

VARIAVEIS

VARIAVEIS

TIPOS

Tipo	Num de bits	Formato i/o	Ínicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
int	32	%d	-2.147.483.648	2.147.483.647
unsigned int	32	%u	0	4.294.967.295
long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
float	32	%f	(+/-)10 ⁻³⁸	(+/-)10 ³⁸
double	64	%lf	(+/-)10 ⁻³⁰⁸	(+/-)10 ³⁰⁸
long double	96			

	bits	mantissa	exponent	sign
character	8	7	0	1
long integer	32	31	0	1
float	32	23	8	1
double	64	52	11	1
long double	96			

VARIÁVEIS

INICIALIZAÇÃO

```
1 #include <stdio.h>
2
3
4 int main(void){
5     int evento ;
6     char corrida;
7     float tempo;
8
9     evento = 5;
10    corrida = 'C';
11    tempo = 27.25;
12
13    printf("O tempo vitorioso na eliminat'oria %c",corrida);
14    printf("\nda competi,c~ao %d foi %f.", evento, tempo);
15
16 return 1;
17 }
```

ATRIBUIÇÃ

```
1 #include <stdio.h>
2
3 int main(void){          DECLARAÇÃO
4
5     int evento = 5 ;
6     char corrida = 'C';
7     float tempo = 27.25;
8
9     printf("O tempo vitorioso na eliminat'oria %c",corrida);
10    printf("\nda competi,c~ao %d foi %f.", evento, tempo);
11
12 return 1;
13 }
```

DECLARAÇÃO

VARIÁVEIS

INICIALIZAÇÃO

```
1 #include <stdio.h>
2
3
4 int main(void){
5     int evento ;
6     char corrida;
7     float tempo;
8
9     evento = 5;
10    corrida = 'C';
11    tempo = 27.25;
12
13    printf("O tempo vitorioso na eliminat'oria %c",corrida);
14    printf("\nda competi,c~ao %d foi %f.", evento, tempo);
15
16 return 1;
17 }
```

ATRIBUIÇÃ

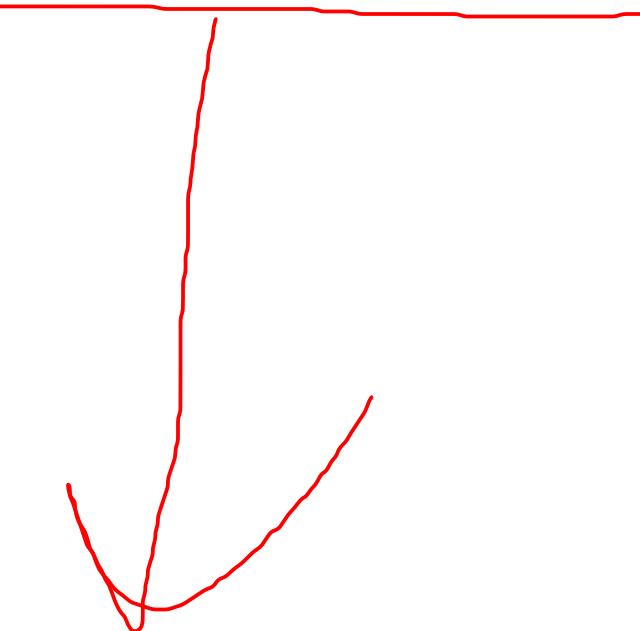
```
1 #include <stdio.h>
2
3 int main(void){          DECLARAÇÃO
4
5     int evento = 5 ;
6     char corrida = 'C';
7     float tempo = 27.25;
8
9     printf("O tempo vitorioso na eliminat'oria %c",corrida);
10    printf("\nda competi,c~ao %d foi %f.", evento, tempo);
11
12 return 1;
13 }
```

DECLARAÇÃO

VARIAVEIS

NOME DAS VARIAVEIS

- O nome das variaveis pode ser qualquer palavra que nao seja uma palavra chave da linguagem.
- E possivel conter um n'umero na palavra: Casal
- Nao é aceitavel iniciar com um numero: 1casa (errado)
- E possivel utilizar subscrito: Casa_da_ana
- Nao pode-se utilizar: { (+ - * / ; . , ?



auto	double	int	struct
break	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

COMPILADORES

2 - COMPILADOR

- Compilar é transformar o código fonte que é texto em linguagem de máquina. Ou em um executável para o sistema operacional executar esse programa.
- O compilador da linguagem C (gcc) , faz a leitura do código fonte que você escreveu, verifica quais são as bibliotecas que você importou e está utilizando no projeto, e , no caso da linguagem C, gera um executável para ao sistema operacional na qual está sendo compilado.
- A linguagem C não é multiplataforma, um programa meu escrito em C e compilado no Linux só irá funcionar no Linux, não irá funcionar no mac ou no windows por exemplo.
 - Agora se pergarmos o código-fonte e compilar lá no windows irá funcionar, é o processo de compilação que transforma o código-fonte em código máquina para o sistema operacional que está utilizando.
- Para compilar um programa pela linha de comando
 - (criar executável)
`gcc nome_do_arquivo -o nome_do_executavel_gerado`
 - (executar executável)
`./programa1.exe`
- No windows basta digitar `p1.exe` ou no powershell `./p1.exe`

ESTRUTURAS **DE DECISÃO**

OPERADORES ARITIMETICOS

Operador	Visualg	Linguagem C
igualdade	=	==
Maior que	>	>
Menor que	<	<
Maior ou igual	>=	>=
Menor ou igual	<=	<=
diferente	<>	!=

**Não esqueça que em C o sinal de igual é atribuição de valor
= (em C) é o mesmo que <- (visualg)**

OPERADORES LÓGICOS

Operador	Visualg	Linguagem C
E	e	&&
Ou	ou	
Não	nao	!

IF

```
se (condição for Verdade) então
    //comando1;
    /*ou bloco de comandos;*/
fim se;
```

```
if (condição) //verdade
{
    //comando1;
    /*ou bloco de comandos;*/
}
```

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

int main()
{
    int x;

    printf("Digite um numero inteiro qualquer\n");
    scanf("%d",&x);
    if ((x>0) && (x<10))
    {
        printf("O numero %d e maior que 0 e menor que 10 a 0\n",x);
    }
    system("pause");
}
```

```
printf("Digite um numero inteiro qualquer\n");
scanf("%d",&x);
if (x>=0)
{
    printf("O numero %d e maior ou igual a 0\n",x);
}
system("pause");
}
```

```
printf("Digite um numero inteiro qualquer\n");
scanf("%d",&x);
if ((x==0) || (x==10))
{
    printf("O numero digitado e = a 0 ou = 10\n",x);
}
system("pause");
}
```

IF...ELSE

```
se (condição for Verdade) então  
    comando1;  
    {ou bloco de comandos};  
senão  
    comando1;  
    {ou bloco de comandos};  
fim se;
```



```
if (condição ) verdade  
{  
    comando1;  
    {ou bloco de comandos};  
}  
else //falso  
{  
    comando1;  
    {ou bloco de comandos};  
}
```

```
printf("Digite um numero inteiro qualquer\n");  
scanf("%d",&x);  
if (x<0)  
{  
printf("O numero %d e negativo\n",x);  
}  
else  
{  
printf("O numero %d e positivo\n",x);  
}  
system("pause");  
}
```

IF'S... ANINHADOS

```
se (condição1 for Verdade) então
  se (condição2 for Verdade) então
    se (condição3 for Verdade) então
      //comando1;
      /*ou bloco de comandos*/
    fim se;
  fim se;
fim se;
```

```
if (condição) //verdade
{
  if (condição2) //verdade;
  {
    if (condição3) //verdade ;
    {
      //comando1
      /*ou bloco de comandos*/
    }
  }
}
```

```
printf("Digite um numero inteiro qualquer\n");
scanf("%d",&x);
if (x>0)
{
  if (x>200)
  {
    if (x<202)
    {
      printf("O numero digitado e 201\n\n");
    }
  }
}
system("pause");
}
```

```
int x;

printf("Digite um numero inteiro qualquer\n");
scanf("%d",&x);
if (x>0)
  if (x>200)
    if (x<202)
      printf("O numero digitado e 201\n\n");
system("pause");
}
```

IF..ELSE ANINHADOS

```
se (condição1 for Verdade) então
    /*bloco de comandos*/
Senao
    se (condição2 for Verdade) então
        /*bloco de comandos*/
    senao
        se (condição3 for
            Verdade) então
                /*bloco de comandos*/
            senao
                /*bloco de comandos*/
            fimse
        fimse
fimse
```

```
if (condição1) //verdade;
{
    /*bloco de comandos*/
}
else if (condição 2) \\\verdade
{
    /*bloco de comandos*/
}
else if (condição 3) //verdade
{
    /*bloco de comandos*/
}
else {
    /*bloco de comandos*/
}
```

```
int x;
printf("Escolha o codigo do produto\n");
printf("1 - Vestuario\n");
printf("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
printf("4 - Produto nao perecivel\n");
scanf("%d",&x);

if (x==1)
{
    printf("Voce quer comprar uma blusa?\n");
} else if(x==2){
    printf("Voce quer comprar um creme dental?\n");
} else if(x==3) {
    printf("Voce quer comprar um kg de carne?\n");

}else if(x==4){
    printf("Voce quer comprar uma lata de oleo ?\n");
}
system("pause");
```

1

```
printf("Escolha o codigo do produto\n");
printf("1 - Vestuario\n");
printf("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
printf("Qualquer outro valor - Produto Indisponível\n");
scanf("%d",&x);

if (x==1)
    printf("Voce quer comprar uma blusa?\n");
else if(x==2)
    printf("Voce quer comprar um creme dental?\n");
else if(x==3)
    printf("Voce quer comprar um kg de carne?\n");

else
    printf("Produto indisponivel ?\n");

system("pause");
```

2

```
printf("Escolha o codigo do produto\n");
printf("1 - Vestuario\n");
printf("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
printf("Qualquer outro valor - Produto Indisponível\n");
scanf("%d",&x);

if (x==1)
    printf("Voce quer comprar uma blusa?\n");
else if(x==2){
    printf("Voce quer comprar um creme dental?\n");
    printf("Voce quer comprar um creme dental11111?\n");
}
else if(x==3)
    printf("Voce quer comprar um kg de carne?\n");

else
    printf("Produto indisponivel ?\n");

system("pause");
```

3

←

Obrigatório a utilização de chaves
Se a condição tiver mais do que 1 instrução

SWITCH

```
escolha (X)
caso 1:
    /*bloco de comandos*/
caso 2:
    /*bloco de comandos*/
caso 3:
    /*bloco de comandos*/
caso Contrário: //pode ser omitido
    /*bloco de comandos*/
fim _escolha;
```

```
switch (X)
{
    case 1:
        /*bloco de comandos*/
        break;
    case 2:
        /*bloco de comandos*/
        break;
    case 3:
        /*bloco de comandos*/
        break;
    default: //pode ser omitido
        /*bloco de comandos*/
        break;
}
```

```
printf("Escolha o codigo do produto\n");
printf("1 - Vestuario\n");
printf("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
scanf("%d",&x);
switch (x)
{
    case 1:
        printf("Voce quer comprar uma blusa?\n");
        break;
    case 2:
        printf("Voce quer comprar um creme dental?\n");
        break;
    case 3:
        printf("Voce quer comprar um kg de carne?\n");
        break;
    default :
        printf("Codigo invalido ?\n");
        break;
}
```

```
#pseudocodigo
escola(variavel)
Inicio
    caso valor1;
    caso valor2;
        instruções
    ..
    caso valorN;
fim
```

Linguagem C

```
switch(variavel){
    case valor1:
        instruções
        break;

    case valor 2:
        instruções
        break;
    default:
        instruções;
}
```

```
#include <stdio.h>

int main()//inicio_main
{
    //declaração_de_variaveis
    int valor;

    //entrada_dados
    printf("Digite um valor de 1 a 7:\n");
    scanf("%d",&valor);

    //processamento_dados
    switch(valor){
        case 1:
            printf("Domingo\n");
            break;
        case 2:
            printf("Segunda\n");
            break;
        case 3:
            printf("Terça\n");
            break;
        case 4:
            printf("Quarta\n");
            break;
        case 5:
            printf("Quinta\n");
            break;
        case 6:
            printf("Sexta\n");
            break;
        case 7:
            printf("Sábado\n");
            break;
        default:
            printf("Valor invalido!\n");
    }
}

//fim_switch
//fim_main
```

ESTRUTURAS **DE REPENÇÃO**

FOR

```
// Estrutura de repetição FOR      SantaKaya, a month ago • ADD(p5.c) ALT(p4.c)

/* Utilizado o FOR
   Faça um programa no qual receba e some 5 numeros inteiros e apresente a soma no final.
*/

#include <stdio.h>

int main(){

    //declaração_variaveis
    int numero, soma = 0; //Inicializa soma para não receber lixo

    //inicio_for
    for (int i = 0; i < 5; i++){//para o int i iniciando em 0; enquanto i < 5; incremente i em 1.

        //entrada_dados
        printf("Informe um numero:\n");
        scanf("%d",&numero);

        //processamento_dados
        soma = soma + numero;
    }//fim_for

    //saída_dados
    printf("A soma eh: %d",soma);

}

return 0;
```

WHILE

```
// Estrutura de repetição while
/*
Utilizado quando voce precisa de m loop, onde não se tenha um numero fixo de elementos, mas que se tenha
um criterio de parada. E antes de iniciar o loop, a condição é checada.

PROBLEMA : Faça um progama no qual receba e some numeros inteiros até que o número de entrada seja 0.
*/
#include <stdio.h>

int main(){
    //declaração_variaveis
    int numero, soma = 0;

    //entrada_dados
    printf("Informe um Numero:\n");
    scanf("%d",&numero);

    //processamento_dados
    while(numero != 0){ //ini_while
        soma = soma + numero;

        //entrada_dados
        printf("Informe um Numero");
        scanf("%d",&numero);
    } //fim_while

    //saída_dados
    printf("A soma eh: %d", soma);
    return 0;
}
```

DO...WHILE

```
//Estrutura de repetição do..while      SantaKaya, a month ago • ADD p6.c

/*
Utilizado quando você precisa de um loop onde não se tenha um numero fixo de elementos mas que
tenha um criterio de parada e a condição de parada é checada após a primeira execução.

PROBLEMA:
Faça um progama, no qual receba e some numeros inteiros até que o numero de entrada seja 0 e apresente
a soma no final:
*/
#include <stdio.h>

int main(){

    //declaração_variavel
    int numero, soma = 0;

    //entrada_dados
    do{
        //entrada
        printf("Informe um numero:\n");
        scanf("%d",&numero);

        //processamento
        soma = soma + numero;
    }while(numero != 0);

    //saída_dados
    printf("A soma eh: %d",soma);

    return 0;
}
```

TIPOS DE **DADOS**

TIPOS NUMERICOS

- Os tipos numericos são subdivididos em duas categorias, os tipos inteiros e os tipos reais.
- Numeros do tipo INTEIRO - 1,2 989...
 - Declaração : int (%d)
- Numeros do tipo REAL - 1.2, 2.3, 90.7 - Declaração : float (%d)
 - Declaração : double(%lf)
 - O double é um float maior, tbm chamado de LONG FLOAT
- Como saber qual variavel declarar?
 - Toda vez que vc for usar uma variavel que recebera numeros decimais, vc a declara como float.
 - Se for numeros inteiros, declara como int.

TIPOS NUMERICOS

- Os tipos numericos são subdivididos em duas categorias, os tipos inteiros e os tipos reais.
- Numeros do tipo **INTEIRO** - 1,2 989...
 - Declaração : int (%d)
- Numeros do tipo **REAL** - 1.2, 2.3, 90.7
 - Declaração : float (%d)
 - Declaração : double(%lf)
 - O double é um float maior, tbm chamado de LONG FLOAT
- Como saber qual variavel declarar?
- Toda vez que vc for usar uma variavel que recebera numeros decimais, vc a declara como float.
- Se for numeros inteiros, declara como int.

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

int main(){
    //declarando_variaveis

    //inteiro
    int numero_inteiro; //7,890...

    //real
    float nota1,nota2; // 23.4,1.23,...9999999
    double media; //23.4, 1.23,...999999999999999

    //entrada_dados
    printf("Qual a primeira nota?\n");
    scanf("%f",&nota1);

    printf("Qual a segunda nota?\n");
    scanf("%f",&nota2);

    //processamento_dados
    media = (nota1 + nota2)/2;

    //saida_dados
    printf("Sua media eh: \n%.2lf",media);

}

return 0;
```

TIPOS ALFANUMERICOS

- São formados por caracteres e strings.
- Na linguagem C não existe o tipo de dados STRING .
- Na Programação tudo que estiver entre aspas duplas "dasd asdasd dasd" é uma string.
- CARACTERE(unidade)
 - Na progama em C, é tudo que possui aspas simples 'c'.
 - Declaração : char(%c)
- Tabela ASCII , diz respeito dos caracteres na computação, inicia do 0 a 127.
- O 0 em char é equivalente ao [null] e assim por diante.- o a = 97 e o z = 128
- Isso significa para a gente que podemos gerar um alfabeto completo com um loop
- Vamos criar um progama chamado loop_alfabeto.c
- Quando mudamos para %d, aparece a contagem dos numeros ate o 122

