MC-102 — Aula 03
Escrita/Leitura e Operações Aritméticas

Alexandre M. Ferreira

IC – Unicamp

2017

Roteiro

A função **scanf**

- Realiza a leitura de dados 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.

A função **scanf**

O programa abaixo é composto de quatro passos:

1. Cria uma variável **n**
2. Escreve na tela "Digite um número:".
3. Lê o valor do número digitado.
4. Imprime o valor do número digitado.

```
#include <stdio.h>
int main(){
  int n;
  printf("Digite um número: ");
  scanf("%d",&n);
  printf("O valor digitado foi %d\n",n);
}
```

Formatos de leitura de variável

Os formatos de leitura são muito semelhantes aos formatos de escrita utilizados pelo **printf**. A tabela a seguir mostra alguns formatos possíveis de leitura

Código	Função
%c	Lê um único caracter
%s	Lê uma série de caracteres
%d	Lê um número decimal
%u	Lê um decimal sem sinal
%ld	Lê um inteiro longo
%f	Lê um número em ponto flutuante
%lf	Lê um double

A função **scanf**

O programa abaixo, lê um caracter, depois um número ponto flutuante e por fim um decimal. Por fim o programa imprime os dados lidos.

```
#include <stdio.h>

int main(){
  char c;
  float b;
  int a;

  printf("Entre com um caracter:");
  scanf("%c", &c);
  printf("Entre com um ponto flutuante:");
  scanf("%f", &b);
  printf("Entre com um número:");
  scanf("%d",&a);

  printf("Os dados lidos foram: %c, %f, %d \n",c,b,a);
}
```

Note que no **scanf**, cada variável para onde será lido um valor, deve ser precedida do caracter **&**.

Exercício

Qual o valor armazenado na variável **a** no fim do programa?

```
int main(void){
    int a, b, c, d;

    d = 3;
    c = 2;
    b = 4;
    d = c + b;
    a = d + 1;
    a = a + 1;

}
```

Exercício

Compile o programa abaixo? Você sabe dizer qual erro existe neste programa?

```
int main(void){
    int a, b;
    double c,d;
    int g;

    d = 3.0;
    c = 2.4142;
    b = 4;
    d = b + 90;
    e = c * d;
    a = a + 1;

}
```

Exercício

- Crie um programa que:
  - Lê um caracter, pula uma linha e imprime o caracter lido.
  - Lê um inteiro, pula uma linha e imprime o inteiro lido.
  - Lê um número ponto flutuante, pula uma linha e imprime o número lido.

Exercício

- Crie um programa que lê dois números **double** e que computa e imprime a soma, a diferença, a multiplicação e divisão dos dois números.

Informações Extras: Constantes Inteiras

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

Informações Extras: Constantes do tipo de ponto flutuante

- Na linguagem 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`
- Um número inteiro ou decimal seguido da letra **e** mais um expoente. Um número escrito dessa forma deve ser interpretado como:

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

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

Informações Extras: caractere

- São, na verdade, variáveis inteiras que armazenam um número associado ao símbolo. A principal tabela de símbolos utilizada pelos computadores é a tabela ASCII (*American Standard Code for Information Interchang*), mas existem outras (EBCDIC, Unicode, etc ..).
- `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.
- Toda constante do tipo caractere pode ser usada como uma constante do tipo inteiro. Nesse caso, o valor atribuído será o valor daquela letra na tabela ASCII.

Informações Extras: Tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 16	caracterees 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	{	—	}	~	

Informações Extras: Obtendo o tamanho de um tipo

O comando `sizeof(tipo)` retorna o tamanho, em bytes, de um determinado tipo. (Um byte corresponde a 8 bits).

Exemplo

```
printf ("%d", sizeof(int));
```

Escreve 4 na tela.

Expressões

- Já vimos que constantes 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 Aritméticas

- Os operadores aritméticos são: +, -, *, /, %
- *expressão* + *expressão* : Calcula a soma de duas expressões.
  Ex: 10 + 15;
- *expressão* - *expressão* : Calcula a subtração de duas expressões.
  Ex: 5 - 7;
- *expressão* * *expressão* : Calcula o produto de duas expressões.
  Ex: 3 * 4;

Expressões

- *expressão* / *expressão* : Calcula a divisão de duas expressões.
  Ex: 4 / 2;
- *expressão* % *expressão* : Calcula o resto da divisão (inteira) de duas expressões.
  Ex: 5 % 2;
- - *expressão* : Inverte o sinal da expressão.
  Ex: -5;

Expressões

Mais sobre o operador resto da divisão: %

- Quando computamos "a dividido por b", isto tem como resultado um valor p e um resto $r < b$ que são únicos tais que

$$a = p * b + r$$

- Ou seja a pode ser dividido em p partes inteiras de tamanho b, e sobrará um resto $r < b$.

