Métodos e Técnicas de Programação
.:: Bases numéricas ::.

Prof. Igor Peretta

2018-2

Contents

1 Bases numéricas

Bases numéricas diferem apenas do "jeito" de contar. A base com que nossa sociedade está acostumada é a base 10, ou seja, são 10 algaritmos. Podemos entender uma base como uma contagem em módulo. Em módulo 10, temos os possíveis valores para o resto da divisão por 10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

1.1 Base 2 (binária)

Dígitos válidos: {0, 1}

Em C: não há representação nativa

```
#include <stdio.h>
void printBits(unsigned char num, int len) {
    if(len != 0) {
printBits(num/2, len-1);
```

```
printf("%d", (num%2));
    }
}
int main() {
    int i, len = 4;
    for(i = 0; i < (1 << len); i++) {
printBits(i,len);
printf(" : %03d\n", i);
    }
    return 0;
}
```

```
0000 : 000
0001 : 001
0010 : 002
0011 : 003
0100 : 004
0101 : 005
0110 : 006
0111 : 007
1000 : 008
1001 : 009
1010 : 010
1011 : 011
1100 : 012
1101 : 013
1110 : 014
1111 : 015
```

1.2 Base 8 (octal)

Dígitos válidos: {0, 1, 2, 3, 4, 5, 6, 7}

Em C: prefixo "0"; exemplo: `0127`; para o `printf` o especificador é "%o"

```
#include <stdio.h>
int main() {
    int i;
    for(i = 0; i < 16; i++)
printf("%02o : %02d\n", i, i);
    return 0;
}
```

```
00 : 00
01 : 01
02 : 02
03 : 03
04 : 04
05 : 05
06 : 06
07 : 07
10 : 08
11 : 09
12 : 10
13 : 11
14 : 12
15 : 13
16 : 14
17 : 15
```

1.3 Base 16 (hexadecimal)

Como é uma base maior do que 10, houve a necessidade de se criar novos dígitos distintos. Foi acordado o uso das letras A a F.

Dígitos válidos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Em C: prefixo "0x"; exemplo: `0x12C`; para o `printf` o especificador é "%X" ou "%x"

```
#include <stdio.h>
int main() {
    int i;
    for(i = 0; i < 20; i++)
printf("%02X : %d\n", i, i);
    return 0;
}
```

```
00 : 0
01 : 1
02 : 2
03 : 3
04 : 4
05 : 5
06 : 6
```

```
07 : 7
08 : 8
09 : 9
0A : 10
0B : 11
0C : 12
0D : 13
0E : 14
0F : 15
10 : 16
11 : 17
12 : 18
13 : 19
```

1.4 Contagem de 0 a 15

dec	bin	oct	hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

<- bit mais significativo -> bit menos significativo

```
#include <stdio.h>
void printBits(unsigned char num, int len) {
    if(len != 0) {
printBits(num/2, len-1);
printf("%d", (num%2));
```

```
    }
}
int main() {
    int i, len = 4;
    printf("bin  | oct | hex : dec \n");
    printf("-----+-----+-----:-----\n");
    for(i = 0; i < (1 << len); i++) {
printBits(i,len);
printf(" | %02o  |  %X  : %02d\n", i, i, i);
    }
    return 0;
}
```

```
bin  | oct | hex : dec
-----+-----+-----:-----
0000 | 00  |  0  : 00
0001 | 01  |  1  : 01
0010 | 02  |  2  : 02
0011 | 03  |  3  : 03
0100 | 04  |  4  : 04
0101 | 05  |  5  : 05
0110 | 06  |  6  : 06
0111 | 07  |  7  : 07
1000 | 10  |  8  : 08
1001 | 11  |  9  : 09
1010 | 12  |  A  : 10
1011 | 13  |  B  : 11
1100 | 14  |  C  : 12
1101 | 15  |  D  : 13
1110 | 16  |  E  : 14
1111 | 17  |  F  : 15
```

1.5 Conversão de binário <-> decimal

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	1	1	0	1
128	64	32	16	8	4	2	1
128	0	0	16	8	4	0	1

soma = $128 + 16 + 8 + 4 + 1 =$ **157**

O reverso sai com módulo:

157%128		= 1
(157-128)%64	29%64	= 0
29%32		= 0
29%16		= 1
(29-16)%8	13%8	= 1
(13-8)%4	5%4	= 1
(5-4)%2	1%2	= 0
1%1		= 1

Resultado: **10011101**

```
#include <stdio.h>
void printBits(unsigned int num, int len) {
    if(len != 0) {
printBits(num/2, len-1);
printf("%d", (num%2));
    }
}
unsigned int bin2dec(char bin[]) {
    int i = 0; unsigned int dec = 0;
    while(bin[i]) {
dec = dec*2 + (bin[i] - '0');
i++;
    }
    return dec;
}
int main() {
    printf("%d\n", bin2dec("10111001"));
    printBits(157,8);
    return 0;
}

185
10011101
```

1.6 Conversão de binário <-> hexadecimal

Cada dígito hexadecimal são exatos 4 bits. A conversão se torna simples:

hex F F 2 5 **bin** 1111 1111 0010 0101

e vice versa:

bin 1000 0001 0000 1101 **hex** 8 1 0 D

```
#include <stdio.h>
unsigned long long int bin2dec(char bin[]) {
    int i = 0; unsigned long long int dec = 0;
    while(bin[i]) {
dec = dec*2 + (bin[i] - '0');
i++;
    }
    return dec;
}
int main() {
    printf("%llX\n", bin2dec("10111001101110011011100110111001"
     "10111001101110011011100110111001"));
    printf("%d\n", 0xB9);
    return 0;
}
```

```
B9B9B9B9B9B9B9B9
185
```

1.7 Conversão de octal <-> hexadecimal

```
#include <stdio.h>
int main() {
    printf("%X\n", 0235);
    printf("%o\n", 0x9D);
    return 0;
}
```

```
9D
235
```