Sistemas de Numeração

Já nos tempos remotos o ser humano sentia a necessidade de quantificar coisas, fossem cabeças de um rebanho, número de inimigos ou qualquer outra informação contável. Todos os seres vivos possuem, de uma maneira ou de outra, a faculdade de comparar, seja ela qualitativa ou quantitativa, e são capazes de avaliar quantidades através de misteriosas sensações de suficiência e qualidades mediante peculiares raízes instintivas de raciocínio. Alimento, luz, ar, companheiros da mesma espécie, inimigos da espécie e outros fatores são medidos em termos de muito, pouco ou suficiente, de acordo com os padrões ou necessidades da espécie.

No ser humano, particularmente, estas faculdades se desenvolveram de maneira acentuada. Sabe-se que não podemos apreender diretamente nenhum número acima de cinco. Além deste número existe apenas o conceito de "muito". Diante desta necessidade, o homem desenvolveu métodos requintados de quantificação. Contar, por exemplo, é um processo de comparar quantidades (principalmente unidades). Com o advento da socialização do ser humano, surgiram sistemas de contagem em planos abstratos, onde já não se dependia da presença física das coisas a serem quantificadas. Os sistemas de numeração inventados foram e ainda são basicamente um meio convencional de expandir a primitiva faculdade de comparar.

Provavelmente, o primeiro sistema a surgir foi o sistema unitário, o sistema baseado em um só dígito. Provavelmente um antigo pastor de ovelhas Neanderthal recorria a desenhos para saber se nenhuma cabeça havia se estraviado. Utilizava como algarismos o desenho do quadrúpede e comparava a quantidade de desenhos com a quantidade de ovelhas. Mais tarde passou a utilizar outro símbolo, pontos por exemplo, p/ designar uma ovelha. Nascia aí, a partir da representação concreta, a representação abstrata e com estas, novos horizontes da matemática. A partir disto, o homem atribuiu símbolos a quantidades maiores, como por exemplo, .=1 (um ponto é igual a uma unidade); ..=2 (dois pontos igual à quantidade dois); ...=3 (três pontos igual à quantidade três). Se o homem não tivesse feito isso, hoje escreveríamos o n°5 como "....." (cinco pontos) ou "11111". Os babilônios utilizavam grupos de luazinhas para representar grandezas de 0 a 9; Os egípcios tinham um, dois e três sinais iguais p/ as grandezas 1, 2 e 3 e um sinal diferente para as grandezas de 4 a 9; os romanos utilizavam sinais I, V, X, C, L, M.

Estes sistemas necessitavam de outros símbolos para quantidades ainda maiores (bilhões, trilhões etc). Presume-se que foram os indianos que primeiramente observaram que, adotando-se uma pequena coleção de símbolos (9 no caso), a posição de um símbolo em relação a outro bastaria para indicar grandezas maiores que o número de símbolos. A idéia foi adotada e propagada pelos árabes, que denominaram símbolos de algarismos (em homenagem ao famoso matemático Al-Khowârizmê). Também foram os inventores do zero, símbolo indispensável ao sistema de numeração por ordens (também chamado de sistema de quantificação por notação posicional). Curiosamente , os árabes não utilizaram sua própria invenção. Foram eles que inventaram os signos ou símbolos (desenhos que representam as quantidades de 0 a 9) que atualmente todo o mundo ocidental usa, enquanto eles, seus inventores, não o utilizam.

Nos sistemas de numeração que adotam o conceito de ordem, temos a primeira ordem representando as unidades com cada unidade representada por um símbolo diferente e em seguida, outras ordens (unitária, dezena, centena, decimal, centesimal etc).

Todos eles foram inventados baseados em 2 conveniências:

a) haver poucos símbolos p/ memorização e

b) possibilitar a representação de quantidades muito grandes.

Sistema decimal

A ordem das unidades contém 10 símbolos (0,1,2,3,4,5,6,7,8,9,), representando as dez grandezas peculiares a este sistema. O número dez (10), formada por dois dos símbolos da ordem unitária, inaugura uma segunda ordem, a das dezenas; o 100 inaugura a 3° ordem, a das centenas e assim por diante.

