

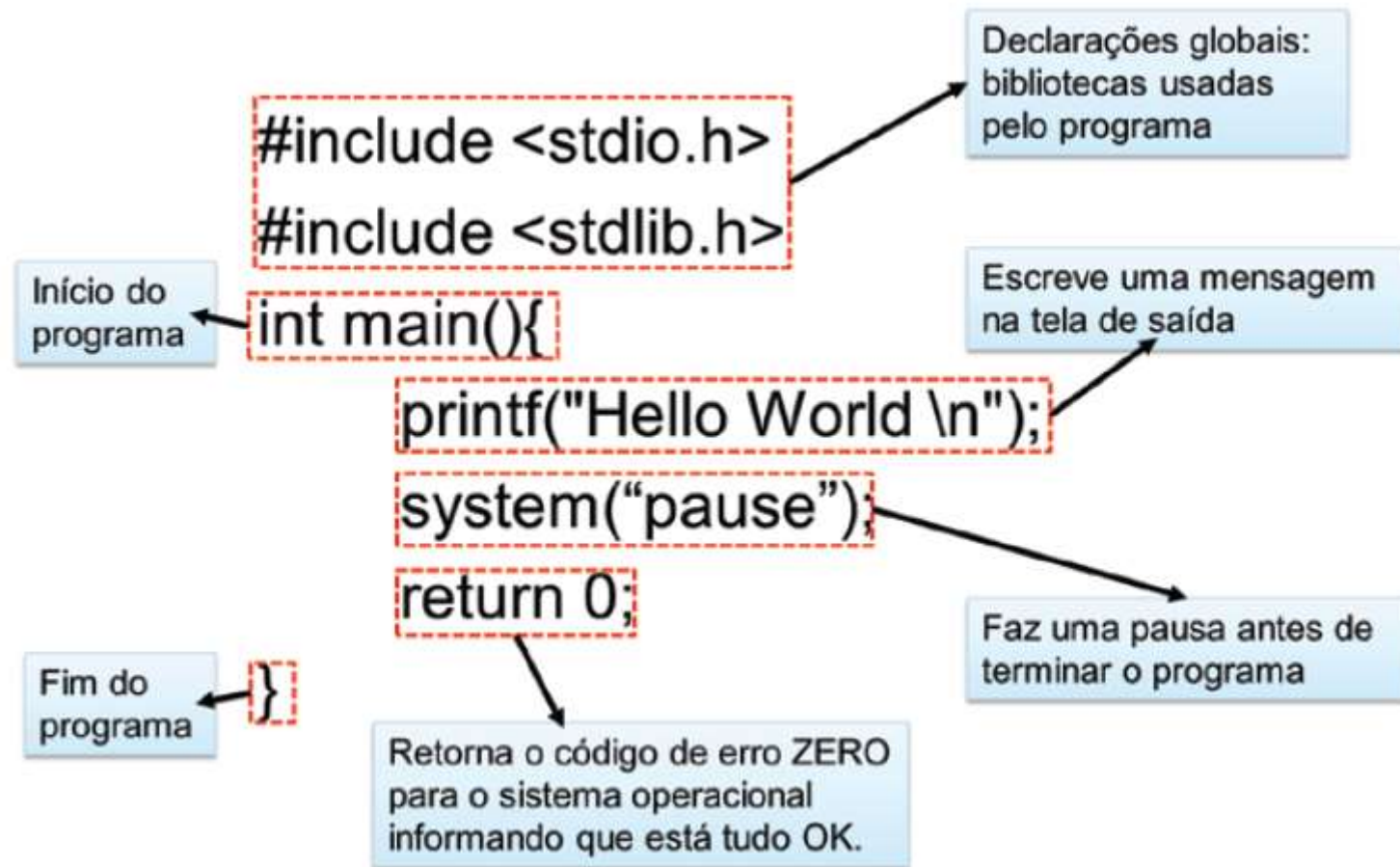
# Linguagem C

Revisão: tipos de dados, comandos de saída e entrada, operadores

Baseado em slides do Prof. Bruno Travençolo

# Primeiro programa em C

---



# Comentários

- ▶ Permitem adicionar uma descrição sobre o programa. São ignorados pelo compilador.

```
*main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      /* a função printf() serve para escrever na
7       tela */
8
9      printf("Hello world!\n");
10
11     // Faz uma pausa no programa (em Windows)
12     system("pause");
13
14     // Retorna 0 (zero) para o sistema operacional
15     return 0; // FIM DO PROGRAMA
16 }
17
```

