

MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2014

Roteiro

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

Variáveis

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 int;`

Variáveis de tipo inteiro

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 de tipo inteiro

- `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 de tipo inteiro

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

Variáveis de tipo inteiro

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, dígitos 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 (maiúscula 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:

{ (+ - * / \ ; . , ?

- A linguagem C diferencia letras minúsculas e maiúsculas:

```
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 símbolo “=” (igual).
- A sintaxe 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;`

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.

- 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).
- Exemplos:

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


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!"

Definindo constantes

- Podemos definir uma constante usando o seguinte comando:

```
#define CONSTANTE valor
```

- Exemplos:

```
#define PI 3.14159
```

```
#define MESES 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 substituiu 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 em qualquer outro lugar onde uma expressão seja necessária).

- Ex.:

```
int a, b;  
a = MESES;  
b = 10;
```

- Ex.:

```
char c, d;  
c = 'F';  
d = PRIMEIRA_LETRA;
```

- Ex.:

```
double e, f;  
e = 2.71828;  
f = PI;
```

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;
```

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

Exemplos incorretos de atribuição

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

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

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.

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

Exemplo:

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

```
#define XXX 3
```

```
int main() {
```

```
    int a, b, c;
```

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

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

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

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

```
    return 0;
```

```
}
```

Constantes do tipo inteiro

- 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 ($A_{16} = 10_{10}$), 0x100 ($100_{16} = 256_{10}$)
- Um número na forma octal (base 8) é precedido de 0:
Ex.: 010 ($10_8 = 8_{10}$)

Constantes do tipo 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:

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

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

Caracteres

- 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 EBCDIC).
- `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.

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	{	—	}	~	

Descobrimo o tamanho de um tipo

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

Exemplo:

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