


# Introdução ao C



# **Estrutura Básica de um programa**

```
int main() {  
    return 0;  
}
```

Nosso código  
vai entre essas  
linhas



```
int main() {  
    return 0;  
}
```

Editor de  
Texto

```
int main() {  
    return 0;  
}
```



Salvar

qualquer\_nome.c  
meu\_codigo.c  
ola\_v.c

Escolha **qualquer  
nome**, mas  
termine com a  
extensão  
**.c**

**E o que mais  
pode ter neste  
código?**

# Variáveis

- Guarda um valor que varia
- $X = 1$ ,  $X = 3$ ,  $X = 5$
- Não é constante
- Usada para guardar dados de entrada
- Usada para te auxiliar no seu algoritmo

X

# Declarando Variáveis

```
int main() {
```

Modificadores

Tipo

Nome

=

Valor Inicial

```
return 0;
```

```
}
```



# Declarando Variáveis

```
int main() {
```

Modificadores

Tipo

Nome

=

Valor Inicial



Opcionais

```
return 0;
```

```
}
```

# Tipos Básicos

# Tipos Básicos

`int`      Números inteiros

`float`    Racionais (6 - 9 casas)

`double`   Racionais com dupla precisão (~15)

`char`     Letras

# Tipos Básicos

int 42

float 2.56f

double 3.14159265359

char 'C'

# Tipos Básicos

1 byte = 8 bits

int 4 bytes

float 4 bytes

double 8 bytes

char 1 byte

# Tipos Básicos

1 byte = 8 bits

int	4 bytes	32 bits
-----	---------	---------

float	4 bytes	32 bits
-------	---------	---------

double	8 bytes	64 bits
--------	---------	---------

char	1 byte	8 bits
------	--------	--------

# Tipos Básicos

`int`            -32768 até 32767

`float`          -3.4E+38 até +3.4E+38

`double`        -1.7E+308 até +1.7E+308

`char`           [Tabela ASCII] OU [-128 até 127]

# Tabela ASCII

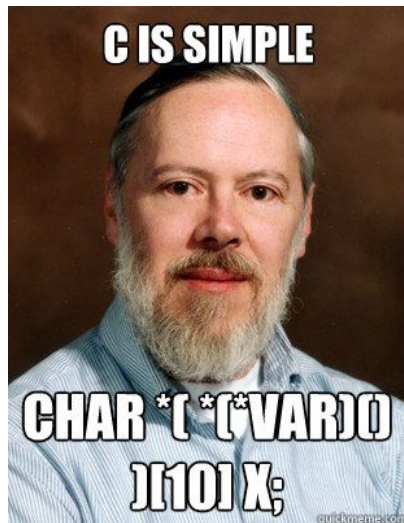
Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	\$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]



**A verdade sobre  
o char**

# Char

- Assume valores numéricos **igual a todos os outros** tipos
- Seus valores numéricos tem uma **representação especial**
  - Seguem a tabela ASCII
  - 65 == A
  - 66 == B
- É possível fazer **operações aritméticas** com eles
  - 'a' + 1 = 'b'
  - 'A' - 'A' = 0
- No fundo são **apenas números**



# Declarando Variáveis

```
int main() {
```

Modificadores

Tipo

Nome

=

Valor Inicial

```
return 0;
```

```
}
```

# Declarando Variáveis

```
int main() {
```

Modificadores

int

Nome

=

Valor Inicial

```
return 0;
```

```
}
```

**Nome de  
variáveis**

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

X

minha\_VaR

varX

varX1

abc42Y

xyz

9xyz

\_xyz

xyz&

xyz#

xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único



X

minha\_VaR

varX

varX1

abc42Y

xyz

9xyz

\_xyz

xyz&

xyz#

xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único



X



minha\_VaR

varX

varX1

abc42Y

xyz

9xyz

\_xyz

xyz&

xyz#

xyz2



# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
varX1  
abc42Y  
xyz  
9xyz  
\_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
abc42Y  
xyz  
9xyz  
\_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
✓ abc42Y  
xyz  
9xyz  
\_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
✓ abc42Y  
✓ xyz  
9xyz  
\_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
✓ abc42Y  
✓ xyz  
✗ 9xyz  
\_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
✓ abc42Y  
✓ xyz  
✗ 9xyz  
✓ \_xyz  
xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓ X  
✓ minha\_VaR  
✓ varX  
✓ varX1  
✓ abc42Y  
✓ xyz  
✗ 9xyz  
✓ \_xyz  
✗ xyz&  
xyz#  
xyz2

# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

✓	X
✓	minha_VaR
✓	varX
✓	varX1
✓	abc42Y
✓	xyz
✗	9xyz
✓	_xyz
✗	xyz&
✗	xyz#
	xyz2



# Nome da Variavel

- Permitido: Letras, números, \_
- Não pode: acentos, outros símbolos
- Não pode: iniciar com números, só com \_ ou letras
- Nome único

- ✓ X
- ✓ minha\_VaR
- ✓ varX
- ✓ varX1
- ✓ abc42Y
- ✓ xyz
- ✗ 9xyz
- ✓ \_xyz
- ✗ xyz&
- ✗ xyz#
- ✓ xyz2

# Declarando Variáveis

```
int main() {
```

Modificadores

**int**

Nome

=

Valor Inicial

```
return 0;
```

```
}
```

# Declarando Variáveis

```
int main() {
```

Modificadores

```
int nota =
```

Valor Inicial

```
return 0;
```

```
}
```

**0 valor é  
simples**

## Declarando Variáveis

```
int main() {  
    int nota = 10;  
    return 0;  
}
```

## Declarando Variáveis

```
int main() {  
    float x = 3.14f;  
    return 0;  
}
```

# Declarando Variáveis

```
int main() {  
  
    float x = 3.14f;  
  
    return 0;  
  
}
```



Sempre deve haver o **f**  
após um número do tipo  
float

## Declarando Variáveis

```
int main() {  
    double x = 3.14;  
    return 0;  
}
```



## Declarando Variáveis

```
int main() {  
    char x = 'c';  
    return 0;  
}
```

# Declarando Variáveis

```
int main() {  
  
    char x = 'c';  
  
    return 0;  
  
}
```

**Sempre** deve haver  
**aspas simples**  
envolvendo o valor do  
**char**

Caso contrário, será  
interpretado como  
número

# Declarando Variáveis

```
int main() {  
    char x = '2';  
    return 0;  
}
```

```
int main() {  
    char x = 2;  
    return 0;  
}
```

Ambos são possíveis e **estão corretos**. Porém **não** representam o **mesmo valor**