Comentário com uma  
ou mais linhas

Início: /\*

Fim: \*/

Comentário em uma  
única linha

Início: //

Fim: sem símbolos, é  
o próprio final da linha

# Variáveis

---

- ▶ Variável em programação:
  - ▶ Posição de memória que armazena uma informação
  - ▶ Pode ser modificada pelo programa
  - ▶ Deve ser **definida** antes de ser usada
- ▶ Declaração de variáveis em C
  - ▶ <tipo de dado> nome-da-variável



# Variáveis

---

## ▶ Nome

- ▶ Deve iniciar com letras ou underscore(\_);
- ▶ Caracteres devem ser letras, números ou underscores;
- ▶ Palavras chaves não podem ser usadas como nomes;
- ▶ Letras maiúsculas e minúsculas são consideradas diferentes (*Case sensitive*)
- ▶ Não pode haver duas variáveis com o mesmo nome (no mesmo escopo)



# Tipos básicos em C

---

- ▶ **char**: um byte que armazena o código de um caractere do conjunto de caracteres local

- ▶ **caracteres sempre ficam entre ‘aspas simples’!**

```
char sexo; // pode receber 'M' ou 'F'
```

```
char UnidadeTemperatura; //pode receber 'C' para Celsius  
                        //ou 'F' para Fahrenheit
```

```
char opcoes; // pode ser '1', '2' , '3' ou '4'
```

- ▶ **int**: um inteiro cujo tamanho depende do processador, tipicamente **16 ou 32 bits**

```
int NumeroAlunos;
```

```
int Idade;
```

```
int NumeroContaCorrente;
```

```
int N = 10; // o variável N recebe o valor 10
```



# Tipos básicos em C

---

- ▶ **Números Reais**

- ▶ **float**: um número real com precisão simples

```
float Temperatura; // pode receber, por exemplo, 23.30
float MediaNotas; // pode receber, por exemplo, 7.98
float TempoTotal; // pode receber 0.0000000032 (s) ou
                  // 3.2000e-009 (notação científica)
                  // que equivale à  $3,2 \times 10^{-9}$ 
```

- ▶ **double**: um número real com precisão dupla

```
double DistanciaGalaxias; // número muito grande
double MassaMolecular; // em Kg, número muito pequeno
double BalancoEmpresa; // valores financeiros
```



# Variáveis

---

Tipo	Bits	Intervalo de valores
char	8	-128 A 127
unsigned char	8	0 A 255
signed char	8	-128 A 127
int	32	-2.147.483.648 A 2.147.483.647
unsigned int	32	0 A 4.294.967.295
signed int	32	-32.768 A 32.767
short int	16	-32.768 A 32.767
unsigned short int	16	0 A 65.535
signed short int	16	-32.768 A 32.767
long int	32	-2.147.483.648 A 2.147.483.647
unsigned long int	32	0 A 4.294.967.295
signed long int	32	-2.147.483.648 A 2.147.483.647
float	32	1,175494E-038 A 3,402823E+038
double	64	2,225074E-308 A 1,797693E+308
long double	96	3,4E-4932 A 3,4E+4932





# Constantes

---

- ▶ Como uma variável, uma constante também armazena um valor na memória do computador.
- ▶ Entretanto, esse valor não pode ser alterado: é constante.
- ▶ Para constantes é obrigatória a atribuição do valor.



# Constantes

---

- ▶ Usando **#define**

- ▶ Você deverá incluir a diretiva de pré-processador **#define** antes de início do código:

```
#define PI 3.1415
```

- ▶ Usando **const**

- ▶ Usando **const**, a declaração não precisa estar no início do código.

```
const double pi = 3.1415;
```



# Constantes char

---

- ▶ A linguagem C utiliza vários códigos chamados códigos de barra invertida.

