

Algoritmos e Programação

A linguagem C

Anderson Fortes



A linguagem C



Linguagens de Programação

Baixo Nível

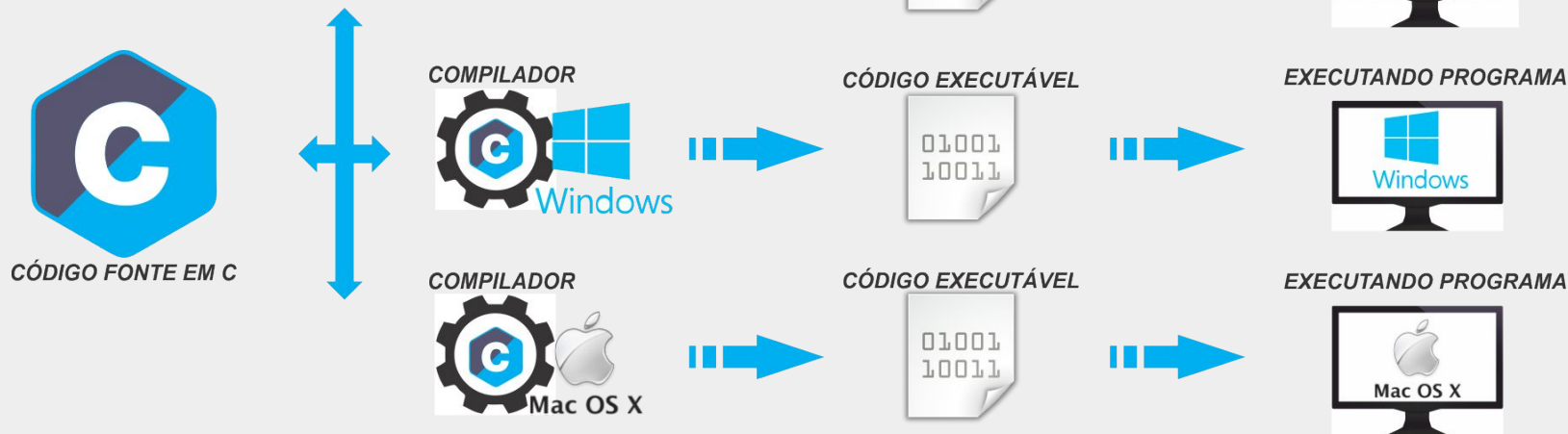
Alto Nível

Assembly

```
C:\DOCUME~1\GAREN~1>debug
-a
1543:0100 jmp 115
1543:0102 db 'Hello world!$'
1543:010F
-a 115
1543:0115 mov ah, 09
1543:0117 mov dx, 102
1543:011A int 21
1543:011C int 20
1543:011E
-h 11E 100
021E 001E
-r cx
CX 0000
:1E
-n hello.com
..
```



Compilação



Faz a Análise se o código está escrito em linguagem correta e se faz sentido.

Ranking 2019



Language Types



Web



Mobile



Enterprise



Embedded

Language Rank

Types

Spectrum Ranking

1. Java	  	100.0
2. C	  	99.2
3. C++	  	95.5
4. Python	 	93.4
5. C#	  	92.2
6. PHP		84.6
7. Javascript	 	84.3
8. Ruby		78.6
9. R		74.0
10. MATLAB		72.6



Por que aprender C?

“...O aprendizado de C é muito importante, senão fundamental, para uma sólida formação em programação.

Naturalmente, existem bons programadores que não conhecem a linguagem C, mas eu acredito que eles serão programadores ainda melhores se aprenderem a programar em C com desenvoltura.”

Por que aprender C?



“

....

Em C, somos obrigados a gerenciar explicitamente a memória que alocamos, podemos manipular diretamente endereços de memória, precisamos entender o conceito de passagem de parâmetro por valor e por referência, etc.”

Por que aprender C?



“

Outras linguagens oferecem suporte nativo às estruturas de dados mais usuais; em C, precisamos implementá-las.

O conhecimento adquirido ao implementar as estruturas nos faz ter um melhor discernimento de qual estrutura devemos usar numa determinada aplicação e nos capacita a adaptar estruturas já existentes quando necessário”

Por que aprender C?



“É bastante trabalhoso lidar com a alocação de memória quando escrevemos aplicações com algoritmos de alto nível.

No entanto, quando lidamos com tarefas de baixo-nível como aquelas que um núcleo (kernel) tem obrigação de desempenhar, como a de copiar um conjunto de bytes para uma placa de rede, torna-se altamente necessário um acesso direto à memória — algo que não é possível fazer com Java.”

