

Introdução à Linguagem de Programação C: Variáveis, Constantes, Expressões, Atribuição, Tipos de dados, Entrada e Saída de Dados

Disciplina de Programação de Computadores I
Universidade Federal de Ouro Preto

Agenda

- Variáveis: Declaração e Atribuição
- Expressões: Variáveis e Constantes
- Tipos de Dados em C
- Entrada e Saída de Dados
- Expressões Aritméticas
- Atribuições Simplificadas
- Modificadores de Tipos
- Conversão de Tipos



Memória e variáveis

- A memória funciona como caixas empilhadas que guardam as informações que um programa manipula.
- Cada caixa tem um tamanho diferente e guarda um tipo específico de informação.
- Os programas acessam as informações em uma caixa através de uma variável, que funciona como um rótulo (ou nome) para esta caixa.
- Uma variável possui, então, um tipo e um nome.

Declaração de Variáveis

- Variáveis precisam ser declaradas antes de serem utilizadas.

Declaração: tipo nome;
 tipo nome₁, nome₂, nome₃, ... ;

Ex: agua jarra;
 biscoito pote;

jarra

agua

pote

biscoito

Atribuição de Variáveis

- Armazenamos um conteúdo em uma variável utilizando o operador de atribuição (=).

Atribuição: nome = conteúdo;

Ex: jarra = 5l de agua;
pote = 10 biscoitos;

jarra

5l

pote

10 biscoitos

Nomes de Variáveis

- Deve começar com uma letra (maiúscula ou minúscula) ou subscrito(_). Nunca pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, números e subscrito.
- Não se pode utilizar como parte do nome de uma variável:

{ } () [] + - * / \ ; . , ?

- Letras maiúsculas e minúsculas são diferentes:

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

Tipos de dados em C

- **int** : representa quantidades inteiras
- **float** / **double** : representa quantidades decimais
- **char** : armazena (APENAS) um caractere; também é um tipo inteiro, já que caracteres possuem representação numérica!
- **void**: representa a ausência de informação sobre o tipo de uma variável
- Em testes, **0** é interpretado como **Falso** e **qualquer número diferente de 0** é interpretado como **verdadeiro**. Sempre usar **inteiros (int e char)** em testes!

Entrada e Saída de Dados

- Geralmente, desejamos que nosso programa receba informações, processe-as e nos mostre um resultado.
- Para obter informações, o computador deve ler dados e armazená-los em uma variável (Entrada de Dados).
- Para mostrar um resultado, o computador deve imprimir dados diretamente ou a partir de uma variável (Saída de Dados).

Saída de Dados

- Para imprimir um texto, utiliza-se o comando **printf**, que possui o texto como parâmetro.

- **Exemplo:**

```
printf("Meu primeiro programa!");
```

Saída: Meu primeiro programa!

- Existem símbolos especiais de formatação:

\n: insere nova linha

\t: insere uma tabulação

- **Exemplo:**

```
printf("Meu primeiro programa!\t Texto deslocado.");
```

Saída: Meu primeiro programa! Texto deslocado.

Saída Formatada de Dados

- Além de texto, o comando **printf** pode imprimir o conteúdo de variáveis ou o valor de constantes.
- São utilizados símbolos especiais dentro do texto para representar o conteúdo da variável que será impresso.
- As variáveis são listadas como parâmetros do comando após o texto, separando-as por vírgula.
- Os símbolos do texto são substituídos pelas variáveis, na ordem em que aparecem, devendo ser do mesmo tipo!

Saída Formatada de Dados - Alguns Modificadores

Sequência	Tipo de Dados	Sequência	Tipo de Dados
%d %i	inteiros	%f %F	ponto flutuante
%u	inteiros sem sinal (números grandes)	%e %E	ponto flutuante em notação científica
%.<num>d	inteiro com <num> dígitos (completa com 0 à esquerda)	%.<num>f	ponto flutuante com <num> casas decimais
%c	um caractere	%s	cadeia de caracteres

Entrada de Dados

- A leitura de dados do teclado é feita pela função **scanf**.
- A função recebe um texto composto por uma sequência de modificadores, indicando os tipos de dados a serem lidos, e uma lista de variáveis, associadas aos modificadores, que receberão os dados lidos.
- **Exemplo:**
 int numero; char letra;
 scanf("%d %c", &numero, &letra);
Entrada: 10 <ENTER> A <ENTER>
Associa 10 à variável numero e 'A' à variável letra.

Variáveis, Constantes e Expressões

- As informações associadas a uma variável podem ser alteradas.
- Constantes são informações que não se alteram.
- Constantes e variáveis possuem os mesmos tipos.
- Expressões são constantes, variáveis ou operações entre estas duas últimas (mais sobre isto depois!).
- A atribuição de uma expressão a uma variável define o valor desta última.

Exemplos de Constantes e Expressões Aritméticas

Constantes:

`int i1 = 10; int i2 = 124957`

`float f1 = 4.2334; double f2 = 343214213.234;`

`char a = 'a'; char A = 'A'; char l_par = '(';`

“Esta constante é uma cadeia(sequência) de caracteres!”