TIPOS ALFANUMERICOS

TIPOS ALFANUMERICOS

```
/*
Alfabeto, tabela ASCII, com loop
97 = a | 122 = z
*/
#include <stdio.h>
#include <stdio.h>

int main(){

//declarando_variaveis

//entrada_dados
for(int i = 97 ; i<=122;i++){
    printf("%d\n",i);
}
//processamento_dados

//saida_dados

return 0;
}
```

```
/*
You, a month ago • ADD(p08.c | p09.c | tipos_num)
Tipos de Dados

TIPOS ALFANUMERICOS
- Caracteres
- Strings
*/
#include <stdio.h>
#include <stdio.h>

int main(){

//declarando_variaveis

char nome[50];//49 char + /0 caractere finalizador.

//entrada_dados
printf("Qual eh o seu nome?\n");
gets(nome);
//processamento_dados

//saida_dados
printf("Seu nome eh %s",nome);

return 0;
}
```

```
/*
Tipos de Dados

- Tipos Alfanumericos
- Caracteres;
    's';
- Strings
    "auihsduiahsduasd asddas asdasd asdd";
* Na linguagem C, não existe o tipo ded dados String.

*/
#include <stdio.h>
#include <stdio.h>

int main(){

//declarando_variaveis
char opcao;

//entrada_dados
printf("Informe uma opcao:\n");
printf("a - Saldo da conta.\n");
printf("b - Extrato da conta.\n");
printf("c - Limite da conta.\n");

scanf("%c",&opcao);
//processamento_dados
if(opcao == 'a'){//inicio_if1
    printf("Seu saldo eh....");
}else if (opcao == 'b'){//inicio_if2
    printf("Extrato da conta eh...");}
else if(opcao == 'c'){//inicio_if3
    printf("Seu limite eh..");}
else{
    printf("Opcao desconhecida..");
}

return 0;
}
```

TIPOS BOOLEANOS

- Os dados booleanos vêm da lógica de Boole, estudado na informática..
- Existem somente 2 tipos de dados (VERDADEIRO | FALSO)
- Em C, não existe um tipo de dados boolean, mas a linguagem C reconhece:
 - 0 como FALSO
 - x != 0 como Verdadeiro
- Geralmente vemos esse formato:

```
4-int main(){  
    int booleano = 1;  
  
    if(booleano == 1){  
        printf("Verdadeiro...");  
    }else{  
        printf("Falso...");  
    }  
  
    return 0;  
}
```

```
//declarando_variaveis  
// variavel inicializada inteira  
//int booleano = 1; //verdadeira  
//int booleano = 0; //falso  
//int booleano = 2; //verdadeiro  
//int booleano = -1; //verdadeiro
```

PULO DO

- Vimos que a linguagem C reconhece o 0(falso) e X!=0(verdadeiro).
- Se booleano = 1, logo X!=0(verdadeiro)

```
4-int main(){  
    int booleano = 1;  
    if(0){  
        printf("Verdadeiro...");  
    }else{  
        printf("Falso...");  
    }  
  
    return 0;  
}
```

TIPOS BOOLEANOS

```
/*
    TIPOS DE DADOS BOOLEANOS
*/

#include <stdio.h>
#include <stdio.h>

int main(){

    //declarando_variaveis
    // variavel inicializada inteira
    //int booleano = 1; //verdadeira
    //int booleano = 0; //falso
    //int booleano = 2; //verdadeiro
    int booleano = -2;

    //entrada_dados
    if(booleano){// A linguagem C reconhece booleano =1 como verdadeiro?
        printf("Verdadeiro..");
    }else{
        printf("Falso..");
    }
    You, a month ago • ALT e ADD
    return 0;
}
```

OPERAÇÕES MATEMATICAS

```
/* OPERAÇÕES MATEMATICAS

SOMAR +
SUBTRAIR -
MULTIPLICAR *
DIVIDIR /
ELEVAR AO QUADRADO X ** 2 ou x*x
módulo (resto da divisão de x por y - par/impar) %

*/
```

```
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S6(tipos_dados)\4-OP_
MATEMATICAS> cmd /c .\"p11.exe"
num1 = 3
num2 = 7
num2 + num1 = 10
num2 - num1 = 4
num2 * num1 = 21
num2 / num1 = 2
num1 ** &num1 = 9
num1 * num1 = 9
num2 num1 = 3 eh impar num2 = 7 eh impar
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S6(tipos_dados)\4-OP_
MATEMATICAS>
```

- Divisão de numeros inteiros não geram o resultado do ponto flutuante, logo temos que aplicar o processo de CASTING.

```
//Dividir numeros inteiros - CAST
res = (float)num1 / (float)num2;
printf("num1 / num2: %f\n",res);
```

- O CAST é a conversão do tipo de um numero para trabalhar na execução de uma função no progama, sem mudar seu tipo declarado.

[(float)int = float]
[(int)float= int]

- Para ao fazer isso voce troca o tipo das variaveis posteriores, logo cuidado. Para resolver, use o cast nas variaveis.

OPERAÇÕES MATEMATICAS

```
//Dividir numeros inteiros - CAST  
res = (float)num1 / (float)num2;  
printf("num1 / num2: %f\n",res);
```

- Faça a multiplicação de num1 * num2 , e me retorne um valor inteiro.
 - (res) : declarado nopal começo como FLOAT.

```
//Multiplicar  
res = num2 * num1;  
printf("num2 * num1 = %d\n", (int)res);
```

OPERAÇÕES MATEMATICAS

```
int main(){  
    //declarando variaveis  
    int num1 = 3, num2 = 7;  
    float res;  
  
    printf("num1 = %d\nnum2 = %d\n", num1, num2);  
    //Soma  
    res = num2 + num1;  
    printf("num2 + num1 = %d\n", (int)res);  
    //Subtrair  
    res = num2 - num1;  
    printf("num2 - num1 = %d\n", (int)res);  
    //Multiplicar  
    res = num2 * num1;  
    printf("num2 * num1 = %d\n", (int)res);  
    //Dividir  
    res = num2 / num1;  
    printf("num2 / num1 = %f\n", res);  
    //Dividir  
    res = num1 / num2;  
    printf("num1 / num2 = %f\n", res);  
    //Dividir numeros inteiros - CAST  
    res = (float)num1 / (float)num2;  
    printf("num1 / num2: %f\n", res);  
    //Elevar ao Quadrado  
    res = num1 ** &num1;  
    printf("num1 ** &num1 = %d\n", (int)res);  
    //Elevar ao quadrado  
    res = num1 * num1;  
    printf("num1 * num1 = %d\n", (int)res);  
    //Modulo - 3/2 = 1 && 3%2= 1(o que sobrou)  
    res = num2 % num1;  
    printf("num2 modulo num1 = %f\n", res);  
    //Modulo - Verificação PAR ou IMPAR  
    if(num1 % 2 == 0){  
        printf("num1 = %d eh PAR\n", num1);  
    }else{  
        printf("num1 = %d eh impar\n", num1);  
    }  
    if(num2 % 2 == 0){  
        printf("num2 = %d eh PAR\n", num2);  
    }else{  
        printf("num2 = %d eh impar\n", num2);  
    }  
  
    return 0;  
}
```

VETORES

DECLARANDO VARIAVEIS

- Vimos que a linguagem C não possui o Tipos String. Mas que podemos criar um ARRAY de caracteres para trabalhar com strings.
- Quando falos de VETOR, estmos falando de um ARRAY UNIDIMENSIONAL, ou seja, uma so dimensão.

char nome[50]

```
//declarando_variaveis

//vetore e string
char nome[50];
//vetores e caracteres
char letras[26];

//vetores de inteiros
int numeros[10];

//vetores e reais
float valores[5];
```

VETORES E STRINGS

- Strings, qualquer coisa entre aspas duplas "asd"

```
//vetore e string
char nome[50];
printf("Qual seu nome?\n");
gets(nome);
printf("Olah %s", nome);
```

VETORES E CARACTERES

- Vimos que 1 caractere é qualquer coisa dentro de aspas simples 'a' 'l'
- O caractere pode ser uma letra e tbm um numerochar l = 'l' char n = 97;
- Na tabela ASCII o numero 97 = a
- Vimos que letras é um vetor de caracteres com 26 posições
 - Primeira posição = 0
 - Ultima = 25(n-1)

```
//vetores e caracteres
char letras[26];
// 'b'
int contador = 0;
for (int i = 97 ; i <= 122 ; i++){
    letras[contador] = i;
    contador++;
}
```

- Como a primeira posição é 0, temos que colocar o valor dela como sendo 0.
- Por isso criamos o contador.
int contador = 0;
letras[contador]=i;
-> Letras[0] ira receber o valor de i = 97 = a

VETORES E CARACTERES

- Depois o contador será incrementado, avançando um posição do vetor letras, letras[1]
- Continua o loop..ate o limite.

```
//imprimindo as letras e seus valores em decimal.  
for (int i = 0 ; i < 26; i++){  
    printf("%d == %c\n",letras[i],letras[i]);  
}
```

- Agora imprimindo as letras:

%d = decimal

%c = caractere

```
Qual seu nome?  
gabi  
Olah gabi  
97 == a  
98 == b  
99 == c  
100 == d  
101 == e  
102 == f  
103 == g  
104 == h  
105 == i  
106 == j  
107 == k  
108 == l  
109 == m  
110 == n  
111 == o  
112 == p  
113 == q  
114 == r  
115 == s  
116 == t  
117 == u  
118 == v  
119 == w  
120 == x  
121 == y  
122 == z  
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S7(vetores_matrizes)\1-VETORES  
> █
```

VETORES E INTEIROS

- Funciona da mesma forma dos vetores de caracteres em questão de posições.
[10] = 0...9 DECLARAÇÃO MANUAL

```
//vetores de inteiros
int numeros[6];//0...5
numeros[0] = 1;
numeros[1] = 3;
numeros[2] = 5;
numeros[3] = 7;
numeros[4] = 9;
numeros[5] = 2;      You, a few
```

VETORES E REAIS

```
//vetores e reais
float valores[5];//0...4
for(int i = 0; i<= 5; i++){
    valores[i] = numeros[0] / 2;      cast
} //fim_for preenchimento      You, a few seconds ago • Uncommitted
                                vetor numeros para preencher
                                problema, valores inteiros retornam inteiros
                                erro

for(int i = 4 ; i > 0; i--){//imprimindo ao contrario.
    printf("%.2f\n",valores[i]);
}
```

```
>
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S7(vetores_matrizes)\1-VETORES
> cd "c:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S7(vetores_matrizes)\1-VETORES"
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S7(vetores_matrizes)\1-VETORES
> cmd /c ."p13.exe"
4.50
3.50
2.50
1.50
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S7(vetores_matrizes)\1-VETORES
> █
```

```
21
22
23 //vetores e reais
24 float valores[5];//0...4
25 for(int i = 0; i<= 5; i++){
26     valores[i] = (float)numeros[i] / (float)2;      cast
27 } //fim_for preenchimento      You, a few seconds ago • Uncommitted changes
28
29 for(int i = 4 ; i > 0; i--){//imprimindo ao contrario.
30     printf("%.2f\n",valores[i]);
31 }
```

MATRIZES

MATRIZES

- Vetores são array uni-dimensionais
- ARRAY UNI(vetores) int numeros[5];
[0][1][2][3][4]
- Matrizes são arrays multi-dimensionais
- ARRAY MULTI(matrizes) int numeros[5][5] [|linhas][|colunas];

```
[00][01][02][03][04]  
[10][11][12][13][14]  
[20][21][22][23][24]  
[30][31][32][33][34]  
[40][41][42][43][44]
```

- 5 Linhas e 5 Colunas
- Numa imagem, quando vemos o pixel, temos a perfeita representação de uma matrix.
512x512

MATRIZES

```
//declarando_variaveis  
  
char nome[3][50];  
  
//entrada_dados  
  
for(int i = 0 ; i < 3; i++){  
    printf("Qual seu nome?\n");| You, 3 minutes ago • Update p14.c  
    gets(nome[i]);// pede o nome ao usuario 3x e guarda nas posições indicadas  
}  
  
for(int i = 0; i < 3; i++){  
    printf("Olah %s\n",nome[i]);// saudação aos nomes digitados.  
}  
  
//processamento_dados  
  
//saída_dados
```

MATRIZES

```
//vetores de inteiros
/* matrix

    [00][01]
    [10][11]
*/
//DECLARAÇÃO
int numeros[2][2];
numeros[0][0] = 1;
numeros[0][1] = 2;
numeros[1][0] = 3;
numeros[1][1] = 4;

//IMPRESSÃO
for(int i = 0; i < 2 ; i++){//for_percorrer_LINHAS
    for(int j = 0; j < 2 ; j++){//for_percorrer_COLUNAS
        printf("%d",numeros[i][j]);
    }
}
```

MATRIZES

```
//vetores de inteiros
/* matrix

    [00][01]
    [10][11]
*/
//DECLARAÇÃO
int numeros[2][2];
numeros[0][0] = 1;
numeros[0][1] = 2;
numeros[1][0] = 3;
numeros[1][1] = 4;

//IMPRESSÃO
for(int i = 0; i < 2 ; i++){//for_percorrer_LINHAS
    for(int j = 0; j < 2 ; j++){//for_percorrer_COLUNAS
        printf("%d",numeros[i][j]);
    }
}
```

MATRIZES

```
//IMPRESSÃO
    for(int i = 0; i < 2 ; i++){//for_percorrer_LINHAS
        for(int j = 0; j < 2 ; j++){//for_percorrer_COLUNAS
            printf("numeros[%d][%d] vale %d\n", i, j,numeros[i][j]);
        }
    }
```

You, a few seconds ago * Uncommitted changes

```
PS C:\Users\dabi\Documents\TEORIA_INDIVIDUAL\UDEM
Y\REP_UDEM\PR00_C\SL7(vetores_matrizes)\2-MATRIZE
$> cd ..\pi5.exe"
numeros[0][0] vale 1
numeros[0][1] vale 2
numeros[1][0] vale 3
numeros[1][1] vale 4
valores[0][0] vale 0.50
valores[0][1] vale 1.00
valores[0][2] vale 1.50
valores[0][3] vale 2.00
valores[0][4] vale 2.50
valores[1][0] vale 3.00
valores[1][1] vale 3.50
valores[1][2] vale 4.00
valores[1][3] vale 4.50
valores[1][4] vale 5.00
valores[2][0] vale 5.50
valores[2][1] vale 6.00
valores[2][2] vale 6.50
valores[2][3] vale 7.00
valores[2][4] vale 7.50
valores[3][0] vale 8.00
valores[3][1] vale 8.50
valores[3][2] vale 9.00
valores[3][3] vale 9.50
valores[3][4] vale 10.00
valores[4][0] vale 10.50
valores[4][1] vale 11.00
valores[4][2] vale 11.50
valores[4][3] vale 12.00
valores[4][4] vale 12.50
```

```
42
43
44
45 //vetores de reais
46 float valores[5][5];
47 int somador = 1; You, a few seconds ago * Uncommitted changes
48
49 for(int i = 0; i < 5; i++){
50     for(int j = 0 ; j < 5; j++){
51         valores[i][j] = (float)somador / 2;
52         somador++;
53     }
54 }
55 //impressão
56 for(int i = 0; i < 5; i++){
57     for(int j = 0; j < 5; j++){
58         printf("valores[%d][%d] vale %.2f\n",i,j,valores[i][j]);
59     }
60 }
61
62
63 //entrada_dados
64 //processamento_dados
65 //saída_dados
```

LINGUAGEM

DE MAQUINA

LINGUAGEM DE MAQUINA

- Os computadores tem sua propria linguagem, chamada de LINGUAGEM BINÁRIA.
- Binario (0 & 1)
- As maquinas entendem 0's e 1's, para elas, o 1 significa que esta passando energia, e o 0, sem energia.
- Ela vai jogando esse chaveamento em com e sem energia, e vai transformando o analogico em digital.
- Os humanos possuem 10 numeros decimais (0...9) , todos os numeros apos esses são COMBINAÇÕES dos anteriores.

Decimais
0
1
2
3
4

LINGUAGEM DE MAQUINA

- Os Computadores utilizam BINARIOS, os chamados de bits.

Decimais	Binários
0	0
1	1
2	10
3	11
4	100

- A representação do numero 0_(decimal) em binario é 0.

- A representação do numero 1_(decimal) em binario é 1.

- A representação do numero 2_(decimal) em binario é 10.

- Combinação(bit_1 , bit_0).

- A representação do numero 3_(decimal) em binario é 11.

- Combinação(bit_1 , bit_1).

- A representação do numero 4_(decimal) em binario é 100.

- Combinação(bit_1 , bit_0,bit_0).

Binário	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

LINGUAGEM DE MAQUINA

- Os Computadores utilizam BINARIOS, os chamados de bits.

Decimais	Binários
0	0
1	1
2	10
3	11
4	100

- A representação do numero 0_(decimal) em binario é 0.

- A representação do numero 1_(decimal) em binario é 1.

- A representação do numero 2_(decimal) em binario é 10.

- Combinação(bit_1 , bit_0).

- A representação do numero 3_(decimal) em binario é 11.

- Combinação(bit_1 , bit_1).

- A representação do numero 4_(decimal) em binario é 100.

- Combinação(bit_1 , bit_0,bit_0).

Binário	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

ARQUITETURA DE COMPUTADORES

- Quando falamos que um computador possui uma arquitetura de 8 bits, significa que ele tem a capacidade de processar até 8 bits por vez no seu ciclo.
 - 16 bits, o dobro do anterior...etc.



- PADRÃO

8 bits = 1 byte

ARQUITETURA DE COMPUTADORES

- NUMERO MAXIMO

- 8 bits

11111111

} 1 byte → número máximo 256 (2^8)

$$8_bits = 1_byte = 256(2^8) \text{ numeros}$$

- Na linguagem C, um dado do tipo INT guar até 4 bytes = 32 bits.