Indo para a parte empolgante!



Instalando Dev C++

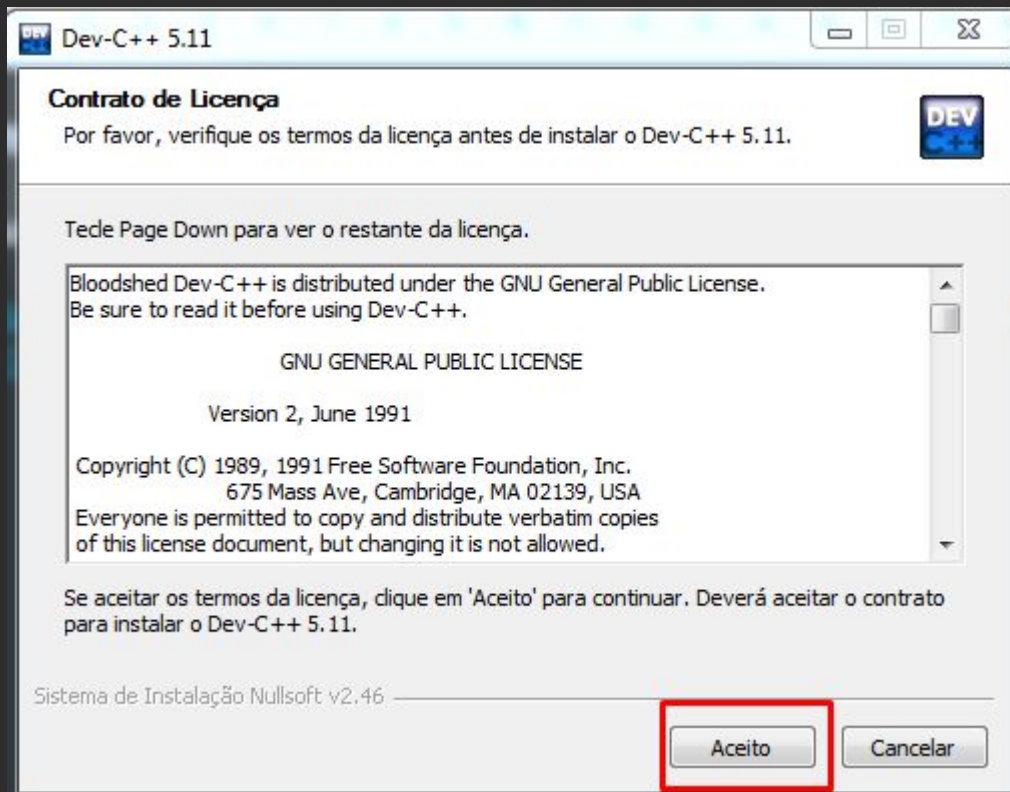


Fazer Download em:

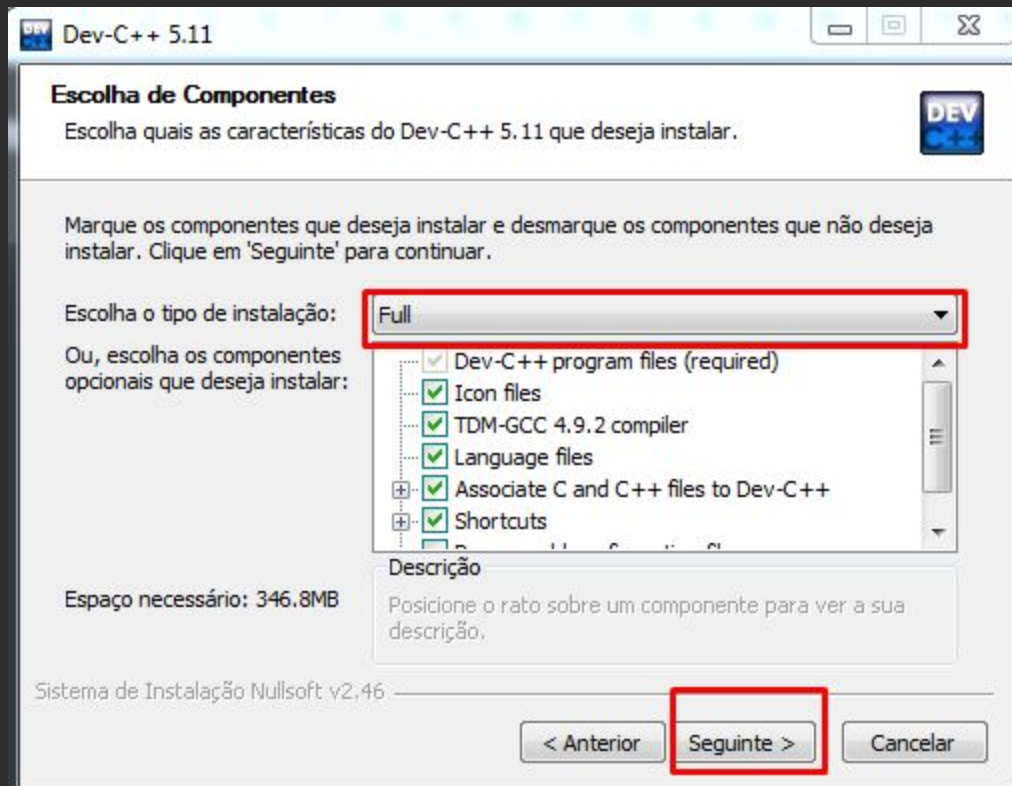
<https://dl.download.it/BR/dev-c.exe?st=YzbIT0QvhjK2dHA48pibhA&e=1654812188>



