BCC 201 - Introdução à Programação

Variáveis e Comandos de Atribuição

Guillermo Cámara-Chávez
UFOP

Estrutura Básica de um programa C I

```
< diretivas do pré-processador >
< declarações globais >;
int main()
{
    < declarações locais >;  /* comentário */
    < instruções >;
    return 0;
}
< outras funções >
```

Estrutura Básica de um programa C II

```cpp
/* Prog. C++: Bom dia */
#include <iostream>
using namespace std;
int main()
{
    cout<<"Bom Dia!!";
    return 0;
}
```

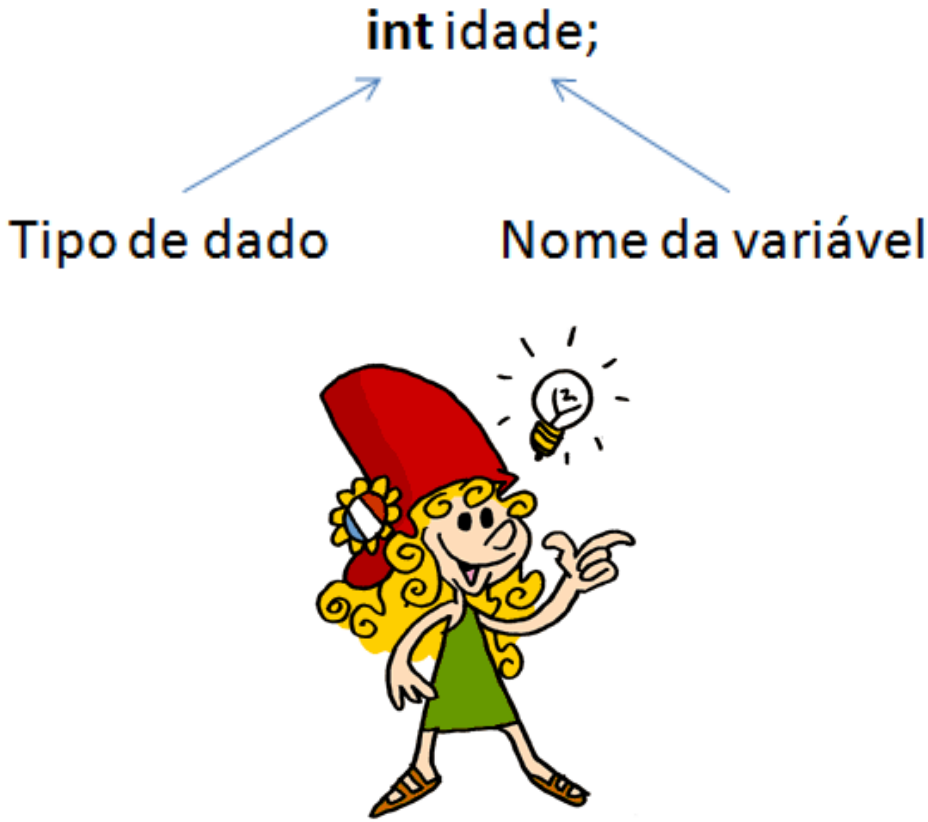
```c
/* Prog. C: Bom dia */
#include <stdio.h>

int main()
{
    printf("Bom Dia!!");
    return 0;
}
```

- *main()* é única, determina o inicio do programa.
- O comando *return* informa ao sistema operacional se o programa funcionou corretamente ou não.

Variáveis

- São locais onde armazenamos valores na memória.
- Toda variável é caracterizada por:
  - um nome: a identifica em um programa,
  - um tipo: determina o que pode ser armazenado naquela variável

Declarando uma variável I



Variáveis inteiras I

- Variáveis utilizadas para armazenar valores inteiros, em formato binário.
  Ex. $13_{10} = 1101_2$
- **int**: inteiro cujo comprimento depende do computador.
- É o inteiro mais utilizado.
- Em computadores Pentium ocupa 32 bits (4 Bytes) e pode armazenar valores de -2.147.483.648 a 2.147.483.647

Variáveis inteiras II

- **unsigned int**: Inteiro cujo comprimento depende do computador e que armazena somente valores positivos. Atualmente ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295
- **long int**: Inteiro que ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647 independente do computador.

