

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

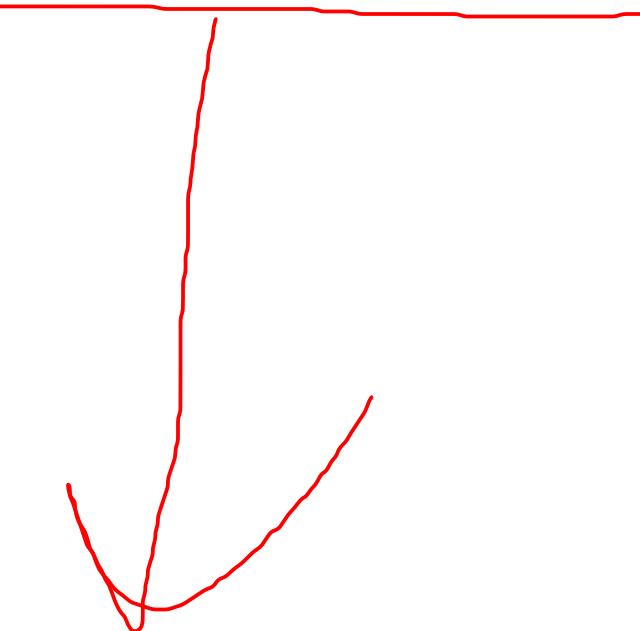
```
1 #include <stdio.h>
2
3 int main(void){          DECLARAÇÃO
4
5     int evento = 5 ;
6     char corrida = 'C';
7     float tempo = 27.25;
8
9     printf("O tempo vitorioso na eliminat'oria %c",corrida);
10    printf("\nda competi,c~ao %d foi %f.", evento, tempo);
11
12 return 1;
13 }
```

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

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 **DEREPENÇÃ*ão***

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

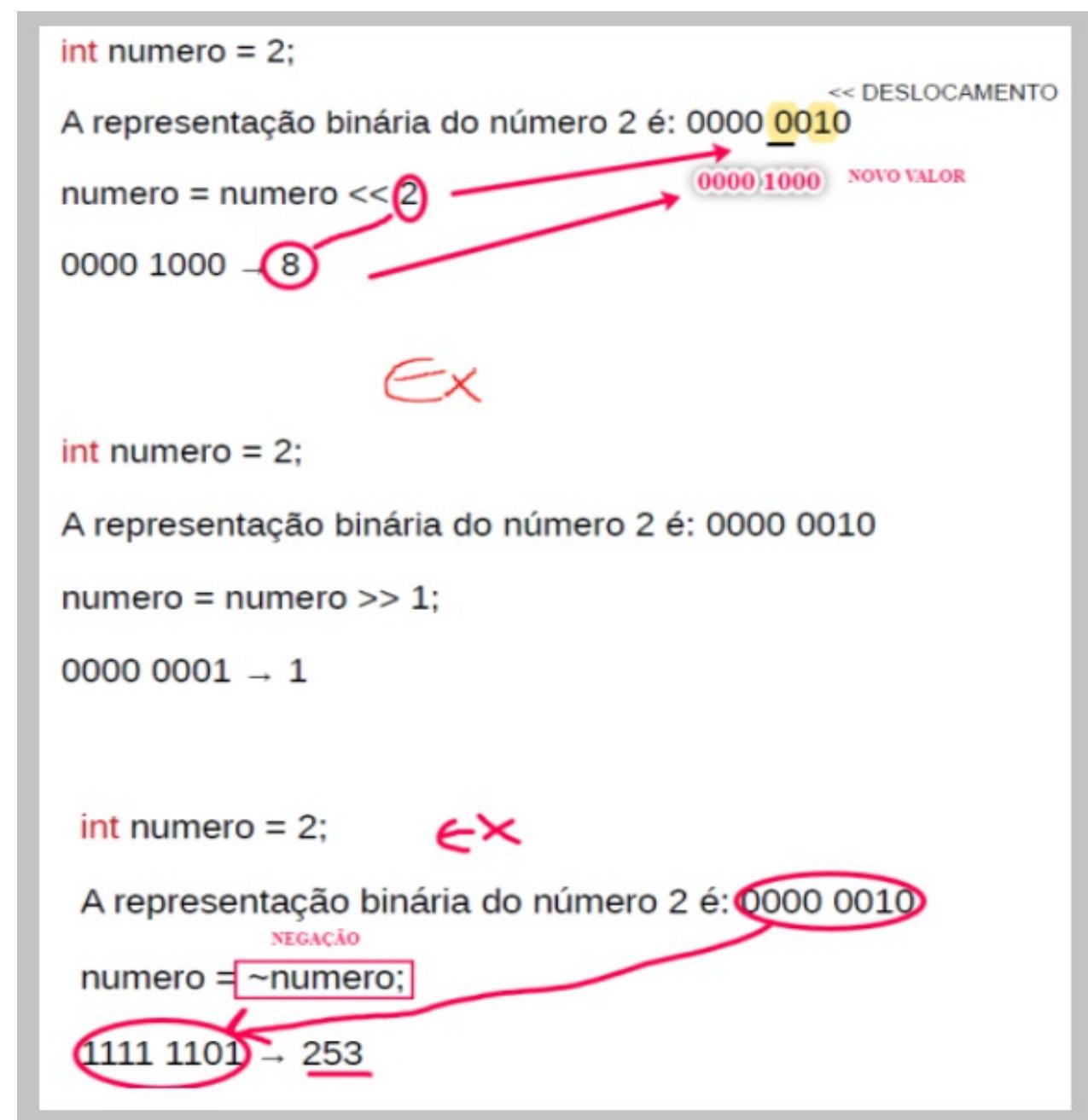
//deslocamento de bit para esquerda
valor = valor >> 1; 0010>>1=0001(1)
printf("Valor vale %d\n",valor);