- NUMERO MAXIMO = 4294967295 (256^4)

USANDO
NUMEROS
BINARIOS

- As vezes precisamos trabalhar a mais baixo nível.
- Imagine uma variável contendo o valor decimal 2, conforme:
int numero = 2;
- A representação binaria do numero 2 é : 0000 0010
- A linguagem C permite que façamos operações em "baixo nível" com variaveis do tipo char, int e long int.

Operador	Ação
<code>~</code>	NOT
<code>>></code>	Deslocamento de bits à direita
<code><<</code>	Deslocamento de bits à esquerda

```

//declarando_variaveis
int valor = 2; 0010
printf("Valor vale %d\n",valor);

//entrada_dados

//processamento_dados
//deslocamento de bit para esquerda
valor = valor << 2; 0010<<2=1000(8)
printf("Valor vale %d\n",valor);

valor = 2;//reset

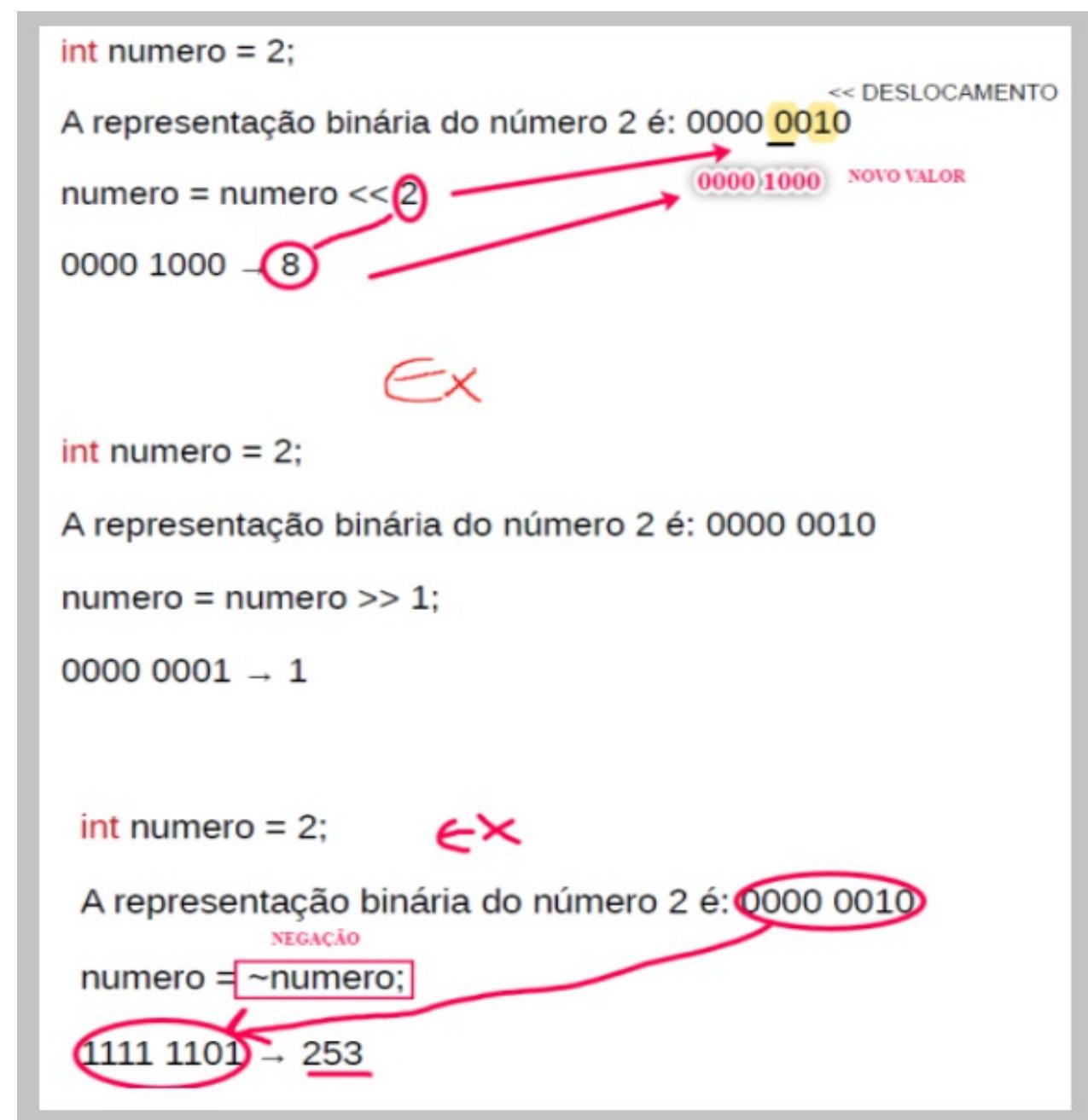
//deslocamento de bit para esquerda
valor = valor >> 1; 0010>>1=0001(1)
printf("Valor vale %d\n",valor);

valor = 2;//reset

//Negação
valor = ~valor; ^0010=1100(-3)???????
printf("Valor vale %d\n",valor);

//saída_dados

```



SOBRE HEXADECIMAIS

- Na base hexadecimal temos 16 algarismos para representar o que precisamos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Lembrando que 1 byte são 8 bits...
256 → 1111 1111 → FF
- Em hexadecimal fica FF (1 byte)
- EX onde são utilizados.

```
.mfp-arrow-left:before,  
.mfp-arrow-left .mfp-b {  
  margin-left: 25px;  
  border-right: 27px solid #3f3f3f;  
}  
  
RGB
```

A diagram illustrating the conversion of the RGB color code to its hex value. It shows the text "RGB" in red, green, and blue letters, with an upward arrow pointing to the hex code "#3f3f3f" in the CSS code above it. The entire diagram is enclosed in a light gray box.

FUNÇÕES

ESCREVENDO FUNÇÕES

main() -> função principal em C

- Dentro da função principal, executamos todas as outras funções que o programa venha a ter.
- printf() é uma função da biblioteca . Uma função que dada a string passada como parâmetro ele imprimirar a string na saída padrão.
- Nos podemos criar nossas próprias funções.

ESTRUTURAS DAS FUNÇÕES

- Tipo de retorno : Tipo de dado que a função ao ser executada irá retornar.
- Nome :
- Parâmetros de entrada (opcional)
- Implementação
- Retorno(opcional)

ESCREVENDO FUNÇÕES

ESTRUTURAS DAS FUNÇÕES

- EXEMPLO

```
void mensagem(){  
    printf("Bem-vindo!");  
}
```

- void = tipo de retorno(vazio), a função simplesmente executa alguma coisa. No caso uma outra função (printf()).
- mensagem = nome
- () = parametro de entrada(opcional)
- { } = implementação
- VOID/INT = retorno, o void não possui. No int = return 0;

```
int soma(int num1, int num2){  
    return num1+num2;  
}
```

- Tipo de retorno : int
- Nome : soma
- Parametros de entrada (opcional): num1 e num 2
- Implementação : {soma}
- Retorno(opcional): soma

```
int soma(int num1, int num2){  
    int res = num1 + num2;  
    return res;  
}
```

ESCREVENDO FUNÇÕES

ESTRUTURAS DAS FUNÇÕES

- EXEMPLO

```
void proximo_char(char caractere){  
    printf("%c", caractere + 1);  
}
```

- Tipo de retorno : void
- Nome : proximo_char
- Parametros de entrada (opcional): char caractere
- Implementação : caractere +1
- Retorno(opcional): imprime caractere.

```
✓ #include <stdio.h>  
#include <stdio.h>  
  
✓ void mensagem(){  
    printf("Bem-vindo!");  
}  
  
✓ int soma(int num1, int num2){  
    return num1+num2;  
}  
  
✓ void proximo_char(char caractere){  
    printf("%c", caractere + 1);  
}
```

USANDO FUNÇÕES

```
void mensagem(){
    printf("Bem-vindo!\n");
}

int soma(int num1, int num2){
    int res = num1 + num2;
    return res; // isso aqui só significa que ele está colocando um valor, e não a imprimindo.
}

void proximo_char(char caractere){ //recebe um caractere e vai imprimir o caractere+1 pela tabela ASCII
    printf("%c", caractere + 1);
}

You, 2 months ago • FINALIZAÇÃO USANDO FUNÇÕES
int main(){
    printf("\n");

    //chamando as funções
    printf("Ola...\n");

    mensagem();

    int retorno = soma(4,6);
    //printf("Retorno = %d\n",retorno);

    printf("Retorno = %d\n", soma(4,6));

    char cara = 'a'; //char cara = 97
    proximo_char(cara);
```

PROTOTIPO DE FUNÇÕES

- Servem para indicar para o main() quais as funções que iremos utilizar dentro dela, que estão implementados(escritos) apos ela.

PROTOTIPO é composto pelas ASSINATURA DAS FUNÇÕES:

- Tipo de retorno
- Nome
- Parametros de entrada

```
int soma(int num1, int num2);
```

- tipo de retorno : int
- Nome : soma
- Parametro de entrada (opcionais): int(num1,num2)

```
//////////////////PROTOTIPOS DE FUNÇÃO/////////  
  
int soma(int num1, int num2);  
  
void mensagem();  
  
int main(){  
    printf("\n");  
  
    //declarando_variaveis  
    int n1, n2, ret;  
  
    //entrada_dados  
    printf("Informe o primeiro numero:\n");  
    scanf("%d",&n1);  
    printf("Informe o segundo numero:\n");  
    scanf("%d",&n2);
```

PROTOTIPO DE FUNÇÕES

```
//processamento_dados
ret = soma(n1,n2);

//saída_dados
printf("A soma de %d com %d eh %d\n", n1, n2, ret);

mensagem();

printf("\n");
return 0;
}

///////////////////////////////FUNÇÕES////////////////

//soma
int soma(int num1, int num2){
    return num1 + num2;
}

//saudação
void mensagem(){
    printf("Bem-vindo!\n");
}
```

ARQUIVOS DE CABEÇALHO

```
/*
 Vamos criar um programa que receba 2 numeros, execute a função de soma e multiplicação e passe esses valores para as variaveis ret_s e ret_m.
*/
You, 2 months ago • ADD e ALT
#include <stdio.h>
#include <stdio.h>
#include "ajuda.h"

int main(){
    printf("\n");

    mensagem();

    //d_v//
    int n1, n2, ret_s, ret_m;

    //e_d//
    printf("Informe o primeiro numero:\n");
    scanf("%d",&n1);
    printf("Informe o segundo numero:\n");
    scanf("%d",&n2);

    //p_d//
    ret_s = soma(n1,n2);
    printf("A soma de %d com %d eh: %d\n", n1, n2, ret_s);
    ret_m = mult(n1,n2);
    printf("A multiplicacao de %d com %d eh: %d", n1, n2, ret_m);
    //s_d//
    printf("\n");
    return 0;
}
```

TP20.C

ARQUIVOS DE CABEÇALHO

```
C ajuda.c  X
S9(funcoes_c) > 4-ARQUIVOS_CABEÇALHOS > C ajuda.c > ...
You, 2 months ago | 1 author (You)
1 // Código de implementação de funções
2 #include <stdio.h>
3 #include <stdlib.h>
4 /////////////////mensagem/////////////
5
6 void mensagem(){
7     printf("Bem vindo... \n");
8 }
9 ////////////////OPERAÇÕES////////////
10 //soma//
11 int soma(int num1, int num2){
12     return num1 + num2;
13 }
14 //multiplicação//
15 int mult(int num1, int num2){
16     return num1 * num2;
17 }
```

The image shows a terminal window displaying a GitHub commit history for a file named 'ajuda.c'. The commit message is 'Código de implementação de funções' (Implementation of functions). The code itself contains two main functions: 'mensagem()' and 'soma()'. A red circle highlights the file name 'ajuda.c' in the commit message. Two red arrows point from the word 'mensagem' in the documentation comment to the start of the 'mensagem()' function definition. Another red arrow points from the word 'mult' in the documentation comment to the start of the 'mult()' function definition.

ARQUIVOS DE CABEÇALHO

H ajuda.h X

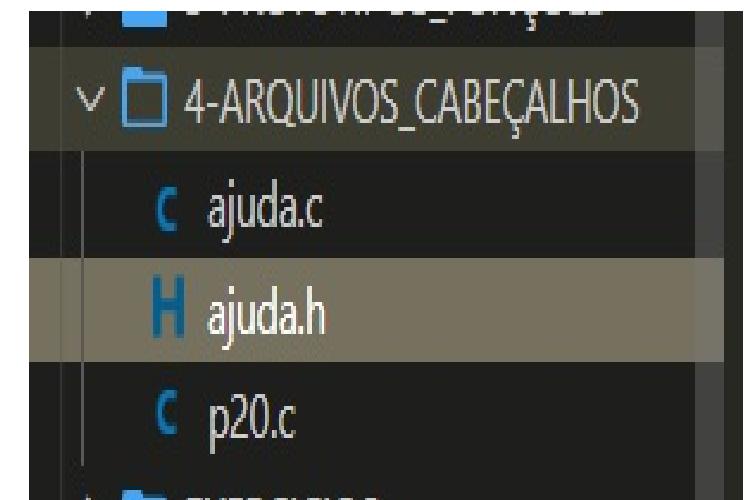
S9(funcões_c) > 4-ARQUIVOS_CABEÇALHOS > H ajuda.h > ...

You, 2 months ago | 1 author (You)

```
1 // Código de implementação de funções You, 2 mo
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ajuda.c"
5
6 void mensagem();
7
8 int soma(int num1, int num2);
9
10 int mult(int num1, int num2);
11
```

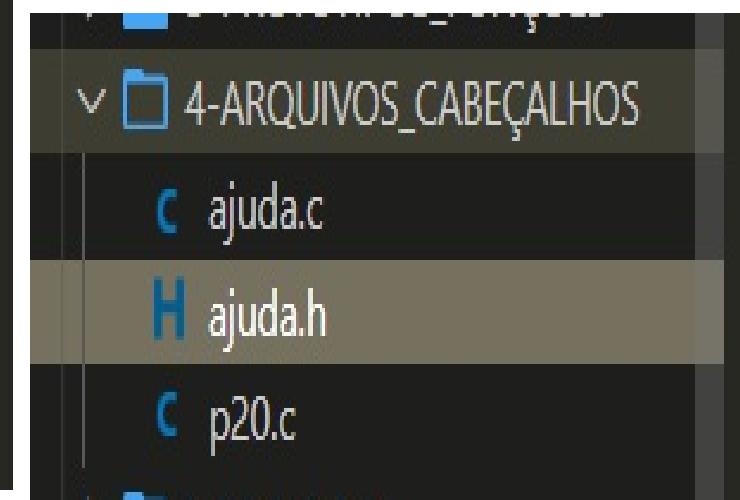
Handwritten annotations in red:

- A red oval highlights the file name "ajuda.h" in the file list.
- A red circle highlights the "#include "ajuda.c"" line in the code editor.
- A large, stylized red arrow points from the highlighted "#include" line towards the file list on the right.
- A red checkmark is placed next to the "ajuda.h" entry in the file list.



ARQUIVOS DE CABEÇALHO

```
H ajuda.h X
S9(funcoes_c) > 4-ARQUIVOS_CABEÇALHOS > H ajuda.h > ...
You, 2 months ago | 1 author (You)
1 // Código de implementação de funções You, 2 mo
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "ajuda.h"
5
6 void mensagem();
7
8 int soma(int num1, int num2);
9
10 int mult(int num1, int num2);
11
```



PONTEIROS

SOBRE PONTEIROS

```
void incrementa(int *contador); // função que recebe um tipo inteiro

int main(){//inicio_main
    printf("\n");
    int contador = 10;
    printf("Antes de incrementar.\n");
    printf("O contador vale :%d\n", contador);
    printf("O endereço de memoria eh: %d\n", &contador);

    incrementa(&contador); //passa o endereço da variavel contador
    printf("Depois de incrementar.\n", contador);
    printf("O contador vale :%d\n", contador);
    printf("O endereço de memoria eh: %d\n", &contador);

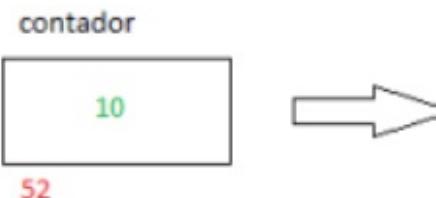
    //declarando_variaveis
    //entrada_dados
    //processamento_dados
    //saida_dados
    printf("\n");
    printf("\n");
    return 0;
}//fin_main

void incrementa(int *contador){// endereço de memoria como parametro de entrada
    printf("O Antes de incrementar.\n");
    printf("O contador vale :%d\n", (*contador));//valor
    printf("O endereço de memoria eh: %d\n", contador);//endereço de memoria

    printf("O Depois de incrementar.\n");
    //valor++;
    printf("O contador vale:%d\n", ++(*contador));
    //printf("O contador vale :%d\n", valor);
    printf("O endereço de memoria eh: %d\n", contador); // não passamos o & pois ja estamos recebendo

}//fim_void
```

SOBRE PONTEIROS



-valor é uma variável criada dentro da função incrementa, logo ela só existe lá dentro. Quando chamamos a função e passamos o contador como parâmetro, estamos dando a valor o valor da variável criada contador, fazendo a PASSAGEM POR COPIA.

