3. Linguagem de Programação C

Prof. Renato Tinós

Departamento de Computação e Matemática
(FFCLRP/USP)

Principais Tópicos

3.3. Conceitos Básicos de C: Expressões, Operadores e Bibliotecas

- 3.3.1. Expressões
- 3.3.2. Conversão de Tipos de Dados
- 3.3.3. Operadores
  - 3.3.3.1. Aritméticos
  - 3.3.3.2. Booleanos
  - 3.3.3.3. Bitwise
  - 3.3.3.4. Atribuição
- 3.3.4. Bibliotecas
- 3.3.5. Comandos Básicos de Entrada e Saída

3.3.1. Expressões

- **Expressões em C são compostas por operadores e operandos**
  - Ex.: *x* = *a* + *5*;
  - Operandos
    - » Pode ser uma variável, constante ou valor retornado por uma função
      - Ex.: a, b, c, 2 e 4
  - Operadores
    - » Definidos pela linguagem e por funções

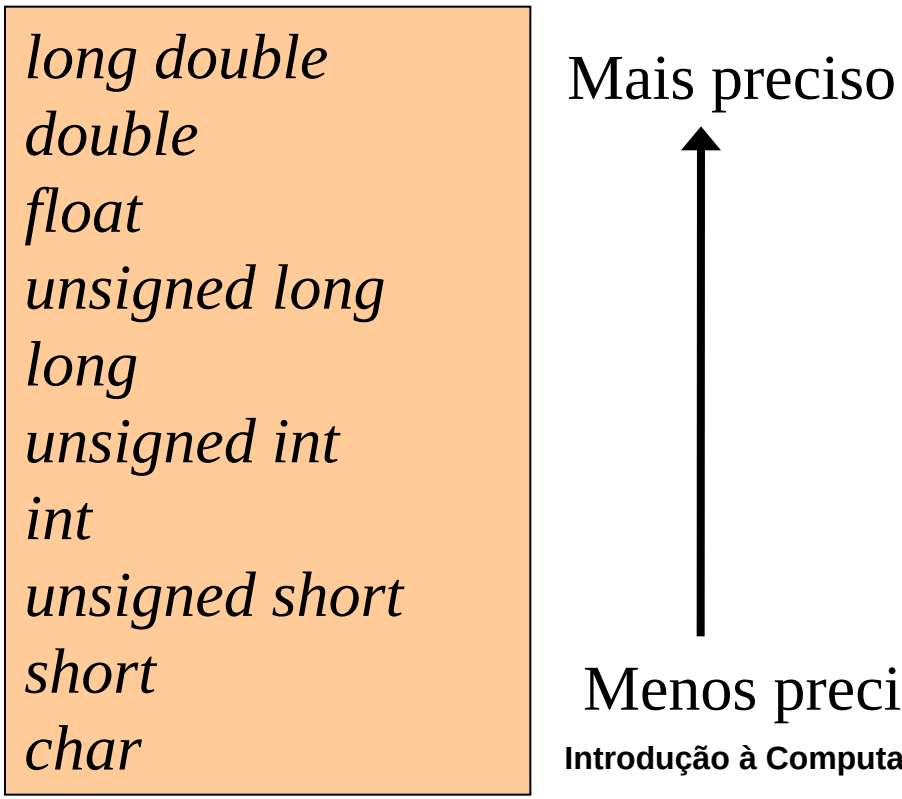
3.3.1. Expressões

- **Servem para**
  - Computar e atribuir valores à variáveis
  - Controlar o fluxo de execução de um programa
- **Ao realizar operações, operador retorna um valor**
  - Valor e seu tipo dependem do operador e do tipo dos seus operandos
    - » Ex.: *x = (-b + sqrt (b * b - 4 * a * c)) / (2 * a)*

