

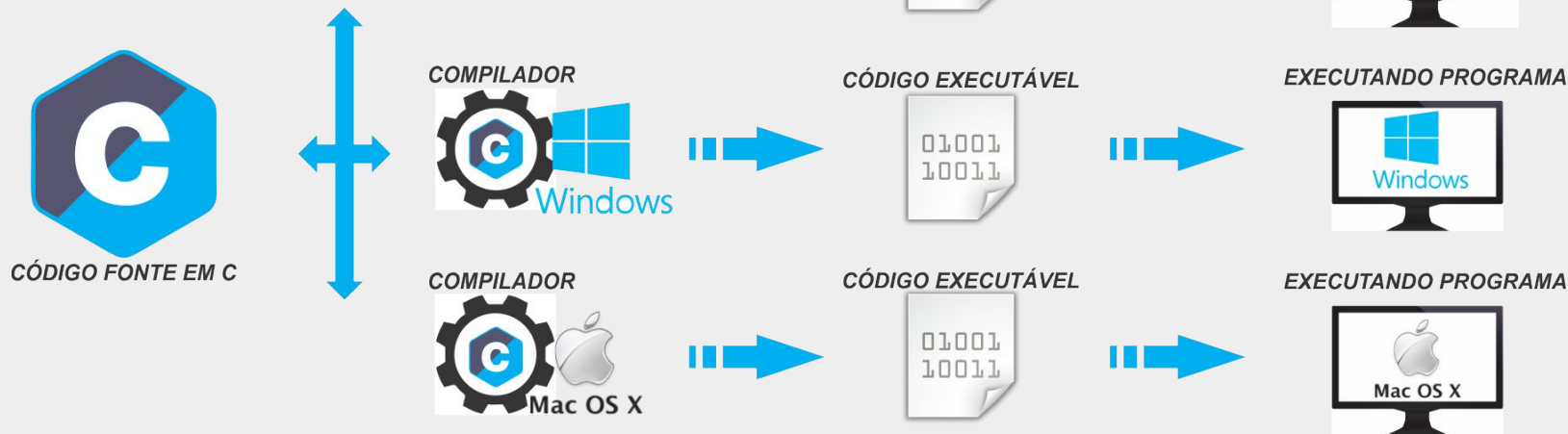
# Algoritmos e Programação

## printf, strings e estruturas de condição

Anderson Fortes



# Compilação



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

# 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>
```

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

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

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

Inclusão de bibliotecas

Função principal - sempre é chamada de main()

# 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

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 - 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.

# 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



# 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

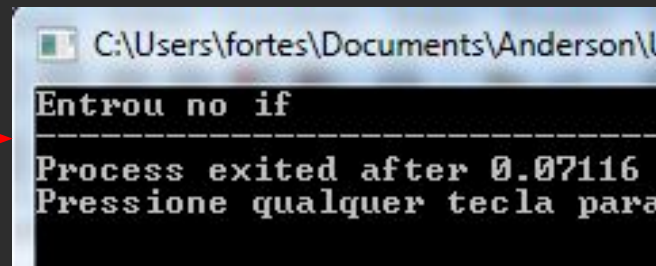
```
if (condição) {  
    //instrução ou instruções para condição verdadeira;  
}
```

**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

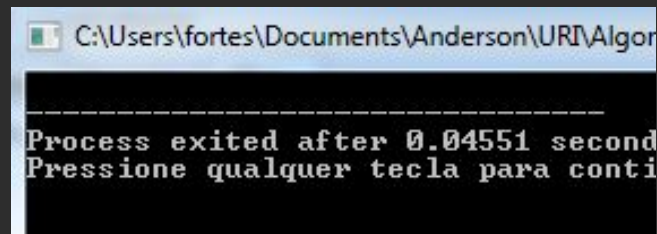
Entrou no if

-----

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 $y$
$x \leq y$	verdade se $x$ for menor ou igual a $y$
$x \geq y$	verdade se $x$ for maior ou igual a $y$

**Não funciona para comparar String.**

**Veremos em seguida como fazer isso.**



Hoje...

# Mais sobre linguagem C

Biblioteca `string.h`, expressões reduzidas e estruturas de repetição

# Biblioteca `<string.h>`

As strings, em C, não são tratadas como os demais tipos de variáveis. A biblioteca `string.h`, contém uma série de funções para manipular strings.

# Biblioteca `<string.h>`

As strings, em C, não são tratadas como os demais tipos de variáveis. A biblioteca `string.h`, contém uma série de funções para manipular strings.

Ela nos ajuda a algumas tarefas:

Copiar strings usando `strcpy`;

Concatenar strings usando `strcat`;

Descobrir o tamanho de uma string usando `strlen()`;

**Comparar** strings usando `strcmp()`;



# Biblioteca `<string.h>` - `strcmp`

Sintaxe:

```
variável do tipo inteiro = strcmp(string1, string2);
```

Compara o conteúdo de duas strings;

Possíveis valores de retorno:

0: conteúdo das strings são iguais

< 0: conteúdo da string1 é menor do que string2

> 0: conteúdo da string1 é maior do que string2

# Biblioteca `<string.h>` - `strcmp`

Sintaxe:

variável do tipo inteiro = `strcmp(string1, string2);`

Compara o conteúdo de duas strings;

Possíveis valores de retorno:

0: conteúdo das strings são iguais

9- Ler a profissão do usuário e o seu salário. Se for técnico, reajustar o salário em 50% .  
Se for estagiário, aumentar o salário em 30% , se não for nenhum dos cargos citados,  
calcular o salário com um aumento de 20%. Informar o salário resultante para o usuário.

> 0: conteúdo da string1 é maior do que string2

# Biblioteca `<string.h>` - `strcpy`

Sintaxe:

```
strcpy(string_destino, string_origem,  
tamanho);
```

Realiza a cópia do conteúdo de uma variável a outra.

Obs: Tanto a variável de destino como a de origem devem ser strings.

# Biblioteca `<string.h>` - `strcpy`

Sintaxe:

`strcpy(string_destino, string_origem, tamanho);`

Realiza a cópia do conteúdo de uma variável a outra.

Obs: Tanto a variável de destino como a de origem devem ser strings.

```
#include <stdio.h>
#include <string.h> //necessário para strcpy

int main (void)
{
    char nome[15];

    strcpy(nome, "Fulano de Tal");
    //strcpy(string_destino, string_origem);
    //note que a string de destino é nome
    //a string de origem é "Fulano de Tal"

    printf("Nome = %s", nome);
    return 0;
}
```

# Biblioteca `<string.h>` - `strncpy`

Sintaxe:

```
strncpy(string_destino,  
string_origem, tamanho);
```

Realiza a cópia do conteúdo de uma variável a outra, **porém, deve ser especificado o tamanho a ser copiado.**

Obs: Tanto a variável de destino como a de origem devem ser strings.

# Biblioteca `<string.h>` - `strncpy`

Sintaxe:

`strncpy(string_destino,  
string_origem, tamanho);`

Realiza a cópia do conteúdo de uma variável a outra, **porém, deve ser especificado o tamanho a ser copiado.**

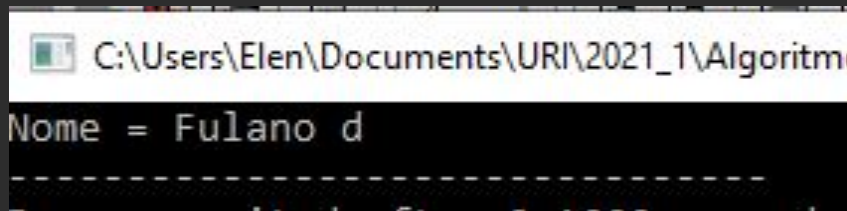
Obs: Tanto a variável de destino como a de origem devem ser strings.

```
#include <stdio.h>
#include <string.h> //necessário para strcpy

int main (void)
{
    char nome[15];

    strncpy(nome, "Fulano de Tal", 8);
    //strcpy(string_destino, string_origem);
    //note que a string de destino é nome
    //a string de origem é "Fulano de Tal"

    printf("Nome = %s", nome);
    return 0;
}
```



C:\Users\Elen\Documents\URI\2021\_1\Algoritmo

Nome = Fulano d

-----

# Biblioteca <string.h> - strcat

strcat

Sintaxe

```
strcat(string_destino, string_origem);
```

Realiza a concatenação do conteúdo de uma variável a outra.

Ambas devem ser strings.

# Biblioteca <string.h> - strcat

strcat

Sintaxe

strcat(string\_destino, string\_origem);

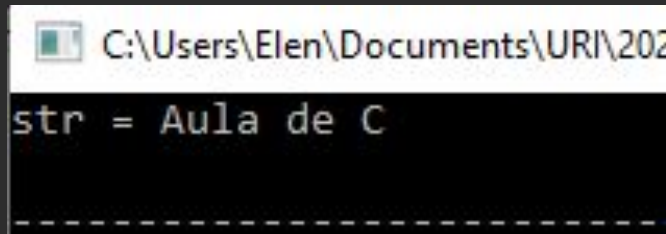
Realiza a concatenação do conteúdo de uma variável a outra.

Ambas devem ser strings.

```
#include <stdio.h>
#include <string.h> //necessário para strcat
int main (void)
{
    char str[10] = "Aula";
    strcat(str, " de C");

    printf("str = %s\n", str);
    //Será exibido curso de C

    return 0;
}
```



C:\Users\Elen\Documents\URI\202

str = Aula de C



# Biblioteca `<string.h>` - `strlen`

Sintaxe:

```
variável do tipo inteiro = strlen(string);
```

Determina o tamanho de uma string.

# Atividades - `<string.h>`

1 - Ler nome e sobrenome de uma pessoa e guardar cada um em uma variável. Declarar uma variável `nome_completo` que recebe a concatenação das duas variáveis. Mostre `nome_completo`.

2 - Complementar o exercício anterior, dizendo quantos caracteres o nome completo possui.

É importante  
dizer que C é  
case sensitive.

# Expressões reduzidas

Expressão	Forma compacta
$x = x + y$	$x += y$
$x = x - y$	$x -= y$
$x = x * y$	$x *= y$
$x = x / y$	$x /= y$
$x = x \% y$	$x \% = y$

Se uma expressão incrementa ou decrementa o valor da variável, podemos então escrevê-la numa forma ainda mais compacta.

ex:  $x = x + 1$

Podemos escrever como

**$x++$**

# Expressões reduzidas

Expressão	Forma compacta
$x = x + y$	$x += y$
$x = x - y$	$x -= y$
$x = x * y$	$x *= y$
$x = x / y$	$x /= y$
$x = x \% y$	$x \% = y$

Se uma expressão incrementa ou decrementa o valor da variável, podemos então escrevê-la numa forma ainda mais compacta.

ex:  $x = x + 1$

$x++$ ,  $x--$

Podemos escrever como

**$x++$**

# Laço de repetição for

*for( inicialização; condição; alteração)*

Ex: Mostrar os números de 10 até 100.

# Laço de repetição for

*for( inicialização; condição; alteração)*

Ex: Mostrar os números de 10 até 100.

```
int main() {  
    int i;  
  
    for (i=10; i<=100; i++){  
        printf("%d \n",i);  
    }  
    return 0;  
}
```

# Laço de repetição for

*for( inicialização; condição; alteração)*

Ex: Mostrar os números de 10 até 100.

```
int main() {  
    int i;  
  
    for (i=10; i<=100; i++){  
        printf("%d \n",i);  
    }  
    return 0;  
}
```

1- Dado um valor n, exiba uma contagem regressiva de n até zero.

2- Leia um número inteiro n e exiba seu fatorial.

3 - Exiba os números ímpares de 100 até 150.



# Laço de repetição while

```
while( condição ) comando;
```

Ex: Mostrar os números de 10 até 100.

# Laço de repetição while

*while( condição ) comando;*

Ex: Mostrar os números de 10 até 100.