- Para que o código funcione na forma que queremos, ou seja, incrementar o contador, temos que passar o endereço da variável contador para a função incrementa. E , na própria função, informar que vamos receber um endereço de memória com um *.

incrementa(&contador)

void incrementa (int *contador){}

& -> fornece o endereço de memória da variável.

```
void incrementa(int *valor){  
    printf("0 Antes de incrementar.\n");  
    printf("0 contador vale :%d\n", valor);  
    printf("0 endereço de memoria eh: %d\n", &valor);  
  
    printf("0 Depois de incrementar.\n");  
    //valor++;  
    printf("0 contador vale:%d\n", ++valor);  
    //printf("0 contador vale :%d\n", valor);  
    printf("0 endereço de memoria eh: %d\n", &valor);  
}
```

```
void incrementa(int *valor); // f  
  
int main(){//inicio_main  
  
    printf("\n");  
  
    int contador = 10;  
  
    printf("Antes de incrementar  
printf("0 contador vale :%d\\n", contador);  
printf("0 endereço de memoria eh: %d\\n", &contador);  
  
    incrementa(&contador); //passagem por copia  
    printf("Depois de incrementar.  
printf("0 contador vale :%d\\n", contador);  
printf("0 endereço de memoria eh: %d\\n", &contador);  
}
```

SOBRE PONTEIROS

- `*valor` = indica que queremos mostrar o valor da variável ponteiro e não seu endereço.- Queremos incrementar tbm esse valor, então na linha de baixo fazemos o mesmo processo com o `*`..

```
//1//  
void incrementa(int *valor){  
    printf("O Antes de incrementar.\n");  
    printf("O contador vale :%d\n", *valor);  
    printf("O endereço de memoria eh: %d\n", &valor);  
  
    printf("O Depois de incrementar.\n");  
    //valor++;  
    printf("O contador vale:%d\n", ++(*valor));  
    //printf("O contador vale :%d\n", valor);  
    printf("O endereço de memoria eh: %d\n", &valor);
```

```
Antes de incrementar.  
O contador vale :10  
O endereço de memoria eh: 6422300  
O Antes de incrementar.  
O contador vale :10  
O endereço de memoria eh: 6422300  
O Depois de incrementar.  
O contador vale:11  
O endereço de memoria eh: 6422300  
Depois de incrementar.  
O contador vale :11  
O endereço de memoria eh: 6422300
```

- Vemos que os endereços das variáveis ainda são diferentes, o que fizemos foi uma PASSAGEM DE VALOR POR REFERENCIA... Onde damos a função o endereço de memoria da variável, e assim a função em vez de criar outra variável, acessa o endereço e altera a variável original.

SOBRE PONTEIROS

```
int main()//Início_main  
printf("\n");
```

&: endereçamento de memória
*: operador de indireção

```
int i, j, *pi, *pj, **ppj;
```

i = 7; valor do dado
j = 10;

```
printf("Os valores de i e j sao: %d, %d\n",i,j);  
printf("Os end. de memoria de i e j sao: %d, %d\n",&i,&j);
```

pi = &i; recebendo como valor o endereço da variável
pj = &j;

```
printf("Os valores de pi e pj sao: %x, %x\n",pi,pj); 315 e 72  
printf("Os end. de memoria de pi e pj sao: %d, %d\n",&pi,&pj); 100 120  
printf("%d %d",*pi,*pj); 7 10  
*mostra o valor da variável ao qual o ponteiro aponta
```

*pi = 20; altera o valor das variáveis que o ponteiro aponta
*pj = 15;

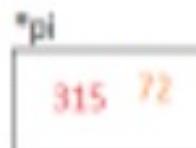
```
printf("Os valores de i e j sao: %d, %d\n",i,j); 20 15  
copia o valor da variável que pi aponta e coloca na variável que pj
```

*pj = *pi; aponta
printf("Os valores de i e j sao: %d, %d\n",i,j); 20 20

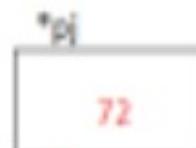
pi = pj; o conteúdo da variável pi é igual ao pj
printf("Os valores de pi e pj sao: %d, %d\n",pi,pj); 72 72

ppj = &pj; ppj vai receber como conteúdo o endereço de pj
**ppj = 70; ** altera o conteúdo de j, pois é um ponteiro de ponteiro

MEMORIA DO COMPUTADOR



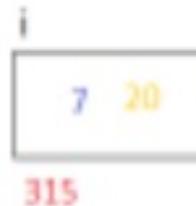
100



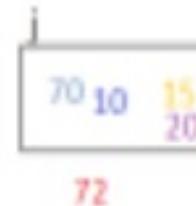
120



250



315



72



USANDO PONTEIROS

- A linguagem C é totalmente dependente de ponteiros.
- Quando falamos de ponteiros, estamos falando em fazer manipulação la no endereço de memoria para onde o valor da variavel foi alocado.
- Trabalhando com endereço de memoria, estamos trabalhando a baixo nível na linguagem. Estamos fazendo acesso ao hardware logico da memoria.
- Vamos criar um programa

```
int main(){//inicio_main
    printf("\n");

    int n; //declarando_variavel - variavel que guarda seu valor na memoria

    int *p; //declarando_variavel(ponteiro) - variavel que guarda um endereço de memoria.

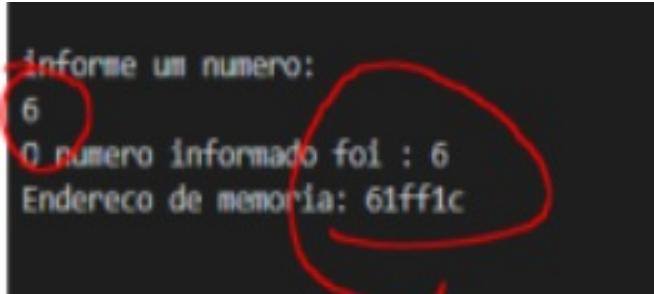
    printf("informe um numero:\n");
    scanf("%d",&n);

    //inicializando_ponteiro
    p = &n; //ponteiro recebe valor informado pelo usuario

    printf("O numero informado foi : %d\n",n);

    printf("Endereco de memoria: %d\n",&n);

    printf("Endereco do ponteiro: %d\n",p); //p hexadecimal
```



```
informe um numero:
6
O numero informado foi : 6
Endereco de memoria: 61ff1c

PS C:\Users\Gabi\Documents\TEORIA_INDIV> cmd /c .\"p22.exe"

informe um numero:
3
O numero informado foi : 3
Endereco de memoria: 61ff1c

PS C:\Users\Gabi\Documents\TEORIA_INDIV>
```

USANDO PONTEIROS

- Alguns sistemas como o windows usando o mesmo espaço alocado, outros reservam um a cada execuçāodo progama.
- Inicializando um ponteiro

```
int n; //declarando_variavel - variavel que guarda seu valor na memoria  
  
int *p; //declarando_variavel(ponteiro) - variavel que guarda um endereco de memoria.
```

- O ponteiro recebe um valor de uma variavel quando passamos o endereco dessa variavel para ele

```
//inicializando_ponteiro  
p = &n; //ponteiro recebe valor informado pelo usuario
```

- O local que o SO alocar para a variavel n deve ser informado para a variavel ponteiro.

```
printf("Endereco do ponteiro: %p\n",p);
```

USANDO PONTEIROS

```
informe um numero:  
6  
O numero informado foi : 6  
Endereco de memoria: 6422296  
Endereco do ponteiro: 0061FF18
```

- Primeiro valor é a forma numerica do endereço de memoria(%d) .
- No segundo estamos imprimindo em hexadecimal (%p).

```
PS C:\Users\Gabriel\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMYPROG_C\SIH(ponteiros)\2-USANDO_PONTEIROS  
> cd /c .\p22.exe  
  
Informe um numero:  
6  
O numero informado foi : 6  
Endereco de memoria: 6422296  
Endereco do ponteiro: 6422296  
  
20  
21  
22 printf("O numero informado foi : %d\n",n);  
23  
24 printf("Endereco de memoria: %d\n",&n);  
25  
26 printf("Endereco do ponteiro: %d\n",p);  
27
```

Os dois são iguais

USANDO PONTEIROS

```
informe um numero:  
6  
O numero informado foi : 6  
Endereco de memoria: 6422296  
Endereco do ponteiro: 0061FF18
```

- Primeiro valor é a forma numerica do endereço de memoria(%d) .
- No segundo estamos imprimindo em hexadecimal (%p).

```
PS C:\Users\Gabriel\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMYPROG_C\SIH(ponteiros)\2-USANDO_PONTEIROS  
> cd /c .\p22.exe  
  
Informe um numero:  
6  
O numero informado foi : 6  
Endereco de memoria: 6422296  
Endereco do ponteiro: 6422296  
  
20  
21  
22 printf("O numero informado foi : %d\n",n);  
23  
24 printf("Endereco de memoria: %d\n",&n);  
25  
26 printf("Endereco do ponteiro: %d\n",p);  
27
```

Os dois são iguais

ARRAYS E PONTEIROS

- Vimos 2 tipos de arrays
 - Vetores (uni-dimensionais)
 - Matrizes (multi-dimensionais)
- Vamos fazer um progama

```
int main(){//inicio_main
    printf("\n");

    int valores[5];

    //recebendo resultado
    for (int i = 0; i < 5;i++){//inicio_for1
        printf("Informe o valor %d/5 valor: ",(i+1));
        scanf("%d",&valores[i]);
    }//fim_for1

    printf("Os valores informados foram: \n");

    //imprimindo resultados
    for (int i = 0; i < 5;i++){//inicio_for2
        printf("%d\n", valores[i]);
    }//fim_for2

    printf("\n");
    printf("\n");
    return 0;
}//fim_main
```

```
Informe o valor 1/5 valor: 2
Informe o valor 2/5 valor: 6
Informe o valor 3/5 valor: 8
Informe o valor 4/5 valor: 1
Informe o valor 5/5 valor: 9
Os valores informados foram:
2
6
8
1
9
```

- O que tem haver ponteiros e arrays ...
- Quando estamos declarando um vetor (array) do tipo inteiro, a linguagem C cria um ponteiro(invisivel) que aponta para o primeiro endereço de memoria do array.
- Quando fazemos a declaração de um vetor, o compilador pega o primeiro endereço de memoria do vetor e coloca um ponteiro apontando para ele. Assim que ele tem o controle de quais são os elementos do array.

ARRAYS E PONTEIROS

```
printf("Endereco de memoria(0): %p\n valor(0) : %d\n", &valores[0], valores[0]);  
printf("Endereco de memoria: %p\n valor(0) : %d", valores, valores[0]);
```

```
printf("Endereco de memoria(1): %p\n valor(0) : %d\n", &valores[1], valores[1]);  
printf("Endereco de memoria: %p\n valor(0) : %d", valores, valores[1]);
```

Endereco de memoria(0): 0061FF04
valor(0) : 1
Endereco de memoria(vetor): 0061FF04
valor(0) : 1

Endereco de memoria(1): 0061FF08
valor(1) : 2
Endereco de memoria(vetor): 0061FF04
valor(1) : 2

```
printf("Endereco de memoria(0): %p\n valor(0) : %d\n", &valores[0], valores[0]);  
printf("Endereco de memoria(vetor): %p\n valor(0) : %d\n", valores, valores[0]);
```

```
printf("Endereco de memoria(1): %p\n valor(1) : %d\n", &valores[1], valores[1]);  
printf("Endereco de memoria(vetor): %p\n valor(1) : %d", valores, valores[1]);
```

Endereco de memoria(0): 0061FF00
valor(0) : 1
Endereco de memoria(vetor): 0061FF00
valor(0) : 1

Endereco de memoria(1): 0061FF04
valor(1) : 2
Endereco de memoria(vetor): 0061FF00
valor(1) : 2

Endereco de memoria(2): 0061FF08
valor(2) : 3
Endereco de memoria(vetor): 0061FF00
valor(2) : 3

Endereco de memoria(3): 0061FF0C
valor(3) : 4
Endereco de memoria(vetor): 0061FF00
valor(3) : 4

Endereco de memoria(4): 0061FF10
valor(4) : 5
Endereco de memoria(vetor): 0061FF00
valor(4) : 5

```
//fim_for2
```

```
for(int i = 0; i < 5; i++){  
    printf("Endereco de memoria(%d): %p\n valor(%d) : %d\n", i, &valores[i], i, valores[i]);  
    printf("Endereco de memoria(vetor): %p\n valor(%d) : %d\n", valores, i, valores[i]);  
}
```

PONTEIROS EM OUTROS LUGARES

- O que ainda precisamos entender para finalizar ponteiros.
- Vamos criar um progama para ver a quantidade de bytes que uma variavel tem.

```
int main(){//inicio_main
    printf("\n");

    int valores[5] = {1, 2, 3, 4, 5}; //declaração_inicialização de um vetor

    for(int i = 0; i < 5; i++){
        printf("O valor %d tem %ld bytes!\n", valores[i], sizeof(valores[i]));
    }
    printf("O valores possui %ld bytes\n", sizeof(valores));

    //utilização do ponteiro do vetor[0]
    printf("valores[0] vale %d e endereço de memoria eh %p\n",valores[0], valores[0]);
    printf("*valores vale %d e endereço de memoria eh %p\n",*valores, *valores);

    // avançando as posições
    printf("*valores + 1) vale %d e endereço de memoria eh %p\n",*(valores+1), *(valores+1));
    printf("*valores + 2) vale %d e endereço de memoria eh %p\n",*(valores+2), *(valores+2));
    printf("*valores + 3) vale %d e endereço de memoria eh %p\n",*(valores+3), *(valores+3));
    printf("*valores + 4) vale %d e endereço de memoria eh %p\n",*(valores+4), *(valores+4));

    printf("\n");
    printf("\n");
    return 0;
}//fim_main
```

PONTEIROS EM OUTROS LUGARES

- O que ainda precisamos entender para finalizar ponteiros.
- Vamos criar um progama para ver a quantidade de bytes que uma variavel tem.

```
int main(){//inicio_main
    printf("\n");

    int valores[5] = {1, 2, 3, 4, 5}; //declaração_inicialização de um vetor

    for(int i = 0; i < 5; i++){
        printf("O valor %d tem %ld bytes!\n", valores[i], sizeof(valores[i]));
    }
    printf("O valores possui %ld bytes\n", sizeof(valores));

    //utilização do ponteiro do vetor[0]
    printf("valores[0] vale %d e endereço de memoria eh %p\n",valores[0], valores[0]);
    printf("*valores vale %d e endereço de memoria eh %p\n",*valores, *valores);

    // avançando as posições
    printf("*valores + 1) vale %d e endereço de memoria eh %p\n",*(valores+1), *(valores+1));
    printf("*valores + 2) vale %d e endereço de memoria eh %p\n",*(valores+2), *(valores+2));
    printf("*valores + 3) vale %d e endereço de memoria eh %p\n",*(valores+3), *(valores+3));
    printf("*valores + 4) vale %d e endereço de memoria eh %p\n",*(valores+4), *(valores+4));

    printf("\n");
    printf("\n");
    return 0;
}//fim_main
```

PONTEIROS EM OUTROS LUGARES

- Quando estudamos tipo de dados, vimos que cada tipo de dado ocupa um espaço da memoria.

valores[5]

int = 4 bytes

1
0

2
1

3
2

4
3

5
4

5

6

0
56

TIPO INTEIRO = 4 BYTES

- BIT 0/1;

- BYTE: 0000 0001

- 4 BYTES :0000 0000 0000 0000 0000 0000 0001 = REP.NUM1

- 4 BYTES :0000 0000 0000 0000 0000 0000 0001 = REP.NUM2

TAMANHO DAS VARIAVEIS

1 - INT VALORES [5] = 4 BYTES * 5 = 20 BYTES

PONTEIROS EM OUTROS LUGARES

- Vamos percorrer o vetor que ja foi inicializado e imprimir o valor da variavel e seu tamanho em `bytes.sizeof(valores[i]) --- %ld`

- Essa função mostra o tamanho da variavel em bytes

- Vemos que todas possuem 4 bytes pois são do tipo inteiro(figura a cima).

```
0 valor 1 tem 4 bytes!
0 valor 2 tem 4 bytes!
0 valor 3 tem 4 bytes!
0 valor 4 tem 4 bytes!
0 valor 5 tem 4 bytes!
0 valores possui 20 bytes
```

- Ao criar um vetor, a linguagem cria um ponteiro com o nome desse vetor e seu endereço. Logo podemos acessar esse vetor pelo ponteiro..