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

3.3.2. Conversão de Tipos de Dados

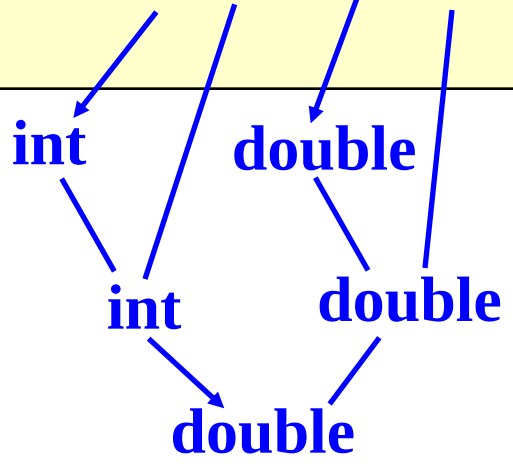
- **Quando variáveis e constantes de tipos de dados diferentes são misturados em uma expressão, elas são convertidas para o tipo mais preciso**



3.3.2. Conversão de Tipos de Dados

- **Exemplo**

```
...
char c;
int i;
float f;
double d;
...
resultado = ( c / i ) - ( f  * d );
...
```



3.3.2. Conversão de Tipos de Dados

- **Conversão de tipos**
  - C permite a conversão explícita de tipos
  - Operador unário com o tipo desejado entre parênteses
    - » Ex. supor que duas variáveis, numerador e denominador, são declaradas como do tipo *int*
    - » res = (double) numerador / denominador
  - Quando conversor converte um tipo para um outro menos preciso, pode haver perda de informação
    - » Ex.: (int) 2.87 = 2

3.3.3. Operadores

- **Executam funções sobre operandos**
  - Unários: utilizam um operando
    - » Pode vir antes do ou após o operando
    - » Ex.: *count++, ++count*
  - Binários: utilizam dois operandos
    - » Operador aparece entre os operandos
    - » Ex.: *count = 0*
  - Ternários: utilizam três operandos (?:)
    - » Cada parte do operador aparece entre dois operandos
    - » Ex.: expr ? op1 : op2

3.3.3. Operadores

- **Operadores de C podem ser divididos em categorias**
  - Aritméticos
  - Booleanos (relacionais e condicionais)
  - Bitwise
  - Atribuição

3.3.3. Operadores

- **Ordem como uma expressão é avaliada é importante**
  - Exemplos: x * z + y; x + y / 100;
  - Ordem em que a expressão será avaliada pode ser definida explicitamente usando parênteses
    - » Exemplo: x + (y / 100)
  - Operadores da linguagem C possuem uma ordem de precedência

3.3.3. Operadores

- **Precedência**

  – Informa ordem de aplicação de operadores na ausência de parênteses

    » Quando dois operadores competem pelo mesmo operando, aplica-se primeiro o de precedência mais alta

    » x = (-b + sqrt (**b * b** - **4 * a * c**)) / (2 * a)

3.3.3. Operadores

Operador	Associatividade
( ) [ ] -> .	esquerda
Operadores unários - -- ++ | & * ~ (type) sizeof	direita
* / %	esquerda
+ -	esquerda
<< >>	esquerda
< <= > >=	esquerda
== !=	esquerda
&	esquerda
^	esquerda
|	esquerda
&&	esquerda
||	esquerda
?:	direita
= =op	direita
,	esquerda

Maior precedência

Menor precedência

3.3.3. Operadores

- **Precedência cresce de baixo para cima**
  - Operadores com maior precedência são avaliados primeiro
  - Operadores na mesma linha têm a mesma precedência
  - Operadores binários (exceto de atribuição) são avaliados da esquerda para a direita
  - Operadores de atribuição são avaliados da direita para a esquerda

3.3.3. Operadores

- **Associatividade**
  - Quando dois operadores têm a mesma precedência, eles são aplicados na ordem especificada por sua associatividade
    - » Indica se o operador segue para a direita ou esquerda
      - Operadores associativos-à-esquerda
        - – Maioria dos operadores de C
        - – Operador mais a esquerda é avaliado primeiro
      - Operador associativos-à-direita
        - – Operador mais a direita é avaliado primeiro

3.3.3.1. Operadores Aritméticos

- **Operadores binários (ou seja, com dois operandos)**

Operador	Uso	Descrição
+	*op1 + op2*	*Adiciona op1 e op2*
-	*op1 - op2*	*Subtrai op2 de op1*
*	*op1 * op2*	*Multiplica op1 por op2*
/	*op1 / op2*	*Divide op1 por op2*
%	*op1 % op2*	*Computa o resto da divisão de op1 por op2*

3.3.3.1. Operadores Aritméticos

- **Operador de divisão para inteiros (/)**
  - Parte decimal é descartada se ambos os operandos forem do tipo inteiro (operação de truncamento)
    - » Ex.: 9 / 4 = 2
  - Para que o resultado não seja truncado, pelo menos um dos operandos deve ser do tipo ponto flutuante
    - » Ex. 9.0 / 4 = 9 / 4.0 = 9.0 / 4.0 = 2.25

3.3.3.1. Operadores Aritméticos

- **Operador resto (%)**
  - Computa o resto da divisão de um número inteiro por um outro número inteiro
    - » Ex. 9 % 4 = 1
    - » Útil para testar se um número é divisível por um outro (resto da divisão é igual a zero)

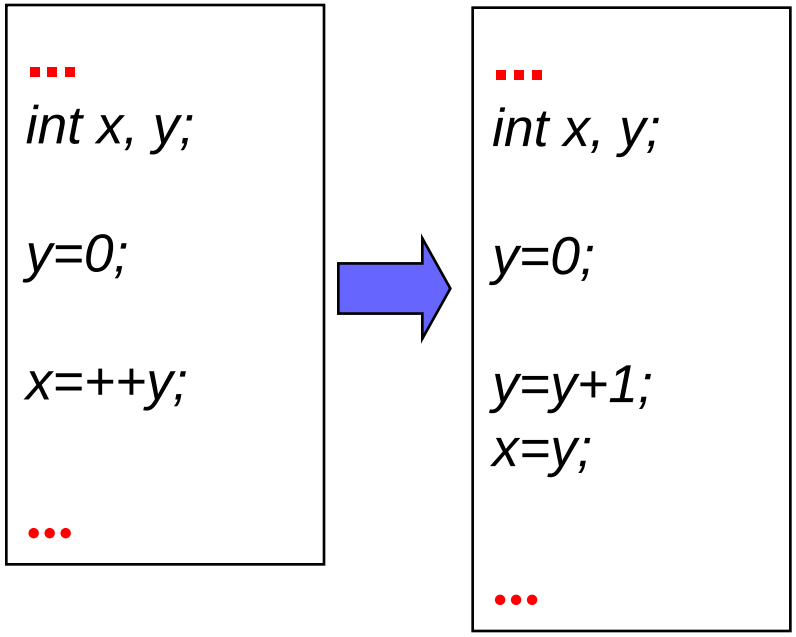
3.3.3.1. Operadores Aritméticos

- **Operadores Unários**

Operador	Uso	Descrição	Equivalência
-	*-op*	*troca o sinal de um operando*	*op= - op*
++	*op++*	*Avalia o valor antes de incrementar; incrementa op de 1*	*op = op+ 1*
	++op	*Incrementa op de 1; avalia o valor após incrementar*	
--	*op--*	*Avalia o valor antes de decrementar; decrementa op de 1*	*op = op-1*
	--op	*Decrementa op de 1; avalia o valor após decrementar*	

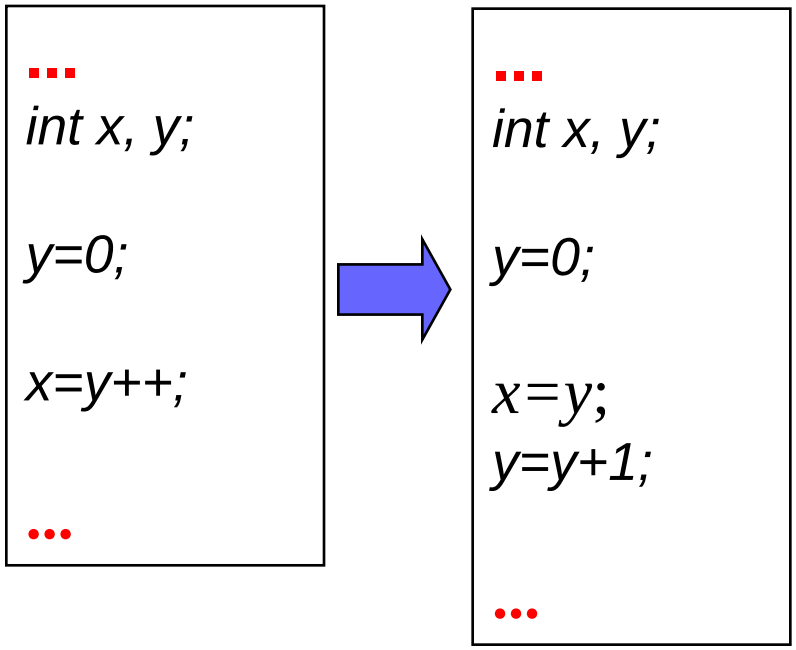
3.3.3.1. Operadores Aritméticos

- **Exemplo**



3.3.3.1. Operadores Aritméticos

- **Exemplo**



3.3.3.1. Operadores Aritméticos

Exercício 3.3.1. O que o seguinte programa imprime na tela?

```
/* Programa: operadores unarios */
# include <stdio.h>

main (){
    int x;

    x = 0;
    printf ("x = %d\n", x++);
    printf ("x = %d\n", x);
    printf ("x = %d\n", ++x);
    printf ("x = %d\n", x++);
    printf ("x = %d\n", x);
}
```

3.3.3.2. Operadores Booleanos

- **C define três classes de operadores que manipulam dados booleanos**
  - Operadores relacionais
  - Operadores lógicos
  - Operador ?:

3.3.3.2. Operadores Booleanos

- **Vários comandos ou operadores da linguagem C utilizam valores lógicos**
  - Falso ou verdadeiro
    - » Valores booleanos
- **Algumas linguagens de programação possuem um tipo específico para estes valores**
  - Tipo Booleano

3.3.3.2. Operadores Booleanos

- **A Linguagem C não utiliza um tipo Booleano**
  - Mas permite o uso do tipo *int* para expressar valores Booleanos
    - » Variável (constante) inteira possui valor igual a 0 → falso
    - » Variável (constante) inteira possui valor diferente de 0 → verdadeiro

3.3.3.2. Operadores Booleanos

- **Expressões e funções lógicas retornam um valor inteiro**
  - Quando o resultado de uma expressão (função) é falso, ela retorna um valor igual a 0
  - Quando o resultado de uma expressão (função) é verdadeiro, ela retorna um valor diferente de 0

3.3.3.2. Operadores Booleanos

- **Operadores Relacionais**
  - Comparam dois valores e determinam o relacionamento entre eles

==	igual
!=	não igual
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

3.3.3.2. Operadores Booleanos

– Como resultado da comparação, retornam um valor inteiro que pode ser
  - 0 (falso)
  - 1 (verdadeiro)

Operador	Uso	Descrição
>	*op1 > op2*	*op1 é maior que op2*
>=	*op1 >= op2*	*op1 é maior ou igual a op2*
<	*op1 < op2*	*op1 é menor que op2*
<=	*op1 <= op2*	*op1 é menor ou igual a op2*
==	*op1 == op2*	*op1 é igual a op2*
!=	*op1 != op2*	*op1 é diferente de op2*

3.3.3.2. Operadores Booleanos

Exercício 3.3.2. O que o seguinte programa imprime na tela?

```
/* Programa: operadores relacionais 1*/
# include <stdio.h>

main ( ) {
    int i=2, j=3;

     printf("%d == %d eh %d \n", i, j, i==j);
     printf("%d != %d eh %d \n", i, j, i!=j);
     printf("%d > %d eh %d \n", i, j, i>j);
     printf("%d < %d eh %d \n", i, j, i<j);
     printf("%d >= %d eh %d\ n", i, j, i>=j);
     printf("%d <= %d eh %d \n", i, j, i<=j);
}
```

3.3.3.2. Operadores Booleanos

– Não confundir = (atribuição) com == (igual a)
  - » Um dos erros mais comuns
  - » Compilador geralmente aceita estes erros
    - Sinal de atribuição dentro de uma expressão vira uma atribuição encaixada

