

Exercício da Lista 18

Implemente o primeiro algoritmo que você estudou: a soma de dois números reais

Construa uma versão considerando ser proibido o uso de variáveis simples. Ao contrário, lhe é autorizado utilizar, unicamente, alocação dinâmica.

Reflita, com cuidado, a sintaxe de scanf( )

Ponteiro como implementação de passagem de parâmetro por referência

Ponteiro em passagem de parâmetro

Como visto, em C, a passagem de parâmetro por referência se dá através de ponteiro.

```
int a=7, b=2;
Teste(a, &b);
printf("%i,%i",a,b);
```

```
void Teste(int x, int *y)
{       x =  x * 2;
      *y = *y * 2;
}
```

Ordenando três caracteres

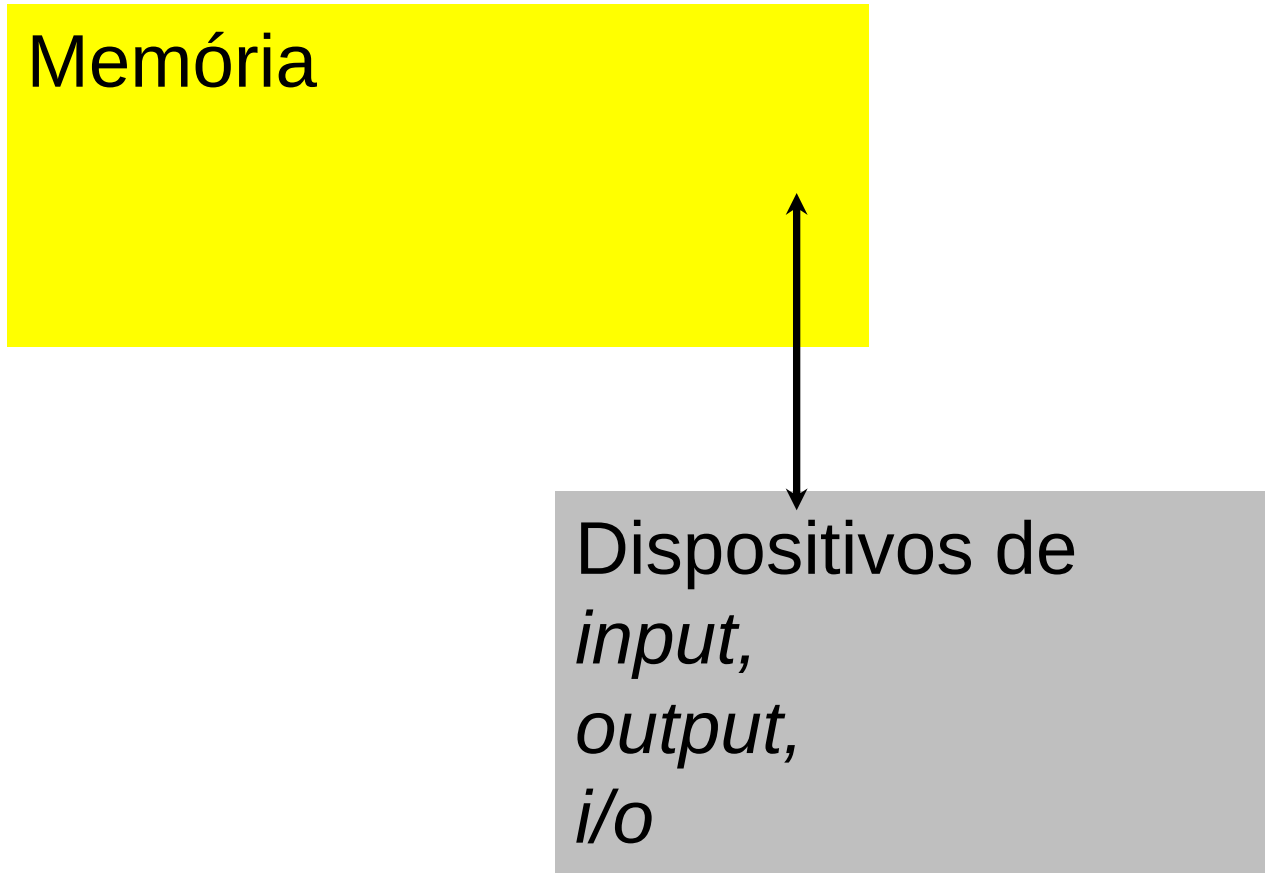
```
int main()
{    printf("Digite 3 letras: ");
     char c1; scanf("%c", &c1);
     char c2; scanf("%c", &c2);
     char c3; scanf("%c", &c3);
     if(c1>c2) Troca(&c1, &c2);
     if(c2>c3) Troca(&c2, &c3);
     if(c1>c2) Troca(&c1, &c2);
     printf("%c,%c ,%c",c1,c2 ,c3);
}
```

```
void Troca(char *x, char *y)
{        char aux = *x;
         *x = *y ;
         *y = aux;
}
```

