Princípios de Desenvolvimento de Algoritmos
MAC122

Prof. Dr. Paulo Miranda
IME-USP

Aula de Revisão:
Variáveis e Atribuições

Introdução

- Memória Principal:
  - Vimos no curso anterior que a **CPU** usa a **memória principal** para guardar as informações que estão sendo utilizadas no momento.
- Objetivos da aula:
  - Entender como os dados são armazenados na memória.
  - Como utilizar a memória principal para guardar dados em linguagem C.
  - Como realizar operações sobre esses dados.

Introdução

- Os computadores digitais trabalham internamente com dois níveis de tensão e portanto as informações são codificadas na forma de números em sistema binário.
- Número:
  - Definições:
    - A relação entre a quantidade e a unidade (Newton)
    - Uma coleção de objetos de cuja natureza fazemos abstração (Boutroux)
    - Número é a representação da pluralidade (Kambly)
  - Representações:
    - **sistema decimal, sistema binário, sistema hexadecimal, numeração romana, etc.**

Sistemas de numeração

Decimal	Romano	Octal	Binário
0		0	0
1	I	1	1
2	II	2	10
3	III	3	11
4	IV	4	100
5	V	5	101
6	VI	6	110
7	VII	7	111
8	VIII	10	1000
9	IX	11	1001
10	X	12	1010

Sistemas de numeração

- O **sistema decimal** é um sistema de numeração de posição que utiliza a base dez.
  - Ex: $123 = 1x10^2 + 2x10^1 + 3x10^0$
- O **sistema binário** é um sistema de numeração posicional que utiliza a base dois.
  - Ex: $101_{(bin)} = 1x2^2 + 0x2^1 + 1x2^0 = 5_{(dec)}$

- **A conversão de base numérica** conserva o valor quantitativo (conceito) apenas altera a simbologia (representação).

Unidades de Armazenamento

- Cada dígito 0 ou 1 é chamado Bit (Binary Digit).
- Fisicamente, o valor de um bit é armazenado como uma carga elétrica acima ou abaixo de um nível padrão em um capacitor dentro de um dispositivo de memória.

- Múltiplos de **bits**
  - Byte = 8 bits
  - Kbyte = 1024 bytes
  - Megabyte = 1024 kbytes
  - Gigabyte = 1024 megabytes
  - Terabyte = 1024 gigabytes

Unidades de Armazenamento

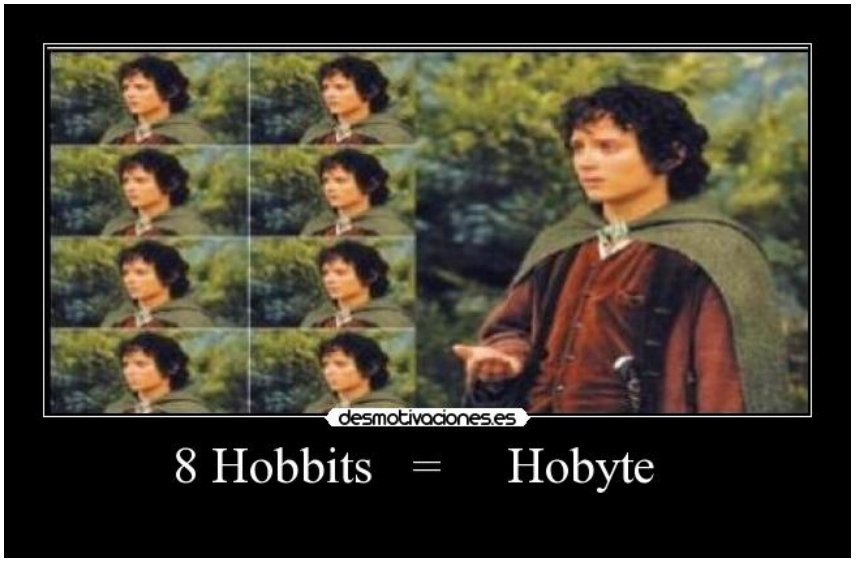
- Cada dígito 0 ou 1 é chamado Bit (Binary Digit).
- Fisicamente, o valor de um bit é armazenado como uma carga elétrica acima ou abaixo de um nível padrão em um capacitor dentro de um dispositivo de memória.