– Podem ser usados apenas para comparar valores de dados atômicos
  - » *int, double, char, etc.*

3.3.3.2. Operadores Booleanos

- **Operadores Lógicos**
  - Utilizam operandos booleanos e retornam resultado booleano

Precedência

Operador	Função	Uso	Retorna verdade (1) se
!	Negação	! op	op é falso (0)
&&	E (And)	op1 && op2	op1 e op2 são ambos verdade (1)
||	Ou (Or)	op1 || op2	op1 ou op2 é verdade (1)

3.3.3.2. Operadores Booleanos

– Tabela Verdade:

A	B	A&&B	A||B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

3.3.3.2. Operadores Booleanos

```
/* Programa: Operadores Lógicos*/
# include <stdio.h>

main () {
    int operacao,
    float a=10.2, b;
    printf("Entre com a operação: ");
    scanf("%d",&operacao);
   printf("Entre com o valor de b: ");
    scanf("%f",&b);

    if ( (operacao==1) && (b != 0.0) )
        a = a / b;
   else
        a = b;

    printf ("Valor da expressao eh: %f \n", a);
}
```

3.3.3.2. Operadores Booleanos

- **Avaliação "Preguiçosa"**
  - Em C as sub-expressões individuais de uma expressão lógica são avaliadas da esquerda para a direita
  - Avaliação termina assim que a resposta puder ser determinada
    - » exp1 && exp 2
    - » exp1 || exp2

3.3.3.2. Operadores Booleanos

- **Operador ?:**
  - Escrito em duas parte e requer três operandos
    - » (condição) ? exp1 : exp2
      - Primeiro avalia a condição
      - Se condição é verdadeira, o resultado do operador é o valor da expressão exp1
      - Se a condição é falsa, o resultado do operador é o valor da expressão exp2
      - Ex.: max = (x > y) ? x : y

3.3.3.2. Operadores Booleanos

```
/* Programa: Operador ?: */
# include <stdio.h>

main () {
    int a, b, resultado;

    a = 37;
    printf("Entre com o valor de b: ");
    scanf("%d",&b);
    resultado = (a > b) ? 1 : 0 ;
    if (resultado==1)
            printf ("O resultado da primeira equacao eh maior que o da segunda");
   else
            printf ("O resultado da segunda equacao eh maior que o da primeira");

}
```

3.3.3.3. Operadores Bitwise

- **Permitem manipular bits dos dados**

Operador	Uso	Descrição
>>	*op1 >> op2*	*move bits de op1 op2 posições p/ direita*
<<	*op1 << op2*	*move bits de op1 op2 posições p/ esquerda*
&	*op1 & op2*	*bitwise and*
|	*op1 | op2*	*bitwise or*
^	*op1 ^ op2*	*bitwise xor*
~	*~ op2*	*Complemento de um*

3.3.3.3. Operadores Bitwise

- **Exemplo**

  – Calcular o valor da expressão:

    » *13 >> 1; // Usar codificação com 8 bits*

  – Resposta:

    » Representação binária do valor 13: 00001101

    » Movendo 00001101 uma posição para a direita: 00000110 = 6

3.3.3.4. Operadores de Atribuição

- **Comandos de atribuição**

  – Atribuição de valores a variáveis em C é construída na forma de uma expressão

    » Ex.: resultado = 4;

  – Converte o tipo do valor do lado direito para o tipo da variável do lado esquerdo

    » Ex. sejam as variáveis n1 e n2 dos tipos *double* e *int,* respectivamente:

      - n1 = 0; // n1 passa a armazenar o valor 0.0
      - n2 = 3.45; // n2 passa a armazenar o valor 3

3.3.3.4. Operadores de Atribuição

- **Operador de atribuição**
  - Pode conter uma expressão do lado direito
    - » Ex.: z = (x + 4) * (y - 3);
  - Atribuições encaixadas
    - » Atribuições dentro de uma expressão maior
      - Devem ser usadas apenas em casos especiais
        - Dificultam a leitura de programas
    - » Ex.: atribuir o mesmo valor a diversas variáveis (atribuições múltiplas)
      - n1 = n2 = n3 = 4; /* equivale a: n1 = (n2 = (n3 = 4)) */

