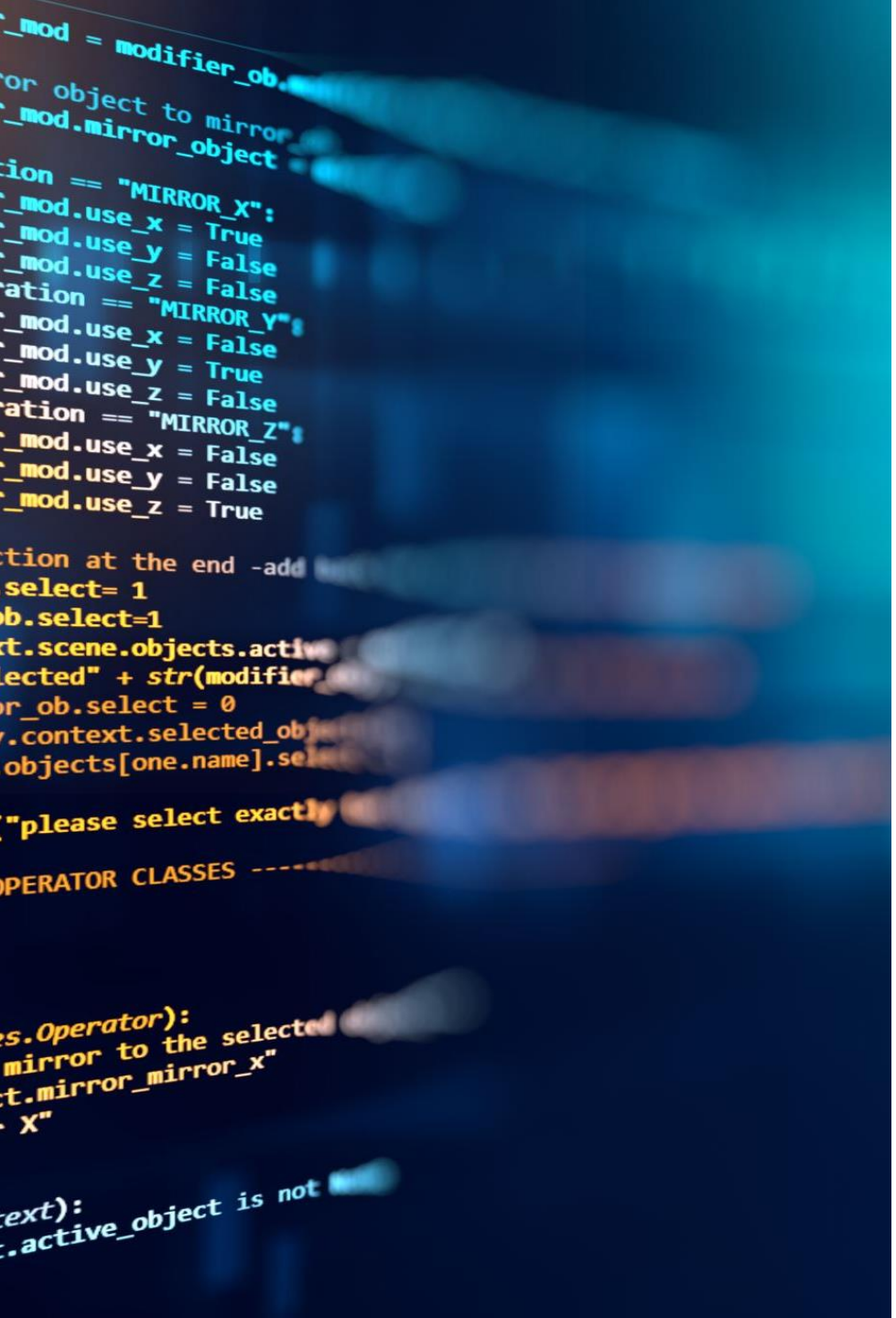


PROGRAMAÇÃO DE COMPUTADORES I

TERCEIRA PARTE

CAPÍTULO II
CONSTANTES, VARIÁVEIS E TIPOS DE DADOS;