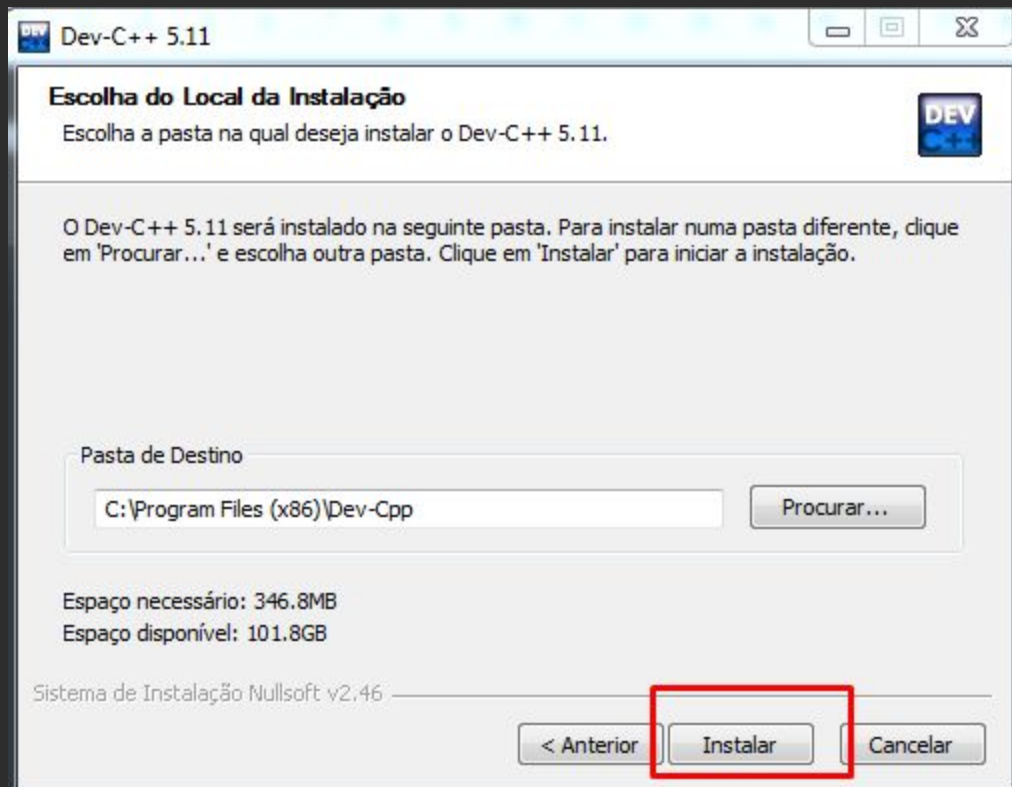
Instalando Dev C++



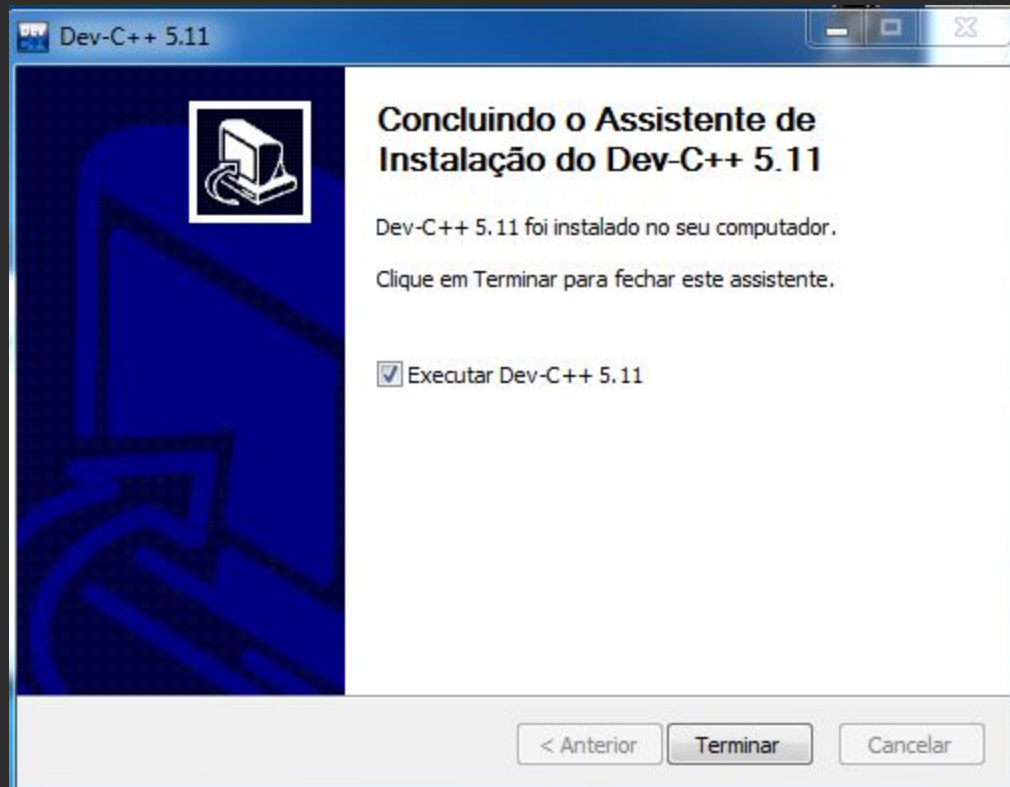
Instalando Dev C++



Instalando Dev C++



Instalando Dev C++



Executando o primeiro programa



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

```
/* Isso é um Comentário  
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {  
    printf("Olá Mundo");  
    return 0;  
}
```

Inclusão de bibliotecas

Executando o primeiro programa



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

Inclusão de bibliotecas

```
/* Isso é um Comentário  
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {  
    printf("Olá Mundo");  
    return 0;  
}
```

Função principal - sempre é chamada de main()

Executando o primeiro programa



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

Inclusão de bibliotecas

```
/* Isso é um Comentário  
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {  
    printf("Olá Mundo");  
    return 0;  
}
```

Função principal - sempre é chamada de main()

Final de toda instrução tem um ;

Executando o primeiro programa



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

```
/* Isso é um Comentário
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {
    printf("Olá Mundo");
    return 0;
}
```

Incluindo as bibliotecas necessárias para fazer o programa "Olá mundo!" no Dev C.

Função principal - sempre é chamada de main()

Final de toda instrução tem um ;



Usando variáveis

