

Variáveis e Atribuições

Computação para Engenharia – Tópico 2

Daniel Guerreiro e Silva

Departamento de Engenharia Elétrica (ENE), Faculdade de Tecnologia (FT)

Roteiro

Variáveis

Principais tipos de variáveis em C++

Constantes literais

Atribuição

Expressões aritméticas

Conversão de tipos

Um exemplo em C++

Variáveis

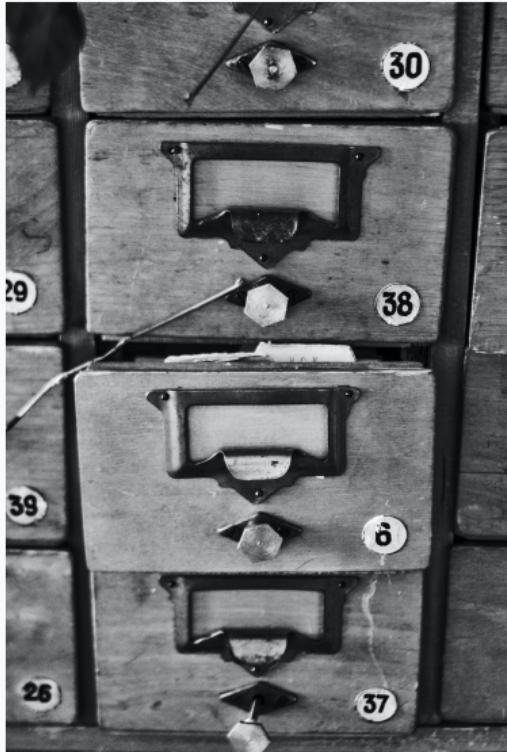
Variáveis

Definição

*Locais onde armazenamos valores na memória. 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.*

Gavetas de um armário

- Programar: estabelecer regras para manipulação de informações na memória principal do computador.
- Memória principal funciona como um conjunto de gavetas com nome, número (endereço) e capacidades de armazenamento diferentes.
- Conteúdo das gavetas pode ser modificado utilizando seu nome ou endereço.



Gavetas de um armário

1. Considere as gavetas **a** e **b**.
2. **Atribua** (guarde o valor) 20 para a gaveta **a** e 30 para a gaveta **b**.
3. Some **a** com **b** e coloque o resultado em **a**.

Gavetas de um armário

Podemos considerar que cada gaveta é uma variável. Escrevendo um algoritmo de forma mais elegante:

1. Sejam **a** e **b** variáveis do tipo inteiro.
2. Faça **a** $\leftarrow 20$ e **b** $\leftarrow 30$.
3. Faça **a** $\leftarrow \mathbf{a} + \mathbf{b}$.

Gavetas de um armário

Traduzindo para a linguagem C++:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     int a, b; /*declaracao das variaveis*/
7
8     /*o sinal = significa ATRIBUICAO, i.e. copiar 20 na variavel a*/
9     a = 20;
10    b = 30;
11
12    a = a + b; /*soma os valores e guarda resultado em a*/
13
14    cout << a << endl; /* imprime resultado no terminal/prompt */
15
16    return 0;
17 }
```

Declarando uma variável

```
int soma;
```

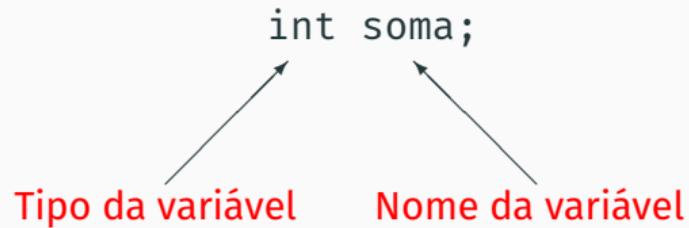
Declarando uma variável

```
int soma;  
↑  
Tipo da variável
```

Declarando uma variável

```
int soma;
```

Tipos da variável Nome da variável

A diagram illustrating the components of a variable declaration. The code 'int soma;' is centered. Two red arrows point from the text 'Tipo da variável' to the word 'int' and from the text 'Nome da variável' to the identifier 'soma'.

The diagram shows the declaration of a variable named 'soma' of type 'int'. The word 'int' is labeled 'Tipo da variável' (Type of variable) and the identifier 'soma' is labeled 'Nome da variável' (Name of variable).

Declarando uma variável