Introdução aos arquivos

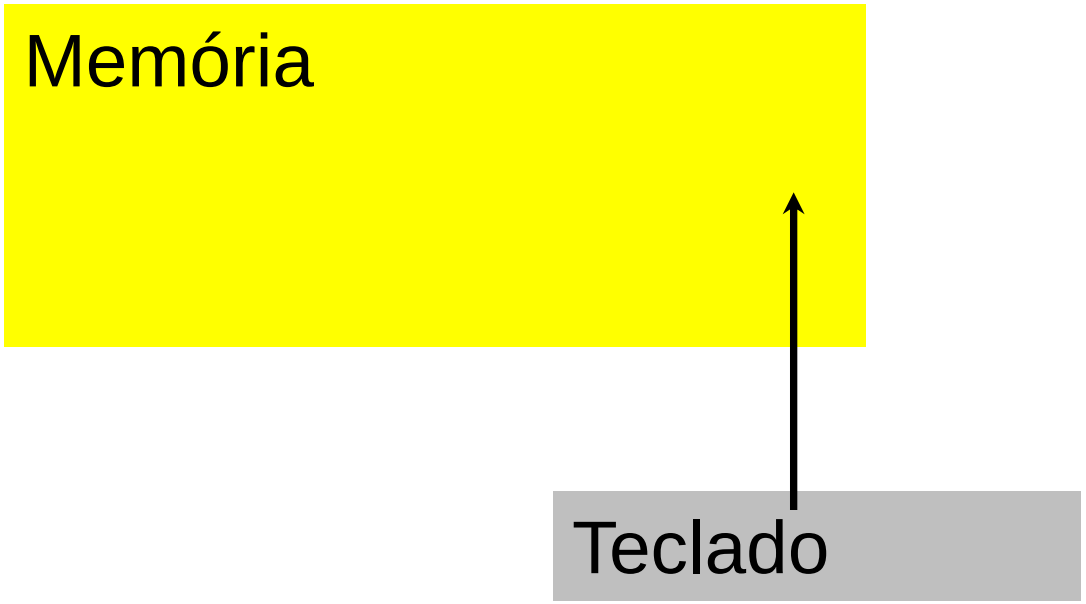
Fluxos de *i/o*

Observe que o dispositivo é apenas um meio



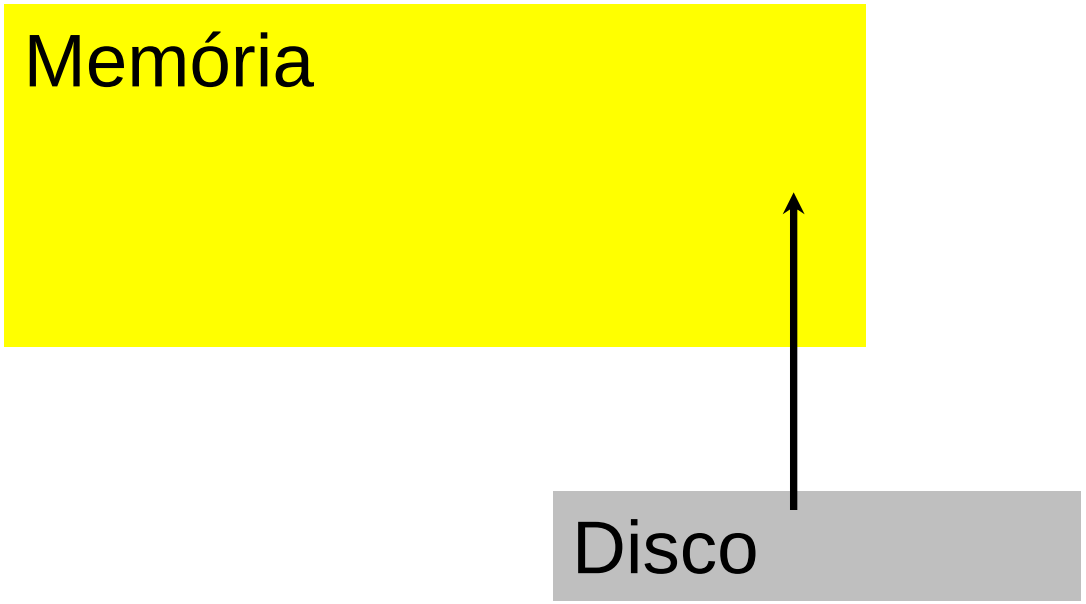
Fluxos de *i/o*

Por exemplo, um fluxo de *input* pode vir do teclado...



