MC-102 — Aula 02
Variáveis, Atribuições e Estrutura Básica de um Programa

Eduardo C. Xavier

Instituto de Computação – Unicamp

2017

Roteiro

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 à 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 processador. É o inteiro mais utilizado. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647.
- **unsigned int:** Inteiro cujo comprimento depende do processador e que armazena somente valores positivos. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.

Variáveis inteiras

- **long int:** Inteiro que ocupa 64 bits em computadores Intel de 64bits, e pode armazenar valores de aprox. -9×10^{18} a aprox. 9×10^{18}.
- **unsigned long int:** Inteiro que ocupa 64 bits em computadores Intel de 64bits, e armazena valores de 0 até aprox. 18×10^{18}.
- **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 de tipo caractere

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

Exemplos de declaração:

- char umaLetra;
- char YorN;

Variáveis de tipo ponto flutuante

Armazenam valores reais. Mas possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real. Exemplos de números em ponto flutuante: 2.1345 ou 9098.123.

- **float:** Utiliza 32 bits, e na prática tem precisão de aproximadamente 6 casas decimais (depois do ponto).
  Pode armazenar valores de $(+/-)10^{-38}$ a $(+/-)10^{38}$
- **double:** Utiliza 64 bits, e na prática tem precisão de aproximadamente 15 casas decimais.
  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;
- double cotaDolar;

Declarando Várias Variáveis

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

- int numVoltas, ano;
- double x1, x2, x3, x4;

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 subscrito.
- Não pode-se utilizar como parte do nome de uma variável:

$$\{ \quad (\quad + \quad - \quad * \quad / \quad \backslash \quad ; \quad . \quad , \quad ?$$

- Letras maiúsculas e minúsculas são diferentes:

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

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`		

Comando de Atribuição

Definição

O comando de atribuição serve para atribuir valores para variáveis.

- 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 para uma variável significa calcular o valor daquela expressão e copiar aquele valor para a variável.

Comando de Atribuição

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

```
soma = a + b;
```

Variável

Expressão

Comando de Atribuição

- Exemplos:

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

Atribuição

- O sinal de igual no comando de atribuição é chamado de **operador de atribuição**.
- Veremos outros operadores mais adiante.

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.

Variáveis e Constantes

Constantes são valores previamente determinados e que por algum motivo, devem aparecer dentro de um 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!"
```

Variáveis e Constantes

- Uma **constante inteira** é um número inteiro como escrito normalmente
  Ex: `10`, `145`, `1000000`
- Uma **constante ponto flutuante** é um número real, onde a parte fracionária vem depois de um ponto
  Ex: `2.3456`, `32132131.5`, `5.0`
- Uma **constante do tipo caractere** é sempre representada por um caractere (letra, dígito, pontuação, etc.) entre aspas simples.
  Ex: `'A'`, `'!'`, `'4'`, `'('`
- 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 ser usada em qualquer outro lugar onde uma expressão seja válida).

- 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;
```

Expressões Simples

```
int a, b;
a = 5;
b = a + 2;
a = a + 10;
```

Quais os valores finais nas variáveis **a** e **b**?

Exemplos de atribuição

- **OBS:** A declaração de uma variável sempre deve ocorrer antes de seu uso.

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

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

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

Qual o valor final na variável **a**?

Exemplos errados de atribuição

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

a b = 10; //Errado! Por quê?
b = -15
d = 90;   //Errado! Por quê?
```

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;

Comando;
.
.
.
Comando;
}
```

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;
}
```

Exercício

Qual o valor armazenado na variável **a** no fim do programa?

```
int main(void){
    int a, b, c, d;

    d = 3;
    c = 2;
    b = 4;
    d = c + b;
    a = d + 1;
    a = a + 1;

}
```

Exercício

Compile o programa abaixo? Você sabe dizer qual erro existe neste programa?

```
int main(void){
    int a, b;
    double c,d;
    int g;

    d = 3.0;
    c = 2.4142;
    b = 4;
    a = 5*b;
    d = b + 90;
    e = c * d;
    a = a + 1;

}
```

Informações Extras: Constantes Inteiras

Possíveis formas de escrita em C de um número inteiro:

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

Informações Extras: Constantes do tipo de ponto flutuante

- Um número só pode ser considerado um número ponto flutuante se tiver uma parte "não inteira", mesmo que essa parte não inteira tenha valor zero.
  Ex: `10.0`, `5.0`
- Um número ponto flutuante pode ser escrito em notação científica também: deve conter um número seguido da letra **e** mais um expoente. Um número escrito dessa forma deve ser interpretado como:

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

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

Informações Extras: caractere

- 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.

Informações Extras: Tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 16	caracterees 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: caractere

- Toda constante do tipo caractere pode ser usada como uma constante do tipo inteiro e vice-versa. Nesse caso, o valor atribuído será o valor daquela letra na tabela ASCII.

```
int a;
char b;

b = 65;
printf("valor de b: %c\n", b);

a = 'C';
printf("valor de a: %d\n", a);
```

Comentários

- É comum incluir comentários em um programa para explicar certas partes do código de tal forma que ajude a sua compreensão.
- Os comentários de um programa são ignorados pelo compilador.
- Em C um comentário de uma linha deve ser escrito depois de //.
- Em C um comentário também pode ser escrito entre /* e */. Neste caso pode haver mais de uma linha de comentário.

```c
#include <stdio.h>

int main(){
  int a; //isto eh um comentario

  int b; /* isto
            tambem eh
            um comentario*/

  a = 2;
  b = 3;
  a = a + b;
}
```