```
int soma;
```

Tipos da variável Nome da variável



Lembre-se: na memória, todos os dados são representados por bits, o tipo da variável indica **o tamanho (número de bits) e a interpretação** que se dá a esse dado.

Principais tipos de variáveis em C++

Variáveis inteiros

Variáveis utilizadas para armazenar valores inteiros, de forma binária.

Por exemplo: $(13)_{10} = (1101)_2$

- `int`: É o inteiro mais utilizado. Em geral ocupa 32 bits (4 bytes) na memória e nesse caso pode armazenar valores de -2.147.483.648 a 2.147.483.647.
- `short int`: Inteiro que ocupa em geral 16 bits (2 bytes) e nesse caso pode armazenar valores de -32.768 a 32.767.
- Há a opção de inteiro sem sinal através do prefixo `unsigned`:
 - `unsigned int`: pode armazenar valores de 0 a 4.294.967.295.
 - `unsigned short int`: pode armazenar valores de 0 a 65.535.

Alguns compiladores consideram apenas 2 bytes para o tipo `int`. Neste caso, o tipo `long` estende para 4 bytes.

Variáveis de tipo caractere

Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos. São, na verdade, variáveis inteiras que armazenam **um código associado ao símbolo que se queira representar**.

- `char`: Armazena um símbolo (no caso, o inteiro correspondente) de tamanho máximo 1 byte. Seu valor pode ir de -128 a 127.
- `unsigned char`: Armazena um símbolo (no caso, o inteiro correspondente) de tamanho máximo 1 byte. Seu valor pode ir de 0 a 255.

Variáveis de tipo caractere

A principal tabela de símbolos utilizada por computadores é a tabela ASCII (*American Standard Code for Information Interchange*).

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	'
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	-	127	7F	177	

Variáveis de tipo ponto flutuante

Armazenam valores reais, da seguinte forma

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

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

- Para o programador, funciona como se ele armazenasse números com casa decimal (Reais).
- O tamanho finito de armazenamento implica em problemas de precisão (arredondamento).
- `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 $(+/-)10^{308}$

O endereço de uma variável

- Toda variável tem um **endereço** de memória associado a ela, que é o local onde o seu conteúdo é armazenado na memória (como se fosse o local de uma gaveta no armário).
 - Este endereço pode ser necessário em algumas funções e operações.
- Normalmente, os endereços literais das variáveis não são conhecidos quando o programa está sendo escrito.
- O endereço de uma variável é dependente do sistema computacional, do compilador C++ que está sendo usado e pode mudar entre diferentes execuções do mesmo programa usando uma mesma máquina.

O endereço de uma variável

- Toda variável tem um **endereço** de memória associado a ela, que é o local onde o seu conteúdo é armazenado na memória (como se fosse o local de uma gaveta no armário).
 - Este endereço pode ser necessário em algumas funções e operações.
- Normalmente, os endereços literais das variáveis não são conhecidos quando o programa está sendo escrito.
- O endereço de uma variável é dependente do sistema computacional, do compilador C++ que está sendo usado e pode mudar entre diferentes execuções do mesmo programa usando uma mesma máquina.

Para acessar o endereço de uma variável de nome x basta escrever no código &x.

Variáveis que guardam endereços

Estas variáveis são chamadas **ponteiros** ou **apontadores**; elas armazenam o endereço de outras variáveis.

- Para cada tipo de dados, existe um tipo para guardar o seu endereço, indicado por * antes do nome da variável.
- `int *addr`: Endereço de uma variável inteira.
- `float *addr`: Endereço de uma variável de ponto flutuante.
- `char *addr`: Endereço de uma variável de caractere.

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.
- Uma variável de letra maiúscula é diferente de uma variável com a mesma letra minúscula.
- Não se pode utilizar como parte do nome de uma variável:

{ (+ - * / \ ; . , ?

Constantes literais

- Constantes literais são valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa (veremos adiante onde elas podem ser usadas).
- Assim como as variáveis, as constantes literais também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo que corresponde a uma sequência de caracteres.
- Exemplos de constantes:

85, 0.10, 'c', "Hello, world!"

Constantes literais inteiiras

- Um número inteiro na base dez, como escrito normalmente
Ex: 10, 145, 1000000
- Um número inteiro na base hexadecimal (base 16), precedido de ox
Ex: 0xA ($A_{16} = 10_{10}$), 0x100 ($100_{16} = 256_{10}$)

Constantes literais do tipo ponto flutuante

- Um número real também na base dez. Para a linguagem C++, um número só pode ser considerado um número real se tiver uma parte “não inteira”, mesmo que essa parte não inteira tenha valor zero. Utilizamos o ponto para separarmos a parte inteira da parte “não inteira”.

Ex: 10.0, 5.2, 3569.22565845

- Um número inteiro ou real seguido da letra **e** e um expoente. Um número escrito dessa forma deve ser interpretado como:

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

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

Constantes literais do tipo caractere

- Uma constante literal do tipo caractere é representada por uma letra (ou qualquer símbolo do teclado) **entre aspas simples**.
Ex: 'A'
- Toda constante literal do tipo caractere pode ser usada como uma constante literal do tipo inteiro. Nesse caso, o valor atribuído será o código correspondente ao caractere na tabela ASCII.

Tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
16																
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	
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	
112	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

Constantes literais do tipo cadeia de caracteres

- É uma sequência de caracteres **entre aspas duplas**
Ex: "Hello, world!"

Atribuição

Atribuição

Definição

Atribuir o valor de uma expressão a uma variável significa calcular o valor da expressão e copiá-lo para o espaço de memória representado por aquela variável.

Atribuição

No exemplo abaixo, a variável soma recebe o valor calculado da expressão a + b. Repare que o comando deve terminar com o ponto-e-vírgula (;