3.3.3.4. Operadores de Atribuição

- **Operador de atribuição**

  - Operador de atribuição básico atribui valor de uma expressão a uma variável
    - » Ex.: *int count = 0; i = i + 2;*
  - A linguagem C permite a combinação da atribuição com operadores binários (atribuição com atalho)
    - » Exemplos:
      - tot += val; // equivale a: tot = tot + val
      - res -= val;
      - x /= 10;
      - salario *= 2;

3.3.3.4. Operadores de Atribuição

Operador	Uso	Equivalente a
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2
<<=	op1 <<= op2	op1 = op1 << op2
>>=	op1 >>= op2	op1 = op1 >> op2

3.3.3.4. Operadores de Atribuição

Exercício 3.3.4. O que o seguinte programa imprime na tela?

```
/* Programa: operadores de atribuicao */
# include <stdio.h>

main (){
    float x=0;
    int y=10;

    x += 2.5;
    printf ("x = %f \n", x);
    x -= 1.5;
    printf ("x = %f \n", x);
    x *= 3.0;
    printf ("x = %f \n", x);
    y %= 2;
    printf ("y = %d\n", y);
}
```

3.3.4. Bibliotecas

- **Difícil escrever programas interessantes sem usar bibliotecas de funções**
  - A cada dia, os programadores dependem mais de bibliotecas de funções
    - » 90% ou mais do código de um programa pode ser formado por códigos de bibliotecas
  - Programador deve saber como:
    - » Escrever novos códigos
    - » Evitar escrita de novos códigos sabendo utilizar as bibliotecas existentes

3.3.4. Bibliotecas

- **Coleções de programas ou funções**
  - Escritas por outros programadores e disponibilizadas
  - Parte do ambiente da linguagem

- **Geralmente cada biblioteca possui um conjunto de funções relacionadas**
  - Exemplos:
    - » Funções matemáticas: *math.h*
    - » Entrada e saída de dados: *stdio.h*

3.3.4. Bibliotecas

- **Para que as funções de uma biblioteca possam ser utilizadas em um programa, a biblioteca tem que ser incluída**
  - Comando #include
  - Ex.: *#include <stdio.h>*
    - » Permite que o programa utilize as funções da biblioteca *stdio.h*

3.3.5. Comandos de Entrada e Saída

- **Biblioteca padrão de entrada e saída (I/O)**
  - *stdio.h*
  - Inclui funções para a entrada (leitura) e saída (escrita) de dados
    - » *printf()*
    - » *scanf()*
    - » *getchar()*
    - » *putchar()*

3.3.5. Comandos de Entrada e Saída

- **Função *printf()***
  - Permite a escrita de dados *na* tela do computador
  - *printf (“texto-de-controle”, argumentos)*
  - Associa valores dos argumentos ao texto de controle

```
/* Programa: comando printf */
# include <stdio.h>

main (){
    int x=0;
    printf (“valor de x = %d \n”, x);
}
```

3.3.5. Comandos de Entrada e Saída

- **Função *printf( )***
  - Texto de controle pode conter
    - » Caracteres que serão exibidos na tela do computador
    - » Código de formatação
      - Indica os formatos usados para imprimir os argumentos
  - Argumentos
    - » Separados por vírgula
    - » Podem ser variáveis, constantes e/ou funções

3.3.5. Comandos de Entrada e Saída

- Códigos de formatação mais comuns

%d **%hd** **%ld**	Estes formatos mostram o valor como um número decimal dos tipos *int*, *short* e *long*, respectivamente. O sinal % pode ser seguido por um número especificando a largura mínima a ser usada na impressão do número Se o número é pequeno demais para preencher toda a largura, espaço extra é adicionado à esquerda, de forma que os números se alinhem à direita. Ex.: *printf ("val = %4d \n", val);* /* reserva 4 espaços para imprimir o valor da variável val */
%f	Utilizado para valores dos tipos *float* ou *double*. Ele exibe valores desses tipos com um ponto decimal. A precisão pode ser especificada indicando quantos dígitos devem ser exibidos à direita do ponto decimal. Ex.: *printf ("val = %5.3f \n", val);* /* imprime um número com 8 caracteres, 3 deles após o ponto */