```
#include<stdio.h>

int main(){

    int i;

    i=0;

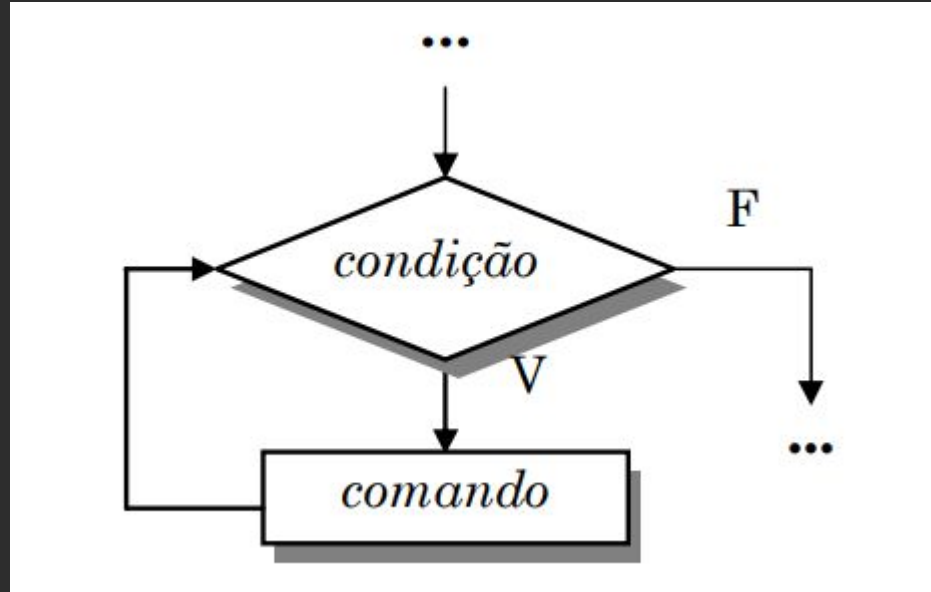
    while (i<=100){
        printf("%i\n",i);
        i++;
    }
}
```

1- Dado um valor n, exiba uma contagem regressiva de n até zero.

2- Leia um número inteiro n e exiba seu fatorial.

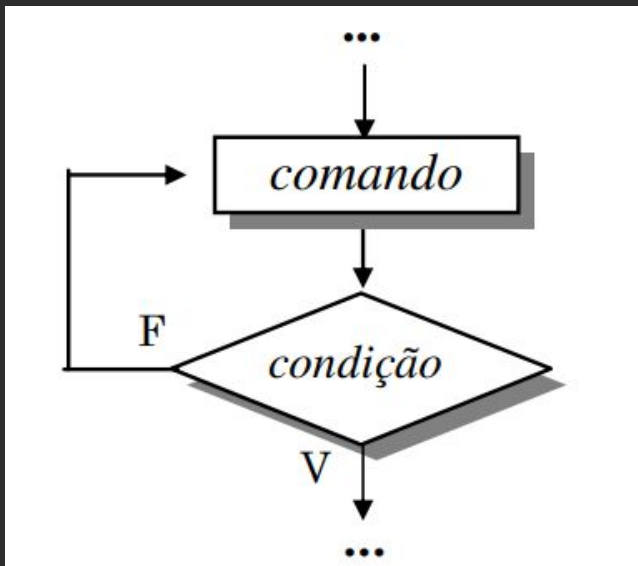
3 - Exiba os números ímpares de 100 até 150.

while e for são estruturas de pré-condição;



# Pós-condição

Podemos obrigar a execução ao menos uma vez, para depois testarmos...



Solicita um número positivo do usuário para calcular a raiz quadrada. Caso ele digite um número negativo, seguir pedindo.

Vamos fazer isso com **while** e com o **do-while**.

# Exercícios - Até o final da aula

Usando while

- 1 - Mostrar os números múltiplos de 5 de 0 a 100.
- 2 - Exibir a tabuada de um número informado pelo usuário.
- 3 – Diga se o número lido é maior ou menor que 100, até o usuário digitar 0.

Usando for

- 4 - Mostrar os números de 100 até 1.
- 5 - Ler um número  $n$  do usuário e mostrar os números de 0 a 100 que são múltiplos de  $n$ .
- 6 - Mostrar os números que estão entre  $n1$  e  $n2$ , sendo que  $n1$  e  $n2$  devem ser lidos do