Variáveis inteiras III

- Geralmente, `int` será o tamanho nativo do processador, ou seja, 32 bits num processador de 32 bits, 16 bits num processador de 16 bits etc.
- Na maioria dos computadores atuais, o tamanho de `int` é igual ao de `long`

Variáveis de tipo caracter I

- Utilizadas para armazenar letras e outro símbolos existentes em textos
- São variáveis inteiras que armazenam um número associado ao símbolo.
- A principal tabela de símbolos utilizada pelo computadores é a tabela ASCII (*American Standard Code for Information Interchange*), mas existem outras (EBCDIC, Unicode, etc.)

Variáveis de tipo caracter II

Tabela Ascii

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Variáveis de tipo caracter III

- **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.

Variáveis de tipo ponto flutuante I

- Armazenam valores reais, da seguinte forma

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

- Para o programador, funciona como se ele armazenasse números na forma decimal.
- Possui problema de precisão (arredondamento).

Variáveis de tipo ponto flutuante II

- `float`: Utiliza 32 bits, sendo 1 para o sinal, 8 para o expoente e 23 para a mantissa. Pode armazenar valores de $(\pm)10^{-38}$ a $(\pm)10^{38}$
- `double`: Utiliza 64 bits, sendo 1 para o sinal, 11 para o expoente e 52 para a mantissa. Pode armazenar valores de $(\pm)10^{-308}$ a $(\pm)10^{308}$

Obtendo o tamanho de um tipo I

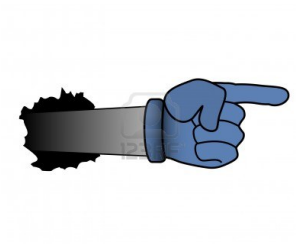
- O comando `sizeof`(tipo) retorna o tamanho, em *bytes*, de um determinado tipo. (Um *byte* corresponde a 8 *bits*).
- Ex. `printf("%d", sizeof(int);` ou `cout<<sizeof(int);`
- Escreve 4 na tela.

Obtendo o tamanho de um tipo II

Mostrar em *Bytes* o tamanho ocupado por variáveis do tipo inteiro, real e caracter.

```
#include <stdio.h>
int main(){
    printf("int      : %d bytes \n", sizeof(int));
    printf("long     : %d bytes \n", sizeof(long int));
    printf("float    : %d bytes \n", sizeof(float));
    printf("double   : %d bytes \n", sizeof(double));
    printf("char     : %d bytes \n", sizeof(char));
    return 0;
}
```

Regras para nomes de variáveis em C I



- Deve começar com uma letra (maiúscula ou minúscula) ou subscrito (_ *underscore*).
- **Nunca** pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, número e subscrito
- Não pode-se utilizar `{ ( + - / \ ; . , ?` como parte do nome de uma variável.

Regras para nomes de variáveis em C II

- C/C++ são uma linguagem *case-sensitive*, ou seja, faz diferença entre nomes com letras maiúsculas e nomes com letras minúsculas: `Peso` e `peso` são diferentes.
- Costuma-se usar maiúsculas e minúsculas para separar palavras `PesoDoCarro`
- Identificadores devem ser únicos no mesmo escopo (não podem haver variáveis com mesmo identificador dentro do mesmo bloco).

Regras para nomes de variáveis em C III

- 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*

Regras para nomes de variáveis em C IV

- Quais dos nomes a seguir são nomes corretos de variáveis? Se não forem corretos, porque não são?