PARTE 2:
VARIÁVEIS CARACTERE E CADEIA DE CARACTERES

Linguagem de Programação C

prof. Marco Villaça

Tipo caractere

- Armazenados nos computadores como codificações numéricas:
  - ✓ ASCII – American Standart Code for Information Interchange – provê 256 caracteres, apenas 128 caracteres de forma não ambígua; os outros 128 dependem da linguagem.
  - ✓ Unicode – inclui os caracteres da maioria das linguagens naturais internacionais.

    O padrão define 17 segmentos de código (planos), cada um com 65536 caracteres (2^{16}), totalizando 1.114.112 caracteres (17 x 2^{16}).

    São codificados (representados em bytes) de diversas formas: UTF-8, UTF-16 e UTF-32.

UTF-8

- UTF - Formato de Transformação Unicode.
- UTF-8 :
  - Sistema de codificação para Unicode em unidades de 1 byte.
  - O mais popular
- O UTF-8 converte um ponto de código (que representa um único caractere em Unicode) em um conjunto de um a quatro bytes:
  - Os primeiros 256 caracteres na biblioteca Unicode - que incluem os caracteres que em ASCII - são representados com um byte.
  - Os caracteres que aparecem posteriormente na biblioteca Unicode são codificados como unidades binárias de dois, três bytes e, eventualmente, quatro bytes.

UTF-8
Vantagens

- A eficiência espacial:
  - Se todo o caractere Unicode fosse representado por 4 bytes, um arquivo de texto em inglês teria 4 vezes o tamanho do mesmo arquivo codificado com UTF-8.
- Compatibilidade com o código ASCII:
  - Os primeiros 128 caracteres da biblioteca Unicode correspondem aos da biblioteca ASCII e o UTF-8 mantém essa correspondência.

Tipo caractere

- O C possui um tipo para representar um único caracter → char.
- O Scilab têm um tipo capaz de armazenar um caractere ou um conjunto de caracteres (uma string).

