

Curso de C

Sistemas Numéricos

Sistemas Numéricos

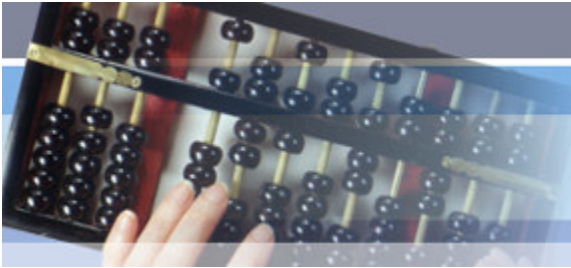
Objetivos:

- Entender como representar números
- Aprender a converter entre representações
- Compreender como o computador representa números na memória
- Ler e escrever em outras bases

Sistemas Numéricos

Roteiro:

- Sistemas numéricos
- Conversão entre sistemas numéricos
- Representação de números
- Escrita e leitura em outras bases



Sistemas numéricos

Sistema decimal de contagem:

- Dez símbolos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

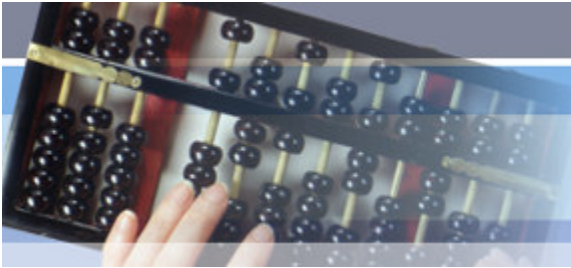
- Contagem:

00, 01, 02, 03, 04, 05, 06, 07, 08, 09,

10, 11, 12, 13, 14, 15, 16, 17, 18, 19,

20, 21, ... Usar os dez símbolos um após o outro

Terminando os dígitos numa coluna, avançar o dígito à esquerda, e assim por diante ...



Sistemas numéricos

Outros sistemas de contagem:

- Romano
 - “I”, “II”, “III”, “IV”, “V”, “VI”, “VII”, “VIII”, “IX” e “X”
 - Base 10
- Dúzia, grossa
 - Base 12
- Segundos, minutos
 - Base 60
- Unário $7 \approx \text{x x x x x x x}$
 - Base 1

Sistemas numéricos

Sistema binário de contagem:

- Dois símbolos:

0, 1

- Contagem:

000, 001,

010, 011,

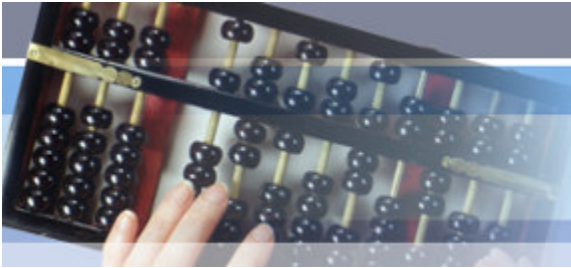
100, 101,

110, 111,

...

Usar os dois símbolos um após o outro

Ao esgotar os símbolos numa coluna,
avançar o símbolo na coluna à esquerda



Sistemas numéricos

Sistema octal de contagem:

- Oito símbolos:
0, 1, 2, 3, 4, 5, 6, 7

- Contagem:

00, 01, 02, 03, 04, 05, 06, 07,
10, 11, 12, 13, 14, 15, 16, 17,
20, 21, ...

Usar os oito símbolos um após o outro
Ao esgotar os símbolos em uma coluna,
avançar o símbolo na coluna à esquerda

Sistemas numéricos

Sistema hexadecimal de contagem:

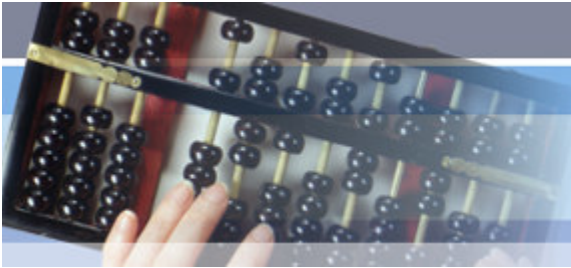
- Dezesesseis símbolos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Contagem:

00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F,
20, 21, ...

Usar os dezesesseis símbolos um após o outro
Ao esgotar os símbolos numa coluna,
avançar o símbolo na coluna à esquerda



Sistemas numéricos

Representação de números:

- Representação não ambígua:

Dígitos _{Base}

- Qual o valor de “101”?

– Decimal: $101_{10} = 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 = 101_{10}$

– Binário: $101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$

– Octal: $101_8 = 1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 = 65_{10}$

– Hexadec.: $101_{16} = 1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0 = 257_{10}$

Sistemas numéricos

Comparação entre sistemas numéricos:

0	0_2	0_8	0_{16}	11	1011_2	13_8	B_{16}
1	1_2	1_8	1_{16}	12	1100_2	14_8	C_{16}
2	10_2	2_8	2_{16}	13	1101_2	15_8	D_{16}
3	11_2	3_8	3_{16}	14	1110_2	16_8	E_{16}
4	100_2	4_8	4_{16}	15	1111_2	17_8	F_{16}
5	101_2	5_8	5_{16}	16	10000_2	20_8	10_{16}
6	110_2	6_8	6_{16}	17	10001_2	21_8	11_{16}
7	111_2	7_8	7_{16}	18	10010_2	22_8	12_{16}
8	1000_2	10_8	8_{16}	19	10011_2	23_8	13_{16}
9	1001_2	11_8	9_{16}	20	10100_2	24_8	14_{16}
10	1010_2	12_8	A_{16}	21	10101_2	25_8	16_{25}

Conversão de Base



Conversão de Base

Conversão para base decimal:

Algoritmo de *multiplicações sucessivas*:

Entrada: Número N , base B :

Dígitos: $A_n A_{n-1} A_{n-2} \dots A_2 A_1 A_0$

Algoritmo:

- Resultado $\leftarrow 0$.
- Para cada dígito A_i (de A_n até A_0):
 - Resultado $\leftarrow \text{Resultado} * B + A_i$

Conversão de Base

Exemplo:

Transformar **1305₈** (octal) para a base decimal:

	Valor	Dígito	Próximo
0	$0 \cdot 8$	+ 1	= 1
1	$1 \cdot 8$	+ 3	= 11
2	$11 \cdot 8$	+ 0	= 88
3	$88 \cdot 8$	+ 5	= 709
4	709	FIM	

Portanto: $1305_8 = 709_{10}$

Conversão de Base

Exemplo:

Transformar **1E31**₁₆ (hexadecimal) para decimal:

	Valor	Dígito	Próximo
0	$0 \cdot 16$	+ 1	= 1
1	$1 \cdot 16$	+ E (14)	= 30 ($16 + 14 = 30$)
2	$30 \cdot 16$	+ 3	= 483
3	$483 \cdot 16$	+ 1	= 7729
4	7729	FIM	

Portanto: 1E31₁₆ = **7729**₁₀

Conversão de Base

Exemplo:

Transformar **00101011₂** (binário) para decimal:

	Valor	Dígito	Próximo
0	$0 \cdot 2$	+ 1	= 1
1	$1 \cdot 2$	+ 0	= 2
2	$2 \cdot 2$	+ 1	= 5
3	$5 \cdot 2$	+ 0	= 10
4	$10 \cdot 2$	+ 1	= 21
5	$21 \cdot 2$	+ 1	= 43
6	43	FIM	

Portanto: $00101011_2 = 43_{10}$



Conversão de Base

Conversão de decimal para outra base:

Algoritmo de *divisões sucessivas*:

Entrada: Número n , base b

Dígitos: $a_m a_{m-1} a_{m-2} \dots a_2 a_1 a_0$

Algoritmo:

- Enquanto $n \neq 0$:
 - $n \leftarrow n \div b$ (resto r)
 - $a_i \leftarrow r$ (de a_0 até a_m).