- Múltiplos de **bits**
  - Byte = 8 bits
  - Kbyte = 1024 bytes
  - Megabyte = 1024 kbytes
  - Gigabyte = 1024 megabytes
  - Terabyte = 1024 gigabytes

Unidades de Armazenamento

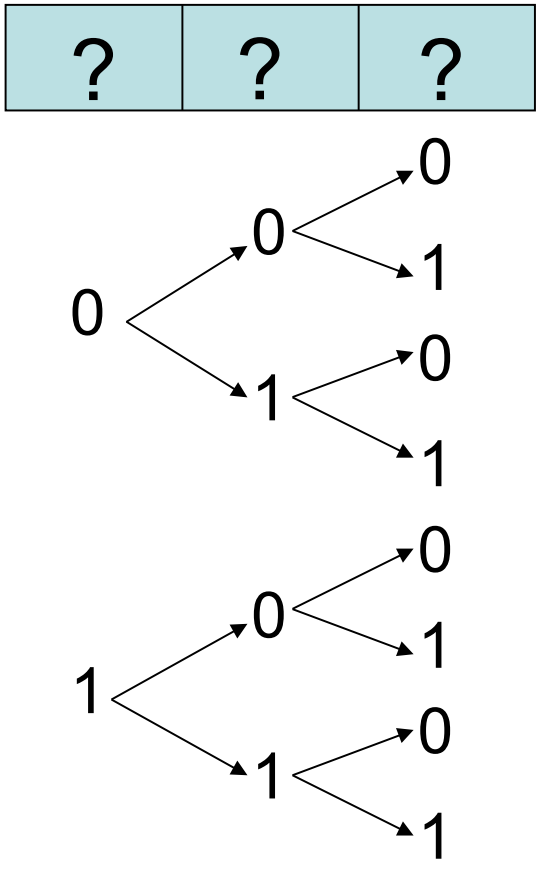
- Cada dígito 0 ou 1 é chamado Bit (Binary Digit).
- Fisicamente, o valor de um bit é armazenado como uma carga elétrica acima ou abaixo de um nível padrão em um capacitor dentro de um dispositivo de memória.

- Múltiplos de **bits**
  - Byte = 8 bits
  - Kbyte = 1024 bytes
  - Megabyte = 1024 kbytes
  - Gigabyte = 1024 megabytes
  - Terabyte = 1024 gigabytes



Unidades de Armazenamento

- Quantas diferentes combinações de 0 e 1 são possíveis usando 3 bits?



Resp: Temos 8 seqüências possíveis.

$$2 \times 2 \times 2 = 2^3$$

Generalizando para N bits temos:

$$2 \times 2 \times \ldots \times 2 = 2^N$$

Variáveis

- Em C, o acesso a memória principal é feito através do uso de variáveis.
- Uma variável é um espaço da memória principal reservado para armazenar dados tendo um nome para referenciar o seu conteúdo.
- O valor armazenado em uma variável pode ser modificado ao longo do tempo.
- Cada programa estabelece o número de variáveis que serão utilizadas.

Variáveis

- Variáveis possuem:
  - **Nome:**
    - Identificador usado para acessar o conteúdo.
    - Formado por caracteres alfanuméricos ou pelo caractere de sublinhar, mas não pode iniciar com números.
    - Não pode ter o mesmo nome de uma palavra-chave de C.
    - Em C letras minúsculas e maiúsculas são diferentes.
  - **Tipo:**
    - Determina a capacidade de armazenamento.
    - Determina a forma como o conteúdo é interpretado.
      - **Ex:** Número real ou inteiro.
  - **Endereço:**
    - Posição na memória principal.

Tipos de variáveis

- O tipo informa a quantidade de memória, em bytes, que a variável irá ocupar e a forma como seu conteúdo será armazenado.