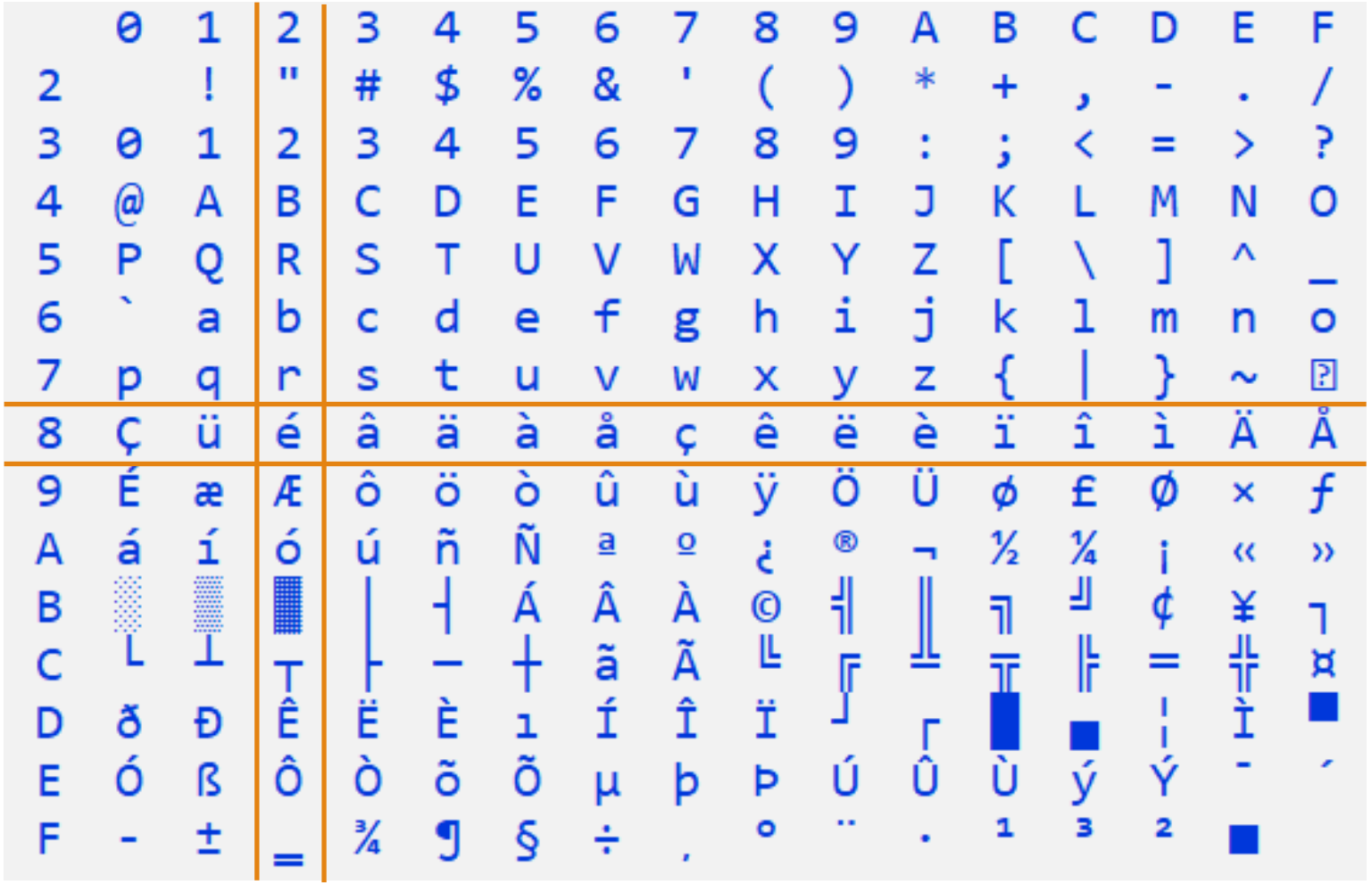
Tipo caractere no C

- Representa um único caractere
- Caracteres entre apóstrofos (char letra =‘a’);
- No printf e no scanf, identificados por %c
- Caracteres especiais antecedidos por “\”
- Exemplos:
  - \n – nova linha;
  - \t–tabulação horizontal.

Conjunto de caracteres ASCII estendido para IBM PC (CP 437 ou MS-DOS LATIN US)

- Uma forma de apresentar caracteres acentuados em ambiente DOS é consultando a tabela a seguir.
- Exemplo:

  é = \x82



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	ø	£	Ø	×	ƒ
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	®	¬	½	¼	¡	«	»
B	▒	▒	▓	|	┤	Á	Â	À	©	╣	‖	╗	╝	¢	¥	┐
C	L	┴	┬	├	─	┼	ã	Ã	╚	╔	╩	╦	╠	=	╬	¤
D	ð	Ð	Ê	Ë	È	ı	Í	Î	Ï	┘	┌	█	▄	¦	Ì	▀
E	Ó	ß	Ô	Ò	õ	Õ	µ	þ	Þ	Ú	Û	Ù	ý	Ý	-	´
F	-	±	‗	¾	¶	§	÷	¸	°	¨	·	¹	³	²	■	

Tipo caractere em C

```
#include <stdio.h>

int main()
{
     char c1 = 'A';     /* caractere 1*/
     char c2 = 'B';     /* caractere 2*/
     printf("%c%c invertido \x82 %c%c\n",c1,c2,c2,c1);
     return (0);
}
```