Expressões Aritméticas:

`int e1 = 10 + 20; int e2 = 123 + e1;`

`float e3 = 12.5 + 1.34; double e4 = 23.12 - e3`

Operadores Aritméticos

Operação	Operador	Expressão Algébrica	Expressão em C
Adição	+	$f + 3$	$f + 3$
Subtração	-	$5 - p$	$5 - p$
Multiplicação	*	$b \cdot m$	$b * m$
Divisão	/	$4 / 3$	$4 / 3$
Resto	%	$4 \bmod 2$	$4 \% 2$

- Operador / retorna a divisão inteira, a menos que um dos operandos seja de ponto flutuante.

Exemplo: $5/2 = 2$ $5.0/2 = 2.5$

- Operador % retorna o resto da divisão inteira.

Exemplo: $5\%2 = 1$

Precedência de operadores

A precedência define a ordem em que as operações devem ser realizadas.

Em C, a ordem das operações é a seguinte:

- expressões entre parênteses, na ordem em que aparecem
- $*$ e $/$, na ordem em que aparecem
- $\%$
- $+$ e $-$, na ordem em que aparecem
- Qual o valor de $5+3*8$? 64 ou 29?
 $5+3*8= 29$ e $(5+3)*8 = 64$

Incremento (++) e Decremento (--)

- Adicionar ou subtrair 1 a uma variável é uma operação muito comum.
- C fornece o operador ++ para adicionar 1 a uma variável: $x = x + 1$; pode ser escrito como $x++$ ou $++x$;
- C fornece o operador -- para subtrair 1 de uma variável: $x = x - 1$; pode ser escrito como $x--$ ou $--x$;
- A instrução $x++$; executa menos códigos de máquina que a instrução $x = x + 1$;

Incremento (++) e Decremento (--): Ordem de Avaliação

- Usar os operadores ++ e -- à esquerda e à direita de uma variável produz resultados diferentes!
- ++x / --x : o valor de x será incrementado / decrementado e, sem seguida, será utilizado;
y = ++x; equivale a x = x + 1; y = x;
y = --x; equivale a x = x - 1; y = x;
- x++ / x-- : o valor de x será utilizado e, em seguida, será incrementado / decrementado;
y = x++; equivale a y = x; x = x + 1;
y = x--; equivale a y = x; x = x - 1;

Atribuições simplificadas

- É possível simplificar atribuições da forma $x = x \text{ <op> } y$

Operador	Expressão	Equivalência
$+=$	$x += y$	$x = x + y$
$-=$	$x -= y$	$x = x - y$
$*=$	$x *= y$	$x = x * y$
$/=$	$x /= y$	$x = x / y$
$\%=$	$x \% = y$	$x = x \% y$

Modificadores de Tipos de Dados

- Os tipos de dados numéricos **char**, **int**, **float** e **double** podem ter seu tamanho e sua representação interna alterados pelos modificadores:
 - **short**: diminui o número de bits utilizado na representação pela metade.
 - **long**: duplica o número de bits utilizado na representação.
 - **signed**: inclui sinal na representação (padrão).
 - **unsigned**: remove o sinal na representação.

Modificadores de Tipos de Dados e a função sizeof

sizeof(Tipo) = tamanho de um elemento de Tipo, em bits

Tipo	Bytes	Bits	Intervalo
signed char	1	8	-128 ~ +127
unsigned char	1	8	0 ~ +255
short int	2	16	-32,768 ~ +32,767
unsigned short int	2	16	0 ~ +65,535
int	4	32	-2,147,483,648 ~ +2,147,483,647
unsigned int	4	32	0 ~ +4,294,967,295
long int	8	64	
long int	8	64	
float	4	32	Precisão simples
double	8	64	Precisão dupla

Conversão de Tipos de Dados

- É possível converter o tipo de expressões e do conteúdo de variáveis fazendo uma atribuição a uma variável de outro tipo.
 - Conversão Implícita: o tamanho do tipo de destino é maior que o do tipo de origem e não há perda de informações.
 - Conversão Explícita: o tamanho do tipo de destino é menor que o do tipo de origem, pode haver perda de informações e é preciso indicar explicitamente a conversão.

Conversão de Tipos de Dados : Exemplos

- Conversão Implícita

- | | | | |
|-----------|--------------|----------|--------|
| int x; | short int y; | y = 10; | x = y; |
| float a; | int b; | b = 10; | a = b; |
| double k; | float z; | z = 2.5; | k = z; |

- Conversão Explícita

- | | | | |
|---|--------------|-----------|--------------------|
| int x; | short int y; | x = 10; | y = (short int) x; |
| float a; | int b; | a = 10.0; | b = (int) a; |
| double k; | int z; | k = 2.5; | z = (int) k; |
| int m = (int) ((float) 2 / (float) 1); | | | |

Referências Bibliográficas

- Material de aula do Prof. Ricardo Anido, da UNICAMP:
<http://www.ic.unicamp.br/~ranido/mc102/>
- Material de aula da Profa. Virgínia F. Mota:
<https://sites.google.com/site/virginiaferm/home/disciplinas>
- DEITEL, P; DEITEL, H. *C How to Program*. 6a Ed. Pearson, 2010.