Conversão de Base

Exemplo:

Transformar **2034** para a base octal:

	Valor		Resto
0	2034 $\div 8 =$ 254	+	2
1	254 $\div 8 =$ 31	+	6
2	31 $\div 8 =$ 3	+	7
3	3 $\div 8 =$ 0	+	3

Portanto: $2034_{10} = 3762_8$

Conversão de Base

Exemplo:

Transformar **1253** para a base hexadecimal:

	Valor	Resto
0	1253 $\div 16 = 78$	5
1	78 $\div 16 = 4$	14 (E)
2	4 $\div 16 = 0$	4

Portanto: $1253_{10} = 4E5_{16}$

Conversão de Base

Exemplo:

Transformar **35** para a base binária:

	Valor				Resto
0	35	$\div 2 =$	17	+	1
1	17	$\div 2 =$	8	+	1
2	8	$\div 2 =$	4	+	0
2	4	$\div 2 =$	2	+	0
2	2	$\div 2 =$	1	+	0
2	1	$\div 2 =$	0	+	1

Portanto: $35_{10} = 100011_2$

Conversão de Base

Conversão entre quaisquer bases:

Mais fácil:

- Converter da base original para base decimal (multiplicações sucessivas)
- Converter da base decimal para outra base (divisões sucessivas).
- Binário, Octal e Hexadecimal: usar casos especiais!

Conversão de Base

Casos especiais:

Converter 00110110_2 para hexadecimal:

Grupos de 4 bits

0	0	1	1	0	1	1	0
3				6			

Converter 00110110_2 para octal:

Grupos de 3 bits

0	0	1	1	0	1	1	0
0		6		6			

Números em C



Números em C

Números em C:

- Decimais: 10, 132, 32179, ...
 - Começam com 1, 2, 3, ..., 9
- Octais: 012, 0204, 076663, ...
 - Começam com 0
- Hexadecimais: 0xA, 0xa, 0x84, 0x7B3, ...
 - Começam com 0x

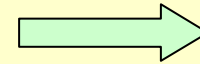
Números em C

No código:

```
int i;
```

```
i = 10;
```

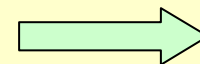
```
printf("i (dec)=%d", i);
```



10

```
i = 010;
```

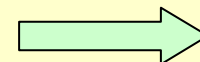
```
printf("i (dec)=%d", i);
```



8

```
i = 0x10;
```

```
printf("i (dec)=%d", i);
```



16

Números em C

Escrita com `printf`:

- Na base 10 (decimal): `%d` ou `%i`
- Na base 8 (octal): `%o`
- Na base 16 (hexadecimal): `%x` ou `%X`

```
int i=22; int j=-1;
printf("i (dec)=%d, i (oct)=%o, i (hex)=%x", i, i, i);
printf("j (dec)=%d, j (oct)=%o, j (hex)=%x, j (HEX)=%X", j, j, j, j);

i (dec)=22, i (oct)=26, i (hex)=16
j (dec)=-1, j (oct)=3777777777, j (hex)=ffffffff, j (HEX)=FFFFFFFF
```

Converte automaticamente para a representação pedida

Números em C

Leitura com `scanf`:

- Na base 10 (decimal): `%d` ou `%i`

```
int i;  
scanf("%d");  
printf("i (dec)=%d", i);
```

22 → 22

- Na base 8 (octal): `%o`

```
int i;  
scanf("%o");  
printf("i (dec)=%d", i);
```

25 → 21

89 → erro

- Na base 16 (hexadecimal): `%x` ou `%X`

```
int i;  
scanf("%x");  
printf("i (dec)=%d", i);
```

2a → 42

Representação de Números





Representação de Números

Bits, Nibbles, Bytes e Palavras:

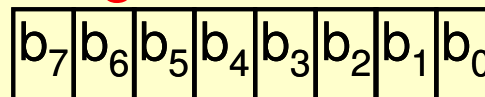
- *Bit*: unidade básica de informação
 - Valores: 0 ou 1
- *Nibble*: 4 bits
 - Valores: 0000, 0001, 00010, ..., 1111
- *Byte*: 8 bits
 - Valores: 0000 0000, ..., 1111 1111
- *Palavra*: 2, 4 ou 8 bytes

Representação de Números

Representação Binária:

- Um byte:

Dígitos binários

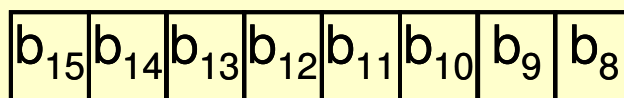


Bit mais
significativo

Bit menos
significativo

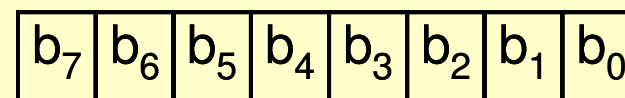
- Uma palavra:

Byte mais significativo

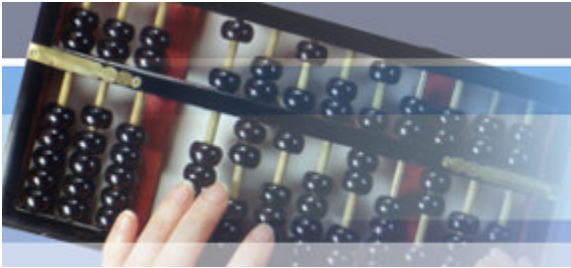


Bit mais
significativo

Byte menos significativo



Bit menos
significativo



Representação de Números

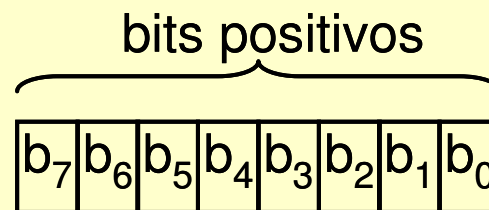
Tipos Binários:

Tipo	Quantidade de memória
<i>char</i> <i>signed char</i> <i>unsigned char</i>	1 byte (8 bits)
<i>(signed) short int</i> <i>unsigned short int</i>	2 bytes (16 bits)
<i>(signed) int</i> <i>unsigned int</i>	4 bytes (32 bits)
<i>(signed) long int</i> <i>unsigned long int</i>	4 bytes (32 bits)
<i>(signed) long long int</i> <i>unsigned long long int</i>	8 bytes (64 bits)

Representação de Números

Número unsigned (*sem sinal*):

Cada bit representa um dígito:



Valor: $b_7 \cdot 2^7 + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$

Palavra com n bits:

Valor mínimo: 0 (todos os bits são 0)

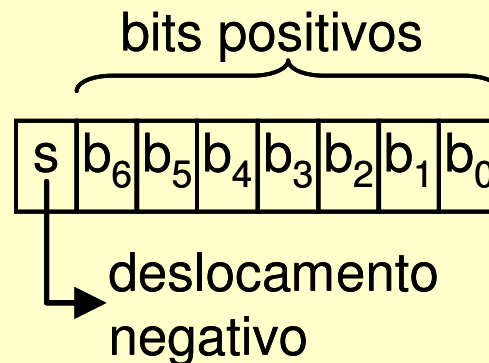
Valor máximo: $2^n - 1$ (todos são 1)

Representação de Números

Número signed (*com sinal*):

Primeiro bit: deslocamento negativo

Outros bits: dígitos



Valor: $-2^{8-1} \cdot s + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$

Palavra com n bits:

Valor mínimo (s é -1, resto é 0): -2^{n-1}

Valor máximo (s é 0, resto é 1): $2^{n-1} - 1$

Representação de Números

Exemplo:

Representar (8 bits): 66

Em binário: 1000010_2

unsigned int:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	1	0	0	0	0	1	0

signed int:

s	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	1	0	0	0	0	1	0

A representação é igual para signed e unsigned

Representação de Números

Exemplo:

Representar (8 bits): 175

Em binário: 10101111_2

unsigned int:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
1	0	1	0	1	1	1	1

signed int:

s	b_6	b_5	b_4	b_3	b_2	b_1	b_0
1	0	1	0	1	1	1	1

Representação errada!

valor correto

$$-128 + 32 + 8 + 4 + 2 + 1 = -81$$

Representação de Números

Exemplo:

Representar: -90 $-90 = -2^7 + 38$

Em binário: $38 = 100110_2$

unsigned int: **impossível**

signed int:

s	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
1	0	1	0	0	1	1	0

Sistemas Numéricos

A background image showing a close-up of a traditional abacus. The abacus has a wooden frame and several vertical rods. Each rod has black beads that can slide up and down. A person's hands are visible, with fingers moving the beads. The image is slightly faded and serves as a background for the text.

- Sinais
- LongShort