Conjunto de caracteres ASCII estendido para IBM PC - Português

- Uma forma de apresentar caracteres acentuados em ambiente DOS, é

incluir no código fonte o arquivo de cabeçalho

```
#include <locale.h>
```

e a linha de comando

```
setlocale(LC_CTYPE,"portuguese"); //character type
```

após a declaração de variáveis no programa principal

Conjunto de caracteres ASCII estendido para IBM PC - Português

- Para ler e apresentar corretamente caracteres acentuados no português, **no Windows** é necessário alterar a página de código de entrada usada pelo console para 1252 (extensão do ISO Latin-1).
- Assim, inclua o arquivo de cabeçalho

```
#include <windows.h>
```

e as linhas de comando

```
SetConsoleCP(1252);
SetConsoleOutputCP(1252);
```

após a declaração de variáveis no programa principal

Conjunto de caracteres ASCII estendido para IBM PC - Português

- A nova tabela ASCII será (ISO Latin -1)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⍰
8	?	?	'	ƒ	"	.	┼	╬	^	%	S	<	O	?	Z	?
9	?	'	'	"	"		-	-	~	T	s	>	o	?	z	Y
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Lendo caracteres a partir do teclado

- `getchar( )`: Lê um caractere do teclado e espera pelo retorno de carro (<enter>);
- `getche( )`: Lê um caractere do teclado com eco e não espera pelo retorno de carro. Não é definido pelo ANSI (no windows usar conio.h);
- `getch( )`: Lê um caractere do teclado sem eco e não espera pelo retorno de carro. Não é definido pelo ANSI (no windows usar conio.h);
- Exemplo:

```
char sexo;
sexo = getche( );
```

EXEMPLO 1

- Escreva um programa que solicite dois caracteres para o usuário, imprima os caracteres digitados e os códigos ASCII correspondentes a estes caracteres.

Tipo cadeia de caracteres ou strings

- Sequência de caracteres
- C não tem um tipo específico (built-in) para strings
  - ✓ Em C Strings são um array de chars (ou seja, um vetor de caracteres)
    - Com a restrição de que o último elemento seja '\0' (NULL) para indicar o fim da string
- O Scilab, por sua vez, trata a cadeia de caracteres como um tipo primitivo

Strings em C

- Vetores em C:
  - ✓ Vetor é um grupo consecutivo de posições de memória de mesmo tipo
  - ✓ Cada item do vetor é um elemento
  - ✓ Número de elementos é a dimensão
- Declaração: `char nome[tam];`
  - ✓ Ex: `char vogais[5];`
- Referência a um elemento pelo índice
  - ✓ `vogais[0] = 'a';`

Strings em C

- MUITO IMPORTANTE!
  - ✓ Em char `vogais[5];`
  - ✓ Os elementos são referenciados de 0 a 4
- Primeiro elemento de um vetor em C tem índice 0

Atribuição de valores para strings em C

- Válido:

```
int main()
{
    char nome[4];
    nome[0] = 'u';
    nome[1] = 'v';
    nome[2] = 'a';
    nome[3] = '\0';
    return (0);
}
```

ou:

```
char nome[4] = “uva”;
```

- Não é válido em C:

```
char nome[4];

int main()
{
    nome = “uva”;
    return (0);
}
```

Erro na compilação!

Atribuição de valores para strings em C

- Atribuição de valor com `strcpy`
- `strcpy` é uma função das bibliotecas padrão do C

```
#include <string.h>
int main()
{
    char nome[4];
    strcpy(nome, “uva”);
    return 0;
}
```

- No printf, string com o especificador %s

Mais funções para strings em C

- Outras funções da biblioteca `<string.h>`

FUNÇÃO	DESCRICÃO
`strcpy (string1, string2)`	Copia a string2 para a string1
`strcat (string1, string2)`	Concatena a string2 no final da string1
`extensao = strlen (string)`	Obtém o número de caracteres da string
`strcmp (string1, string2)`	Retorna 0 se string1 igual a string2 Retorna <0 se os 1º caractere que não corresponde tem menor valor em string1 do que em string2 Retorna >0 se os 1º caractere que não corresponde tem maior valor em string1 do que em string2

Lendo strings pelo teclado

- `fgets(string,tamanho,stdin);`
- Lê de `stdin` (entrada padrão) para a cadeia de caracteres `string` até a quantidade de caracteres "`tamanho - 1`" ser lida ou até uma nova linha ('\n')
- String terminada com '\0'
- A função `gets(string)` não é recomendada (*deprecated* – descontinuada) pois pode acontecer do usuário digitar mais caracteres do que o definido, causando um erro no programa.

