

MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2013

Roteiro

- 1 Variáveis
- 2 Atribuição
- 3 Estrutura Básica de um Programa em C
- 4 Algumas Informações Extras

Definição

Variáveis são locais da memória onde são armazenados valores.

- Uma variável é caracterizada por dois atributos:
 - ▶ um nome que identifica a variável em um programa.
 - ▶ um tipo que determina o que pode ser armazenado naquela variável.

Declarando uma variável

Uma variável é declarada da seguinte forma:

```
Tipo_Variável Nome_Variável;
```

Exemplos válidos:

- `int soma;`
- `float preco_abacaxi;`
- `char resposta;`

Exemplos inválidos:

- `soma int;`
- `float preco_abacaxi;`

Variáveis inteiras

Variáveis utilizadas para armazenar valores inteiros.

Exemplos: 13 ou 1102 ou 24.

Os seguintes tipos da linguagem C servem para armazenar números inteiros, sendo que os valores armazenáveis em cada tipo variam de acordo com o tipo do computador e o sistema operacional (os valores exibidos correspondem a máquinas 64 bits executando Linux ou Mac OS):

- `int`: ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647.
- `unsigned int`: ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.

Variáveis inteiras

- `short int`: ocupa 16 bits e pode armazenar valores de -32.768 a 32.767 .
- `unsigned short int`: ocupa 16 bits e pode armazenar valores de 0 a 65.535 .
- `long int`: ocupa 64 bits e pode armazenar valores entre $-2^{-63} \cong -9.2 \times 10^{18}$ e $2^{63} - 1 \cong 9.2 \times 10^{18}$.
- `unsigned long int`: ocupa 64 bits e pode armazenar valores entre 0 e $2^{-64} - 1 \cong 1.8 \times 10^{19}$.

Variáveis inteiras

Exemplos de declaração de variáveis inteiras:

- `int numVoltas;`
- `int ano;`
- `unsigned int quantidadeChapeus;`

Exemplos inválidos:

- `int int numVoltas;`
- `unsigned int ano;`

Variáveis inteiras

Pode-se declarar diversas variáveis de um mesmo tipo, basta separar cada uma delas por uma vírgula:

- `int numVoltas, ano;`
- `unsigned int a, b, c, d;`

Variáveis de tipo caractere

Variáveis utilizadas para armazenar letras, números e outros símbolos existentes em textos. Esse tipo de variável guarda apenas um caractere.

Exemplos: 'A', 'c', '7' ou '*'.

Exemplos de declaração:

- `char umaLetra;`
- `char VouF;`

Variáveis de tipo ponto flutuante

Armazenam valores reais, da seguinte forma:

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

Mantissa é parte fracionária da representação numérica, escrita em binário como $1.b_1b_2b_3 \dots b_n$, onde n é o número de bits utilizados.

Ex: $0.15625 = (-1)^0 \times 1.25 \times 2^{-3}$, onde $1.25 = 1.01$ na base binária.

- Variáveis de ponto flutuante possuem problemas de precisão numérica, pois há uma quantidade limitada de memória para armazenar um número real.

Variáveis de tipo ponto flutuante

- `float`: utiliza 32 bits, sendo 1 bit para o sinal, 8 bits para o expoente e 23 bits para a mantissa. Note que com os 8 bits do expoente conseguimos representar 256 números inteiros, no entanto, os valores 00000000 e 11111111 são reservados, respectivamente, para zero e infinito. Assim, os valores do expoente variam entre -126 e 127 . Logo, o menor número (em módulo) representável é $1 \times 2^{-126} \cong 1.7 \times 10^{-38}$, assim como o maior número é $2 \times 2^{127} \cong 3.4 \times 10^{38}$.
- `double`: utiliza 64 bits, sendo 1 para o sinal, 11 para o expoente e 52 para a mantissa. Note que com os 11 bits do expoente conseguimos representar 2048 números inteiros, no entanto, os valores 00000000 e 11111111 são reservados para zero e infinito, respectivamente. Assim, os valores do expoente variam entre -1022 e 1023 . Logo, o menor número (em módulo) representável é $1 \times 2^{-1022} \cong 2.2 \times 10^{-308}$, assim como o maior número é $2 \times 2^{1023} \cong 1.8 \times 10^{308}$.
- A precisão numérica de variáveis do tipo `float` é de 7 casas decimais, enquanto do tipo `double` é de 15 casas decimais.

Variáveis de tipo ponto flutuante

Exemplos de declaração de variáveis de tipo ponto flutuante.