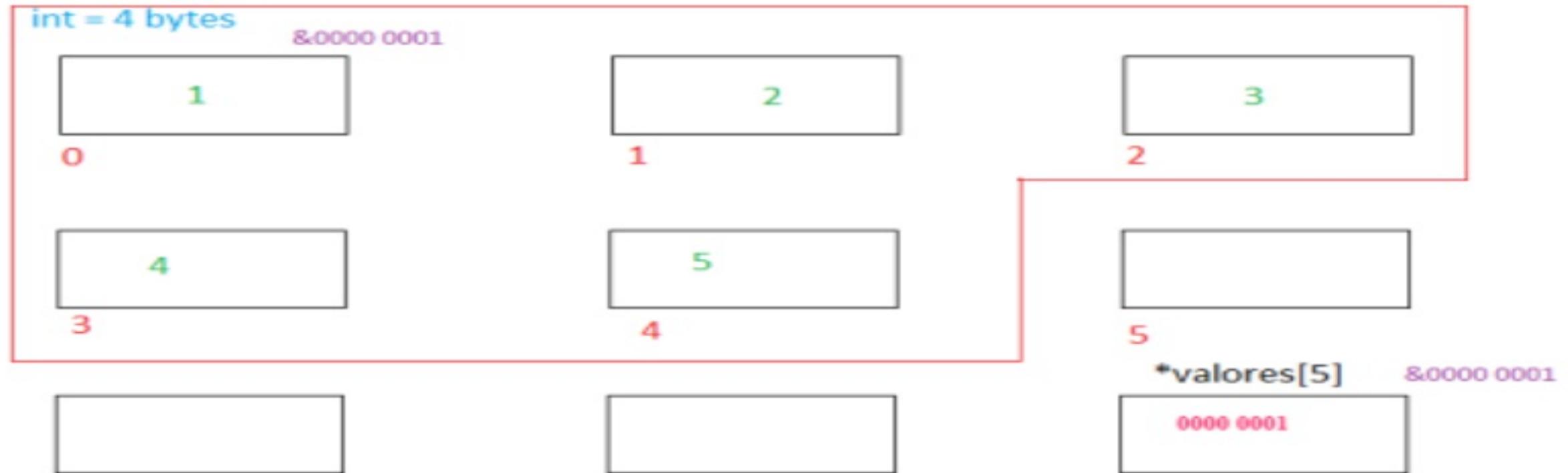
```
//utilização do ponteiro do vetor[5]
printf("valores[0] vale %d e endereço de memoria eh %p\n",valores[0], valores[0]);
printf("*valores vale %d e endereço de memoria eh %p\n",*valores, *valores);
```

```
0 valor 1 tem 4 bytes!
0 valor 2 tem 4 bytes!
0 valor 3 tem 4 bytes!
0 valor 4 tem 4 bytes!
0 valor 5 tem 4 bytes!
0 valores possui 20 bytes
valores[0] vale 1 e endereço de memoria eh 00000001
*valores vale 1 e endereço de memoria eh 00000001
```

PONTEIROS EM OUTROS LUGARES

MEMORIA DO COMPUTADOR

VALORES[5]



- Para acessar as outras posições

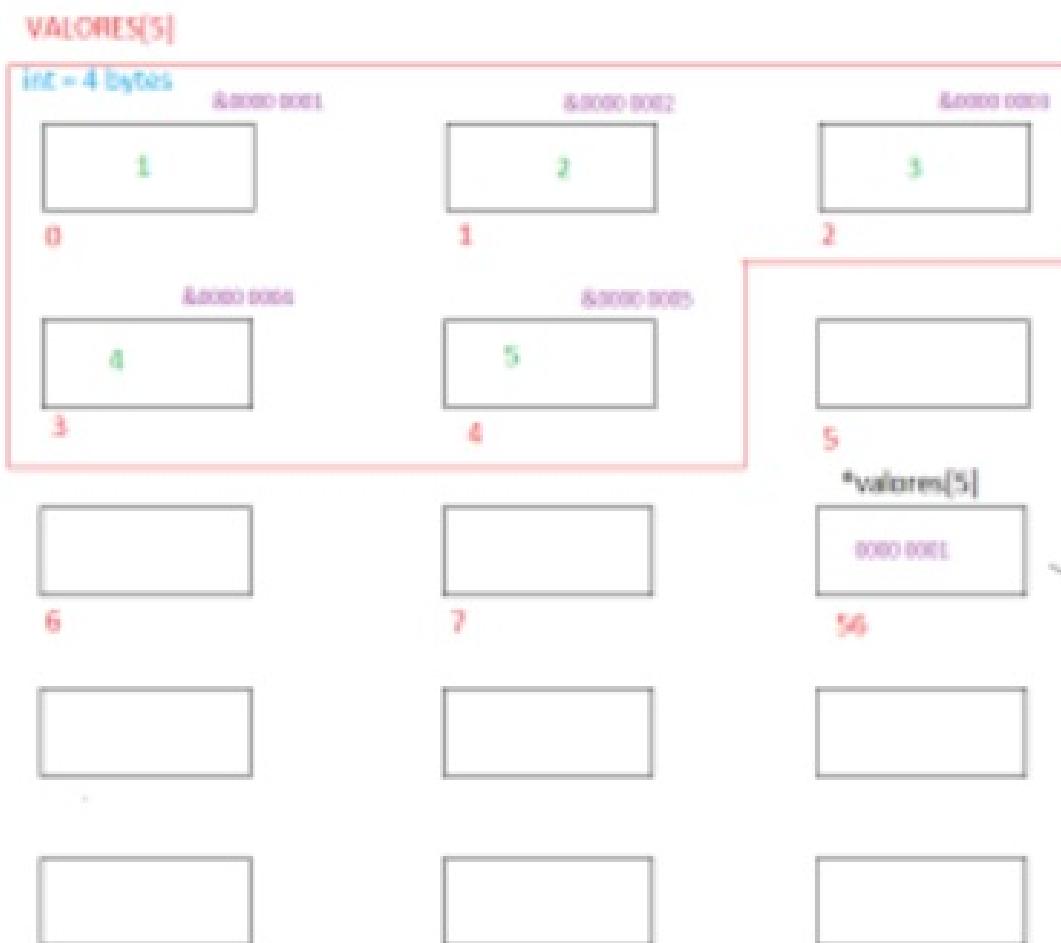
```
// avançando as posições
printf("(valores + 1) vale %d e endereco de memoria eh %p\n", *(valores+1), *(valores+1));
printf("(valores + 2) vale %d e endereco de memoria eh %p\n", *(valores+2), *(valores+2));
printf("(valores + 3) vale %d e endereco de memoria eh %p\n", *(valores+3), *(valores+3));
printf("(valores + 4) vale %d e endereco de memoria eh %p\n", *(valores+4), *(valores+4));
```

```
*valores vale 1 e endereco de memoria eh 00000001
*(valores + 1) vale 2 e endereco de memoria eh 00000002
*(valores + 2) vale 3 e endereco de memoria eh 00000003
*(valores + 3) vale 4 e endereco de memoria eh 00000004
*(valores + 4) vale 5 e endereco de memoria eh 00000005
```

PONTEIROS EM OUTROS LUGARES

- Estamos somando ao conteudo do ponteiro(`end_vetor`) a 1, ou seja, o proximo endereço, percorrendo assim o vetor utilizando o ponteiro.
- Quando fazemos a inicialização de um vetor, o computador aloca um espaço de memoria contínuo(um lado do outro)

MEMORIA DO COMPUTADOR



TIPO INTEIRO = 4 BYTES

-bit 0 / 1;
-byte 0000 0001
-4 bytes 0000 0000 0000 0000 0001 = Rep. num 1

-4 bytes 0000 0000 0000 0000 0000 0000 0000 0000 0010 = Rep. num 2

TAMANHO DAS VARIÁVEIS

$1 - \text{int valores[5]} = 4\text{bytes} * 5 = 20\text{ bytes}$

*valores[5] 0000 0001 + 1 → *valores[5] 0000 0002 56

PONTEIROS EM OUTROS LUGARES

- Estamos somando ao conteudo do ponteiro(`end_vetor`) a 1, ou seja, o proximo endereço, percorrendo assim o vetor utilizando o ponteiro.
- Quando fazemos a inicialização de um vetor, o computador aloca um espaço de memoria continuo(um lado do outro)

MEMORIA DO COMPUTADOR



TIPO INTEIRO = 4 BYTES

-bit 0 / 1;
-byte 0000 0001
-4 bytes 0000 0000 0000 0000 0001 = Rep. num 1

-4 bytes 0000 0000 0000 0000 0000 0000 0000 0000 0010 = Rep. num 2

TAMANHO DAS VARIAVEIS

$1 - \text{int valores[5]} = 4\text{bytes} * 5 = 20\text{ bytes}$

*valores[5] 0000 0001 + 1 → 0000 0002

ENTRADA E

SAÍDA

MANIPULANDO ENTRADA E SAIDA

- Para trabalharmos criando arquivos, não precisamos de nenhuma biblioteca especial. A propria linguagemC possui todos os recursos necessarios.
- Para criarmos ou ler arquivos, precisamos criar uma variavel de tipo especifico FILE.

```
int main(){  
    FILE *arq;
```

- Essa variavel, é um ponteiro que aponta para um arquivo(FILE).
- Para trabalharmos tanto com leitura quanto com escrita de arquivos, ou seja, colocarmos informações nesse arquivo, precisamos ter a declaração dessa variavel.

```
//fopen(nome-do-arquivo, forma-de-abertura);  
//w - abrir o arquivo para escrita  
//r - abrir o arquivo para leitura (não podemos editar)  
//wa - abrir o arquivo para adição de conteúdo (existir - conteúdo add nas linhas abaixo.)
```

- Para abrir um arquivo para trabalho usamos uma função que recebe 2 parametros de entrada(nome_arquivo|forma_como_quer_abrir);
- Se o arquivo não existir, será criado um.
- Caso ja exista,ele será sobreescrito com um novo zerado.- Sempre que finalizarmos a manipulação de um arquivo, devemos fecha-lo.

MANIPULANDO ENTRADA E SAIDA

- Para trabalharmos criando arquivos, não precisamos de nenhuma biblioteca especial. A propria linguagemC possui todos os recursos necessarios.
- Para criarmos ou ler arquivos, precisamos criar uma variavel de tipo especifico FILE.

```
int main(){
```

```
FILE *arq;
```

- Essa variavel, é um ponteiro que aponta para um arquivo(FILE).
- Para trabalharmos tanto com leitura quanto com escrita de arquivos, ou seja, colocarmos informações nesse arquivo, precisamos ter a declaração dessa variavel.

```
//fopen(nome-do-arquivo, forma-de-abertura);
//w - abrir o arquivo para escrita
//r - abrir o arquivo para leitura (não podemos editar)
//wa - abrir o arquivo para adição de conteúdo (existir - conteúdo add nas linhas abaixo.)
```

- Para abrir um arquivo para trabalho usamos uma função que recebe 2 parametros de entrada(nome_arquivo|forma_como_quer_abrir);
- Se o arquivo não existir, será criado um.

MANIPULANDO ENTRADA E SAIDA

- Caso ja exista,ele será sobreescrito com um novo zerado.
- Sempre que finalizarmos a manipulação de um arquivo, devemos fecha-lo.

```
#include <stdio.h>
//FILE == ARQUIVO

int main(){
    FILE *arq;

    //fopen(nome-do-arquivo, forma-de-abertura);
    //w - abrir o arquivo para escrita
    //r - abrir o arquivo para leitura (não podemos editar)
    //wa - abrir o arquivo para adição de conteúdo (existir - conteúdo add nas linhas abaixo.)
    arq = fopen("arquivo.txt","w");

    //Sempre que finalizarmos a manipulação de um arquivo, devemos fecha-lo.
    //fclose(nome-da-variavel)
    fclose(arq);

    return 0;
}
```

You, 2 months ago • add alt

LEITURA DE ARQUIVOS

- Vamos aprender a ler arquivos.

```
#include <stdio.h>

int main(){
    FILE *arq;
    arq = fopen("arquivo.txt", "r")

    return 0;
}
```

- Sempre que utilizamos o fopen, independente do modo de abertura (arq) pode vir nulo ou não.
 - Por isso temos que fazer uma checagem desse arquivo, pois só podemos continuar o programa ou fazer algo com esse arquivo se ele não for nulo.
 - A variável poderia ser nula, caso por exemplo ele não exista.

```
if(arq){//sempre importante fazer essa verificação.
    printf("Achei o arquivo!");
}else{
    printf("Não achei o arquivo!")
}
```

- Queremos fazer a leitura de um arquivo e imprimir o que está escrito.

```
if(arq){//sempre importante fazer essa verificação.
    while((c = getc(arq)) != EOF){
        printf("%c", c)
    }
}else{
    printf("Não achei o arquivo!")
}
```

- WHILE = Estamos utilizando a função getc() = get caractere arquivo .
 - Vai pegar cada um dos caracteres do arquivo, 1 por 1 sucessivamente, até o EOF = end of file e imprimir

LEITURA DE ARQUIVOS

The screenshot shows a code editor with two tabs open. The left tab, titled 'arquivo.txt', contains the following text:

```
c: > Users > Gabi > Documents > TEORIA_INDIVIDUAL > UDEMY  
You, 2 months ago | 1 author (You)  
1 10  
2 20  
3 30  
4 40  
5
```

The right tab, titled 'p26.c', contains the following C code:

```
c: > Users > Gabi > Documents > TEORIA_INDIVIDUAL > UDEMY > REP_UDEMY > PROG  
You, 2 months ago | 1 author (You)  
1 #include <stdio.h>  
2  
3 int main(){  
4     FILE *arq;  
5     char c;  
6  
7     arq = fopen("arquivo.txt","r");  
8  
9     if(arq){//sempre importante fazer essa verificação.  
10        while((c = getc(arq)) != EOF){// EOF = AND OF FILE  
11            printf("%c",c);  
12        }  
13    }else{  
14        printf("Nao achei o arquivo!");  
15    }  
16  
17    fclose(arq);  
18  
19    return 0;  
20}
```

```
PS C:\Users\Gabi\Documents\TEORIA_IND  
IVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(en  
trada_saida_em_C)\2-LEITURA_DE_ARQUIV  
OS> & .\"p26.exe" 10  
20  
30  
40
```

- Repare que pegamos caractere por caractere.

LEITURA DE ARQUIVOS

- Lembre-se que o fopen na opção r, so ira funcionar se o arquivo existir.

The terminal window shows the following session:

```
S> cd "c:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(entrada_saida_em_C)\2-LEITURA_DE_ARQUIVOS"
S C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(entrada_saida_em_C)\2-LEITURA_DE_ARQUIVOS
S> & .\p26.exe
ao achei o arquivo!
S C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(entrada_saida_em_C)\2-LEITURA_DE_ARQUIVOS
S>
```

The code editor shows a C program with line numbers and syntax highlighting. A red circle highlights the `fopen` call in line 8, and another red circle highlights the file name "Xarquivo.txt" in the same line.

```
You, seconds ago | 1 author (You)
1 #include <stdio.h>
2
3 int main(){
4
5     FILE *arq;
6     char c;
7
8     arq = fopen("Xarquivo.txt", "r");
9
10    if(arq){//sempre importante fazer e
11        while((c = getc(arq)) != EOF){/
```

LEITURA DE ARQUIVOS

- Queremos agora pegar linha a linha, em vez de caractere por caractere

```
int main(){  
  
    FILE *arq;  
    char nome[10], *resultado;  
  
    arq = fopen("arquivo.txt","r");  
  
    if(arq){  
        while(!feof(arq)){//feof - file end of file , ira fazer o loop enquanto n chegar ao fim de arquivo  
            resultado = fgets(nome, 10, arq); //fgets - file gets  
            printf("Resultado: %d\n", resultado);  
            if(resultado){  
                printf("%s", nome);  
            }  
        }  
    }else{  
        printf("Nao achei o arquivo.");  
    }  
  
    fclose(arq);  
    return 0;|      You, 2 months ago • 2 - ADD MOD
```

- fgets(nome-da-variavel-que-ira-colocar, numero_de_caracteres_pegos, nome_arquivo);
- Queremos pegar os 10 primeiros caracteres e colocar na variavel que criamos nome[10] e depois vamos imprimir.
- Gerando uma linha a mais que não da para ver.
- Para ter certeza de o que esta sendo impresso(pegó) é o que nos queremos, vamos utilizar a variavel resultado para receber o valor que fgets pega. E só vamos imprimir se o resultado for verdadeiro. Ou seja,vamos fazer uma verificação primeiro desse resultado

LEITURA DE ARQUIVOS

- Queremos agora pegar linha a linha, em vez de caractere por caractere

```
int main(){  
  
    FILE *arq;  
    char nome[10], *resultado;  
  
    arq = fopen("arquivo.txt","r");  
  
    if(arq){  
        while(!feof(arq)){//feof - file end of file , ira fazer o loop enquanto n chegar ao fim de arquivo  
            resultado = fgets(nome, 10, arq); //fgets - file gets  
            printf("Resultado: %d\n", resultado);  
            if(resultado){  
                printf("%s", nome);  
            }  
        }  
    }else{  
        printf("Nao achei o arquivo.");  
    }  
  
    fclose(arq);  
    return 0;|      You, 2 months ago • 2 - ADD MOD
```

- fgets(nome-da-variavel-que-ira-colocar, numero_de_caracteres_pegos, nome_arquivo);
- Queremos pegar os 10 primeiros caracteres e colocar na variavel que criamos nome[10] e depois vamos imprimir.
- Gerando uma linha a mais que não da para ver.
- Para ter certeza de o que esta sendo impresso(pegó) é o que nos queremos, vamos utilizar a variavel resultado para receber o valor que fgets pega. E só vamos imprimir se o resultado for verdadeiro. Ou seja,vamos fazer uma verificação primeiro desse resultado

LEITURA DE ARQUIVOS

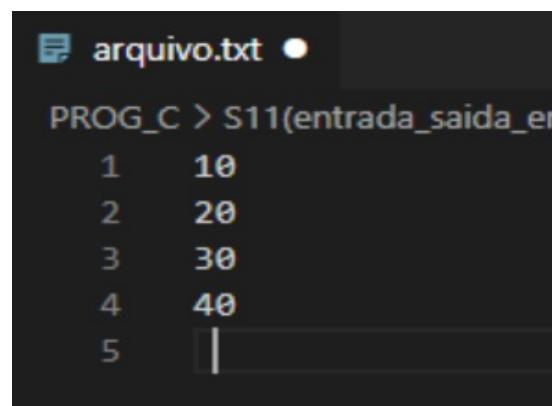
```
if(arq){  
    while(!feof(arq)){//feof - file and of file , ira fazer o lo  
        resultado = fgets(nome, 10, arq);//fgets - file gets  
        if(resultado){  
            printf("%s", nome);  
        }  
    }  
}else{  
    printf("Nao achei o arquivo.");  
}
```

```
if(arq){  
    while(!feof(arq)){//feof - file and of file , ira fa  
        resultado = fgets(nome, 10, arq);//fgets - file  
        printf("Resultado: %d\n", resultado);  
        if(resultado){  
            printf("%s", nome);  
        }  
    }  
}else{  
    printf("Nao achei o arquivo.");  
}
```

- Esse 0 é o que estava gerando a linha a mais, como estamos fazendo uma checagem do resultado (if2) o zero não entra mais.