Lendo strings pelo teclado com `fgets`

- O enter presente na string causa uma troca de linha inconveniente ao se utilizar a função printf.
- Para retirá-lo, uma possibilidade é o uso da função `strcspn(string, key)`.
- Esta função retorna o número de caracteres da `string` antes dos caracteres que formam a string `key`. No caso empregamos:

  `string[strcspn(string, "\n")] = '\0'.`

  Substituindo o '`\n`' pelo '`\0`'

EXEMPLO 2

Faça um programa que defina uma variável para nome

- Coloque seu nome na variável usando `fgets;`
- E mostre na tela o seu nome e a quantidade de letras que ele contém.

EXEMPLO 3

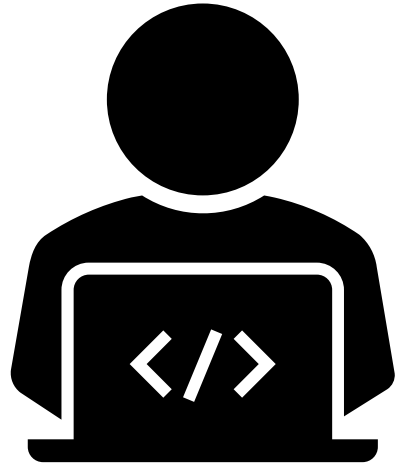
Fazer um programa que pergunte ao usuário:

- O nome e armazene em uma variável.
- O sobrenome armazene em outra variável.
- Juntar as variáveis em uma única variável `nome_completo;`
- Apresentar o resultado na tela

Exercício 1

- Fazer um programa que pergunte ao usuário seu nome completo, guarde em uma variável e separe e imprima seu primeiro nome utilizando outra variável.
- Pesquise e utilize as funções `strcspn` e `strncpy`
- Exemplo de uso

```
Digite seu nome completo: Washington Luis Pereira de Souza
Bem-vindo, Washington
Process returned 0 (0x0)   execution time : 64.237 s
Press any key to continue
```



CAPÍTULO II
CONSTANTES, VARIÁVEIS E TIPOS DE DADOS;

PARTE 2:
VARIÁVEIS CARACTERE E CADEIA DE CARACTERES
E/S FORMATADAS

Linguagem de Programação C

prof. Marco Villaça

Tipo caractere no Scilab

- Mesmo tipo para um caractere e para uma cadeia de caracteres.
- Strings são escritas entre aspas, simples ou duplas. Elas até podem iniciar com aspas duplas e terminá-las com aspas simples.

```
--> a = "Programação I";
```

- Uma operação comum com strings é a concatenação. Seu efeito é combinar dois operandos do tipo caracter em uma única cadeia resultante, de extensão igual a soma das extensões dos operandos originais .

Tipo caractere no Scilab

- No Scilab a concatenação utiliza o mesmo símbolo da adição numérica, o **“+”**.

```
-->a = 'Programação de ';
-->b = 'Computadores I';
-->Unidade = a + b
 Unidade  =
Programação de Computadores I.
```

Tipo caractere no Scilab

- Lembre-se que strings podem ser lidas pelo comando input, utilizando o parâmetro **s** na sequência de chamada.

```
-->Nome = input("Digite seu nome: ","s");
Digite seu nome: Marco

-->Nome
 Nome  =

 Marco
```

Utilizando o `scanf` para caracteres e strings

- `scanf( )` também pode ser utilizada para ler caracteres `(%c)`.
- A exemplo de `getchar( )`, ela guarda a entrada em um buffer, o que faz necessário apertar a tecla `<enter>` para a entrada do caractere
- `scanf( )`, pode, ainda ser usada para ler uma string `(%s)`. Entretanto, ao contrário de `gets( )`, `scanf( )` lê a string até encontrar um espaço em branco.

Utilizando o `scanf` para strings

- Para ler uma string com espaço basta usar:

```
scanf("%[^\n]s", variavel);
```

- Isto fará `scanf( )` ler até se teclar `<enter>`.
- É possível, também, limitar os caracteres que podem ser lidos. O comando abaixo, por exemplo, lê todos os caracteres do alfabeto e o espaço:

```
scanf("%[a-z A-Z]s", variavel);
```

Limpeza do buffer

- Ao se utilizar `scanf( )` com caracteres e strings, o `<enter>` digitado permanecerá no buffer, fazendo com que ele possa, em algumas situações, pular um próximo `scanf( )`.
- Para que isto nunca ocorra, deve-se limpar o buffer antes da próxima utilização de `scanf( )`.
- Com esse fim no Windows/ Linux utiliza-se:

```
fflush(stdin) / __fpurge(stdin);
```

CÓDIGO PARA ENTRADA E SAÍDA FORMATADAS

`%[flags][largura][.precisão][conversão]especificador`

"%-+ 0 w.p m c" – c é o especificador de tipo

d, i – inteiro

u – sem sinal

c – caractere

s – string

f – float e double (printf)

float(scanf)

e, E – exponencial

lf – double(scanf)

O – octal

x, X-hexadecimal

P – ponteiro

g, G – melhor formato para float e double

CÓDIGO PARA ENTRADA E SAÍDA FORMATADAS

%[**flags**][**largura**][**.precisão**][**conversão**]especificador

“%-+ 0 w.p m c”

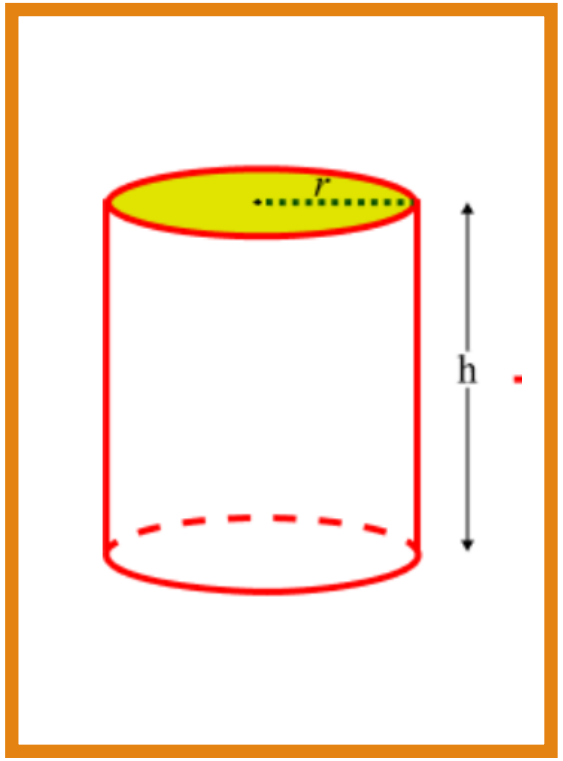
- ✓ -: justificado à esquerda;
- ✓ +: apresenta com sinal;
- ✓ espaço: apresenta espaço se sem sinal;
- ✓ 0: preenche com zeros à esquerda;
- ✓ w: largura mínima do campo
- ✓ p: casas decimais
- ✓ m: caractere de conversão:

  h – short l, L – long

CÓDIGO PARA ENTRADA E SAÍDA FORMATADAS

- Exemplo

```
double x = 300.45;
printf("x = %+10.3f",x);
// Saída: x =     +300.450
printf("x = %-10.3f",x);
// Saída: x = 300.450
printf("x = %010.3f",x);
// Saída: x = 0000300.450
printf("x = %+10.3e",x);
// Saída: x = +3.004e+002
```



EXEMPLO 1

- Escrever um programa para calcular a área de um cilindro:

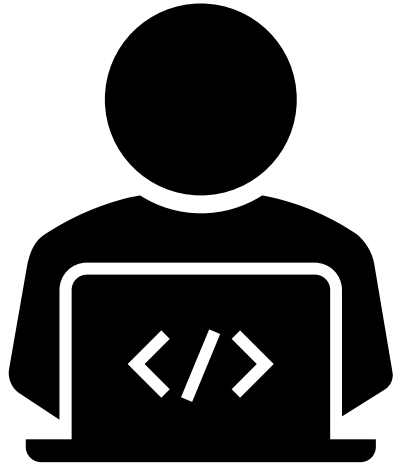
$$A = A_B + A_L = 2\pi r^2 + 2\pi r \cdot h$$

$$A = 2\pi r(r + h)$$

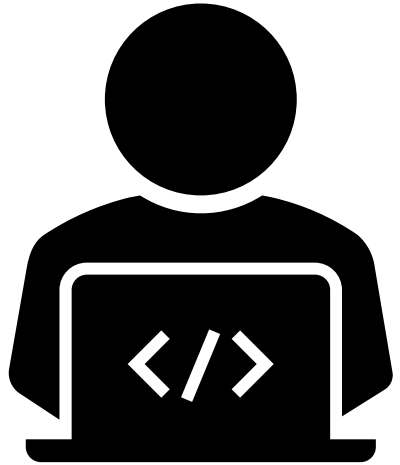
- Mostrar o resultado na tela:
- Justificado à esquerda
- Com no mínimo 12 campos
- Com duas casas decimais

Exercício 2

- Calcular número total de minutos a partir da entrada horas e minutos
- Exemplo: 1 h 25 min → 85 min.



- Calcular número de horas e minutos a partir da entrada minutos
- Exemplo: 85 min → 1 h 25 min



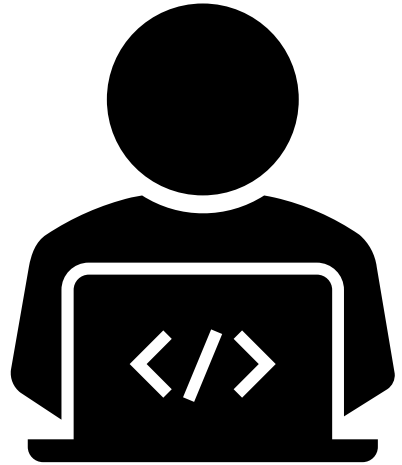
Exercício 3

Exercício 4

- Elabore um programa que peça uma letra minúscula ao usuário e imprima o caractere e a letra maiúscula correspondente a este caractere.
- Presuma, por enquanto, que o usuário realmente digite uma letra minuscula.
- No C

  Pesquise e utilize a função disponível na biblioteca `ctype` do ANSI C.
- No Scilab:

  Pesquise no menu ajuda em **Cadeias de Caracteres (Strings)**



Bibliografia e crédito das figuras

OUALLINE, S. Practical C Programming. 3a ed. O'Reilly, 1997.

SEBESTA, R. Conceitos de Linguagens de Programação. 5a ed. Porto Alegre: Bookman, 2003.

FORBELLONE, A. L. V.; Eberspacher, H. F. Lógica de Programação – A construção de Algoritmos e Estrutura de Dados. 2ª. Ed. São Paulo: Pearson/Makron Books, 2000.

ASHLEY, Stephen. The Fundamentals of C. 1a ed. Kindle Edition.

http://help.scilab.org/docs/6.1.0/pt_BR/index.html

Compiler, assembler, linker and loader: a brief story. Disponível em: http://www.tenouk.com/ModuleW.html

http://www.programmingbasics.org