3.3.5. Comandos de Entrada e Saída

- Códigos de formatação mais comuns

%g	Este formato também é usado para valores dos tipos *float* ou *double,* sendo semelhante ao formato %f quando o número cabe em um espaço pequeno. Números cuja magnitude é ou muito grande ou muito pequena (ex. 27000000.0 ou .00000000006) são representados mais compactamente usando notação científica (ex.: 2.7e+7 ou 6.0e-11). O formato %g pode também incluir um campo largura e precisão, embora a precisão no formato %g especifique o número de dígitos significativos (ao invés dos números a direita do ponto decimal)
%c	Utilizado para valores do tipo *char. Exibe apenas um caracter* Ex.: *printf ("letra= %c\n", var);* /* imprime um caracter */

3.3.5. Comandos de Entrada e Saída

- Outros códigos de formatação

%s	string de caracteres
%i	decimal (como o %d)
%e ou %E	notação científica
%o	octal
%x	hexadecimal
%%	exibe o sinal %
%p	exibe um ponteiro

3.3.5. Comandos de Entrada e Saída

- **Códigos especiais:**
  - Não são impressos, mas executam determinadas ações
  - Exemplos:

\n	passa para a próxima linha
\t	tab
\b	volta um caractere
\\	barra
\”	imprime aspas
\f	passa para a próxima folha
\0	nulo

3.3.5. Comandos de Entrada e Saída

- **Função *scanf( )***
  - Permite a leitura de dados do teclado
  - *scanf (“texto-de-controle”, argumentos)*
  - Associa valores do texto de controle aos argumentos

```
/* Programa: comando scanf */
# include <stdio.h>

main ( ){
        int x;
        printf (“Entre com o valor de x: ”);
        scanf(“%d”, &x);
}
```

3.3.5. Comandos de Entrada e Saída

- **Função *scanf( )***
  - Complemento da função *printf( )*
  - Texto de controle contém
    - » Código de formatação
      - Indica os formatos usados para ler os argumentos
  - Argumentos
    - » Separados por vírgula
    - » Endereços de variáveis (&x)

3.3.5. Comandos de Entrada e Saída

- **Função *scanf( )***
  - Operador de endereço: &
  - Endereço do primeiro byte ocupado pela variável

```
/* Programa: comando scanf 2*/
# include <stdio.h>

main(){
    int val;
    printf("Digite um numero: ");
    scanf("%d", &val);
    printf("\n o  numero eh: %d", val);
    printf("\n o  endereco eh: %d", &val);
}
```

3.3.5. Comandos de Entrada e Saída

- **Função *scanf( )***

  - Pode-se ler mais de um valor através de *scanf*
    - » Ex.: *scanf(“%d,%d”, &a,&b);*
    - » Deve ser evitado pois pede a entrada dos dados da mesma forma que aparece no comando
  - Para *strings*, não é necessário utilizar o operador &
    - » *scanf(“%s”, var);*

3.3.5. Comandos de Entrada e Saída

Exercício 3.3.5. Escrever um algoritmo e um programa em C que imprima a média aritmética de três números fornecidos pelo usuário?

Exercício 3.3.6. Escreva um algoritmo e um programa em C que, dados os dois catetos, imprima o valor da hipotenusa de um triângulo retângulo (dica: use a função *sqrt()* da biblioteca *math.h*).

Exercício 3.3.7. Escreva um algoritmo e um programa em C que leia uma temperatura na escala Celsius (ºC) e imprima essa temperatura na escala Fahrenheit (ºF). A fórmula de conversão é $^{\circ}F = \frac{9}{5}{}^{\circ}C + 32$

3.3.5. Comandos de Entrada e Saída

Exercício 3.3.8. As raízes de uma equação quadrática da forma $ax^2+bx+c=0$ são reais se e somente se o discriminante dado por b^2-4ac for maior ou igual a zero. Escreva um algoritmo e um programa em C que leia os valores dos coeficientes *a, b* e *c* e imprima o valor do discriminante.