OUTRA SITUAÇÃO

- Imagine que tenhamos um arquivo somente com numero, e precisamos fazer alguma operação com eles..
- Por exemplo somar



LEITURA DE ARQUIVOS

```
int main(){

    //declarando
    FILE *arq;
    int num, soma, resultado;

    soma = 0;

    //abrir o arquivo
    arq = fopen("arquivo.txt","r");

    //fazer a verificação se vai conseguir abrir
    if(arq){
        //verdadeiro, precisamos percorrer o arquivo
        while(!feof(arq)){
            resultado = fscanf(arq,"%d",&num);
            //ver o que a variavel resultado esta recebendo.
            printf("Resultado: %d\n", resultado);

            You, 2 months ago • ALT NAMES
            if(resultado == 1){
                soma = soma + num;
            }

        }
    }else{
        printf("FNF");
    }
    printf("\n\n\n\n A soma dos numeros encontrados eh: %d\n\n\n\n",soma);

    fclose(arq);
    return 0;
}
```

- Resultado

A soma dos numeros encontrados eh: 140

- if{}else{} - checagem de arquivo
- while(){} - percorrer o arquivo.
- Descrição:Na linha 1 do arquivo.txt temos o valor 10(num=10), A variavel soma(0) será somada por elamesma mais a variavel num(10) , guardando o novo valor soma(10).
- Essa sequencia se repetirá ate o EOF (end of file).
- Depois ira imprimir o valor da soma.

LEITURA DE ARQUIVOS

- O total seria 100, esta dando errado, pois a ultima linha esta sendo impressa, ou seja, antes do EOF existe uma linha vazia que não eh vista, temos que fazer uma verificação para que a ultima linha não apareça com a ajuda da variavel RESULTADO.
- Resultado ira receber o que o scanf conseguir.
- E depois fazemos uma verificação, se resultado for = 1, faz a soma.

```
A soma dos numeros encontrados eh: 100
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(entrada_saida_em_C)\2-LEITURA_DE_ARQUIVOS> []

13 //fazer a verificação se vai conseguir abrir
14 if(arq){
15     //verdadeiro, precisamos percorrer o arquivo
16     while(!feof(arq)){
17         resultado = fscanf(arq,"%d",&num);
18         //ver o que a variavel resultado esta recebendo
19         //printf("Resultado: %d\n", resultado);
20
21         if(resultado == 1){
22             soma = soma + num;
23         }
24     }
25 } else{
```

- Vamos entender porque RESULTADO == 1

```
Resultado: 1 PRIMEIRO NUMERO=10 15 if(arq){
Resultado: 1 16     //verdadeiro, precisamos percorrer o arquivo
Resultado: 1 17     while(!feof(arq)){
Resultado: 1 18         resultado = fscanf(arq,"%d",&num);
Resultado: 1 19         //ver o que a variavel resultado esta r
Resultado: -1 20         printf("Resultado: %d\n", resultado);
21
22         if(resultado == 1){
23             soma = soma + num;
24     }
25 } else{
```

- (1) = significa que ele encontrou 1 numero naquele local.
- (-1) = significa nenhum numero encontrado.
- Por isso precisamos fazer a verificação de efetuar a soma se o resultado for igual a 1. Porque assim se ele não encontrar um numero, a variavel não será substituida, ou seja, mantem o valor final.

LEITURA DE ARQUIVOS

- O total seria 100, esta dando errado, pois a ultima linha esta sendo impressa, ou seja, antes do EOF existe uma linha vazia que não eh vista, temos que fazer uma verificação para que a ultima linha não apareça com a ajuda da variavel RESULTADO.
- Resultado ira receber o que o scanf conseguir.
- E depois fazemos uma verificação, se resultado for = 1, faz a soma.

```
A soma dos numeros encontrados eh: 100
PS C:\Users\Gabi\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S11(entrada_saida_em_C)\2-LEITURA_DE_ARQUIVOS> []

13 //fazer a verificação se vai conseguir abrir
14 if(arq){
15     //verdadeiro, precisamos percorrer o arquivo
16     while(!feof(arq)){
17         resultado = fscanf(arq,"%d",&num);
18         //ver o que a variavel resultado esta recebendo
19         //printf("Resultado: %d\n", resultado);
20
21         if(resultado == 1){
22             soma = soma + num;
23         }
24     }
25 } else{
```

- Vamos entender porque RESULTADO == 1

```
Resultado: 1 PRIMEIRO NUMERO=10 15 if(arq){
Resultado: 1 16     //verdadeiro, precisamos percorrer o arquivo
Resultado: 1 17     while(!feof(arq)){
Resultado: 1 18         resultado = fscanf(arq,"%d",&num);
Resultado: 1 19         //ver o que a variavel resultado esta r
Resultado: -1 20         printf("Resultado: %d\n", resultado);
21
22         if(resultado == 1){
23             soma = soma + num;
24     }
25 } else{
```

- (1) = significa que ele encontrou 1 numero naquele local.
- (-1) = significa nenhum numero encontrado.
- Por isso precisamos fazer a verificação de efetuar a soma se o resultado for igual a 1. Porque assim se ele não encontrar um numero, a variavel não será substituida, ou seja, mantem o valor final.

ESCRITA EM ARQUIVOS

- Vamos criar um arquivo e colocar dados nele.

```
int main(){
    FILE *arq;
    char fruta[10], resultado;

    arq = fopen("frutas.txt", "w");//w = write | OVERWIRTE IF FILE EXIST

    if(arq){
        printf("Informe uma fruta, ou 0 para sair: \n");
        fgets(fruta, 10, stdin);//STDDIN = STANDARD INPUT
        while(fruta[0] != '0'){

            fputs(fruta,arq);//coloca a fruta no arquivo.

            printf("Informe uma fruta, ou 0 para sair: \n");
            fgets(fruta, 10, stdin);
        }
    }else{
        printf("\nArquivo não criado.\n");
    }

    fclose(arq);
    return 0;
}
```

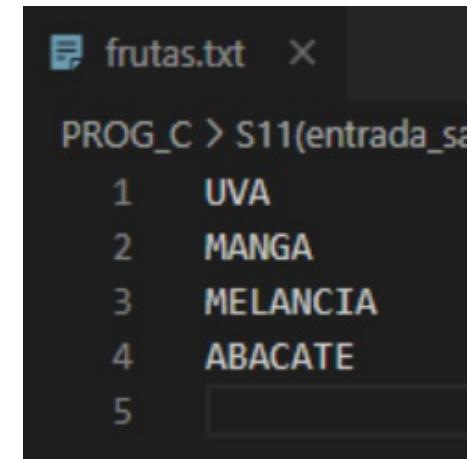
```
Informe uma fruta, ou 0 para sair:
manga
Informe uma fruta, ou 0 para sair:
uva
Informe uma fruta, ou 0 para sair:
melancia
Informe uma fruta, ou 0 para sair:
abacate
Informe uma fruta, ou 0 para sair:
0
```

"w" cria o arquivo. ou o sobre escreve por um novo...

- stdin : guarde no vetor fruta, ate 10 caracteres, da entrada padrao(teclado).

- Enquanto fruta na posição 0 for diferente de 0, guarde no vetor fruta[10].

- fputs : guarde fruta no arquivo.



ESCRITA EM ARQUIVOS

- ENTRADAS + LINHA EM BRANCO

Modo	Significado
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"rb"	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
"wb"	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
"ab"	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
"r+"	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
"w+"	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
"a+"	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"r+b"	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
"w+b"	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário

S11 - ENTRADA E SAIDA EM C

1 - MANIPULANDO E/S

MEMORIA DO COMPUTADOR

FILE *arq;



fopen "w"

3



ANOTAÇÕES

```
#include <stdio.h>
//FILE == ARQUIVO

int main(){
    FILE *arq;

    //fopen(nome-do-arquivo, forma-de-abertura);
    //w - abrir o arquivo para escrita
    //r - abrir o arquivo para leitura (não podemos editar)
    //wa - abrir o arquivo para adição de conteúdo (existir - conteúdo add nas linhas abaixo)
    arq = fopen("arquivo.txt","w");

    //Sempre que finalizarmos a manipulação de um arquivo, devemos fechá-lo.
    //fclose(nome-da-variável)
    fclose(arq);

    return 0;
}
```

W - SE O ARQUIVO NÃO EXISTE, ELE É CRIADO.

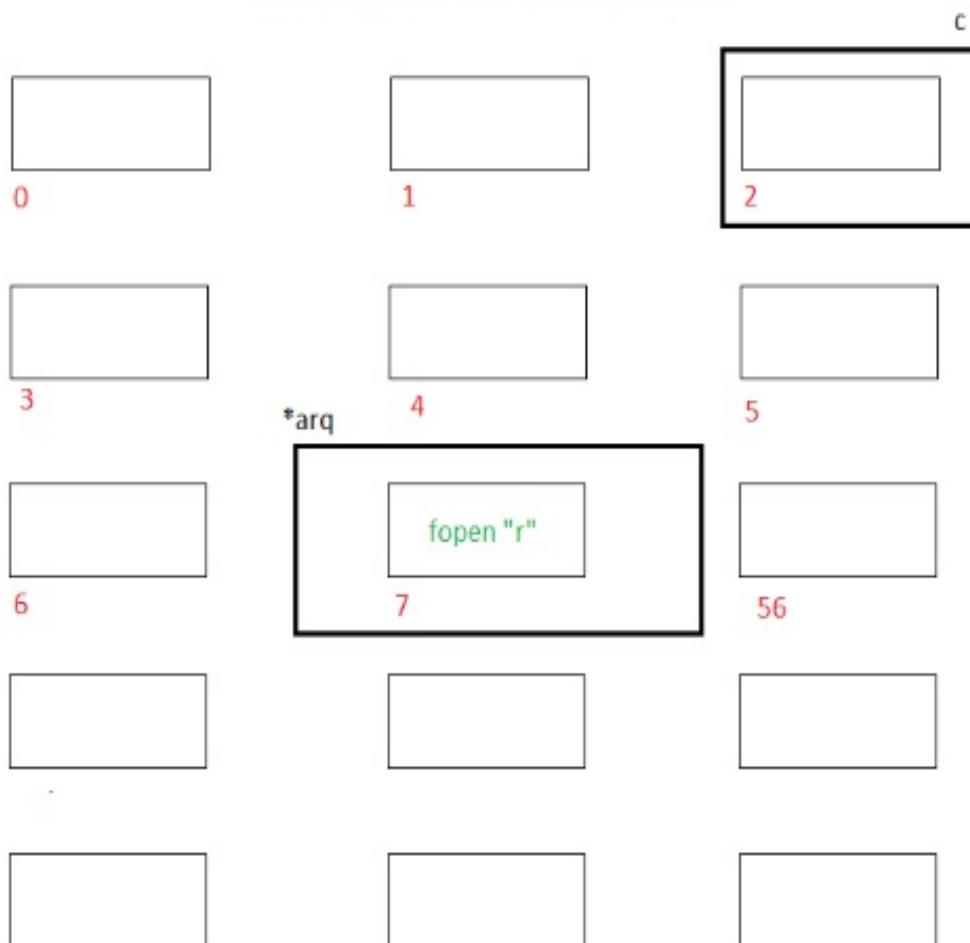
~BASICAMENTE~

- Para manipular arquivos temos que criar um tipo específico de dados (FILE) com um ponteiro (*arq).
- O ponteiro irá receber o endereço da função que abre o arquivo.

S11 - ENTRADA E SAIDA EM C

2 - LEITURA DE ARQUIVOS

MEMORIA DO COMPUTADOR



ANOTAÇÕES

```
1 #include <stdio.h>
2
3 int main(){
4
5     FILE *arq;
6     char c;
7
8     arq = fopen("arquivo.txt", "r");
9
10    if(arq){//sempre importante fazer essa verificação.
11        while((c = getc(arq)) != EOF){// EOF = END OF FILE
12            printf("%c",c);
13        }
14    }else{
15        printf("Nao achei o arquivo!");
16    }
17
18    fclose(arq);
19    return 0;
20}
```

BASICAMENTE

- Cria FILE e CHAR;
- ADD CONTEUDO_arq (fopen "r")
- if: Faz a verificação para ver se existe um arquivo.
- while : Percorre pelo arquivo ate EOF.

FUNÇÕES

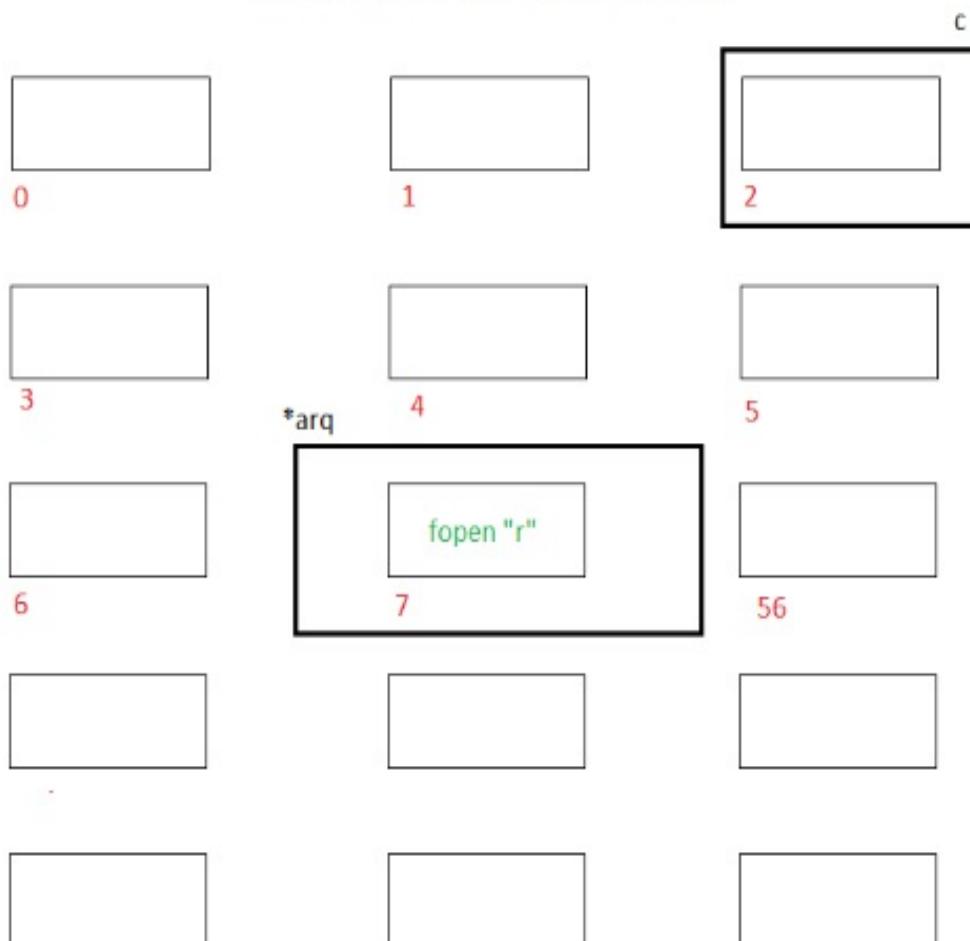
c = getc(arq) != EOF

get_(pega) c_(caractere) de arq(arquivo) enquanto nao(!=) EOF(end of file)

S11 - ENTRADA E SAIDA EM C

2 - LEITURA DE ARQUIVOS

MEMORIA DO COMPUTADOR



ANOTAÇÕES

```
1 #include <stdio.h>
2
3 int main(){
4
5     FILE *arq;
6     char c;
7
8     arq = fopen("arquivo.txt", "r");
9
10    if(arq){//sempre importante fazer essa verificação.
11        while((c = getc(arq)) != EOF){// EOF = END OF FILE
12            printf("%c",c);
13        }
14    }else{
15        printf("Nao achei o arquivo!");
16    }
17
18    fclose(arq);
19    return 0;
20}
```

~BASICAMENTE~

- Cria FILE e CHAR;
- ADD CONTEUDO_arq (fopen "r")
- if: Faz a verificação para ver se existe um arquivo.
- while : Percorre pelo arquivo ate EOF.