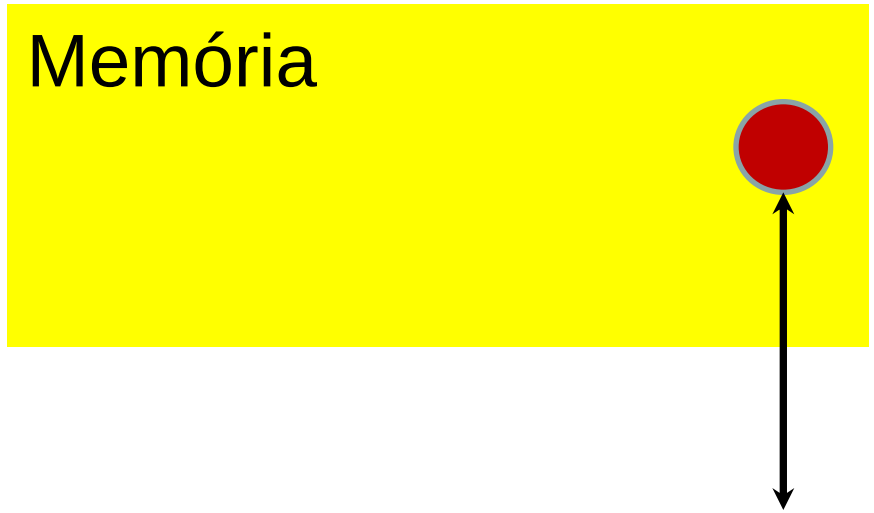
Fluxos de *i/o*

...ou de um arquivo armazenado em disco



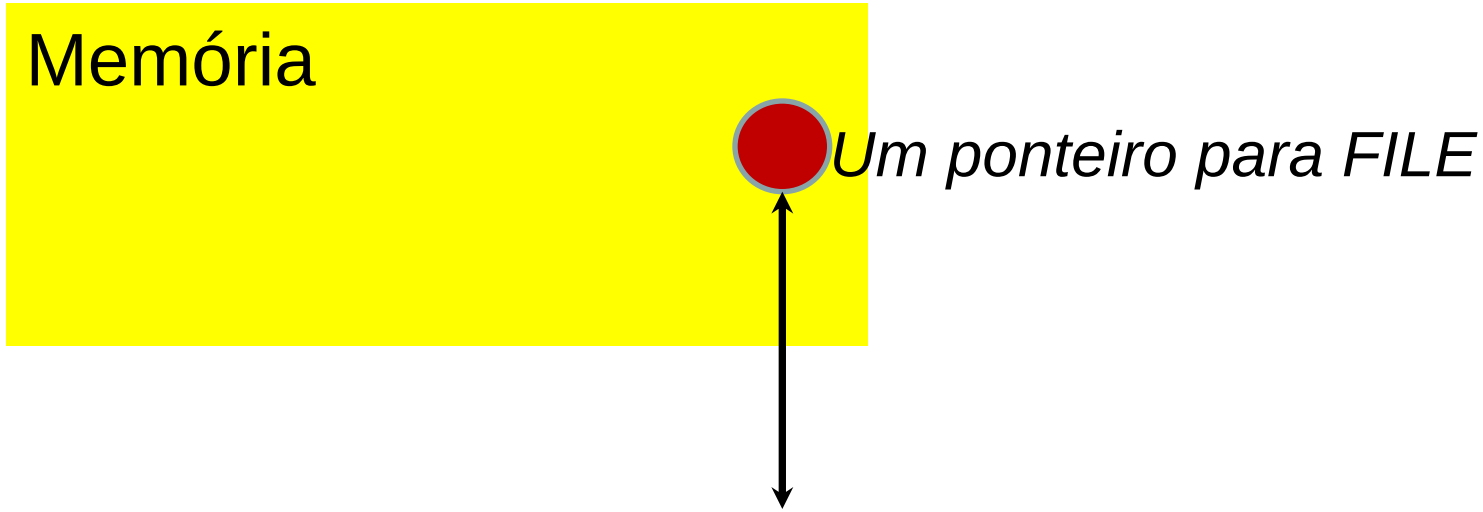
Fluxos de *i/o*

Logo, é necessário existir um objeto para estabelecer o contato com um fluxo de dados