3ab	a3b	fim	int
`\meu`	`_A`	`n_a_o`	papel-branco
`a*`	`c++`	`*nova_variavel`	

Regras para nomes de variáveis em C V

- Exemplo 1:

```
int a, b, SomaGeral;
a = 3; // a recebe o valor 3
b = a * 2; // b recebe o dobro do valor de a
SomaGeral = a + b + 2; // c recebe a soma de a e b + 2
```

Regras para nomes de variáveis em C VI

- Exemplo 2:

```c
/* Exemplo de programa em C */
//  Arquivo de cabeçalho (header)
#include <stdio.h>
int main()
{
    int contador;  // declarações simples
    float PrecoDoQuilo;
    double TaxaDeCambio;
    char LetraDigitada;
    // Pode colocar mais de uma variável na
    // na mesma linha
    int IdadeManoel, IdadeJoao, IdadeMaria;
    double  TaxaDoDolar,     // Também pode trocar
            TaxaDoMarco,     // de linha no meio
            TaxaDoPeso,
            TaxaDoFranco;
.......
}
```

Regras para nomes de variáveis em C VII

- Exemplo 3: inicialização de variáveis

```
// Exemplo de programa em C
//  Arquivo de cabeçalho (header)
#include <stdio.h>
int main()
{
    // declara e inicializa com Zero
    int NroDeHoras = 0;
    // declara e inicializa com 10.53
    float PrecoDoQuilo = 10.53;
    double  TaxaDoDolar = 1.8,
            TaxaDoMarco = 1.956,
            TaxaDoPeso = 1.75,
            TaxaDoFranco = 0.2;


.......
    return 0;
}
```

Constantes

- Constantes 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 também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo *string*, que corresponde a uma seqüência de caracteres.
- Exemplos de constantes: 85, 0.10, 'c', "Meu primeiro programa"

Constantes inteiras

- Um número na forma decimal, como escrito normalmente
  Ex: 10, 145, 1000000
- Um número na forma hexadecimal (base 16), precedido de 0x
  Ex: $0xA$ ($0xA_{16} = 10_2$), $0x100$ ($0x100_{16} = 256_2$)
- Um número na forma octal (base 8), precedido de 0
  Ex: 010 ($0x10_8 = 8_2$)

Constantes de tipo flutuante

- Um número decimal. Para a linguagem C/C++, um número só pode ser considerado um número decimal 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

Constantes de tipo caracter I

- Uma constante do tipo caracter é sempre representado por uma letra entre aspas simples.
  Ex: 'A'
- 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.

Constantes de tipo caracter II

```
#include <stdio.h>
int main(){
    int i;
    for (i = 60; i < 120; i++)
        printf("%d %c \n", i, i);
    return 0;
}
```

Constantes de tipo *string* I

- Uma constante do tipo é um texto entre aspas duplas
  Ex: “Meu primeiro programa”
- Um *string* ou cadeia de caracteres é um conjunto de caracteres.
- Toda cadeia finaliza com o caracter nulo (`'\0'`)

Constantes de tipo *string* II

```
#include <stdio.h>
int main(){
    char nome[30];
    nome[0] = 'c';      nome[1] = 'a';
    nome[2] = 'd';      nome[3] = 'e';
    nome[4] = 'i';      nome[5] = 'a';
    printf("%s: \n", nome);
    nome[5] = '\0';
    printf("%s: \n", nome);
    return 0;
}
```

Exemplo de declaração de constantes I

```
// Não se coloca ponto-e-vírgula após o valor
#define LARGURA_MAXIMA 50
#define NRO_DE_DIAS_DA_SEMANA 7
#define NRO_DE_HORAS_DO_DIA 24
#define VALOR_DE_PI  3.1415

int main ()
{
    int TotalDeHoras;
    const int r = 100;

    TotalDeHoras = 10 * NRO_DE_DIAS_DA_SEMANA *
                   NRO_DE_HORAS_DO_DIA;
.......
    return 0;
}
```

Escrevendo o conteúdo de uma variável na tela em C I

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando *printf*.
- Utilizamos um símbolo no texto para representar que aquele trecho deve ser substituído por uma variável e, no final, passamos uma lista de variáveis ou constantes, separadas por vírgula.