- `float salario;`
- `float resultado, cotacaoDolar;`
- `double a, b, c;`

Regras para nomes de variáveis em C

- Deve começar com uma letra (maíuscula ou minúscula) ou subscrito(_). Nunca pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, números e subscrito.
- Os seguintes caracteres não podem ser utilizados como parte do nome de uma variável:

{ (+ - * / \ ; . , ?

- Letras maiúsculas e minúsculas são diferentes:

```
int c;  
int C;
```

Regras para nomes de variáveis em C

As seguintes palavras, conhecidas como *palavras reservadas*, já têm um significado na linguagem C e, por esse motivo, não podem ser utilizadas como nome de variáveis:

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

Comando de atribuição

Definição

O comando de atribuição serve para atribuir valores para variáveis.

- O comando de atribuição em C é o sinal =
- A sintaxe do uso do comando é:

variável = valor;

- Exemplos:

```
int a;  
float c;  
a = 5;  
c = 67.89505456;
```

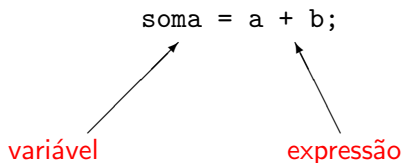
Comando de atribuição

- O comando de atribuição pode conter expressões do lado direito:
 $\text{variável} = \text{expressão};$
- Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor em uma determinada variável.
- Exemplos:

```
int a;  
float c;  
a = 5 + 5 + 10;  
c = 67.89505456 + 8 - 9;
```


Comando de 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

Atribuição e Constantes

Constantes são valores previamente determinados e que não se alteram ao longo do programa.

- 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!"

Atribuição e Constantes

- Uma *constante inteira* é um número na forma decimal, como escrito normalmente
Ex: 10, 145, 1000000
- Uma *constante ponto flutuante* é um número real, em que a parte fracionária é precedida por um ponto.
Ex: 2.3456, 32132131.5, 5.0
- Uma *constante do tipo caractere* é sempre representado por uma letra entre apóstrofos (aspas simples).
Ex: 'A'
- Uma *constante do tipo string* é um texto entre aspas duplas.
Ex: "Hello, world!"

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).

- Ex1:

```
int a;  
a = 10;
```

- Ex2:

```
char b;  
b = 'F';
```

- Ex3:

```
double c;  
c = 3.141592;
```

Expressões simples

Uma variável também é uma expressão e pode ser atribuída a outra variável.

Ex:

```
int a, b;  
a = 5;  
b = a;
```

Exemplos de atribuição

- Sempre antes de usar uma variável, esta deve ter sido declarada.

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

```
a = 10;  
b = -15;  
f = 10.0;  
g = 12.6;  
h = 'A';
```

```
a = b;  
f = a;  
a = b + f + g;
```

Exemplos incorretos de atribuição

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

```
a b = 10;  
b = -15  
d = 90;
```


Estrutura básica de um programa em C

A estrutura básica é a seguinte:

Declaração de bibliotecas usadas

```
int main() {  
    Declaração de variáveis  
  
    Comandos  
    ...  
    Comandos  
  
    return 0;  
}
```

Estrutura básica de um programa em C

Exemplo:

```
#include <stdio.h>

int main() {
    int a, b, c;

    a = 7 + 9;
    b = a + 10;
    c = b - a;

    printf("%d\n", c);

    return 0;
}
```

Informações extras: constantes inteiras

- Um número na forma decimal é escrito normalmente:
Ex: 10, 145, 1000000
- Um número na forma hexadecimal (base 16) é precedido de 0x:
Ex: 0xA ($0xA_{16} = 10$), 0x100 ($0x100_{16} = 256$)
- Um número na forma octal (base 8) é precedido de 0:
Ex: 010 ($0x10_8 = 8$)

Informações extras: 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 esta parte não inteira tenha valor zero. Utiliza-se o ponto para separar 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:

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

Ex: 2e2 ($2e2 = 2 \cdot 10^2 = 200.0$)

Informações extras: caractere

- Um caractere é armazenado como um valor inteiro. A tabela padrão de símbolos utilizada pelos computadores é a tabela ASCII (*American Standard Code for Information Interchange*), entretanto, há outras (EBCDIC, Unicode, etc).
- `char`: armazena um valor inteiro entre -128 a 127, sendo que os valores de 0 a 127 são associados a símbolos da tabela ASCII.
- Toda constante do tipo caractere pode ser usada como uma constante do tipo inteiro. Nesse caso, o valor atribuído será o valor daquele caractere na tabela ASCII.

Informações extras: tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 16	Caracteres de Controle															
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	{	—	}	~	

Informações extras: obtendo o tamanho de um tipo

O comando `sizeof(tipo)` retorna o tamanho, em bytes, de um determinado tipo.

Exemplo:

```
printf("%lu\n", sizeof(int));
```