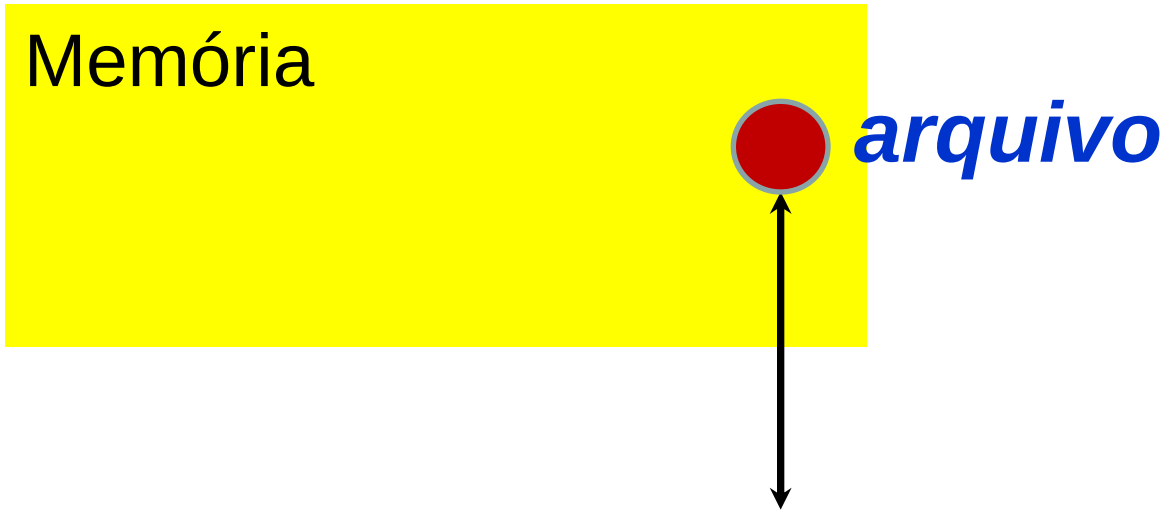
Em C, o tipo FILE provê este serviço

Uma ponteiro do tipo FILE aponta para um arquivo



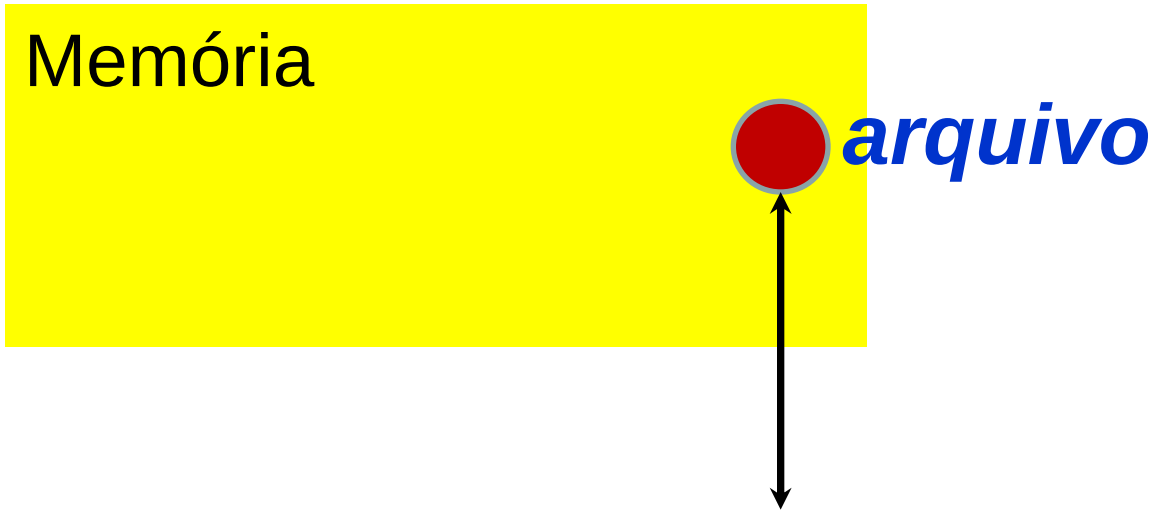
```
#include <stdio.h>

...
FILE* arquivo;
```



```
#include <stdio.h>

...
FILE* arquivo = fopen("nome", "modo");
```



Modos de abertura de um arquivo com fopen():