TIPO	BIT	BYTES	ESCALA
char	8	1	-128 a 127
short	16	2	-32768 a 32767
int	32	4	-2147483648 a 2147483647
float	32	4	3.4E-38 a 3.4E+38
double	64	8	1.7E-308 a 1.7E+308
void	0	0	sem valor
unsigned char	8	1	0 a 255
unsigned int	32	4	0 a 4294967295

Tabela ASCII

- ASCII é uma padronização onde cada carácter é manipulado sob forma de código binário.

SIMB	DEC	BINÁRIO
3	51	00110011
4	52	00110100
5	53	00110101
6	54	00110110
7	55	00110111
8	56	00111000
9	57	00111001
:	58	00111010
;	59	00111011

SIMB	DEC	BINÁRIO
<	60	00111100
=	61	00111101
>	62	00111110
?	63	00111111
@	64	01000000
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100

Tabela ASCII

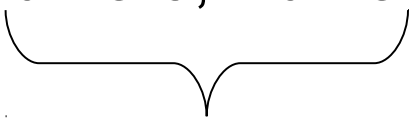
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Declaração de variáveis

- Sintaxe: <tipo> <nome> [=valor];
- Ex:

```
int ano = 1980;
float salario = 970.0;
char letra = 65;  /* 'A' é o valor 65.  */
int numero, Numero; /* C é Case Sensitive.  */
```



É possível declarar mais de uma variável do mesmo tipo de uma única vez, separando seus nomes por vírgulas.

Exemplos

```
#include <stdio.h>
int main(){
   int a;
   unsigned int b;
   short c;
   char g;

   a = 10;          /* Correto. */
   b = -6;          /* Errado.  */
   c = 100000;      /* Errado.  */
   g = 'e';         /* Correto. */
   g = e;           /* Errado.  */
   return 0;
}
```

Exemplos

```
#include <stdio.h>
int main(){
   int a = 10, b = -30;
   float c;
   char d = '4'; /* '4' é o valor 52. */
   c = a;   /* converte para float e copia 10.0 para "c". */
   c = a + 1.8; /* atribui valor 11.8 para "c".   */
   b = c;   /* converte para int truncando e copia 11 para "b".  */
   b = a + b;   /* soma 10 e 11, e copia 21 para "b".   */
   a = a + d;   /* soma 10 e 52, e copia 62 para "a".   */
   a = 0.2 + c;  /* soma 0.2 e 11.8 e copia 12 para "a".  */
   a = 0.2 + (int)c; /* converte "c" para 11 antes, soma 0.2 e
                        trunca novamente para 11 e copia 11
                        para "a". */
   return 0;
}
```

Operadores Aritméticos

- C oferece 6 operadores aritméticos binários (operam sobre dois operandos) e um operador aritmético unário (opera sobre um operando).

Binários	
=	Atribuição
+	Soma
-	Subtração
*****	Multiplicação
/	Divisão
%	Módulo (resto da divisão)
Unário	
-	Menos unário

Precedência	**Operador**
1	**-** unário
2	*** / %**
3	**+ -**

O uso de parênteses altera a ordem de prioridade das operações.

Ex:

(a + b)* 80 ≠ a + b * 80

A função printf()

- A função **printf()** é uma das funções de E/S (entrada e saída). Ela escreve o texto passado no interior dos parênteses (argumento da função) na saída padrão (terminal/monitor).
  - **Ex:** printf(“MC 102XY\n”);
- Para imprimir o conteúdo de uma variável, esta também deve ser passada como argumento da função. Na parte do texto deve ser inserido um código de formatação.
  - **Ex:** printf(“texto %*codigo*”,variável);

Códigos de formatação

Código	Formato
%c	Caractere simples (char)
%d	Decimal (int)
%e	Notação científica
%f	Ponto flutuante (float)
%g	%e ou %f (o mais curto)
%o	Octal
%s	Cadeia de caracteres
%u	Decimal sem sinal
%x	Hexadecimal
%ld	Decimal longo
%lf	Ponto flutuante longo (double)
%p	Ponteiro