

MC-102 — Aula 02

Variáveis e Atribuições

Instituto de Computação – Unicamp

Segundo Semestre de 2007



Variáveis

Definição

Variáveis são locais onde armazenamos valores na memória. Toda variável é caracterizada por um nome, que a identifica em um programa, e por um tipo, que determina o que pode ser armazenado naquela variável.

Roteiro

- 1 Variáveis
- 2 Constantes
- 3 Atribuição
- 4 Estrutura básica de um programa em C



Armários da BC

- Programar: estabelecer regras para manipulação de informações na memória principal do computador.
- Memória principal funciona como um conjunto de armários com nome, número (endereço) e capacidades de armazenamento diferentes.
- Conteúdo dos armários pode ser modificado utilizando seu nome ou endereço.



Armários da BC

- 1 Considere os armários A e B (que, na linguagem C, são diferentes dos armários a e b).
- 2 Atribua 20 para o armário A e 30 para o armário B .
- 3 Some A com B e coloque o resultado em A .

Armários da BC

Traduzindo para a linguagem C:

```
int main() {  
    int A,B;  
    A = 20;  
    B = 30;  
    A = A + B;  
    printf("%d\n",A); /* imprime na tela o resultado. */  
}
```

Armários da BC

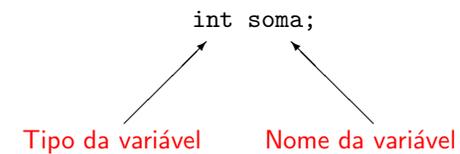
Podemos considerar que cada armário é uma variável. Escrevendo um algoritmo de forma mais elegante:

- 1 Sejam A e B variáveis do tipo inteiro.
- 2 Faça $A \leftarrow 20$ e $B \leftarrow 30$.
- 3 Faça $A \leftarrow A + B$.

Declarando uma variável

`int soma;`

Tipo da variável Nome da variável



Variáveis inteiras

- Variáveis utilizadas para armazenar valores inteiros, em formato binário.
Ex: $13_{10} = 1101_2$
- `int`: Inteiro cujo comprimento depende do computador. É o inteiro mais utilizado. Em computadores *Pentium*, ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647.
- `unsigned int`: Inteiro cujo comprimento depende do computador e que armazena somente valores positivos. Em computadores *Pentium*, ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.

Variáveis inteiras

- `long int`: Inteiro que ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647, independente do computador.
- `unsigned long int`: Inteiro que ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295, independente do computador.
- `short int`: Inteiro que ocupa 16 bits e pode armazenar valores de -32.768 a 32.767.
- `unsigned short int`: Inteiro que ocupa 16 bits e pode armazenar valores de 0 a 65.535.

Variáveis de tipo caracter

- Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos.
- São, na verdade, variáveis inteiras que armazenam um número associado ao símbolo. A principal tabela de símbolos utilizada pelos computadores é a tabela ASCII (*American Standard Code for Information Interchange*), mas existem outras (EBCDIC, Unicode, etc ..).
- `char`: Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de -128 a 127.
- `unsigned char`: Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de 0 a 255.

Variáveis de tipo ponto flutuante

- Armazenam valores reais, da seguinte forma

$$(-1)^{\text{signal}} \cdot \text{mantissa} \cdot 2^{\text{expoente}}$$

Ex: $0.5 = (-1)^0 \cdot 1 \cdot 2^{-1}$

- Para o programador, funciona como se ele armazenasse números na forma decimal.
- Possuem problemas de precisão (arredondamento).
- `float`: Utiliza 32 bits, sendo 1 para o sinal, 8 para o expoente e 23 para a mantissa. Pode armazenar valores de $(+/-)10^{-38}$ a $(+/-)10^{38}$
- `double`: Utiliza 64 bits, sendo 1 para o sinal, 11 para o expoente e 52 para a mantissa. Pode armazenar valores de $(+/-)10^{-308}$ a $(+/-)10^{308}$

O endereço de uma variável

- Toda variável tem um endereço de memória associado a ela. Esse endereço é o local onde essa variável é armazenada no sistema (como se fosse o endereço de um armário na BC).
- Normalmente, o endereço das variáveis não são conhecidos quando o programa é escrito.
- O endereço de uma variável é dependente do sistema computacional e também da implementação do compilador C que está sendo usado.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

Variáveis que guardam endereços

- Armazenam o endereço de outras variáveis.
- Para cada tipo de dados, existe um tipo para guardar o seu endereço, indicado por * antes do nome da variável.
- `int *endereço`: Endereço de uma variável inteira.
- `float *endereço`: Endereço de uma variável de ponto flutuante.
- `char *endereço`: Endereço de uma variável de carácter.
- Estas variáveis são chamadas **apontadores**.

Obtendo o tamanho de um tipo

O comando `sizeof(tipo)` retorna o tamanho, em bytes, de um determinado tipo. (Um byte corresponde a 8 bits).