Ainda uma especulação. Muito provavelmente foi o fato de termos 10 dedos nas mãos que influenciou a escolha de nossa espécie pelo sistema decimal, o que pode, sob certos aspectos, ser considerado como um fato infeliz, pois o sistema decimal não é, em absoluto, o melhor de todos. O sistema de base 12 seria muito mais vantajoso devido ao menor número de divisões quebradas que resulta.

Notação Posicional

A posição que um algarismo ocupa em relação aos demais e a base do sistema em questão nos fornece todos subsídios necessários para o entendimento e representação de uma grandeza ou quantidade. Todos os sistemas de numeração conhecidos têm uma notação definida, igual para várias bases, que torna possível a identificação de qualquer número baseado somente nos algarismos adotados pela base e nas posições que ocupam entre si.

Por exemplo: no número 1962 temos o algarismo um (1) na posição que indica milhares, o 9 na posição indicativa de centenas, o 6 na de dezenas e o 2 na posição de unidades. Assim sabemos que o n°1962 é igual a 1x1000+ 9x100+ 6x10+2x1= 1000+900+60+2=1962.

Podemos escrever ainda, usando uma outra forma de representar a mesma coisa, que 1962 é igual a:

$1x10^3+ 9x10^2+ 6x101+2x10^0$ pois $1000 = 10^3$; $100 = 10^2$; $10 =10^1$ e $1=10^0$.

O nº 1962 está escrito na base dez, i.e., no sistema decimal. A rigor podemos generalizar para qualquer base:

Seja "b" a base de representação de um número e A, B, C, D, E,... os símbolos dos algarismos deste sistema, então o número

.... EDCBA na base "b", escrito convencionalmente como

$$EDCBA_b$$

representa a grandeza $E.b^4+ D.b^3+ C.b^2+ B.b^1+ A.b^0$.

Veja o exemplo.

$1234_{10}= 1x10^3 + 2x10^2+ 3x10^1+ 4x10^0= 1000+200+30+4= 1234$

0 mesmo "número", numa base hipotética 5, representaria uma quantidade ou grandeza diferente:

$1x5^3+2x5^2+3x5^1+4x5^0 = 125+50+15+4= 194$ (diferente de 1234).

Para representarmos na base 5 a grandeza 1234, precisaríamos do número 14414_5, pois:

$1x5^4+4x5^3+4x5^2+1x5^1+4x5^0 = 625+500+100+5+4= 1234$

O processo para a determinação deste número será tratado mais adiante.

Sistema Binário

Os atuais computadores processam suas operações em um sistema diferente do decimal, o sistema binário.

O sistema binário, como o nome já diz, tem dois algarismos aos quais damos geralmente os símbolos 0 e 1, que correspondem a qualquer conjunto dual, como por exemplo: não e sim; falso e verdadeiro; desligado e ligado; negativo e positivo, falso e verdadeiro, etc. Nos circuitos lógicos, 0 e 1 representam respectivamente níveis de tensão baixa e alto ou estados de saturação e corte de transistores. Daí, uma outra designação comum: L e H (Low e High levels do inglês: baixo e alto níveis de tensão).

Na base 2, o número decimal 11 (e grandeza ou quantidade 11) é representado por 1011_2.

Vê-se que na base 2 foram necessários 4 sinais-algarismos para representar a grandeza 11, que no sistema decimal é representado por apenas dois sinais.

A explicação é simples. Na primeira posição do sistema decimal podemos representar 10 grandezas (0 a 9) ao passo que na posição correspondente do sistema binário só podemos representar duas (0 e 1). Se à maior grandeza, da posição de unidades decimais, acrescentarmos uma unidade, a primeira ordem (ou posição) volta à grandeza menor e a próxima ordem é incrementada uma unidade. Este é todo o segredo que envolve a passagem do nº 9 p/ 10_{10}. O 9 volta para o zero (menor quantidade) e a posição das dezenas (anteriormente neste caso ocupada pelo algarismo zero, que não precisamos representar) é incrementada para 1, fornecendo o número 10.