Declarando Variáveis - inteiros



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

```
/* Isso é um Comentário  
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {
```

```
    int a,b,soma;
```

```
    a = 10;
```

```
    b = 15;
```

```
    soma = a+b;
```

```
    printf("A soma de %i com %i eh igual a %i.\n",a,b,soma);
```

```
    return 0;
```

```
}
```

Variáveis inteiras



Declarando Variáveis - inteiros



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

```
/* Isso é um Comentário  
em bloco*/
```

```
//Isso é um comentário de u1 linha
```

```
int main() {  
    int a,b,soma;
```

```
    a = 10;
```

```
    b = 15;
```

```
    soma = a+b;
```

```
    printf("A soma de %i com %i eh igual a %i.\n",a,b,soma);
```

```
    return 0;
```

```
}
```

Variáveis inteiras

Impressão de Inteiros

Declarando Variáveis - Real



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

```
int main() {
```

```
    float nota1, nota2;
```

```
    float media;
```

```
    nota1 = 8.5
```

```
    nota2 = 6.5
```

```
    media = (nota1+nota2)/2
```

```
    printf("A media: %f \n", media);
```

```
}
```

Variáveis de ponto flutuante

Impressão de floats

Declarando Variáveis - Real



```
#include <stdio.h>

int main() {
    float nota1, nota2;
    float media;
    nota1 = 8.5
    nota2 = 6.5

    media = (nota1+nota2)/2

    printf("A media: %f \n", media);
}
```

Escrever esse programa
no Dev C.