Exemplo

```
printf ("%d", sizeof(int));  
Escreve 4 na tela (Pentium).
```

Regras para nomes de variáveis em C

- **Deve** começar com uma letra (maiuscula ou minuscula) ou subscrito(_). **Nunca** pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, números e subscrito.
- Não pode-se utilizar como parte do nome de uma variável:
{ (+ - * / \ ; . , ?

Regras para nomes de variáveis em C

As seguintes palavras já tem um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

```
auto      double   int       struct   break
enum      register typedef  char     extern
return    union     const     float    short
unsigned  continue for       signed   void
default   goto     sizeof    volatile do
if        static   while
```

Constantes

- Constantes são valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa (veremos adiante onde elas podem ser usadas).
- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo `string`, que corresponde a uma sequência de caracteres.
- Exemplos de constantes:

```
85, 0.10, 'c', "Hello, world!"
```

Constantes inteiras

- Um número na forma decimal, como escrito normalmente
Ex: 10, 145, 1000000
- Um número na forma hexadecimal (base 16), precedido de 0x
Ex: 0xA ($0xA_{16} = 10_{10}$), 0x100 ($0x100_{16} = 256_{10}$)
- Um número na forma octal (base 8), precedido de 0
Ex: 010 ($010_8 = 8_{10}$)

Constantes do tipo de ponto flutuante

- Um número decimal. Para a linguagem C, um número só pode ser considerado um número decimal se tiver uma parte “não inteira”, mesmo que essa parte não inteira tenha valor zero. Utilizamos o ponto para separarmos a parte inteira da parte “não inteira”.
Ex: 10.0, 5.2, 3569.22565845
- Um número inteiro ou decimal seguido da letra `e` e um expoente. Um número escrito dessa forma deve ser interpretado como:

$$\text{numero} \cdot 10^{\text{expoente}}$$

```
Ex: 2e2 (2e2 = 2 · 102 = 200.0)
```

Constantes do tipo caracter

- Uma constante do tipo caracter é sempre representado por uma letra entre aspas simples.
Ex: 'A'
- Toda constante do tipo caracter pode ser usada como uma constante do tipo inteiro. Nesse caso, o valor atribuído será o valor daquela letra na tabela ASCII.

Constantes do tipo string

- Uma constante do tipo string é um texto entre aspas duplas
Ex: "Hello, world!"

Constantes: Tabela ASCII

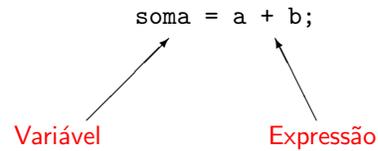
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Caracteres de Controle															
16																
32		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[/]	^	_
96	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	-	}	~	

Atribuição

Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor para uma determinada variável.

Atribuição

No exemplo abaixo, a variável soma recebe o valor calculado da expressão a + b



Atribuição

- O operador de atribuição é o sinal de igual (=)

À esquerda do operador de atribuição deve existir somente o nome de uma **variável**.

=

À direita, deve haver uma **expressão** cujo valor será calculado e armazenado na variável

Expressões Simples

- Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou em qualquer outro lugar onde uma expressão seja necessária).
Ex: a = 10;
- Uma variável é uma expressão.
Ex: a = b;
- O endereço de uma variável também é uma expressão, e é obtido colocando-se o símbolo & antes do nome da variável.
Ex: endereco = &a;

Exemplos de atribuição

```
int a,b;  
float f,g;  
char h;  
int *endereco;
```

```
a = 10;  
b = -15;  
f = 10.0;  
h = 'A';
```

```
a = b;  
f = a;
```

```
endereco = &a;
```

Estrutura básica de um programa em C

```
// Diretivas do pré-processador
#include <stdio.h>
#define MAX 100

// Declarações de variáveis globais
int a,b;

// Declarações de funções
float calcula_raiz_quadrada(int n) {
    // Declarações de variáveis locais
    ...
    // Código para calcular raiz quadrada
    ...
    return valor_calculado
}

main() {
    // Declarações de variáveis locais
    int n;
    float resultado;

    // Código do programa principal
    printf("Digite um número inteiro para calcular sua raiz quadrada: ");
    scanf("%d", &n);
    resultado = calcula_raiz_quadrada(n);
    printf("A raiz quadrada de %d é %f", n, resultado);

    return 0;
}
```

Exercício

```
#include <stdio.h>

main() {
    int primeiro, segundo, auxiliar;

    printf("Digite o primeiro número inteiro: ");
    scanf("%d", &primeiro);
    printf("Digite o segundo número inteiro: ");
    scanf("%d", &segundo);
    auxiliar = primeiro;
    primeiro = segundo;
    segundo = auxiliar;

    printf("primeiro=%d\n", primeiro);
    printf("segundo=%d\n", segundo);

    return 0;
}
```

Exercício

Como fazer a mesma coisa que o algoritmo anterior faz, mas usando apenas duas variáveis?