Exemplos: descubra os próximos números para: a) 1234_5; b) 1234_6; c) 12_3 ; d)12_{10} e e)19_{10}

(Respostas: a)1240/b)1235/c)20/d)13/e)20nas respectivas bases)

Como o sist. Binário só tem dois algarismos, na 1° ordem podemos representar 2. Se utilizarmos a primeira e Segunda ordens, podemos representar até 4 grandezas, com as 3 primeiras posições ou ordens, podemos representar até 8 grandezas e assim por diante. Note a relação:

$$2^{\text{n° da ordem}}.$$

Por exemplo, com um número binário de 10 posições ou ordens podemos representar até 2^{10}= 1024 grandezas (da grandeza zero até a grandeza 1023), ao passo que dez posições no sistema decimal representaria 10^{10}= 10.000.000.000 ou dez bilhões de grandezas diferentes.

O sistema binário é usado em computadores devido à maior facilidade de manipular somente duas grandezas. No caso dos computadores, precisamos ter somente "tensão presente" ou "tensão nula" ou corte e saturação de transistores.

Sistema Octal

Como já diz o nome, é o sistema de base 8 e, consequentemente, contém 8 algarismos (0,1,2,3,4,5,6 e 7). É utilizado por ser um sistema que tem relação direta com o sistema binário. Veremos esta relação quando tratarmos de transformação entre bases. Neste sistema, a grandeza 8 é representada por 10_8 pois $1x8^1+0x8^0= 8+0$

Sistema Hexadecimal

Do hexa=6 e deci=10, sistema numérico de base 16. Tem 16 algarismos que são: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. A,B,C,D, E,e,F fazem o papel das grandezas 10,11,12,13,14,15. Usamos as letras maiúsculas pela necessidade de termos que representar cada uma destas grandezas com um único algarismo.

O sistema Hexadecimal é um sistema muito utilizado em computadores. Neste sistema a grandeza 16 é representada por 10_H ou 10_{16} pois $1x16^2 +0x16^0= 16+1$.

Ex: que grandeza representa o número $1AC_H$?
Solução:

Sol: $1x16^2+Ax16^1+Cx16^0=1x16^2+10x16^1+12x16^0$, uma vez que A e C representam 10 e 12 respectivamente
$=256+160+12=428=428_{10}$

Neste ponto é conveniente perceber a diferença entre a grandeza 428 e a representação decimal da mesma, 428_{10}. A primeira representa uma quantidade de objetos ou coisas, enquanto a segunda é somente uma forma de representação daquela quantidade.

Um exemplo mais simples:

> *Temos três ovelhas num pasto. A grandeza é 3 (que representa a quantidade de objetos-ovelha) é representada pelos números 3_{10}, 11_2, 3_8 e 3_H; nos sistemas decimal, binário, octal e hexadecimal, respectivamente.*

Resumo de Sistemas Numéricos

Os Sistemas Numéricos representam grandezas e/ou quantidades.

Sistema	Base	algarismos	maior representação na 1° ordem
decimal	10	0,1,2,3,4,5,6,7,8,9	9
binário	2	0,1	1
octal	8	0,1,2,3,4,5,6,7	7
hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	15 ou F

Conversão entre Bases: Decimal® binária, octal, hexadecimal:

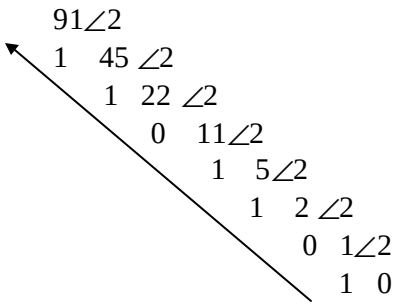
Técnica usual de transformação: Parte inteira

> *Para converter um número decimal inteiro em um número de base "b", basta executar sua divisão aproximada por "b", sucessivamente até que o enésimo dividendo não possa mais ser dividido por b, é ler os restos de trás para diante.*

Veja o exemplo:

$91_{10} \rightarrow X_2$ $\quad N \angle b \Rightarrow N = q.b +$

Onde q, b e r são inteiros e N é a parte inteira do número decimal



$X_2 = 1011011_2$

No exemplo, *o divisor é sempre a base para a qual se quer converter o número decimal; o último quociente inteiro passa a ser dividendo da próxima divisão. O processo continua até que o dividendo seja menor que o divisor (a base), quando então passa a ser o último "resto".*

Parte Fracionária

O processo é diferente para a parte fracionária. Tomemos a seguinte exemplo:

$91,6_{10}$ ® X_2

A parte inteira do número é convertida conforme o processo já demonstrado e obtemos o nº 1011011_2. A parte fracionária, 0,6, é convertida da seguinte maneira:

Multiplica-se a parte fracionária (multiplicando) pela base "b" (multiplicador), neste caso o 2, e separa-se a parte inteira do produto. O resultado obtido da subtração da parte inteira do produto passa a ser o próximo multiplicando. Faz-se sucessivamente esta operação até que consiga uma precisão satisfatória. Lê-se os algarismos separados de cima para baixo.

Veja o exemplo

$0,6_{10} \rightarrow X_2$ $\qquad 0,6_{10} \rightarrow X_{8\ (\text{exercício})}$

0,6x2=1,2
menos a parte inteira (1)=0,2
vezes 2= 0,4
menos a parte inteira (0) = 0,4
x2 = 0,8
menos a parte inteira (0)= 0,8
x 2 = 1,6
menos a parte inteira (1)= 0,6
x 2 = 1,2
menos a parte inteira (1)=0,2 e assim por diante

Lendo de cima para baixo teremos 10011, então $0,6_{10}=10011_2$. Se fizermos uma conferência, descobriremos que $0,10011_2$ é igual a:

$1x2^{-1}+0x2^{-2}+0x2^{-3}+1x2^{-4}+1x2^{-5}=1/2+1/16+1/32=19/32=0,59375,$

portanto, como podemos perceber, teremos sempre diferenças de precisão entre bases.

Exercícios: transforme os números decimais 0,5; 0,2, 0,25, 0,8 e 0,99 em números binários.

Binário, Octal, Hexadecimal® Decimal

Este processo serve para converter qualquer base para a base decimal.

O processo deriva da notação posicional comum a todos os sistemas de numeração que utilizam ordens. Tomemos como exemplo o número real:

$$XYZ, WK_B$$

onde X, Y ,Z, W e K são algarismos da base "b" e Z é o algarismo da 1^0 ordem, Y da 2^0 e X da terceira ordem da parte inteira. W e K são algarismos das ordens fracionárias. Podemos dizer que cada um destes algarismos é multiplicado por um peso que depende da posição em que se encontra e da base em que esta expresso o número. Assim, os pesos dos sistemas numéricos ordenados serão sempre:

$$\ldots\ b^4\ b^3\ b^2\ b^1\ b^0, b^{-1}\ b^{-2}\ b^{-3}\ b^{-4}\ \ldots$$

e o n^0 genérico acima será:

$$X.b^2 + Y.b^1 + Z.b^0 + W.b^{-1} + K.b^{-2}$$

com $b^0 = 1$. Os exemplos práticos a seguir tornam isto mais claro.

Ex.1

Pesos das posições 6 5 4 3 2 1 0 -1 -2 -3 -4 -5

$1\ 0\ 1\ 1\ 0\ 1\ 1,\ \ 1\ 0\ 0\ 1\ 1_2 \rightarrow X_{10}$

$1x2^6+0x2^5+1x2^9+1x2^3+0x2^2+1x2^1+1x2^0+1x2^{-1}+0x2^{-2}+0x2^{-3}+1x2^{-4}+1x2^{-5} =$
$64+0+16+8+0+2+1+1/2+0+0+1/16+1/32 = 91,59375 \text{ ou } @\ 91,6_{10}$

Ex2:

$13^A,C_H \rightarrow X_{10}$

na base 16, A=10 e C=12, então:

$1x16^2+3x16^1+10x16^0+12x16^{-1} = 256+48+10+12/16=314,75_{10}$

Ex3;

$265,41_8 \rightarrow X_{10}$

$= 2x8^2+6x8^1+5x8^0+4x8^{-1}+1x8^{-2} = 128+48+5+4/8+1/64\ @\ 181,51563_{10}$

Conversão binário, octal, hexad. Entre si

Sendo 2, 8 e 16 potências de 2, as conversões entre os sistemas binário, octal e hexadecimal são imediatas, como se poderá ver.

