MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2015

Roteiro

Variáveis

Definição

Variáveis são porções de memória usadas para armazenar valores.

- Uma variável é caracterizada por dois atributos:
  - um nome que identifica a variável em um programa.
  - um tipo que determina os valores que podem ser armazenados naquela variável.

Declarando uma variável

- Uma variável é declarada da seguinte forma:

```
tipo_da_variável nome_da_variável;
```

- Exemplos válidos:
  - `int soma;`
  - `float preco_abacaxi;`
  - `char resposta;`
- Exemplos inválidos:
  - `soma int;`
  - `float int;`

Variáveis de tipo inteiro

- Variáveis utilizadas para armazenar valores inteiros.
  - Exemplos: `1102, -13, 24`.
- Os seguintes tipos da linguagem C servem para declarar variáveis para armazenar números inteiros, sendo que os valores armazenáveis variam de acordo com o tipo do computador e do sistema operacional.
- Os valores exibidos correspondem a máquinas de 64 bits executando Linux ou Mac OS.

Variáveis de tipo inteiro

- `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.
- `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} ($\approx -9.2 \times 10^{18}$) e $2^{63} - 1$ ($\approx 9.2 \times 10^{18}$).
- `unsigned long int`: ocupa 64 bits e pode armazenar valores entre 0 e $2^{64} - 1$ ($\approx 1.8 \times 10^{19}$).

Variáveis de tipo inteiro

- Exemplos válidos de declaração de variáveis inteiras:
  - `int numVoltas;`
  - `int ano;`
  - `unsigned int quantidadeChapeus;`
- Exemplos inválidos de declaração de variáveis inteiras:
  - `int int numVoltas;`
  - `unsigned int;`

Variáveis de tipo inteiro

- Podemos declarar diversas variáveis de um mesmo tipo numa única linha de código, basta separar cada uma delas por uma vírgula:
  - `int numVoltas, ano;`
  - `unsigned int a, b, c, d;`
  - `long int x, y, z;`

Variáveis de tipo caractere

- Variáveis utilizadas para armazenar letras, dígitos e outros símbolos existentes em textos. Esse tipo de variável armazena apenas um caractere.
  - Exemplos: `'A'`, `'c'`, `'7'`, `'*'`.
- Exemplos de declaração:
  - `char letra;`
  - `char VouF;`

Variáveis de tipo ponto flutuante

- Armazenam valores reais, da seguinte forma:

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

- Mantissa é parte fracionária da representação numérica, escrita em binário como $1.b_1 b_2 b_3 \ldots b_n$, onde n é o número de bits utilizados.
  - Exemplo: $0.15625 = (-1)^0 \times 1.25 \times 2^{-3}$, onde 1.25 é representado por 1.01 na base binária.
- Variáveis de ponto flutuante podem apresentar 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 2^{-126} ($\approxeq 1.7 \times 10^{-38}$) e o maior número é 2×2^{127} ($\approxeq 3.4 \times 10^{38}$).
- `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 00000000000 e 11111111111 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 1×2^{-1022} ($\approxeq 2.2 \times 10^{-308}$) e o maior número é 2×2^{1023} ($\approxeq 1.8 \times 10^{308}$).
- 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 válidos de declaração de variáveis de ponto flutuante:
  - `float salario;`
  - `float resultado, cotacaoDolar;`
  - `double a, b, c;`
- Exemplos inválidos de declaração de variáveis de ponto flutuante:
  - `unsigned float teste;`
  - `double float;`

Regras para nomes de variáveis em C

- Nomes de variáveis devem começar com uma letra (maiúscula ou minúscula) ou um subscrito (_).
- Nomes de variáveis não podem começar com um número ou com qualquer outro símbolo além daqueles mencionados acima.
- Nomes de variáveis podem conter letras maiúsculas, letras minúsculas, números e subscrito.
  - Exemplos:

```
int digito1, digito_2, pr1m31r0V4l0r, Ultimo_Valor;
double codigo171, _temp, form_DS160, formDS117;
```

- A linguagem C diferencia letras minúsculas e maiúsculas.
  - Exemplos:

```
int c, C;
double numero, Numero, NUMERO;
```

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 símbolo "=" (igual).
- A sintaxe do comando é:

```
nome_da_variável = valor;
```

- Exemplo:

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

Comando de atribuição

- O comando de atribuição pode envolver expressões:

```
nome_da_variável = expressão;
```

- À esquerda do operador de atribuição (=) deve existir o nome de uma variável.
- À direita do operador de atribuição (=) deve existir um valor ou uma expressão cujo valor será calculado e armazenado na variável.
- Atribuir o valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor em uma determinada posição de memória (variável).
- Exemplo:

```
int a;
float c;
a = 5 + 5 + 10;
c = 67.89505456 + 18 - 9;
```

Constantes

Definição

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

Constantes

- Uma constante inteira é um número na forma decimal.
  - Exemplos: `10`, `-145`, `2015`.
- Uma constante do tipo ponto flutuante é um número real, em que a parte fracionária é precedida por um ponto.
  - Exemplos: `2.3456`, `32132131.5`, `-5.0`.
- Uma constante do tipo caractere é sempre representada por um caractere entre apóstrofos (aspas simples).
  - Exemplos: `'A'`, `'?'`, `'2'`.
- Uma constante do tipo string é um texto entre aspas duplas.
  - Exemplos: `"Hello, world!"`, `"Uma frase qualquer"`.