Código	Comando
\a	som de alerta (bip)
\b	retrocesso (backspace)
\n	nova linha (new line)
\r	retorno de carro (carriage <b>return</b> )
\v	tabulação vertical
\t	tabulação horizontal
\'	apóstrofe
\"	aspa
\\	barra invertida (backslash)
\f	alimentação de folha (form feed)
\?	símbolo de interrogação
\0	caractere nulo (cancela a escrita do restante)



# Tipos Booleanos em C

---

- ▶ Um tipo booleano pode assumir dois valores:
  - ▶ verdadeiro ou falso (true ou false)
- ▶ Na linguagem C não existe o tipo de dado booleano.
- ▶ Para armazenar esse tipo de informação, use-se uma variável do tipo `int` (número inteiro)
  - ▶ Valor 0 significa falso / números + ou - : verdadeiro
- ▶ Exemplos:

```
int AssentoOcupado = 1; // verdadeiro (o assento está
                        // ocupado)
int PortaAberta = 0; // falso (a porta está fechada)
```



# Atribuição

---

- ▶ **Operador de Atribuição: =**
  - ▶ nome\_da\_variável = expressão, valor ou constante;



O operador de atribuição "=" armazena o valor ou resultado de uma expressão contida à sua **direita** na variável especificada à sua **esquerda**.

- ▶ **Ex.:**

```
int main( ){  
    int x = 5; /* em pseudolinguagem  
               representamos assim: x <- 5 */  
    int y;  
    y = x + 3;  
}
```

- ▶ A linguagem C suporta múltiplas atribuições
  - ▶ x = y = z = 0;



# Conversões de Tipos na Atribuição

---

## ▶ Atribuição entre tipos diferentes

- ▶ O compilador converte automaticamente o valor do lado direito para o tipo do lado esquerdo de “=”
- ▶ Pode haver perda de informação
- ▶ Ex:

```
int x; char ch; float f;
```

```
ch = x; /* ch recebe 8 bits menos significativos de x */
```

```
x = f; /* x recebe parte inteira de f */
```

```
f = ch; /* f recebe valor 8 bits convertido para real */
```

```
f = x; /* f recebe valor 16 bits convertido para real */
```



# Modeladores (Casts)

---

- ▶ Um modelador é aplicado a uma expressão.
- ▶ Força o resultado da expressão a ser de um tipo especificado.
  - ▶ (tipo) expressão
- ▶ Ex:
  - ▶ (float) x;
  - ▶ (int) x \* 5.25;



# Modeladores (Casts)

---

► **Ex:**

```
int num;  
float f;  
num = 10;  
f = num/7;  
printf ("%f \n", f);  
f = (float)num/7;  
printf ("%f", f);
```

► **Resultado**

**1.000000**

**1.428571**





# Comando de saída

---

## ▶ **printf()**

- ▶ Mostra o valor contido em uma variável
- ▶ Utilizar especificadores de formato (*format specifiers*)
  - ▶ Subsequências iniciando com o símbolo de porcentagem: %
- ▶ Um símbolo é usado para cada tipo de variável a ser mostrada
  - ▶ Por ex., a letra **f** é usada para mostrar valores do tipo **float**

```
float TemperaturaAtual = 34.5;  
printf("A temperatura atual é: ");  
printf("%f", TemperaturaAtual);  
printf(" graus Celsius");
```

