



# Variáveis

# Operadores aritméticos



## INF110 – Programação I

Prof. Alcione/André Gustavo  
DPI/UFV – 2020/1



# + Na aula passada...



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     cout << "Hello World!" << endl;
7     return 0;
8 }
```

Cabeçalho

Corpo do programa



## Na aula passada...

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     int prova1, prova2, prova3, notafinal;
7
8     //Ler nota da prova 1, prova 2, prova 3
9     cout << "Digite sua nota na prova 1: ";
10    cin >> prova1;
11    cout << "Digite sua nota na prova 2: ";
12    cin >> prova2;
13    cout << "Digite sua nota na prova 3: ";
14    cin >> prova3;
15
16    //Calcular a nota final: somar as 3 notas
17    notafinal = prova1 + prova2 + prova3 ;
18
19    //Escrever a nota final
20    cout << "Sua nota final foi ";
21    cout << notafinal << endl ;
22
23    return 0;
24 }
```

Declaração  
de variáveis

Entrada

Processamento

Saída

# + Declaração de variáveis

## ■ Sintaxe

- `[modificador] tipo identificador;`

# + Identificadores



- Um identificador em C++ só pode conter letras, dígitos e \_
  - E não pode começar com dígito!
  - Nem ter espaço em branco
  - Assim, `3D`, `2Teste` e `Nota final` não são válidos
- Letras somente a-z e A-Z; não podem ser usadas letras acentuadas, cedilhas, ...
  - Assim, `média` e `endereço` não são válidos
- C++ é *case sensitive*, diferencia maiúsculas e minúsculas
  - Desta forma, as variáveis `Soma`, `soma` e `SOMA` podem coexistir e são variáveis diferentes

# + Identificadores



- Palavras-chave não podem ser usadas como identificadores
  - auto, static, register, signed, unsigned, long, short
  - void, int, float, double, char,
  - if, else, while, do, for, return, ...
- Dicas
  - Escolher identificadores significativos (legibilidade)
  - Evite abreviações (legibilidade)
  - Evite identificadores que comecem com sublinhado
  - Evite identificadores com mais de 31 caracteres (portabilidade)

# + Tipos



- A linguagem C++ tem 4 tipos básicos
  - int
  - float
  - double
  - char
  - bool
- O programador pode ainda criar outros tipos com `typedef`
  - (veremos futuramente)

# + int

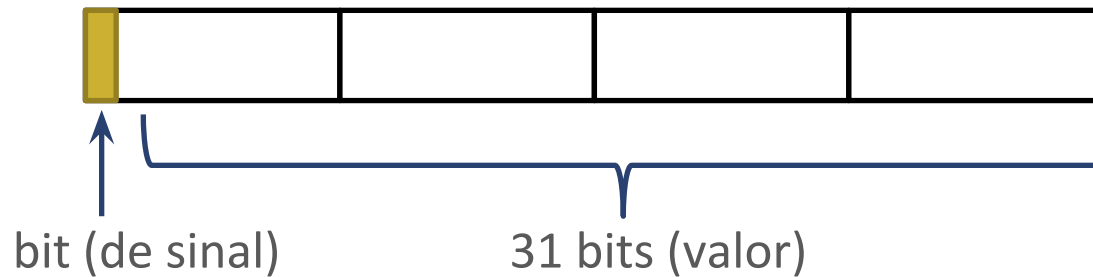


- Números inteiros
- Os limites dependem do tamanho (número de bytes), e o tamanho depende do processador e compilador
- Valores são lidos e escritos na base decimal
- Constantes podem ser atribuídas nas bases decimal, octal e hexadecimal

# + int



- Geralmente 4 bytes
- O primeiro bit é usado para sinal: 1 negativo, 0 positivo



- $-2^{31}$  a  $2^{31}-1$

# + float e double



- Números reais
  - Também chamados de números de ponto flutuante
- `float`: precisão simples
- `double`: precisão dupla
- A parte decimal deve ser separada por ponto, e não vírgula!
- Pode-se usar notação científica

# + float e double

- Padrão IEEE 754

- **float** (4 bytes)