~FUNÇÕES~

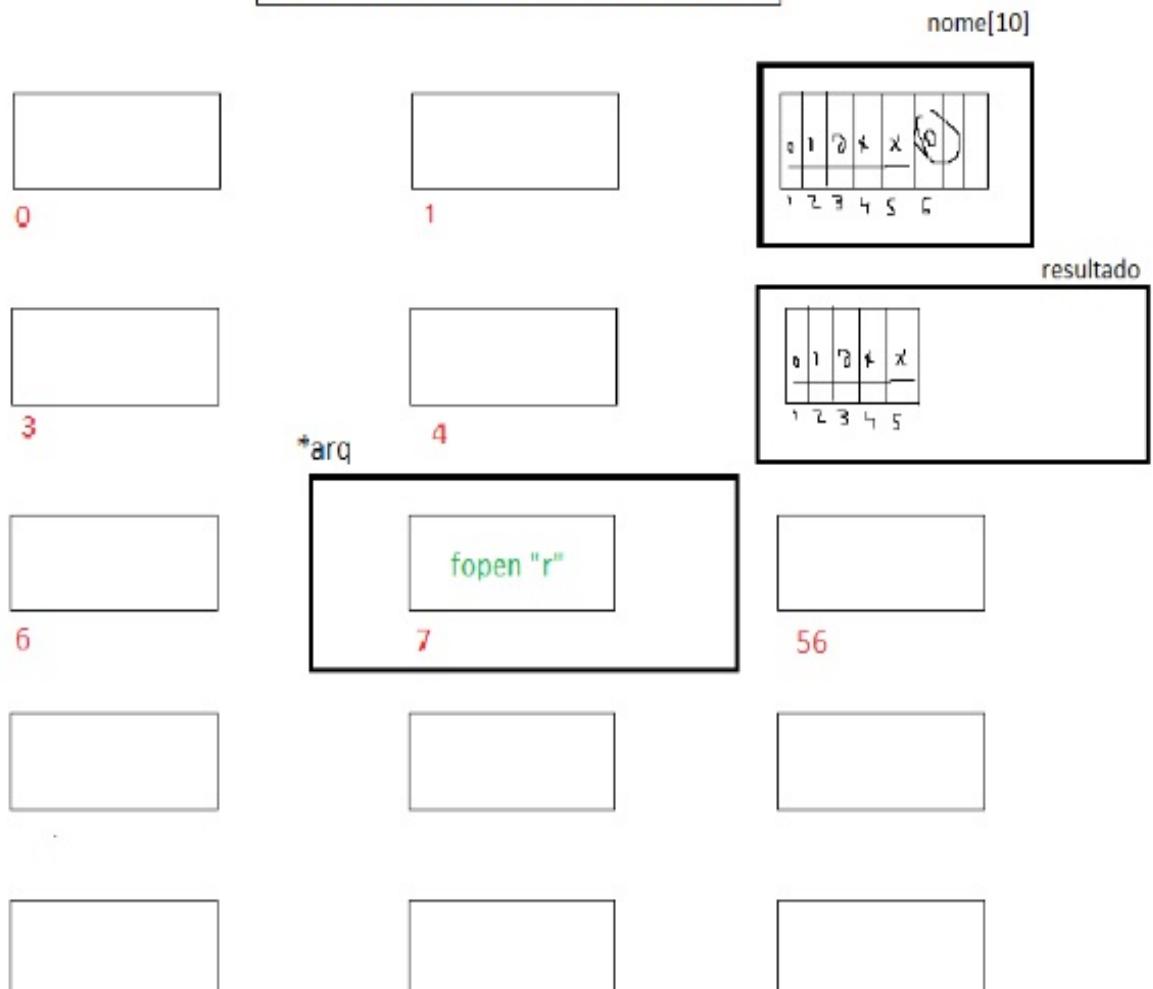
c = getc(arq) != EOF

get_(pega) c_(caractere) de arq(arquivo) enquanto nao(!=) EOF(end of file)

S11 - ENTRADA E SAIDA EM C

2.1- LEITURA DE ARQUIVOS

MEMÓRIA DO COMPUTADOR



ANOTAÇÕES

```
#include <stdio.h>

int main(){

    FILE *arq;
    char nome[10], *resultado;

    arq = fopen("arquivo.txt","r");

    if(arq){
        while(!feof(arq)){//feof - file end of file , ira fazer o loop enquanto n chegar ao fim de arquivo
            resultado = fgets(nome, 10, arq);//fgets - file gets
            printf("Resultado: %d\n", resultado);
            if(resultado){}
                printf("%s", nome);
            }
        }
    else{
        printf("Nao achei o arquivo.");
    }

    fclose(arq);
    return 0;
}
```

~BASICAMENTE~

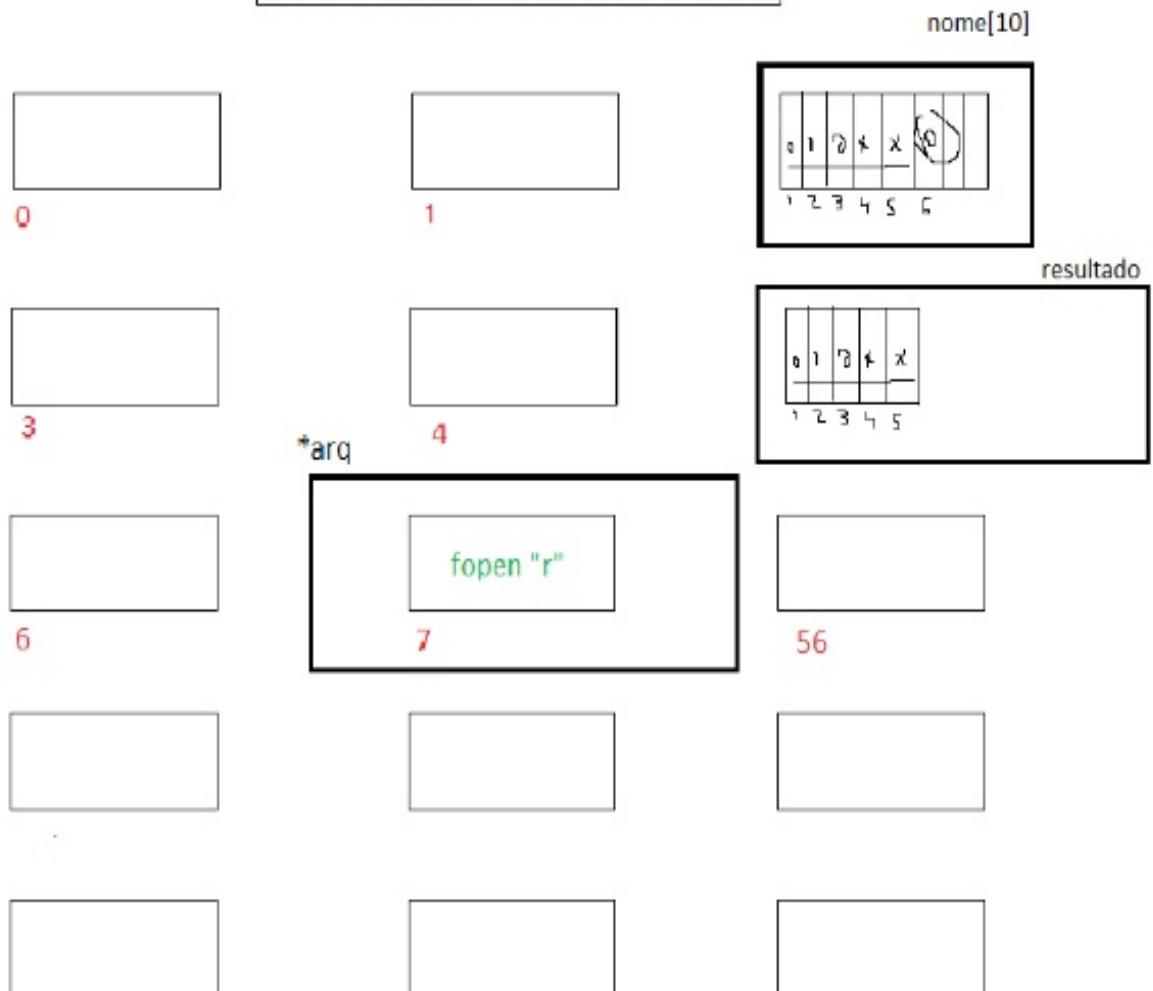
- Criamos um vetor[10] e um ponteiro `*resultado`.
- Na verificação, `resultado`(conteúdo) irá receber a função `gets(busca)` que possui como parâmetros (`nome-da-variável-que-ira-colocar`, `numero_de_caracteres_pegos`, `nome_arquivo`);
- Depois irá imprimir o resultado percorrendo assim o arquivo linha por linha

VERIFICAÇÃO = Linha invisível

S11 - ENTRADA E SAIDA EM C

2.1- LEITURA DE ARQUIVOS

MEMÓRIA DO COMPUTADOR



ANOTAÇÕES

```
#include <stdio.h>

int main(){

    FILE *arq;
    char nome[10], *resultado;

    arq = fopen("arquivo.txt","r");

    if(arq){
        while(!feof(arq)){//feof - file end of file , ira fazer o loop enquanto n chegar ao fim de arquivo
            resultado = fgets(nome, 10, arq);//fgets - file gets
            printf("Resultado: %d\n", resultado);
            if(resultado){}
                printf("%s", nome);
            }
        }
    }else{
        printf("Nao achei o arquivo.");
    }

    fclose(arq);
    return 0;
}
```

~BASICAMENTE~

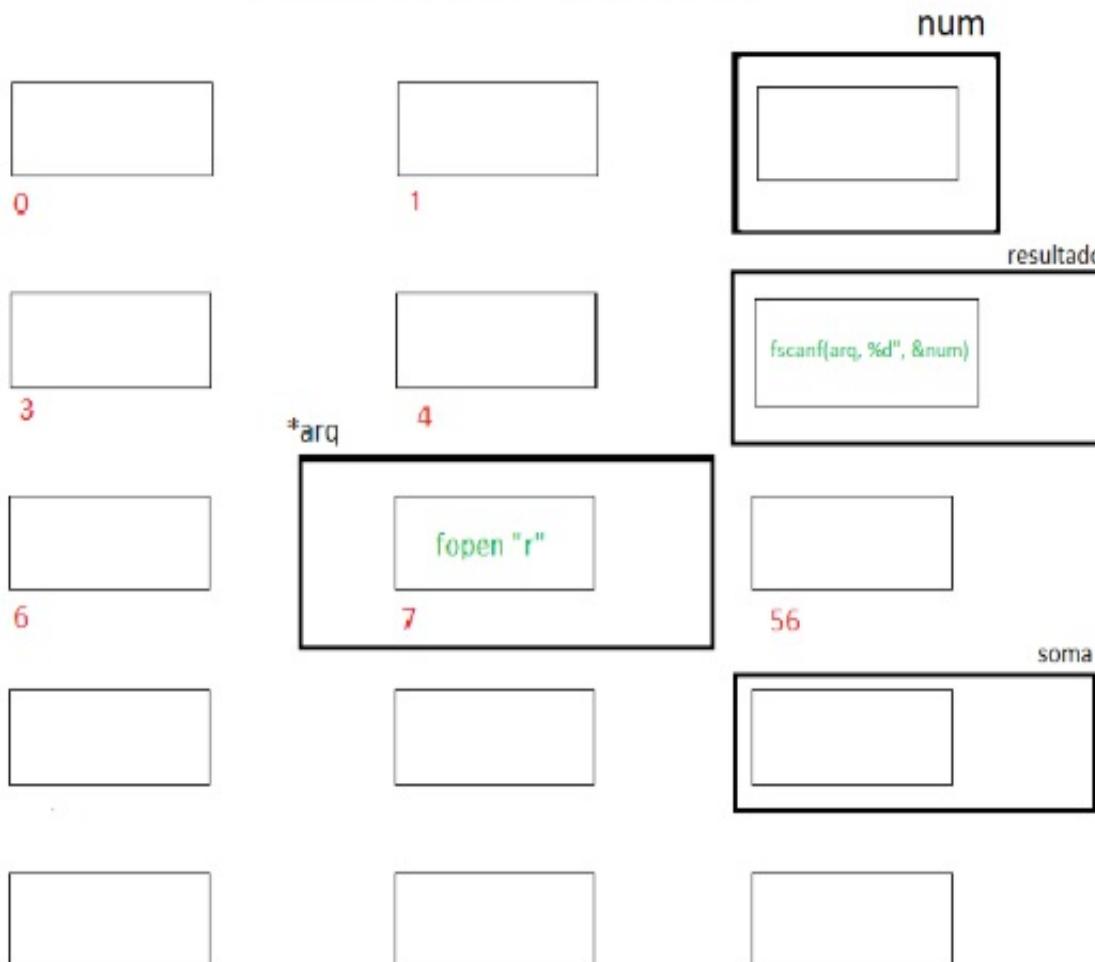
- Criamos um vetor[10] e um ponteiro `*resultado`.
- Na verificação, `resultado`(conteúdo) irá receber a função `gets(busca)` que possui como parâmetros (`nome-da-variável-que-ira-colocar`, `numero_de_caracteres_pegos`, `nome_arquivo`);
- Depois irá imprimir o resultado percorrendo assim o arquivo linha por linha

VERIFICAÇÃO = Linha invisível

S11 - ENTRADA E SAÍDA EM C

2.2 - LEITURA DE ARQUIVOS

MEMÓRIA DO COMPUTADOR



ANOTAÇÕES

```
int main(){
    //declarando
    FILE *arg;
    int num, soma, resultado;

    soma = 0;

    //abrir o arquivo
    arg = fopen("arquivo.txt", "r");

    //fazer a verificação se vai conseguir abrir
    if(arg){
        //verdadeiro, precisamos percorrer o arquivo
        while(!feof(arg)){
            resultado = fscanf(arg, "%d", &num);
            //ver o que a variável resultado está recebendo.
            printf("Resultado: %d\n", resultado);

            You, an hour ago + 2 - ADD MOD
            if(resultado == 1){
                soma = soma + num;
            }
        }
    }else{
        printf("FNF");
    }
    printf("\n\n\nA soma dos números encontrados eh: %d\n\n\n",soma);

    fclose(arg);
    return 0;
}
```

"BASICAMENTE"

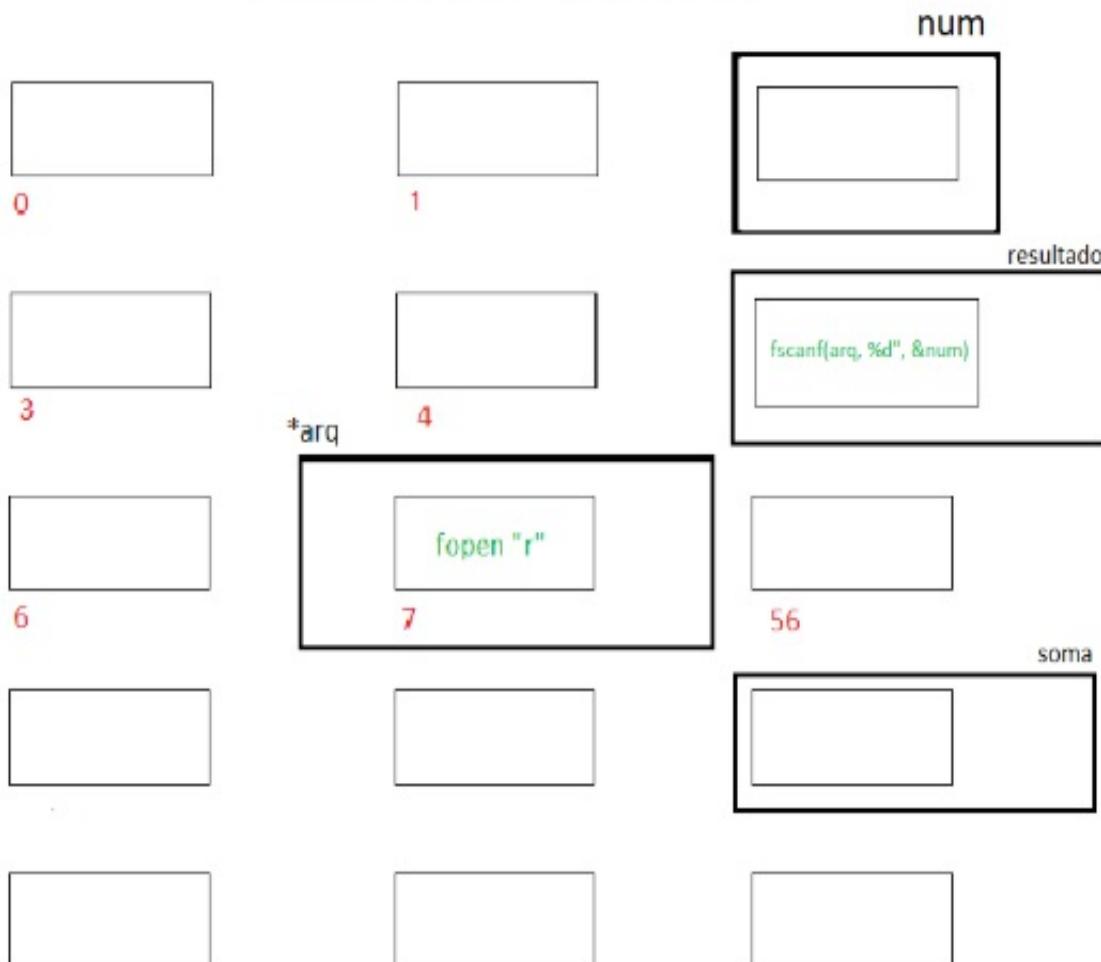
- Guarda em resultado a função fscanf, que vai receber do arquivo(arg) um inteiro(%d) e irá guardar no endereço de memória da variável num (&num).
- Impressão resultado
- Se resultado == 1 (se achar algum número), faça a soma.
- Impressão soma

VERIFICAÇÃO = LINHA

S11 - ENTRADA E SAÍDA EM C

2.2 - LEITURA DE ARQUIVOS

MEMÓRIA DO COMPUTADOR



ANOTAÇÕES

```
int main(){
    //declarando
    FILE *arg;
    int num, soma, resultado;

    soma = 0;

    //abrir o arquivo
    arg = fopen("arquivo.txt", "r");

    //fazer a verificação se vai conseguir abrir
    if(arg){
        //verdadeiro, precisamos percorrer o arquivo
        while(!feof(arg)){
            resultado = fscanf(arg, "%d", &num);
            //ver o que a variável resultado está recebendo.
            printf("Resultado: %d\n", resultado);

            You, an hour ago + 2 - ADD MOD
            if(resultado == 1){
                soma = soma + num;
            }
        }
    }else{
        printf("FNF");
    }
    printf("\n\n\n A soma dos números encontrados eh: %d\n\n\n",soma);

    fclose(arg);
    return 0;
}
```

"BASICAMENTE"

- Guarda em resultado a função fscanf, que vai receber do arquivo(arg) um inteiro(%d) e irá guardar no endereço de memória da variável num (&num).
- Impressão resultado
- Se resultado == 1 (se achar algum número), faça a soma.
- Impressão soma

VERIFICAÇÃO = LINHA

STRUCT

DEFININDO TIPOS

- STRUCT é uma estrutura de dados. E com ele, criamos nossa propria estrutura.
- Uma das formas de utilizar struct, eh ja no momento da declaração dela, declararmos uma ou mais variaveisque queremos fazer a utilização

```
struct st_aluno{  
    char matricula[10];  
    char nome[100];  
    char curso[50];  
    int ano_nascimento;  
};
```

```
#include <stdio.h>  
#include <string.h>  
  
struct st_aluno{  
    char matricula[10];  
    char nome[100];  
    char curso[50];  
    int ano_nascimento;  
};  
  
int main(){  
  
    struct st_aluno aluno1;  
  
    printf("Informe a matricula do aluno:\n");  
    fgets(aluno1.matricula,10,stdin);  
  
    printf("Informe o nome do aluno:\n");  
    fgets(aluno1.nome,100,stdin);  
  
    printf("Informe o curso do aluno:\n");  
    fgets(aluno1.curso,50,stdin);  
  
    printf("Informe o ano de nascimento do aluno:\n");  
    scanf("%d",&aluno1.ano_nascimento);  
  
    printf("-----Dados do aluno-----\n");  
    printf("Matricula :%s\n", aluno1.matricula);  
    printf("Nome do Aluno:%s\n", aluno1.nome);  
    printf("Curso :%s\n", aluno1.curso);  
    printf("Ano de Nascimento:%d\n", aluno1.ano_nascimento);  
  
    return 0;  
}
```

- A partir do struct temos acesso a uma estrutura completa para se trabalhar, a partir de uma variavelfazemos acesso tanto para colocar dados quanto para fazer leitura de dados.

DEFININDO TIPOS

DECLARANDO VARIÁVEL NA PROPRIA STRUCT

```
struct st_aluno{  
    char matricula[10];  
    char nome[100];  
    char curso[50];  
    int ano_nascimento;  
}aluno1,aluno2;
```

- Em alguns casos podemos precisar de varios alunos, então podemos declarar um array de alunos, a declaração pode ser feita tanto na struct quanto fora dela

```
struct st_aluno{  
    char matricula[10];  
    char nome[100];  
    char curso[50];  
    int ano_nascimento;  
}alunos[5];
```

```
int main(){  
    struct st_aluno alunos[5];
```

- Para o programa poder funcionar precisamos aplicar uma repetição.

- Sempre que temos um scanf e depois eh seguido de uma string, temos o problema de receber o enter, resolvemos isso colocando getchar(); depois do scanf

DEFININDO TIPOS

```
struct st_aluno{
    char matricula[10];
    char nome[100];
    char curso[50];
    int ano_nascimento;
}alunos[5];//array 0...4

int main(){

    //struct st_aluno alunos[5];

    for(int i = 0; i < 5; i++){
        printf("Informe a matricula do aluno:\n");
        fgets(alunos[i].matricula,10,stdin);

        printf("Informe o nome do aluno:\n");
        fgets(alunos[i].nome,100,stdin);

        printf("Informe o curso do aluno:\n");
        fgets(alunos[i].curso,50,stdin);

        printf("Informe o ano de nascimento do aluno:\n");
        scanf("%d",&alunos[i].ano_nascimento);
        getchar();

    }//fim_for

    for(int i = 0; i < 5; i++){
        printf("-----Dados do aluno %d-----\n", (i+1));
        printf("Matricula :%s\n", alunos[i].matricula);
        printf("Nome do Aluno:%s\n", alunos[i].nome);
        printf("Curso :%s\n", alunos[i].curso);
        printf("Ano de Nascimento:%d\n", alunos[i].ano_nascimento);
    }
}
```

DEFININDO TIPOS

```
struct st_aluno{
    char matricula[10];
    char nome[100];
    char curso[50];
    int ano_nascimento;
}alunos[5];//array 0...4

int main(){

    //struct st_aluno alunos[5];

    for(int i = 0; i < 5; i++){
        printf("Informe a matricula do aluno:\n");
        fgets(alunos[i].matricula,10,stdin);

        printf("Informe o nome do aluno:\n");
        fgets(alunos[i].nome,100,stdin);

        printf("Informe o curso do aluno:\n");
        fgets(alunos[i].curso,50,stdin);

        printf("Informe o ano de nascimento do aluno:\n");
        scanf("%d",&alunos[i].ano_nascimento);
        getchar();

    }//fim_for

    for(int i = 0; i < 5; i++){
        printf("-----Dados do aluno %d-----\n", (i+1));
        printf("Matricula :%s\n", alunos[i].matricula);
        printf("Nome do Aluno:%s\n", alunos[i].nome);
        printf("Curso :%s\n", alunos[i].curso);
        printf("Ano de Nascimento:%d\n", alunos[i].ano_nascimento);
    }
}
```

DEFININDO TIPOS

- Podemos criar structs de structs
- Vamos montar uma estrutura para contato, e outra estrutura com um array de 100 contatos.

```
You, 2 months ago | 1 author (You)
struct st_contato{
    char nome[100];
    int ano_nascimento;
    char telefone[20];
    char email[100];
}; //array 0...4

You, 2 months ago | 1 author (You)
struct st_agenda{
    struct st_contato contatos[100];
}agenda;

int main(){
    //struct st_aluno alunos[5];

    for(int i = 0; i < 3; i++){
        printf("Informe o nome:\n");
        fgets(agenda.contatos[i].nome, 100, stdin);

        printf("Informe o ano de nascimento:\n");
        scanf("%d", &agenda.contatos[i].ano_nascimento);
        getchar();

        printf("Informe o telefone:\n");
        fgets(agenda.contatos[i].telefone, 50, stdin);

        printf("Informe O EMAIL:\n");
        fgets(agenda.contatos[i].email, 100, stdin);

    } //fim_for

    printf("=====AGENDA DE CONTATO===== \n");

    for(int i = 0; i < 3; i++){
        printf("-----CONTATO: %d-----\n", (i+1));
        printf("Nome :%s\n", strtok(agenda.contatos[i].nome, "\n"));
        printf("Telefone :%s\n", strtok(agenda.contatos[i].telefone, "\n"));
        printf("Email:%s\n", strtok(agenda.contatos[i].email, "\n"));
        printf("Ano de Nascimento:%d\n", strtok(agenda.contatos[i].ano_nascimento, "\n"));
    }

    return 0;
}
```

- O fgets faz com que na hora de salvar o que eh pedido se salve tbm o enter depois da pessoa digitar, por isso o espaço na resposta, para corrigir isso usamos um comando na impressão.

strtok

- Faz parte da biblioteca string e a função ele eh, se no primeiro printf tiver um enter(\n), removemos ele e imprime sem o \n

USANDO TYPEDEF

- A gente pode em determinados momentos querer dar um apelido ou redefinir tipos. E podemos usar o `typedef` para isso.
- Ex: Podemos querer ter um dado do tipo float chamado notas

```
int main(){  
  
    typedef float nota;  
  
    //declarando_variaveis  
    nota prova1 = 7.0;  
    nota prova2 = 6.0;  
  
    nota soma = prova1 + prova2;  
  
    printf("A soma das notas eh: %f",soma);  
  
    return 0;  
}
```

- Basicamente redefinimos o nome do tipo de dado (float) para NOTA

USANDO UNION

- Ja sabemos que quando fazemos uma declaração de uma variavel, um espaço de memoria é alocado paraguardar o conteudo da mesma.
- Cada tipo de dado ocupa um tamnho de espaço na memoria, conseguimos verificar o quanto é esse espaço utilizando uma função chamada SIZEOF().

```
#include <stdio.h>

int main(){
    printf("\n");

    int numero = 42;
    float nota = 7.9;
    char letra = 'd';

    printf("A variavel 'numero' tem valor %d e ocupa %ld bytes em memoria.\n",numero,sizeof(numero));
    printf("A variavel 'nota' tem valor %.2f e ocupa %ld bytes em memoria.\n",nota,sizeof(nota));
    printf("A variavel 'letra' tem valor %c e ocupa %ld bytes em memoria.\n",letra, sizeof(letra));

    printf("\n");
    return 0;
}
```

You, a few seconds ago • Uncommitted changes

- Tanto faz voce passar o nome da variavel quanto seu tipo como parametro no sizeof. A variavel ocupa oespaço em memoria relativo ao dado.
- Outro detalHe é a colocação do %ld = long interger

```
A variavel 'numero' tem valor 42 e ocupa 4 bytes em memoria.
A variavel 'nota' tem valor 7.90 e ocupa 4 bytes em memoria.
A variavel 'letra' tem valor d e ocupa 1 bytes em memoria.
```

USANDO UNION

Vamos criar um código.

```
#include <stdio.h>
#include <string.h> //acesso as funções de string

You, a few seconds ago | 1 author (You)
union pessoa{
    char nome[100];// 1 char = 1 byte -> 1 * 100 = 100 bytes
    int idade;// 4 bytes
    //-> total de 104 bytes.
};

int main(){
    printf("\n");

    union pessoa pes;//union(nome[100],idade)

    strcpy(pes.nome, "Angelina Jolie"); //copiando a string para a variável pes.nome
    printf("Dados de %s\n",pes.nome);

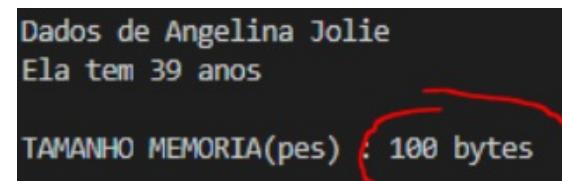
    pes.idade = 39;
    printf("Ela tem %d anos\n",pes.idade);

    printf("\nTAMANHO MEMORIA(pes) : %ld bytes",sizeof(pes));

    printf("\n");      You, a few seconds ago • Uncommitted changes
    printf("\n");
    return 0;
}
```

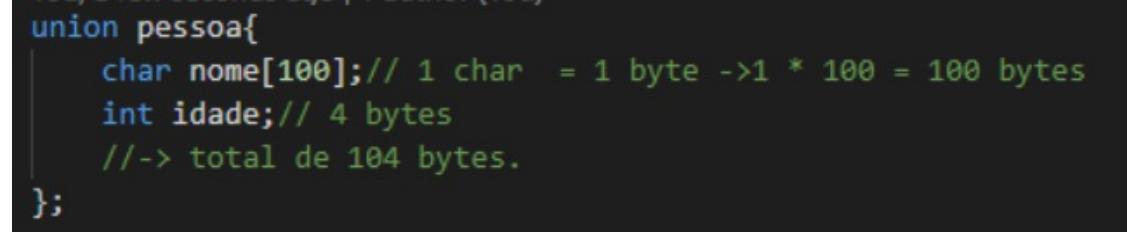
USANDO UNION

- A union possui a mesma estrutura da struct, podemos criar a variavel tanto ao criar a union quanto dentro do main.
- strcpy é uma função que copia uma string para a variável desejada.
- Depois veremos quanto a variável possui de tamanho, pelos cálculos deveria vir 4 bytes.



Dados de Angelina Jolie
Ela tem 39 anos
TAMANHO MEMORIA(pes) : 100 bytes

- Quando declaramos uma union, não importa quantas variáveis colocamos ali dentro, ela sempre irá ocupar somente o espaço da maior dessas variáveis.



```
union pessoa{  
    char nome[100];// 1 char = 1 byte -> 1 * 100 = 100 bytes  
    int idade;// 4 bytes  
    //=> total de 104 bytes.  
};
```

USANDO UNION

- Assim, ela utiliza o espaço para outras variaveis, sobrescrevendo o anterior...

```
int main(){
    printf("\n");

    union pessoa pes; //union(nome[100],idade)

    strcpy(pes.nome, "Angelina Jolie"); //copiando a string para a variavel pes.nome
    printf("Dados de %s\n",pes.nome);

    pes.idade = 39;
    printf("Ela tem %d anos\n",pes.idade);

    printf("Dados de %s\n",pes.nome);  

    printf("\nTAMANHO MEMORIA(pes) : %ld bytes",sizeof(pes));

    printf("\n");
    printf("\n");
    return 0;
}
```

You, a few seconds ago • Uncommitted changes

```
Dados de Angelina Jolie
Ela tem 39 anos
Dados de ' '
TAMANHO MEMORIA(pes) : 100 bytes
```

USANDO UNION

- Assim, ela utiliza o espaço para outras variaveis, sobrescrevendo o anterior...

```
int main(){
    printf("\n");

    union pessoa pes; //union(nome[100],idade)

    strcpy(pes.nome, "Angelina Jolie"); //copiando a string para a variavel pes.nome
    printf("Dados de %s\n",pes.nome);

    pes.idade = 39;
    printf("Ela tem %d anos\n",pes.idade);

    printf("Dados de %s\n",pes.nome);  

    printf("\nTAMANHO MEMORIA(pes) : %ld bytes",sizeof(pes));

    printf("\n");
    printf("\n");
    return 0;
}
```

You, a few seconds ago • Uncommitted changes

```
Dados de Angelina Jolie
Ela tem 39 anos
Dados de ' '
TAMANHO MEMORIA(pes) : 100 bytes
```

USANDO UNION

- Ao tentar imprimir novamente o nome que foi adicionado,a memoria alocada não possui mais esseconteudo, logo não imprime...
- [0]
- [angelina jolie]
- [39] , no final, so tem o ultimo valor.

OUTRO EXEMPLO

```
PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S12(struct_en C)\3-UNION> cd ..\\"p36.exe"
0 valor de num1 eh : 9
0 valor de num2 eh : 9
0 valor de num3 eh : 9
0 valor de num4 eh : 9
0 valor de num5 eh : 9
PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDEMY\REP_UDEMY\PROG_C\S12(struct_en C)\3-UNION>
```

```
3 union numeros{
4     int num1, num2, num3, num4, num5;
5 }n;
6
7
8
9 int main(){
10
11     n.num1 = 1;
12     n.num2 = 3;
13     n.num3 = 5;
14     n.num4 = 7;
15     n.num5 = 9;
16
17     printf("\n0 valor de num1 eh : %d\n",n.num1);
18     printf("\n0 valor de num2 eh : %d\n",n.num2);
19     printf("\n0 valor de num3 eh : %d\n",n.num3);
20     printf("\n0 valor de num4 eh : %d\n",n.num4);
21     printf("\n0 valor de num5 eh : %d\n",n.num5);
22
23
24
25
26     return 0;
27 }
```

USANDO UNION

- Ao tentar imprimir novamente o nome que foi adicionado,a memoria alocada não possui mais esseconteudo, logo não imprime...
- [0]
- [angelina jolie]
- [39] , no final, so tem o ultimo valor.

OUTRO EXEMPLO

```
PS C:\Users\valdi\Documents\TEORIA_INDIVIDUAL\VIDEOMAPAS\REP_LDEM\006_C512\struct_en C:\US-UNION> cmd /c .\p36.exe
valor de num1 eh : 1
valor de num2 eh : 3
valor de num3 eh : 5
valor de num4 eh : 7
valor de num5 eh : 9
valor de num1 eh : 9
valor de num2 eh : 9
valor de num3 eh : 9
valor de num4 eh : 9
valor de num5 eh : 9
"n" está ocupando 4 bytes de memoria.
PS C:\Users\valdi\Documents\TEORIA_INDIVIDUAL\VIDEOMAPAS\REP_LDEM\006_C512\struct_en C:\US-UNION>
```

```
1 union numeros{
2     int num1, num2, num3, num4, num5;
3 }n;
4
5
6
7
8
9 int main(){
10
11     n.num1 = 1;
12     printf("\n0 valor de num1 eh : %d\n",n.num1);
13     n.num2 = 3;
14     printf("\n0 valor de num2 eh : %d\n",n.num2);
15     n.num3 = 5;
16     printf("\n0 valor de num3 eh : %d\n",n.num3);
17     n.num4 = 7;
18     printf("\n0 valor de num4 eh : %d\n",n.num4);
19     n.num5 = 9;
20     printf("\n0 valor de num5 eh : %d\n",n.num5);
21
22     printf("\n0 valor de num1 eh : %d\n",n.num1);
23     printf("\n0 valor de num2 eh : %d\n",n.num2);
24     printf("\n0 valor de num3 eh : %d\n",n.num3);
25     printf("\n0 valor de num4 eh : %d\n",n.num4);
26     printf("\n0 valor de num5 eh : %d\n",n.num5);
27
28     printf("\n\n" está ocupando %ld bytes de memoria.\n",sizeof(n));
29
30
31     return 0;
32 }
```

USANDO ENUM