Escrevendo o conteúdo de uma variável na tela em C II

- Ex.
  `printf ("A variável %s contém o valor %d", "a", a);`
- Imprime: A variável *a* contém o valor 10
- Nesse caso, `%s` deve ser substituído por uma variável ou constante do tipo *string* enquanto `%d` deve ser substituído por uma variável do tipo inteiro.

Formatos inteiro I

- %d: escreve um número inteiro na tela sem formatação

  Ex: `printf("%d", 10);`

  Imprime 10

Formatos inteiro II

- %< `número` >d: escreve um inteiro na tela, preenchendo com espaços a esquerda para que ele ocupe pelo menos < `número` > casas na tela.

  Ex: `printf("%4d", 10);`
  imprime ␣␣10

- %0< `número` >d: escreve um inteiro na tela, preenchendo com zeros a esquerda para que ele ocupe pelo menos < `número` > casas na tela.

  Ex: `printf("%04d", 10);`
  imprime 0010

Formatos inteiro III

- A letra *d* pode ser substituída pelas letras *u* e *l*, ou as duas, quando desejamos escrever variáveis do tipo *unsigned* ou *long*, respectivamente.

  Ex: `printf("%d", 4000000000);`

  escreve -294967296 na tela, enquanto que

  `printf("%u", 4000000000);`

  escreve 4000000000

Formatos ponto flutuante I

- `%f`: escreve um ponto flutuante na tela, sem formatação

  Ex: `printf("%f", 10.0);`

  imprime 10.000000

Formatos ponto flutuante II

- %e: escreve um ponto flutuante na tela, em notação científica
  - Ex: `printf("%e", 10.02545);`
  - imprime $1{,}002545e+01$

Formatos ponto flutuante III

- %< tamanho >.< decimais >f: escreve um ponto flutuante na tela, com tamanho < tamanho > e < decimais > casas decimais.
- O ponto utilizados para separar a parte inteira da decimal, também conta no tamanho

  Ex: `printf("%6.2f", 10.0);`
  ␣10.00

Formatos ponto flutuante IV

- A letra *f* pode ser substituída pelas letras *lf*, para escrever um *double* ao invés de um *float*

  Ex: `printf(%6.2lf), 10.0);`

  imprime ␣10.00

Formato caracter

- %c: escreve uma letra.

  Ex. `printf("%c", 'A');`
  imprime a letra A

  Note que `printf("%c", 65);` também imprime a letra A.

Formato *string* I

- %s: escreve um *string*

  Ex. `printf ("%s", "Meu primeiro programa");`
  imprime Meu primeiro programa

A função *scanf* I

- realiza a leitura de um texto a partir do teclado
- parâmetros:
  - uma *string*, indicando os **tipos das variáveis** que serão lidas e o formato dessa leitura.
  - uma **lista de variáveis**
- aguarda que o usuário digite um valor e atribui o valor digitado à variável

```
scanf("%d", &idade);
```

A função *scanf* II

Elaborar um programa para calcular e exibir a média aritmética de duas notas

A função *scanf* III

```c
#include<stdio.h>
int main()
{
    float nota1, nota2, media;
    printf("Digitar duas notas: ");
    scanf("%f %f", &num1, &num2);
    media = (nota1 + nota2) / 2.0;
    printf("A media eh: %f ", media);
    return 0;
}
```

A função *scanf* IV

mostrar a média utilizando duas casas para a parte inteira (deve preencher com zeros à esquerda para que ocupe pelo menos duas casas) e com duas casas decimais.

A função *scanf* V

```c
#include<stdio.h>
int main()
{
    float nota1, nota2, media;
    printf("Digitar duas notas: ");
    scanf("%f %f", &num1, &num2);
    media = (nota1 + nota2) / 2.0;
    printf("A media eh: %05.2f ", media);
    return 0;
}
```

FIM