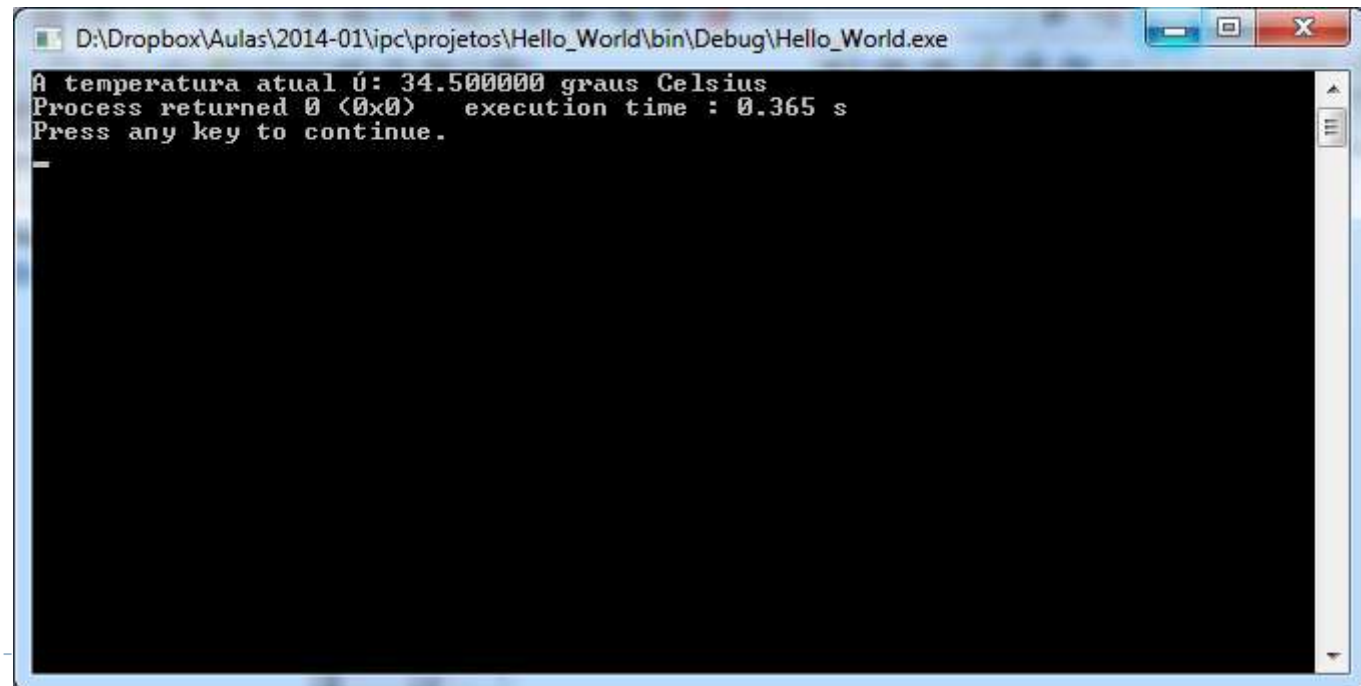


# Comando de saída

---

## ► printf()

```
float TemperaturaAtual = 34.5;  
printf("A temperatura atual é: ");  
printf("%f", TemperaturaAtual);  
printf(" graus Celsius");
```

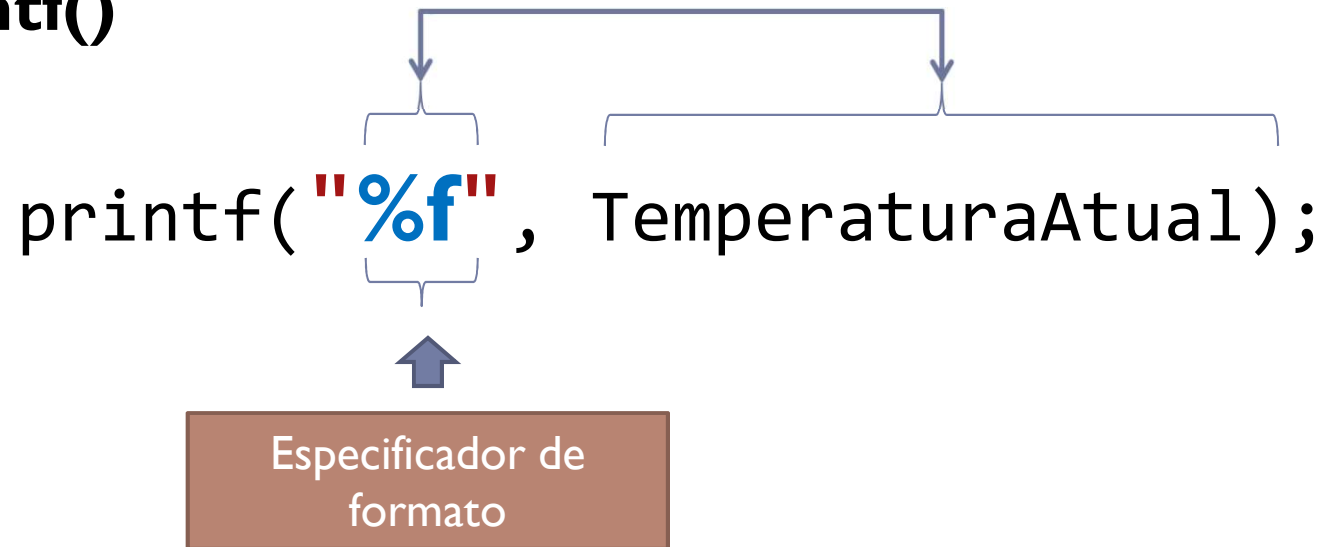


A screenshot of a Windows command prompt window. The title bar shows the file path: D:\Dropbox\Aulas\2014-01\ipc\projetos\Hello\_World\bin\Debug\Hello\_World.exe. The window contains the following text: "A temperatura atual é: 34.500000 graus Celsius", "Process returned 0 (0x0) execution time : 0.365 s", and "Press any key to continue.". The text is displayed in a monospaced font on a black background.