**Colocando para  
funcionar**

# Compilação

```
gcc -o prog -W -Wall -Wshadow -pedantic exercicio.c
```

# Compilação

Compilador



**gcc** -o prog -W -Wall -Wshadow -pedantic exercicio.c

# Compilação

Compilador

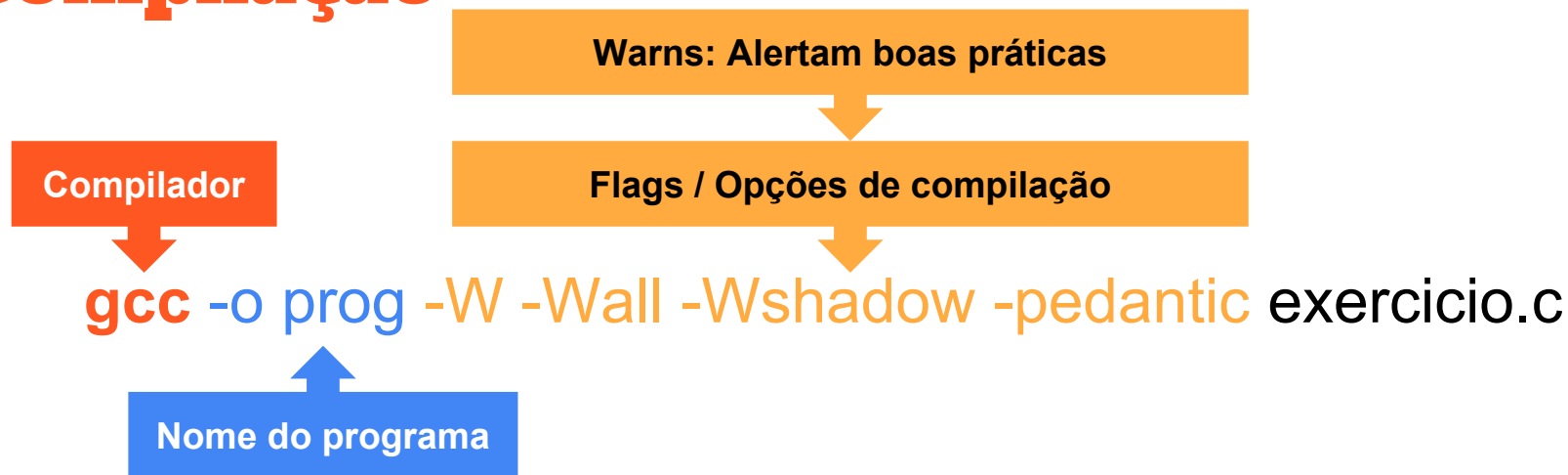


**gcc** -o prog -W -Wall -Wshadow -pedantic exercicio.c



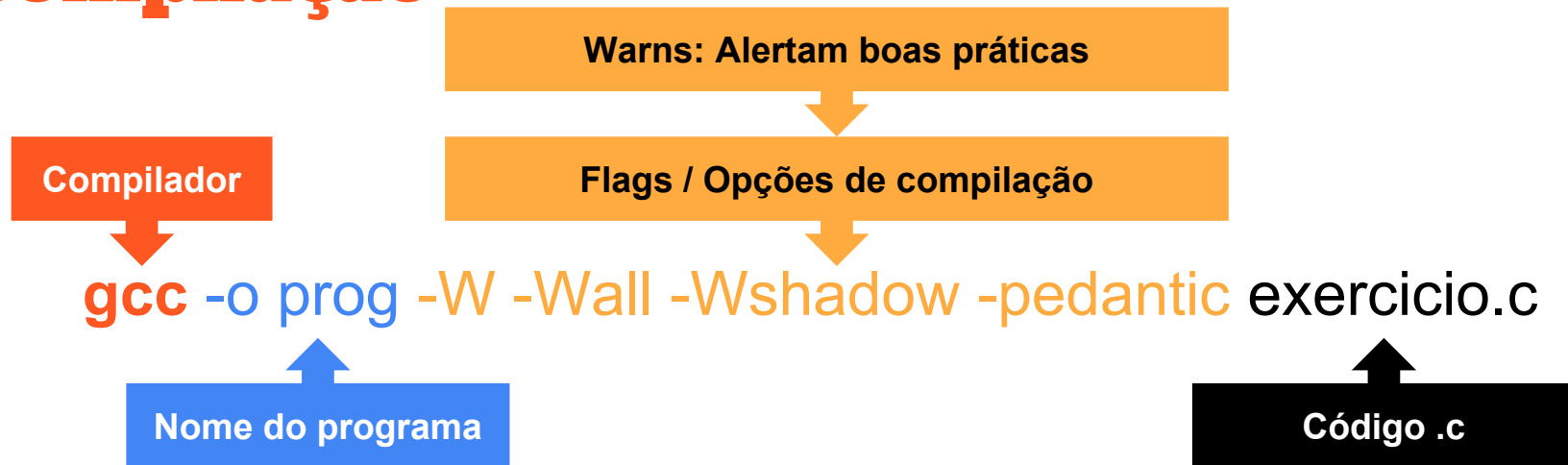
Nome do programa

# Compilação





# Compilação



# O que pode mudar?

Warns: Alertam boas práticas



Flags / Opções de compilação



Compilador



**gcc** -o my\_abc -W -Wall -Wshadow -pedantic exercicio.c

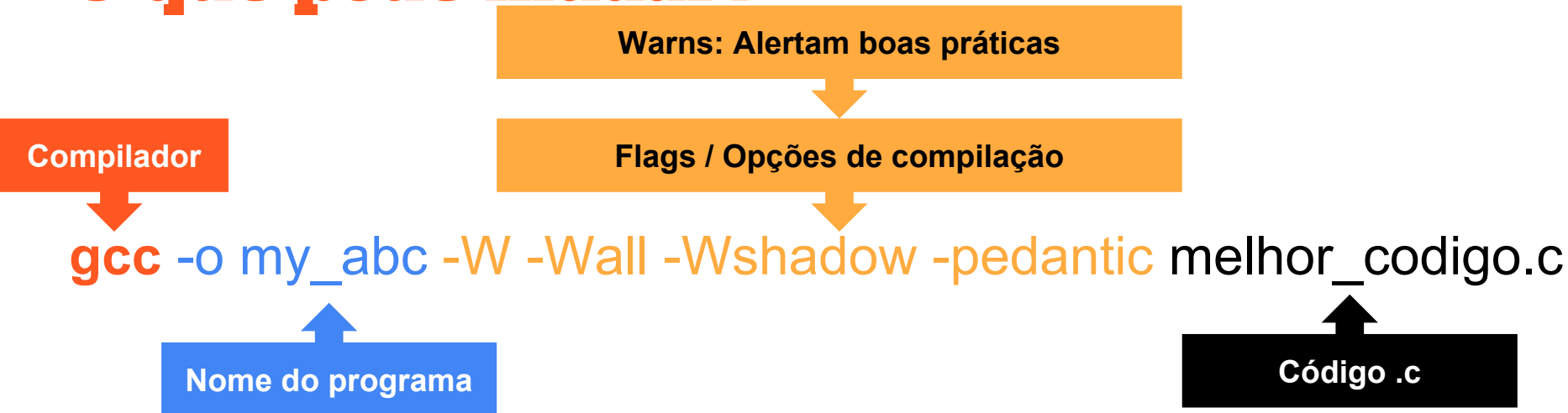
Nome do programa



Código .c




# O que pode mudar?



# O que pode mudar?

```
gcc -o my_abc -W -Wall -Wshadow -pedantic melhor_codigo.c
```



Nome do  
programa

Código .c

**Testem: crie um  
codigo com 3 variáveis.  
E compilem.**

```
gcc -o prog -W -Wall -Wshadow -pedantic exercicio.c
```



**Testem: crie um  
codigo com 3 variáveis.  
E compilem.**

# Declarando Variáveis

```
int main() {
```

Modificadores

```
int nota = 10;
```

```
return 0;
```

```
}
```

# Modificadores de Tamanho

int            long, long long

float

double

char

Apenas o int possui  
modificadores de  
tamanho



# Modificadores de Tamanho

`int` -32767 to 32767

`long int` -2147483647 to 2147483647

`long long int` -9223372036854775807 to 9223372036854775807

# Modificadores de Sinal

`int`      signed, unsigned

`float`

`double`

`char`      signed, unsigned

O float e o double não  
possuem modificador de  
sinal

# Quanto cabe?

char

-127 to 127

unsigned char

0 to 255

# Quanto cabe?

Se o modificador de sinal for omitido, o padrão é o modificador signed

signed char

-127 to 127

unsigned char

0 to 255

# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

0 0 0 0 0 0 0 0

# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

1 1 1 1 1 1 1 1

# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

1 1 1 1 1 1 1 1

255

# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

Bit de sinal  
1 = negativo  
0 = positivo

1 1 1 1 1 1 1 1



# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

Bit de sinal  
1 = negativo  
0 = positivo

1 1 1 1 1 1 1 1

# Quanto cabe?

1 byte = 8 bits

signed char

-127 to 127

