

MC-102 — Aula 03

Variáveis, Atribuições e Estrutura Básica de um Programa

Instituto de Computação – Unicamp

Primeiro Semestre de 2012

Roteiro

1 Variáveis

2 Atribuição

3 Estrutura de um Programa em C

Variáveis

Definição

Variáveis são locais onde armazenamos valores. 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.

- Durante a execução do programa, um pedacinho da memória corresponde a variável.

Declarando uma variável

Declara-se da seguinte forma: **Tipo_Variável Nome_Variável;**

Exemplos corretos:

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

Exemplos incorretos:

- `soma int;`
- `float preco_abacaxi`

Variáveis inteiras

Variáveis utilizadas para armazenar valores inteiros. Ex: 13 ou 1102 ou 24.

Abaixo temos os tipos da linguagem C que servem para armazenar inteiros:

- **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 inteiras

Exemplos de declaração de variáveis inteiras:

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

Exemplos Inválidos:

- `int int numVoltas;`
- `unsgned int ano;`

Variáveis inteiras

Você pode declarar várias variáveis de um mesmo tipo. Basta separar as variáveis por vírgula:

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

Variáveis de tipo caracter

Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos. OBS: Guarda apenas um caracter.

Exemplos de declaração:

- `char umaLetra;`
- `char YOuN;`

Variáveis de tipo ponto flutuante

Armazenam valores reais, da seguinte forma

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

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

- Possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real.
- **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 $(+/-)10^{308}$

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 subcrito(_). **Nunca** pode começar com um número.
- Pode conter letras maíusculas, minúsculas, números e subcrito.
- Não pode-se utilizar como parte do nome de uma variável:

{ (+ - * / \ ; . , ?

- Letras maiúsculas e minúsculas são diferentes (**case sensitive**):

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

Regras para nomes de variáveis em C

As seguintes palavras (**palavras reservadas**) 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		

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:

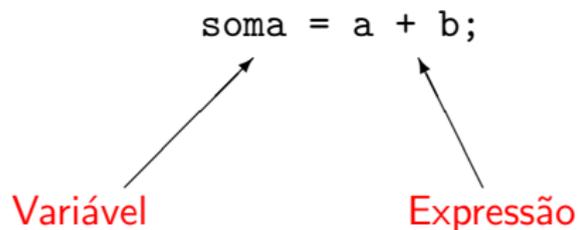
variável = expressão ;

- Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor para 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 por algum motivo, devem aparecer dentro de um programa. Ex: $\pi = 3,141592$

- 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, onde a parte fracionário vem depois de um ponto
Ex: 2.3456, 32132131.5, 5.0
- Uma **constante do tipo character** é sempre representado por uma letra entre 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:

```
const int a = 10;  
int b;  
b = a;
```

- Ex2:

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

- Ex3:

```
const double pi = 3.141592;  
double d;  
d = pi+2;
```

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

- **OBS:** 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;  
h = 'A';
```

```
a = b;  
f = a;  
a = (b+f+a);
```

Exemplos errados 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

Declaração de variáveis

```
int main(){
```

Declaração de variáveis

Comandos

.
. .
.

Comandos

}

Estrutura Básica de um Programa em C

Exemplo:

```
#include <stdio.h>
```

```
int main(){
```

```
    int a;
```

```
    int b,c;
```

```
    a = 7+9;
```

```
    b = a+10;
```

```
    c = b-a;
```

```
}
```

Informações Extras: Caracter

- 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 Interchang*), 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.
- 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.

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. (Um byte corresponde a 8 bits).

Exemplo

```
printf ("%d", sizeof(int));
```

Escreve 4 na tela (*Pentium*).

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;
}
```