# Comando de saída

---

## ► printf()



Alguns tipos de saída

`%c` – escrita de **um** caractere

`%d` – escrita de números inteiros

`%f` – escrita de número reais

`%s` – escrita de **vários** caracteres



# Comando de saída

---

## ▶ **printf()**

- ▶ Podemos mostrar mais de uma variável
  - ▶ Sintaxe: **printf**("format", **arg1**, ...)
  - ▶ A sintaxe diz que podemos ter vários argumentos (arg1, arg2, ...)

// cálculo do IMC (índice de massa corporal):  $\text{Peso(kg)} / (\text{Altura}^2)$

```
float peso = 82.5;
```

```
float altura = 1.70;
```

```
float IMC;
```

```
IMC = peso / (altura*altura);
```

```
printf("Peso: %f, Altura: %f, IMC: %f", peso, altura, IMC);
```

Hand-drawn arrows in blue ink showing the mapping of variables to format specifiers in the printf statement. An arrow points from 'peso' to '%f' in 'Peso: %f'. Another arrow points from 'altura' to '%f' in 'Altura: %f'. A third arrow points from 'IMC' to '%f' in 'IMC: %f'. There is also a long curved arrow from the first '%f' to the second '%f'.

# Conversão de tipos

---

```
int Inteiro;
```

```
float Real;
```

```
Real = 1/3;
```

```
printf("%f \n",Real); // resposta 0.00000
```

```
Real = 1/3.0;
```

```
printf("%f \n",Real); // resposta 0.33333
```

```
Inteiro = 3;
```

```
Real = 1/Inteiro;
```

```
printf("%f \n",Real); // resposta 0.00000
```

```
Real = 1/(float)Inteiro;
```

```
printf("%f \n",Real); // resposta 0.33333
```

```
Real = 2.9;
```

```
Inteiro = Real;
```

```
printf("%d \n",Inteiro); // resposta 2
```



# Comando de entrada

---

- ▶ Em C, o comando que permite ler dados da entrada padrão (no caso o teclado) é o `scanf()`
- ▶ **`scanf()`**
- ▶ Sintaxe: **`scanf("format",&name1,...)`**
  - ▶ `format` – especificador de formato da entrada que será lida
  - ▶ `&name1, &name2, ...` – endereços das variáveis que irão receber os valores lidos



# Comando de entrada

---

- ▶ Temos, igual ao comando printf, que especificar o tipo (formato) do dado que será lido
- ▶ scanf(“tipo de entrada”, lista de variáveis)



- ▶ Alguns “tipos de entrada”
  - ▶ %c – leitura de **um** caractere
  - ▶ %d – leitura de números inteiros
  - ▶ %f – leitura de float
  - ▶ %s – leitura de **vários** caracteres



# Comando scanf() - Exemplo

---

```
scanf("%f",&peso); // leia um valor real (do tipo float) e armazene no endereço da variável peso
```

- ▶ O símbolo & indica qual é o endereço da variável que vai receber os dados lidos
  - ▶ peso – variável peso
  - ▶ &peso – endereço da variável peso





# Comando scanf() - Exemplo

---

```
// declaração das variáveis
```

```
float peso;
```

```
float altura;
```

```
float IMC;
```

```
// Obtendo os dados do usuário
```

```
printf("Informe o peso: ");
```

```
scanf("%f",&peso);
```

```
printf("Informe a altura: ");
```

```
scanf("%f",&altura);
```

```
// calculando o ICM e mostrando o resultado
```

```
IMC = peso / (altura*altura);
```

```
printf("Peso: %f, Altura: %f, IMC: %f", peso, altura, IMC);
```



# Comando de entrada

---

- ▶ **Ex:**

- ▶ Leitura de um único valor

```
int x;
```

```
scanf("%d",&x);
```

- ▶ Podemos ler mais de um valor em um único comando

```
int x,y;
```

```
scanf("%d%d",&x,&y);
```

- ▶ Obs: na leitura de vários valores, separar com espaço, TAB, ou Enter.