Exemplos:
5%2 tem como resultado o valor 1.
15%3 tem como resultado o valor 0.
1%5 tem como resultado o valor 1.
19%4 tem como resultado o valor 3.

Expressões

No exemplo abaixo, quais valores serão impressos?

```
#include <stdio.h>

int main(){

  printf("%d \n", 27%3);
  printf("%d \n", 4%15);
}
```

Expressões

Mais sobre o operador /

- Quando utilizado sobre valores inteiros, o resultado da operação de divisão será inteiro. Isto significa que a parte fracionária da divisão será desconsiderada.
  - 5/2 tem como resultado o valor 2.
- Quando pelo menos um dos operandos for ponto flutuante, então a divisão será fracionária. Ou seja, o resultado será a divisão exata dos valores.
  - 5.0/2 tem como resultado o valor 2.5.

Expressões

No exemplo abaixo, quais valores serão impressos?

```
#include <stdio.h>

int main(){
  int a=5, b=2;
  float c=5.0, d=2.0;

  printf("%d \n",a/b);
  printf("%f \n", a/d);
  printf("%f \n", c/d);
}
```

Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo:
  `a = b *( (2 / c)+(9 + d * 8) );`

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

Precedência

- Precedência é a ordem na qual os operadores serão avaliados quando o programa for executado. Em C, os operadores são avaliados na seguinte ordem:
  - * e /, na ordem em que aparecerem na expressão.
  - %
  - + e -, na ordem em que aparecerem na expressão.
- Exemplo: 8+10*6 é igual a 68.

Alterando a precedência

- (*expressão*) também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.
  - $5 + 10 \% 3$ é igual a 6
  - $(5 + 10) \% 3$ é igual a 0
- Você pode usar quantos parênteses desejar dentro de uma expressão.
- Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

Incremento(++) e Decremento(−−)

- É muito comum escrevermos expressões para incrementar/decrementar o valor de uma variável por 1.

```
a = a + 1;
```

- Em C, o operador unário ++ é usado para incrementar de 1 o valor de uma variável.

```
a = a + 1; é o mesmo que a++;
```

- O operador unário −− é usado para decrementar de 1 o valor de uma variável.

```
a = a - 1; é o mesmo que a--;
```

Exercício

- Crie um programa que:
  - Lê um caracter, pula uma linha e imprime o caracter lido.
  - Lê um inteiro, pula uma linha e imprime o inteiro lido.
  - Lê um número ponto flutuante, pula uma linha e imprime o número lido.

Exercício

- Crie um programa que lê dois números **double** e que computa e imprime a soma, a diferença, a multiplicação e divisão dos dois números.

Informações Extras: Incremento(++) e Decremento(−−)

Há uma diferença quando estes operadores são usados à esquerda ou à direita de uma variável e fizerem parte de uma expressão maior:

- **++a** : Neste caso o valor de **a** será incrementado antes e só depois o valor de **a** é usado na expressão.
- **a++**: Neste caso o valor de **a** é usado na expressão maior, e só depois é incrementado.
- A mesma coisa acontece com o operador −−.

O programa abaixo imprime "b: 6".

```
#include <stdio.h>

int main(){
  int a=5, b, c;

  b = ++a;

  printf(" b: %d \n",b);
}
```

Já o programa abaixo imprime "b: 5".

```
 #include <stdio.h>

int main(){
  int a=5, b, c;

  b = a++;

  printf(" b: %d \n",b);
}
```

Informações Extras: 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
```

Informações Extras: Atribuições simplificadas

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

Informações Extras: Conversão de tipos

- É possível converter alguns tipos entre si.
- Existem duas formas de fazê-lo: implícita e explícita:
- Implícita
  - Capacidade (tamanho) do destino deve ser maior que a origem senão há perda de informação.
    Ex.: `int a; short b; a = b;`
    Ex: `float a; int b=10; a = b;`
- Explícita:
  - Explicitamente informa o tipo que o valor da variável ou expressão é convertida.
    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

Informações Extras: 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 `3.33333333`

Informações Extras: comentários

- O código fonte pode conter comentários direcionados unicamente ao programador. Estes comentários devem estar delimitados pelos símbolos `/*` e `*/`, e são ignorados pelo compilador.

Exemplo

```
#include <stdio.h>

/* Este é o meu primeiro programa. */
//Isto tambem é um comentário
int main() {
  printf("Hello, world!\n");
}
```

- Comentários são úteis para descrever o algoritmo usado e para explicitar suposições não óbvias sobre a implementação.