Definindo constantes

- Podemos definir uma constante usando o seguinte comando:

```
#define CONSTANTE valor
```

- Exemplos:

```
#define PI 3.14159
#define MESES_POR_ANO 12
#define DIAS_DA_SEMANA 7
#define TAMANHO 100
#define PRIMEIRA_LETRA 'A'
```

Definindo constantes

- Constantes devem ser definidas logo no começo do programa, após as declarações das bibliotecas, de forma que elas possam ser usadas em qualquer ponto do programa.
- O tipo da constante é implicitamente definido pelo tipo do seu valor. Por exemplo, `PI` é uma constante de tipo flutuante, enquanto `MESES` é uma constante inteira.
- Ao se definir uma constante (usando `#define`) não é alocado um espaço de memória para ela. O compilador simplesmente substitui o valor da constante em todo lugar que ela é usada no programa.

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 permitida.
- Exemplo:

```
int ano, bienio;
ano = MESES_POR_ANO;
bienio = 2 * MESES_POR_ANO;
```

- Exemplo:

```
char letra1, letra2;
letra1 = 'F';
letra2 = PRIMEIRA_LETRA;
```

- Exemplo:

```
double raio, circunferencia;
r = 2.25;
c = 2 * PI * r;
```

Expressões simples

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

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

Exemplos corretos de atribuição

- Exemplo:

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

a = 10;
b = -15;
f = 8.1;
g = 12.6;
h = 'A';

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

- Qual o valor da variável "a" após a última atribuição? Resposta: -17

Exemplos incorretos de atribuição

- Exemplo:

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

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

- Sempre antes de usar uma variável, esta deve ter sido declarada.
- Note que não é possível afirmar com certeza o valor da variável `f` após a execução deste trecho de programa, já que a variável `g` não foi inicializada.

Declarando e inicializando variáveis

- Uma variável é dita não inicializada se ela ainda não recebeu um valor, desde que ela foi declarada.
- Note que mesmo variáveis não inicializadas possuem um valor: aquele previamente armazenado na posição de memória onde a variável foi alocada.
- É possível declarar e inicializar uma ou mais variáveis numa única linha de código.
- Exemplos:
  - `int dia = 30, mes = 6, ano = 2012;`
  - `double parcial, total = 0.0;`
  - `float juros, saldo = 0.00, limite = 1000.00;`

Estrutura básica de um programa em C

```
declaração de bibliotecas

definição de constantes

int main() {
  declaração de variáveis;

  comando;
  comando;

  ...

  comando;

  return 0;
}
```

Estrutura básica de um programa em C

```
#include <stdio.h>

#define XXX 3

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

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

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

  return 0;
}
```

Constantes do tipo inteiro

- Um número na forma decimal é escrito normalmente:
  - Exemplos: `1234`, `-13`, `42`.
- Um número na forma hexadecimal (base 16) é precedido de `0x`:
  - Exemplos: `0xA` ($A_{16} = 10_{10}$), `0x100` ($100_{16} = 256_{10}$)
- Um número na forma octal (base 8) é precedido de `0`:
  - Exemplos: `010` ($10_8 = 8_{10}$), `03737` ($3737_8 = 2015_{10}$)

Constantes do tipo ponto flutuante

- Uma constante numérica é considerada do tipo ponto flutuante se possuir uma parte decimal, mesmo que esta parte tenha valor zero. Utiliza-se o ponto para separar a parte inteira da parte decimal.
  - Exemplos: `10.0`, `5.25`, `3569.22565845`, `2.`, `.3`
- Outra forma de definir uma constate do tipo ponto flutuante é usando um número (inteiro ou decimal) seguido da letra 'e' e de outro número (inteiro ou decimal). Uma constante escrita dessa forma deve ser interpretada da sequinte forma:

$$\text{mantissa} \times 10^{expoente}$$

  - Exemplos: `2e2` ($= 2 \times 10^2 = 200.0$), `1.7e-3` ($= 1.7 \times 10^{-3} = 0.0017$)

Constantes do tipo 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 tabelas similares (como, por exemplo, EBCDIC, *Extended Binary Coded Decimal Interchange Code*).
- Uma varíavel do tipo `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 usado será o correspondente ao caractere na tabela ASCII.
  - Exemplo:

```
int x;
x = 'A' + 5;
```

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	{	—	}	~	

Obtendo o tamanho de um tipo

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

```
#include <stdio.h>

int main() {

  printf("%lu\n", sizeof(int));
  printf("%lu\n", sizeof(char));
  printf("%lu\n", sizeof(double));
  printf("%lu\n", sizeof(float));

  return 0;
}
```