Impressão de floats

Declarando Variáveis - Caractere



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

```
int main(){
```

```
char letra;
```

```
letra = 'J';
```

```
printf("a letra eh %c!\n", letra);
```

```
return 0;
```

```
}
```

Variável caractere

Imprimindo um caractere.

Essa forma só funciona para um único caractere. Veremos como trabalhar com palavras (String).



Ler dados do usuário

Lendo dados do Usuário

```
#include <stdio.h>

int main() {
    int anos, meses, dias;
    int total_dias;

    printf("Digite os anos: \n");
    scanf("%i",&anos);

    printf("Digite os meses: \n");
    scanf("%i",&meses);

    printf("Digite os dias: \n");
    scanf("%i",&dias);

    total_dias = anos*365 + meses*30 + dias;
    printf("voce ja viveu %i dias! \n", total_dias);

    return 0;
}
```

Aguarda o usuário digitar e salva na variável.

“%i” - significa que estamos lendo um inteiro

&anos - passamos o endereço de memória da variável anos.

Lendo dados do Usuário

```
#include <stdio.h>

int main() {
    int anos, meses, dias;
    int total_dias;

    printf("Digite os anos: \n");
    scanf("%i",&anos);

    printf("Digite os meses: \n");
    scanf("%i",&meses);

    printf("Digite os dias: \n");
    scanf("%i",&dias);

    total_dias = anos*365 + meses*30 + dias;
    printf("voce ja viveu %i dias! \n", total_dias);

    return 0;
}
```

Escrever um programa que leia a altura e base de um retângulo e mostre a sua área.

lendo um inteiro

&anos - passamos o endereço de memória da variável anos.

String - Vetor de caracteres

```
#include <stdio.h>

int main() {

    char nome[10];

    printf("Digite seu nome: ");
    scanf("%s",&nome);

    printf("Bem vindo, %s",nome);

    return 0;
}
```

Uma string é um vetor de caracteres..


“%s” - significa que estamos lendo uma string

A função printf :Caracteres de controle

Caracter de controle		Efeito
	\a	soa o alarme do microcomputador
	\b	o cursor retrocede uma coluna
	\f	alimenta página na impressora
Mais usado	\n	o cursor avança para uma nova linha
	\r	o cursor retrocede para a primeira coluna da linha
	\t	o cursor avança para próxima marca de tabulação
	\"	exibe uma única aspa
	\'	exibe um único apóstrofo
	\\	exibe uma única barra invertida

A função printf : Formatando a saída

```
int a = 678;  
float b = 112.3456;  
  
printf("\n| %10d |", a);  
printf("\n| %12d |", a);  
printf("\n| %12.3f |", b);  
printf("\n| %.2f |", b);
```



C:\Users\fortes\Documents\Anderson\URI\Algo

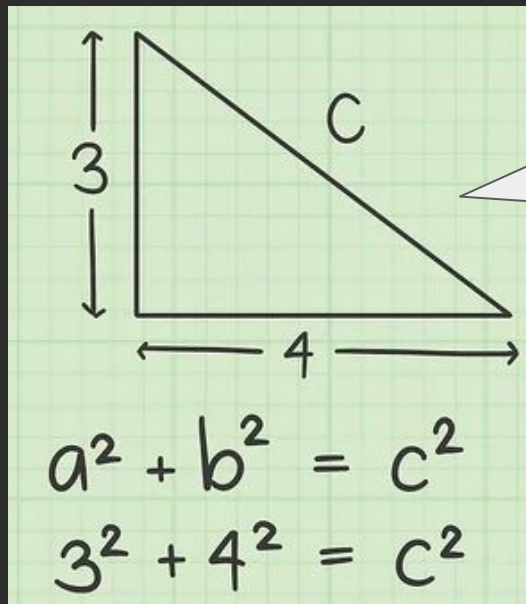
```
|          678 |  
|         678 |  
|        112.346 |  
| 112.35 |
```

Operadores

Operador	Resultado
+	soma de dois números quaisquer
-	diferença entre dois números quaisquer
*	produto de dois números quaisquer
/	quociente da divisão de dois números
%	resto da divisão de dois número inteiros

Exercício

Receber do usuário os valores representando comprimento de dois catetos e mostrar o comprimento da hipotenusa.