```
soma = a + b;  
      ↑       ↑  
Variável Expressão
```

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

Expressões Simples

- Uma constante literal também é uma expressão e, como tal, pode ser atribuída a uma variável.
Ex: $a = 10;$
- Uma variável é uma expressão.
Ex: $a = b;$

Exemplos de atribuição

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

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

```
a = b;  
f = a;
```

Expressões aritméticas

Expressão aritmética

- Já vimos que constantes literais e variáveis são expressões.
- Uma expressão também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizadas para fazer “cálculos” sobre os valores das variáveis.

Exemplo

$$a + b$$

Calcula a soma de a e b

Expressões

< expressao > + < expressao >: Calcula a soma de duas expressões.

Ex: c = a + b;

< expressao > - < expressao >: Calcula a subtração de duas expressões.

Ex: c = a - b;

*< expressao > * < expressao >*: Calcula o produto de duas expressões.

Ex: c = a * b;

Expressões

< expressao > / < expressao >: Calcula o quociente de duas expressões.

Ex: `c = a / b;`

< expressao > % < expressao >: Calcula o resto da divisão inteira de duas expressões.

Ex: `c = a % b;`

- *< expressao >*: Inverte o sinal da expressão.

Ex: `a = -b;`

Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo:
 $a = b + 2 + c;$

Qual o valor da expressão **5 + 10 % 3**?

E da expressão **5 * 10 % 3**?

Precedência

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C++, os operadores são calculados na seguinte ordem:
 1. * e /, na ordem em que aparecerem na expressão
 2. %
 3. + e -, na ordem em que aparecerem na expressão

Alterando a precedência

- (*< expressao >*) também é uma expressão, que calcula o resultado da expressão dentro dela para só então permitir que as outras expressões executem. Deve ser utilizada para definir a precedência de execução conforme a conveniência do programador.

Ex: $5 + 10 \% 3$ retorna 6, enquanto $(5 + 10) \% 3$ retorna o

- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.

Incremento(++) e Decremento(--)

Operadores de incremento e decremento simplificam a tarefa de incrementar ou decrementar o valor da variável ao qual estão associados em uma unidade.

Exemplo

C++

incrementa o valor da variável c em uma unidade, isto é, equivale a fazer $c=c+1$.

- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra.

Incremento(++) e Decremento(--)

Operador à esquerda da variável: primeiro a variável é incrementada, depois a expressão retorna o seu valor.

Exemplo

```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5     int a = 10;
6
7     cout << ++a;
8
9     return 0;
10 }
```

Imprime 11

Incremento(++) e Decremento(--)

Operador à direita da variável: primeiro a expressão retorna o valor da variável, e depois a variável é incrementada.

Exemplo