Binário $\rightarrow$ octal

$8=2^3 \Rightarrow$ separa-se o número binário em grupos de 3 e transforma-os diretamente p/ a base 8

Ex: 10110101_2 para a base oito

10ê110ê101$_2$

101$_2$*= 5 ; 110=6 ; 10=2* *Þ 265*$_8$

Binário $\rightarrow$ Hexadecimal

$16=2^4 \Rightarrow$ separa-se o número binário em grupos de 4 algarismos e transforma-os diretamente p/ a base 16

Ex: $1011|0101_2$

1011ê0101$_2$

2= 010

6=110 ® 110010$_2$

Hexa $\rightarrow$ binário.

Cada algarismo hexadec. Gera a mesma grandeza em um grupo de 4 algarismos binários

$BF1_{16} \rightarrow X_2$

1= 0001$_2$

F= 1111$_2$

B= 1011$_2$ *® 101111110001*$_2$

A conversão de um número X na base genérica b1 para um em outra base b2 é efetuada através da conversão do primeiro número X_{b1} para a base 10 e da base 10 para a base b2.

Exercícios propostos

1) $1990_{10} \rightarrow X_2$

2) $10101010_2 \rightarrow X_{10}, X_{8,} X_{16}$

3) $AB2,C_H \rightarrow X_{10,}\ X_2$

4) $40,25_{10} \rightarrow X_H$

5) $541_7 \rightarrow X_9$

6) $F8_H \rightarrow X_4$

7) $110111_2 + 728 \rightarrow X_{10}$

8) $F8_H - 26_8 \rightarrow X_{10}$

9) $20_3 + 40_5 \rightarrow X_{10}$

10) $27_{10} - 110_2 \rightarrow X_H$

11) $100_2 \times 14_H \rightarrow X_{10}$

Antes de passarmos aos códigos, é necessário conceituarmos alguns vocábulos doravante utilizados.

bit→ (pron. *bit*)O vocábulo surgiu da contração abreviada de "binary digit" do inglês e representa os valores possíveis que uma variável lógica (binária) pode assumir, 0 e 1.

byte→ (pron. *ba'it)* grupo ou palavra de 8 bits (ex: 010111010)

nybble→ (pron. *ni'bôu*) grupo ou palavra de 4 bits (ex: 0111)

word= palavra→ (pron. *uô.rd*). Palavra é qualquer conjunto de bits que contém ou representa um item de informação

Ex: Se 01 faz com que um sistema gire um motor em um sentido, 10 gira-o noutro sentido e 00 desliga o motor, então 01,10 e 00 são palavras porque carregam em si uma "informação".

Bit também pode ser o nome de uma palavra de um único bit.

(origens: do inglês: bit=mordidinha; nybble=mordida moderada e Byte=grande mordida)

Códigos

Veremos a seguir alguns códigos baseados em dígitos binários. É importante saber a diferença entre os códigos e os sistemas numéricos. Um sistema numérico representa quantidades ou grandezas e os códigos estão relacionados com informações (que podem ser grandezas, também, se assim o desejarmos), transmissão destas informações e sua interpretação por sistemas lógicos.

Código BCD

Do inglês "*Binary Coded Decimal*" ou Código decimal codificado em binário.

Características: Representa as grandezas decimais (0,1...9); utiliza uma palavra de 4 bits (*nybble*); Os pesos de cada posição dos bits são iguais ao sistema binário, ou seja 2^3,2^2,2^1.2^0=8,4,2,1; usa-se só 10 das 16 possíveis combinações de um nybble para representar as quantidades de zero a nove.

Decimal	BCD natural pesos				BCD 3 em excesso			
	8	4	2	1				
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0?

No código "BCD 3 em excesso" a correspondência não é direta. Nota-se que os *nybbles* têm sempre três unidades em excesso. Isto é feito de modo a facilitar os ajustes decimais necessários após operações de soma lógica.

Código (refletido) de Gray

Características: Neste código, sempre teremos
a) uma distância (veja definição abaixo) entre palavras imediatamente adjascentes igual a um e
b) em qualquer ponto de desenvolvimento do código, teremos entre a primeira e última palavras uma distância também igual a um.

Distância entre palavras: *Distância entre duas palavras digitais é o número de bits diferentes (quando comparados em suas respectivas posições) entre elas. Assim, dizemos que a distância entre as palavras 100101000 e 100111100 é dois porque existe diferenças entre os quinto e sétimo bits (contando da esquerda para a direita neste caso).*

Veremos mais tarde a vantagem deste código quando estudarmos os mapas de Karnaugh, onde a utilização deste código tornará possível a visualização por adjascência geométrica de termos que também estão adjacentes matematicamente.

Desenvolvimento do código de Gray

Primeiramente, faz-se uma coluna dos valores possíveis de um bit:

0
1

em seguida, imaginamos um espelho imaginário e refletimos bits p/ baixo do espelho.

0
1
———
1
0

Colocamos, então, "zeros" à esquerda dos bits que estão acima do "espelho" e "uns" à esquerda daqueles que estão abaixo do espelho:

00
Zeros→ {01
———
11
uns→ {10

temos agora o código de Gray para 4 palavras. O processo é repetido até que se consiga o número de palavras requerido.

Exercício: desenvolver o código de Gray p/ 8 palavras

Resposta:
000
001
011
010
110
111
101
100

Perceba as características do código. A quinta palavra do código, 110, difere somente no primeiro bit da 4^0 palavra, 010, e somente no 3^0 bit da 6^0 palavra, 111, porque estão adjacentes. A distância entre a Quinta e Quarta e entre a Quinta e Sexta palavras é um. A última palavra, 100, também só difere em um bit da primeira, 000.

A seguir, resume-se o processo de desenvolvimento do código em forma de fluxograma:

Algorítimo para determinação do código de Gray

1-Faz-se a coluna das palavras possiveis somente com um único bit (o em cima e 1 em baixo)

2-Imagina-se um espelho por baixo da coluna e copia-se os digitos acima, como se estivessem sendo refletidos por um espelho plano.

3-coloca-se zeros à esquerda das palavras que ficaram acima do espelho imaginário e uns à esquerda das palavras que ficaram abaixo do mesmo.

4-repete-se os passos 2 e 3 até que se tenha conseguido o código com o comprimento (número de palavras ou número de bits/palavra) desejado.

Exercício: desenvolva o código Gray para o comprimento de 32 palavras (ou 5 bits).

Código Biquinário

Este código é estudado somente para demonstrar um exemplo de um código onde os pesos das posições e a organização das mesmas são drasticamente diferentes do que foi visto até o momento.

Características: Código de 7 bits divididos em dois grupos: o primeiro com 2 e o segundo com 5 bits; cada palavra tem somente dois "uns", um em cada grupo; cada posição tem um peso, como se pode ver Na tabela.

Tem a desvantagem de precisar de muitos bits para cada palavra, e em contrapartida, a vantagem de ser um código de fácil detecção de erro, pois a presença de somente um ou mais de dois dígitos "1" em cada palavra ou a ausÊncia de um "1" ou presença de mais de um "1" em cada grupo acusa erro de transmissão-recepção de dados ou informações.

Veja a seguir a relação entre o código biquinário e o decimal.

Biquinário			**Pesos**				
Dec	**5**	**0**	**4**	**3**	**2**	**1**	**0**
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

Código ASCII

Do inglês, "American Standard Code For Information Interchange", ou Código Padrão Americano para troca de Informações. Este código utiliza 7 bits para representação das informações (caracteres, sinais gráficos e comandos) mais um bit de paridade. O bit de paridade é inserido na palavra ASCII como um meio para possibilitar a detecção de erros de transmissão, recepção e/ou interpretação de informações. A poucos anos atrás, a chance de ocorrer o erro em 1 bit da palavra 8 bits era de uma em 10.000 e a de ocorrerem 2 erros na mesma palavra, de 1 em 100.000.000. Este bit de paridade evitaria o primeiro tipo de erro, uma vez que o segundo tipo tinha uma chance pouco significativa de ocorrer. Atualmente o volume, complexidade e velocidade com que os dados e informações são transmitidos são tão grandes que se faz necessário a utilização de mecanismos bem mais potêntes para a detecção de erros.