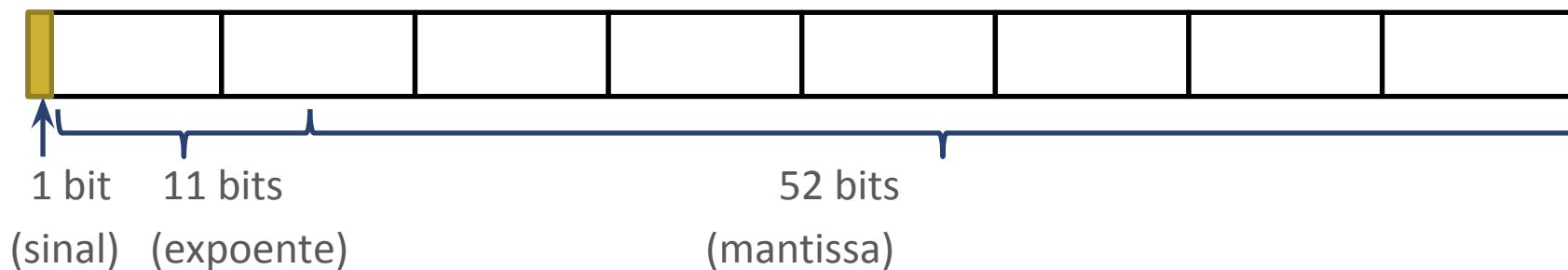
- Aprox.  $-1.2 \times 10^{-38}$  a  $3.4 \times 10^{38}$
- Precisão de 6 casas decimais

# + float e double



- Padrão IEEE 754

- **double** (8 bytes)



- Aprox.  $2.2 \times 10^{-308}$  a  $1.7 \times 10^{308}$
- Precisão de 15 casas decimais

## + char



- Um byte
- Normalmente um código numérico de um caractere da tabela ASCII

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☹	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(	072	H	104	h
009	(tab)	HT	041	)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♪	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▲	DLE	048	0	080	P	112	p
017	▼	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019	!!	DC3	051	3	083	S	115	s
020	π	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↑	CAN	056	8	088	X	120	x
025	↓	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[	123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093	]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	□

# + char



- Um byte
- Normalmente um código numérico de um caractere da tabela ASCII
- Pode ser usado para guardar inteiros pequenos
- Constantes caracteres usam aspas simples!
  - `char resposta = 's';`
- Podem ser usados inteiros, mas fica mais difícil compreender
  - `char resposta = 115;`

## + bool

- true/false

- Obs.: em C não existia tipo bool; era usado tipo inteiro, com 0 para falso e 1 (ou qualquer coisa diferente de 0) para verdadeiro.

# + Modificadores

- signed
- unsigned
- long
- short





# Tipos inteiros

Tipo	Intervalo	Memória (bytes)*
int	-2.147.483.648 ... 2.147.483.647	4
unsigned int	0 ... 4.294.967.295	4
short int	-32.768 ... 32.767	2
char	-128 ... 127	1
unsigned char	0 ... 255	1
long int	-2.147.483.648 ... 2.147.483.647	4
long long int	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807	8
unsigned long long int	0 ... 18,446,744,073,709,551,615	8

\*valores típicos, depende do processador, sistema operacional, compilador, ...

# + Operadores aritméticos

■ + - \* /

■ Soma, subtração, multiplicação e divisão

# + Operadores aritméticos

- % (mod)
- Retorna o resto da divisão
  - Só para operandos inteiros!
- Exemplos
  - $5 \% 2 \Rightarrow 1$
  - $4 \% 2 \Rightarrow 0$
  - $17 \% 3 \Rightarrow 2$
  - $6 \% 9 \Rightarrow 6$
  - $7 \% 0 \Rightarrow \text{ERRO}$

# + Operadores – precedência

- Da maior para a menor precedência

Operador	Significado	Ordem
( )	Parênteses	De dentro pra fora, da esquerda para direita
* / %	Multiplicação, divisão, resto	Da esquerda para direita
+ -	Adição, Subtração	Da esquerda para direita
=	Atribuição	Da direita para esquerda

# + Operadores – precedência

## ■ Exercícios

Expressão	Resultado
$10 + 10 / 2$	15
$(10 + 10) / 2$	10
$10 + (10 / 2)$	15
$10 + 10 \% 2$	10
$1 + 4 / 2 + 2 * 2$	7
$1 + 4 / (2 + 2) * 2$	3

- Na dúvida, use parênteses!

# + Operadores

- O tipo do resultado depende do tipo dos operandos
  - No caso, o tipo mais abrangente
  - Ex.:
    - Se `A` é `int` e `B` é `long int`, `A + B` será `long int`
    - Se `A` é `float` e `B` é `int`, `A + B` será `float`
- **CUIDADO!!!**
  - Assim como `int + int` dá resultado `int`
  - Temos que `int / int` dá resultado `int`



# Operadores



## ■ CUIDADO!!!

- Se `A` é `int`, e vale 15, então `A/2` dá como resultado 7 (por quê?)
- Alternativas de solução
  - Declarar `A` como `float` ou `double`
  - Mudar a expressão para `A/2.0` (por que funciona?)
  - Type casting (conversão explícita): `(double)A/2`

- Exercício – que valor sairá na tela?

```
int a;  
float b;  
a = 15;  
b = a / 2;  
cout << b;
```



# Continua em:

## Entrada, Saída e formatação