Dica: Existe uma função pronta que calcula a raiz quadrada :D

chama-se **sqrt()**

tbm existe a de exponenciação, chama-se **pow(base,expoente)**

Basta importar a biblioteca **math.h**

Estrutura de condição

O if faz o papel do SE

```
if (condição) {  
    //instrução ou instruções para condição verdadeira;  
}
```

Estrutura de condição

O if faz o papel do SE

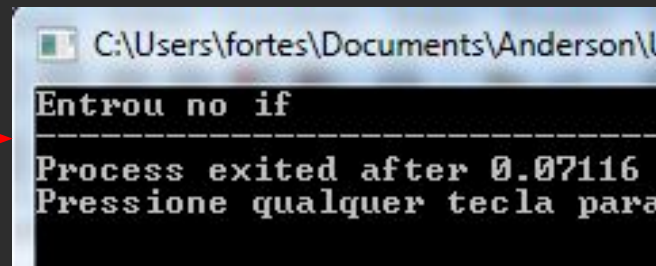
```
if (condição) {  
    //instrução ou instruções para condição verdadeira;  
}
```

IMPORTANTE: Em C, não existe um tipo específico para a representação de valores lógicos.

Zero representa **falso** e **qualquer outro** valor representa **verdadeiro**..

Estrutura de condição

```
if (1) {  
    printf("Entrou no if");  
}
```



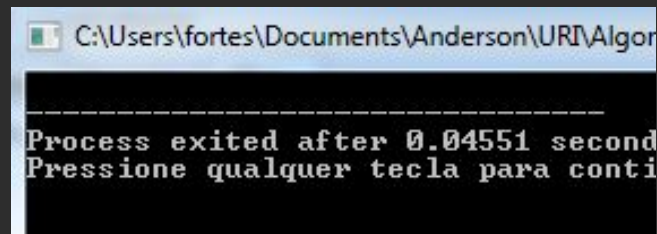
C:\Users\fortes\Documents\Anderson\U

Entrou no if

Process exited after 0.07116

Pressione qualquer tecla para

```
if (0) {  
    printf("Entrou no if");  
}
```



C:\Users\fortes\Documents\Anderson\URI\Algor

Process exited after 0.04551 second

Pressione qualquer tecla para conti

Comparações

$a = b$ -> atribuição (a recebe b)
 $a == b$ -> comparação (a é igual a b?)

Operador relacional	Resultado
$x = y$	verdade se x for igual a y
$x != y$	verdade se x for diferente de y
$x < y$	verdade se x for menor que y
$x > y$	verdade se x for maior que y
$x \leq y$	verdade se x for menor ou igual a y
$x \geq y$	verdade se x for maior ou igual a y

Comparações



Operador relacional	Resultado
$x == y$	verdade se x for igual a y
$x != y$	verdade se x for diferente de y
$x < y$	verdade se x for menor que y
$x > y$	verdade se x for maior que y
$x <= y$	verdade se x for menor ou igual a y
$x >= y$	verdade se x for maior ou igual a y

Não funciona para comparar String.

Veremos em seguida como fazer isso.

Comparações

```
#include <stdio.h>

int main() {

    printf("%d %d %d", 5<6, 6>5, 10>11);

    return 0;
}
```

Digitar, executar e discutir.

estrutura if-else

```
if (4>5) {  
    printf("Entrou no if");  
}  
else{  
    printf("Entrou no else");  
}
```

estrutura if-else

```
if (4>5) {  
    printf("Entrou no if");  
}  
else{  
    printf("Entrou no else");  
}
```

3- Ler dois números e imprimir o maior deles.

estrutura if-else

```
if (4>5) {  
    printf("Entrou no if");  
}  
else{  
    printf("Entrou no else");  
}
```

3- Ler dois números e imprimir o maior deles.

13- Ler um número e imprimir se ele é par ou ímpar. Dica: a função mod representa o resto da divisão inteira de um número por outro.

Ex.: $3 \bmod 2 = 1$, pois 3 dividido por 2 é igual a 1, sobrando 1

Atividades para entrega

- 1 - Criar um programa que receba um número inteiro e diga qual é a metade desse número.
- 2- Criar um programa que receba 3 notas, calcule a média e diga se o aluno está aprovado (média ≥ 7) ou reprovado.
- 3 - Escreva um algoritmo para ler um valor e escrever se ele está no intervalo entre 1 e 10 ou não está.
- 4 - Ler um número e imprimir se ele é positivo, negativo ou zero.