## Limpendo o buffer

---

- ▶ Antes de usar um scanf com “%c” faça
- ▶ **setbuf(stdin, NULL);**
- ▶ Esse comando limpa o buffer de entrada

```
char letra;  
setbuf(stdin, NULL);  
scanf("%c", &letra);
```



# Comando de entrada

---

## ► **getchar()**

- Comando que realiza a leitura de um único caractere

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      char c;
05      c = getchar();
06      printf("Caractere: %c\n", c);
07      printf("Codigo ASCII: %d\n", c);
08      system("pause");
09      return 0;
10  }
```



# Operadores

---

## ► Operadores Unários

Op	Uso	Exemplo
+	mais unário ou positivo	+X
-	menos unário (número oposto)	-X
!	NOT ou negação lógica	!x
&	Endereço	&x

Existem outros operadores que serão vistos em outras aulas



# Operador Unário

---

```
int num;  
int oposto_num;  
num = -10;  
oposto_num = -num;  
printf("Num. %d => Valor oposto %d", num, oposto_num);
```

> Saída: Num. -10 => Valor oposto 10



# Operador Unário

---

```
int ligado;  
ligado = 1; // valor booleano (verdadeiro)  
  
printf("Ligado: %d; Desligado %d", ligado, !ligado);
```

> Saída: Ligado: 1; Desligado 0



# Operadores

---

## ► Binários

Operador	Descrição	Exemplo
+	Adição de dois números	$z = x + y;$
-	Subtração de dois números	$z = x - y;$
*	Multiplicação de dois números	$z = x * y;$
/	Quociente de dois números	$z = x / y;$
%	Resto da divisão	$z = x \% y;$





# Operadores Relacionais e Lógicos

---

- ▶ Operadores relacionais: comparação entre variáveis
- ▶ Esse tipo de operador retorna *verdadeiro* (1) ou *falso* (0)

Op	Descrição	Exemplo
>	Maior do que	Idade > 6
>=	Maior ou igual a	Nota >= 60
<	Menor do que	Valor < Temperatura
<=	Menor ou igual a	Velocidade <= MAXIMO
==	Igual a	Opcao == 'a'
!=	Diferente de	Opcao != 's'



# Operadores Relacionais e Lógicos

---

## ▶ Exemplos

Expressão	Resultado
▶ <code>x=5; x &gt; 4</code>	verdadeiro (1)
▶ <code>x=5; x == 4</code>	falso (0)
▶ <code>x=5; y=3; x != y</code>	verdadeiro (1)
▶ <code>x=5; y=3; x != (y+2)</code>	falso (0)



# Operadores Relacionais e Lógicos

---

- ▶ Operadores lógicos: operam com valores lógicos e retornam um valor lógico *verdadeiro* (1) ou *falso* (0)

Op	Função	Exemplo
&&	AND (E)	(c >= '0' && c <= '9')
	OR (OU)	(a == 'F'    b != 32)
!	NOT	!continuar



# Importante

---

- ▶ Símbolo de atribuição **=** é diferente, muito diferente, do operador relacional de igualdade **==**

```
int Nota;
```

```
Nota = 50; // Nota recebe 50
```

```
// Erro comum em C:
```

```
// Teste se a nota é 60
```

```
// Sempre entra na condição
```

```
if (Nota = 60) {  
    printf("Você passou raspando!!");  
}
```

```
// Versão Correta
```

```
if (Nota == 60) {  
    printf("Você passou raspando!!");  
}
```



# Expressões

---

- ▶ Expressões são combinações de variáveis, constantes e operadores.

- ▶ Exemplos:

`Anos = Dias/365.25;`

`i = i+3;`

`c= a*b + d/e;`

`c= a*(b+d)/e;`



# Operadores

---

## ► Operadores Unários

- : menos unário ou negação	-x
! : NOT ou negação lógica	!x
&: endereço	&x
*: conteúdo (ponteiros)	(*x)
++: pré ou pós incremento	++x ou x++
-- : pré ou pós decremento	-- x ou x --



# Operadores

---

- ▶ **Ex:**

```
int x,y;  
x = 10;  
y = x++;  
printf("%d \n",x);  
printf("%d \n",y);  
y = ++x;  
printf("%d \n",x);  
printf("%d \n",y);
```

- ▶ **Resultado**

```
11  
10  
12  
12
```