O ASCII foi adotado pela maioria dos fabricantes de computadores do mundo. A IBM foi uma exceção, desenvolvendo seu próprio código; o EBCDIC.

Devido ao fato de que nem o ASCII nem o EBCDIC não contam os caracteres da nossa língua, o "ç", ã, e é por exemplo, a Secretaria Especial de Informática, a SEI, aprovou no primeiro semestre de 1986 um projeto para a criação do código BRASCII (ASCII com caracteres em português ou o ASCII brasileiro), para servir de padrão para equipamentos nacionais.

Código ASCII

O código ASCII conta com 128 palavras, 2^7, que vão de 000000_2 a 1111111_2, ou 00_H a $7F_H$ OU 00_H A $127_{D.}$ Está dividido como mostrado a seguir:

Decimal	Hexadecimal	Função
0 32	0 20	Instruções de controle
33 47	21 2F	Sinais gráficos
48 57 58 64	30 39 3^A 40	n^0's 0→ 9 Sinais gráficos
65 90	41 5^A	Letras Maiúsculas
91 96	5B 60	Sinais Gráficos
97 122	61 7^A	Letras minúsculas
123 127	7B 7F	Sinais gráficos
128 165	80 A5	Letras especiais: à, ç, ü, *f*, ⌐,...
166 255	A6 FF	Caracteres especiais: α, β, Ω, Φ,≥, ...

A seguir, temos a tabela detalhada do código.

Código ASCII	Caractere	Código ASCII	Caractere	Código ASCII	Caractere	Código ASCII	Caractere
00	NUL	10	DLE	20	SP	30	**0**
01	SOH	11	DC1(X-ON)	21	!	31	**1**
02	STX	12	DC2(TAPE)	22	"	32	**2**
03	ETX	13	DC3(X-OFF)	23	#	33	**3**
04	EOT	14	DC4	24	$	34	**4**
05	ENQ	15	NAK	25	%	35	**5**
06	ACK	16	SYN	26	&	36	**6**
07	BEL	17	ETB	27	'	37	**7**
08	BS	18	CAN	28	(	38	**8**
09	HT	19	EM	29	)	39	**9**
0A	LF	1A	SUB	2A	*	3A	:
0B	VT	1B	ESC	2B	+	3B	;
0C	FF	1C	FS	2C	,	3C	<
0D	CR	1D	GS	2D	-	3D	=
0E	SO	1E	RS	2E	.	3E	>
0F	SI	1F	US	2F	/	3F	?
Código ASCII	Caractere	Código ASCII	Caractere	Código ASCII	Caractere	Código ASCII	Caractere
40	@	50	**P**	60	,	70	**p**
41	**A**	51	**Q**	61	**a**	71	**q**
42	**B**	52	**R**	62	**b**	72	**r**
43	**C**	53	**S**	63	**c**	73	**s**
44	**D**	54	**T**	64	**d**	74	**t**
45	**E**	55	**U**	65	**e**	75	**u**
46	**F**	56	**V**	66	**f**	76	**v**
47	**G**	57	**W**	67	**g**	77	**w**
48	**H**	58	**X**	68	**h**	78	**x**
49	**I**	59	**Y**	69	**i**	79	**y**
4A	**J**	5A	**Z**	6A	**j**	7A	**z**
4B	**K**	5B	[	6B	**k**	7B	{
4C	**L**	5C	\	6C	**l**	7C	|
4D	**M**	5D	]	6D	**m**	7D	} (ALT MODE)
4E	**N**	5E	^ (↑)	6E	**n**	7E	~
4F	**O**	5F	_ (←)	6F	**o**	7F	DEL (RUB OUT

Pesquisa: Códigos EBCDIC e BRASCII

Exercício: representar as quantidades de 0 a 31 em todos os códigos estudados.