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
```



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 code to its corresponding definition at line 6 and from the word 'mult' in the code to its corresponding definition at line 15.

ARQUIVOS DE CABEÇALHO

H ajuda.h X

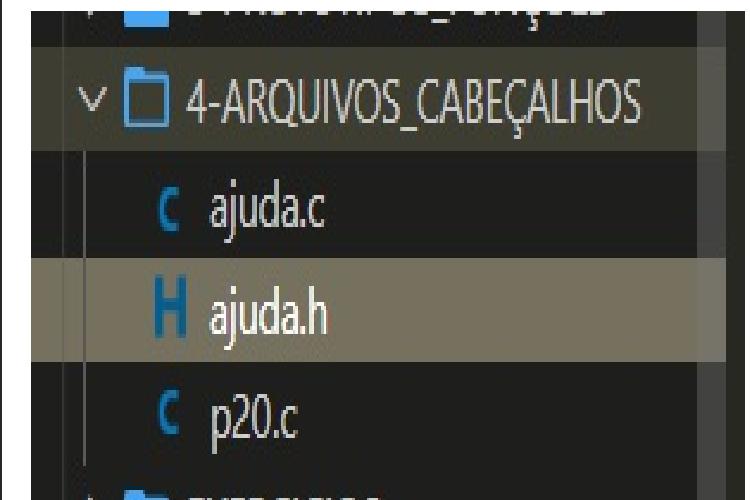
S9(funcões_c) > 4-ARQUIVOS_CABEÇALHOS > H ajuda.h > ...

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 red arrow points from the highlighted "#include" line in the code editor towards the "ajuda.c" file in the sidebar.
- A red circle highlights the "ajuda.c" file in the sidebar.



ARQUIVOS DE CABEÇALHO

The screenshot shows a terminal window with a dark background. At the top, there is a red oval highlighting the file name 'ajuda.h'. Below the file name, the path 'S9(funcoes_c) > 4-ARQUIVOS_CABEÇALHOS > ajuda.h > ...' is displayed. The main content of the terminal is a C code listing:

```
You, 2 months ago | 1 author (You)
1 // Código de implementação de funções
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
```

Red handwritten annotations are present in the terminal window:

- A large red arrow points from the word "ajuda.h" at the top left towards the "#include "ajuda.c"" line.
- A red circle highlights the "#include "ajuda.c"" line.
- A red circle highlights the "ajuda.c" file in the file tree on the right.
- A red arrow points from the "ajuda.c" file in the file tree towards the "#include "ajuda.c"" line in the terminal.