w Cria um fluxo para escrita no arquivo - *sobrescreve*

a Cria um fluxo para escrita no arquivo – *ao final*

r Cria um fluxo para leitura do arquivo – *NULL se não houver*

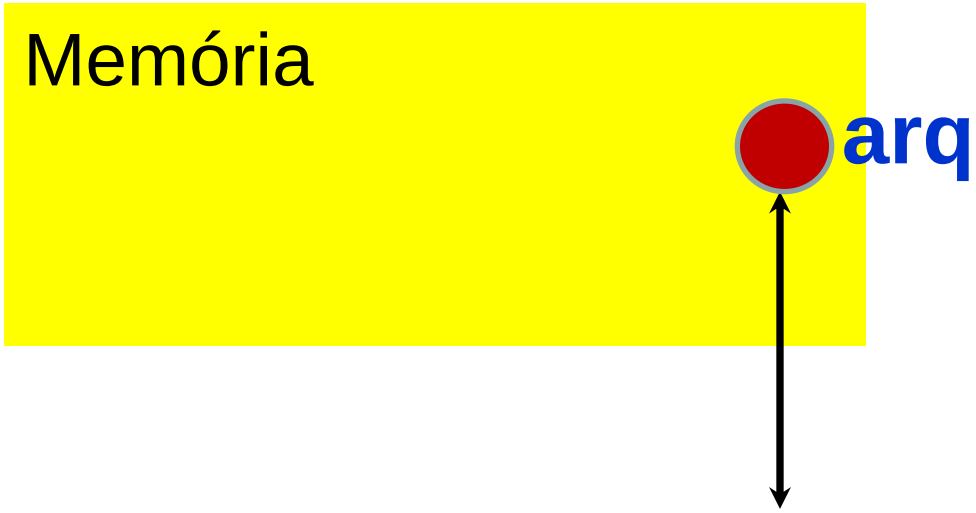
w+ Acrescenta permissão para leitura

a+ Acrescenta permissão para leitura

r+ Acrescenta permissão para escrita

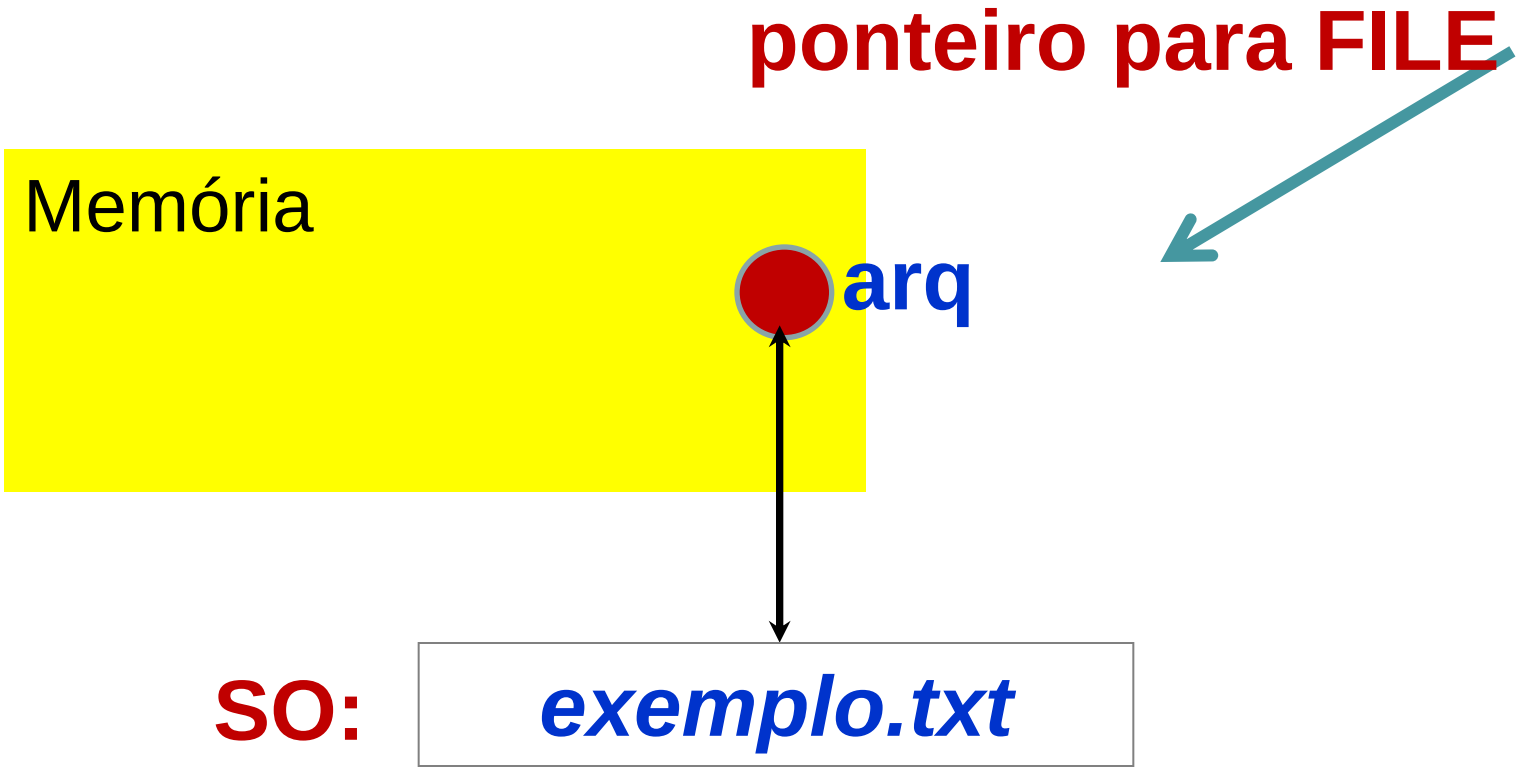
```
#include <stdio.h>

...
FILE* arq = fopen("exemplo.txt", "w");
```



Criando um novo arquivo

FILE* arq = fopen(*"exemplo.txt"*, *"w"*);



fprintf() – Acrescenta o ponteiro para FILE

```
printf(          “Hello World!\n");
fprintf(stdout, “Hello World!\n");

FILE *arq = fopen( "exemplo.dat", "w");
int x=10;
printf(          "%i\n", x);
fprintf(stdout, "%i\n", x);
fprintf(     arq, "%i\n", x);
```

fprintf() – Acrescenta o ponteiro para FILE

```
printf(          “Hello World!\n");
fprintf(stdout, “Hello World!\n");

FILE *arq = fopen( "exemplo.dat", "w");

for(int x=10; x<=20; x+=2){
    fprintf(arq, "%i\n", x);
}
```

fclose() – Fecha o arquivo

```
FILE *arq;
arq = fopen( "exemplo.dat", "w");

for(int x=10; x<=20; x+=2){
    fprintf(arq, "%i\n", x);
}

fclose(arq);
```