unsigned char

0 to 255

Bit de sinal  
1 = negativo  
0 = positivo

1 1 1 1 1 1 1 1

127

# Quanto cabe?

long long int

-9223372036854775807 to 9223372036854775807

unsigned long long int

0 to 18446744073709551615

# Posso misturar modificadores? Sim!

unsigned long long int

signed long int

unsigned char

signed char

# Declarando Variáveis

```
int main() {
```

Modificadores

```
int nota = 10;
```

```
return 0;
```

```
}
```

## Declarando Variáveis

```
int main() {  
    unsigned int nota = 10;  
    return 0;  
}
```

**Testem: crie um  
codigo com 3 variáveis  
com modificadores. E  
compilem.**

# Recapitulando

<div>int</div> <div>float</div> <div>double</div> <div>char</div>	<div>long, long long</div>	<div>unsigned, signed</div> <div>unsigned, signed</div>	<div>42</div> <div>3.4f</div> <div>5.1231</div> <div>'A'</div>
Tipo	Tamanho		Sinal
Modificadores			
Exemplo			



**O que dá pra  
fazer com as  
variáveis?**

# Operadores Aritiméticos

+ Soma

- Subtração

\* Multiplicação

/ Divisão

% Módulo (Resto da Divisão)

$$x = 1 + 2$$

$$x = 4 - 2$$

$$x = 11 * 3$$

$$x = 44 / 2$$

$$x = 15 \% 4$$

`x = 1 + 2`

`x = 4 - 2`

`x = 11 * 3`

`x = 44 / 2`

`x = 15 % 4`

`int`

`float`

`double`

`char`

$x = 'A' + 2$

$x = 'b' - 'a'$

$x = '.' * '!'$

$x = '2' / '3'$

$x = 'b' \% '#'$

int

float

double

char

Os caracteres serão interpretados como o número que eles ocupam na tabela ASCII

$x = 1 + 2$

$x = 4 - 2$

$x = 11 * 3$

$x = 44 / 2$

$x = 15 \% 4$

int

float

double

char

**Divisão inteira:** Ignora a parte fracionária. Não é a mesma coisa que arredondamento

$x = 1 + 2$

$x = 4 - 2$

$x = 11 * 3$

$x = 44.0 / 2.0$

$x = 15 \% 4$

int

float

double

char

**Divisão comum**

$x = 1 + 2$

$x = 4 - 2$

$x = 11 * 3$

$x = 44 / 2$

$x = 15 \% 4$

int

float

double

char

A operação de **módulo (%)** só funciona com tipos que **não possuem** parte fracionária.  
(Tipos Integrais)



# Operadores Aritiméticos

**++** Incremento ( $x = x + 1$ )

**--** Decremento ( $x = x - 1$ )

x++

x--

```
int x = 50;  
x++;
```

int  
float  
double  
char

```
int x = 29;  
x--;
```

x++

x--

int  
float  
double  
char

```
int x = 50;  
x++;  
// 51
```

```
int x = 29;  
x--;  
// 28
```

# Fim++