```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5     int a = 10;
6
7     cout << a++ << endl;
8     cout << a;
9
10    return 0;
11 }
```

Imprime 10 seguido de 11.

Atribuições simplificadas

Uma expressão da forma

$$a = a + b$$

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

$$a += b$$

Atribuições simplificadas

Comando	Exemplo	Corresponde a:
<code>+=</code>	<code>a += b</code>	<code>a = a + b;</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a - b;</code>
<code>*=</code>	<code>a *= b;</code>	<code>a = a * b;</code>
<code>/=</code>	<code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	<code>a %= b;</code>	<code>a = a % b;</code>

Conversão de tipos

Conversão de tipos

É possível converter alguns tipos entre si. Existem duas formas de fazê-lo:

1. Implícita

- Capacidade (tamanho) do destino deve ser maior que a origem
Ex.: int a; short int b; a = b;
- Operações entre int e float **sempre** convertem para float

2. Explícita:

- Aplicável a variáveis e expressões
Ex. a = (int)((float)b / (float)c);
- Não modifica o tipo “real” da variável, só o valor de uma expressão.
Ex. int a; (float)a=1.0; ← **Errado**

Um uso da conversão de tipos

A operação de divisão (/) possui dois modos de operação de acordo com os seus argumentos: inteira ou de ponto flutuante.

- Se os dois argumentos forem inteiros, acontece a divisão inteira. A expressão `10 / 3` tem como valor 3.
- Se **um** dos dois argumentos for de ponto flutuante, acontece a divisão de ponto flutuante. A expressão `1.5 / 3` tem como valor 0.5.

Quando se deseja obter o valor de ponto flutuante de uma divisão (não-exata) de dois inteiros, basta converter um deles para ponto flutuante:

Exemplo

A expressão `10 / (float) 3` tem como valor 3.3333333

Um exemplo em C++

Problema

Construa um programa que lê dois números inteiros digitados no teclado, daí calcula e imprime no terminal a média deles.

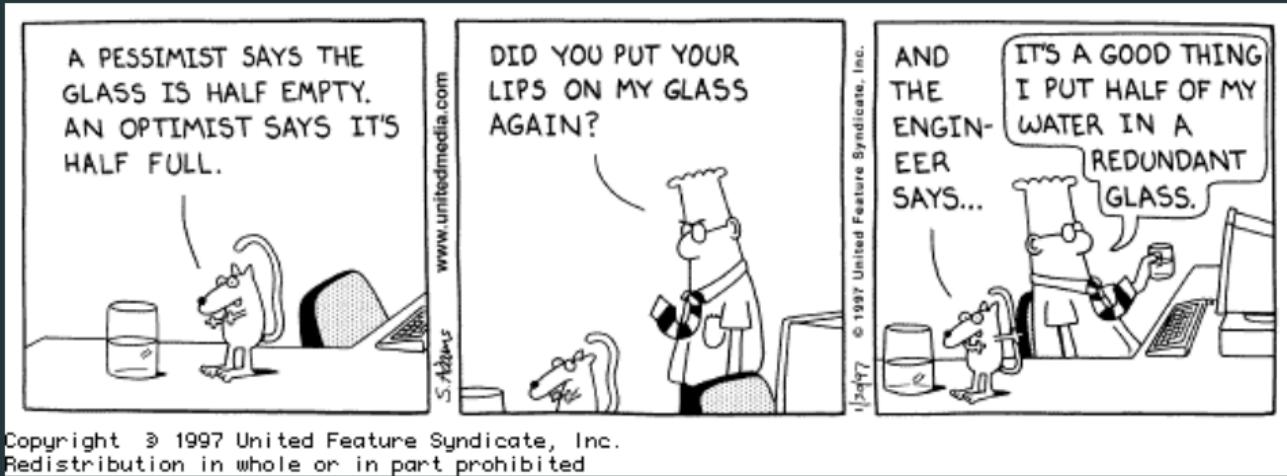
Algoritmo

1. Leia o número a
2. Leia o número b
3. Calcule a média $(a + b)/2$
4. Imprima o resultado

Uma implementação em C++

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int a, b, c; //declaracao das variaveis
6
7     cout << "Digite o primeiro termo: ";
8     cin >> a; //leitura de um valor da entrada
9
10    cout << "Digite o segundo termo: ";
11    cin >> b; //leitura do segundo valor da entrada
12
13    c = (a + b)/2; //atribuicao
14    cout << "A media e: " << c << endl;
15
16    return 0;
17 }
```

Até o próximo tópico...



Copyright © 1997 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited