



PROGRAMAÇÃO DE COMPUTADORES I

SEGUNDA PARTE

“A tecnologia moderna é capaz de realizar a produção sem emprego. O diabo é que a economia moderna não consegue inventar o consumo sem salário.”

Herbert de Souza (Betinho)

CAPÍTULO II
CONSTANTES, VARIÁVEIS
E TIPOS DE DADOS;

PARTE 1:
VARIÁVEIS INTEIRAS E
REAIS
ENTRADA E SAÍDA DE
DADOS

Programação de Computadores I

prof. Marco Villaça

VARIÁVEIS

- De forma simples, uma variável é um “pedaço” nomeado da memória de um computador onde se guardam valores.
- Em uma definição mais formal, uma variável é uma abstração de uma ou mais células de memória de um computador
- Como a memória guarda informações, a variável pode ser vista como um recipiente de dados.
- Vamos lembrar como “batizar” esse espaço de memória (dar um nome e definir o tipo)
- E como incluir (atribuir) um valor



VARIÁVEIS

- Uma variável pode ser caracterizada por 6 atributos:
 - nome
 - endereço
 - tipo
 - valor
 - tempo de vida
 - escopo

absolute; z-index:
px #ccc}.gbrtl .gb
play:block;positi
ty:1;*top:-2px;*l
p:-4px\0/;left:-6
ox;display:inline
lay:block;list-st
lock;line-height:
nter;display:bloc
e;z-index:1000}.gb
ding-right:9px}#gb
rl//

NOME DE VARIÁVEIS

- Nome: cadeia de caracteres que identifica uma entidade (variáveis, constantes, rotinas, etc.) de um programa.

DECLARAÇÃO DE VARIÁVEIS

- **Declaração:** nome e tipo (valor e comentário)
- **Declaração explícita:** é uma instrução em um programa que lista nome de variáveis e especifica que elas são de um tipo particular
- Em C:
 - `int num_dias; /* número de dias de aula */`
 - No Scilab para declarar uma variável basta lhe atribuir um valor
- O C e o Scilab são **case sensitive**, ou seja, diferenciam letras maiúsculas de minúsculas).
 - Em `int Mes, mes;`
Mes e mes são variáveis diferentes.

DECLARAÇÃO DE VARIÁVEIS

- **Declaração implícita:** é um meio de associar variáveis a tipos por convenções em vez de instruções
 - O Fortran é uma linguagem que dispõe de declarações implícitas.
- Declaração implícita no Fortran:
 - Nomes iniciados com **i, j, k, l, m, n** são implicitamente declarados como do tipo **integer**, caso contrário são do tipo real;
 - Prejudiciais a legibilidade, variáveis deixadas de declarar acidentalmente recebem tipo padrão.
- **O C não utiliza declaração implícita.**

A decorative graphic on the left side of the slide featuring a pile of 3D, light blue letters and numbers scattered on a white surface, casting soft shadows.

DECLARAÇÃO DE VARIÁVEIS

- Em C os caracteres válidos para declarar uma variável são:
 - Letras (a - z; A - Z).
 - Numerais (0 - 9).
 - O caractere underscore “_”
- O Scilab ainda permite o uso dos caracteres:
 - '#', '!', '\$' e '?'
- Em C os 31 primeiros caracteres são significativos.
- No Scilab, apenas os 24 primeiros

DECLARAÇÃO DE VARIÁVEIS

- Tanto em C, quanto no Scilab, **não** vale:
 - Começar uma variável com número: 1a_nota
 - Utilizar espaço: media final
 - Usar operadores: media*final
- No C, não é permitido utilizar palavras reservadas como nome de variáveis

COMENTÁRIOS

- Comentários

- Trechos (mais de uma linha): entre `/*` e `*/`;

- Uma única linha: `//`.

- Na declaração de variáveis

- RUIM, por não fornecer informação relevante:

```
int n_dias; /* define n_dias como inteira */
```

- MELHOR:

```
int n_dias; /* número de dias de aula */
```

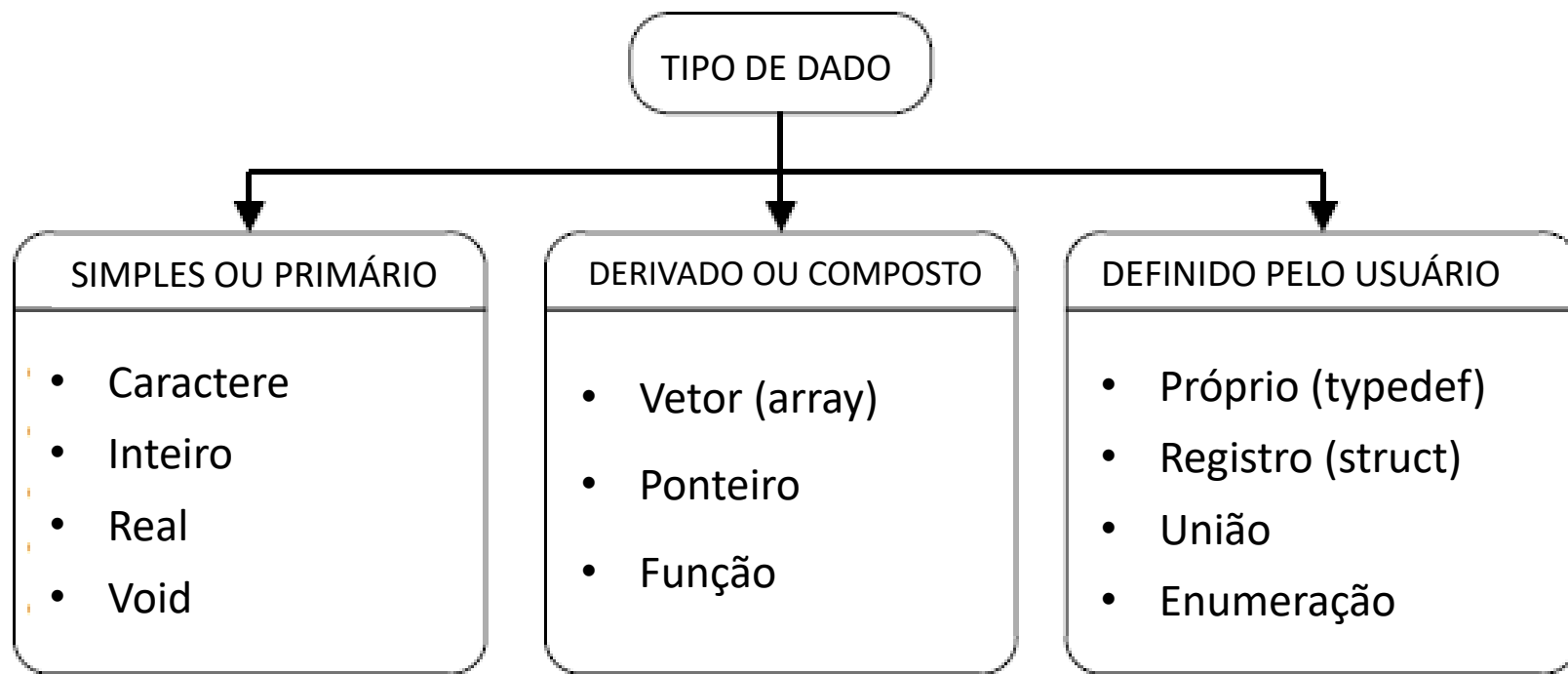
ENDEREÇO DE VARIÁVEIS

- Endereço: endereço de memória ao qual ela está associada
- Pode haver dois nomes para um mesmo endereço (aliases ou apelidos)
 - o valor da variável muda com uma atribuição a qualquer um de seus nomes
 - desvantagem: legibilidade
- Será discutido posteriormente, como os aliases são criados no C.



TIPOS DE DADOS

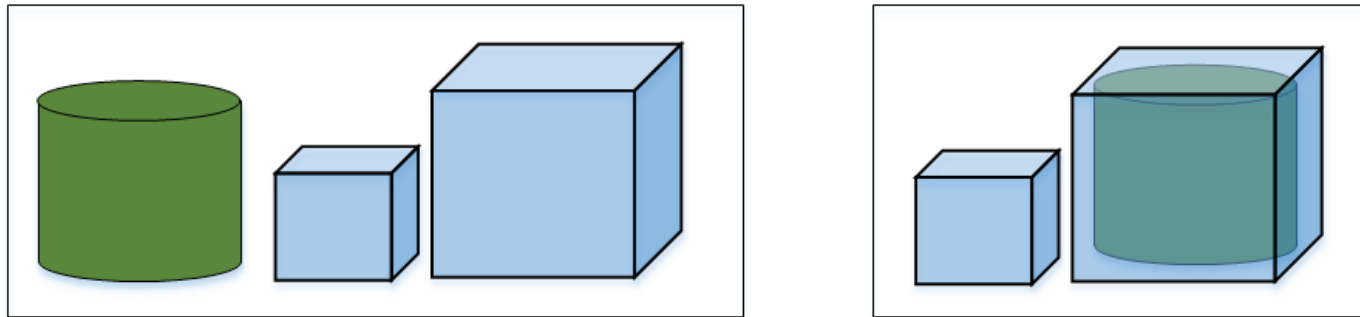
- No mundo real, usamos diferentes tipos de dados, como números inteiros, reais e caracteres.
- Para armazenar dados ou valores, precisamos de variáveis.
- A questão é, quantos blocos de memória são necessários para armazenar um valor?
- Veremos que se deve declarar uma variável instruindo o compilador a alocar o número adequado de blocos.



TAXONOMIA DOS TIPOS DE DADOS

TIPOS DE DADOS

- Na vida real os recipientes se adequam ao propósito. Similarmente, nas LP existem recipientes diferentes para armazenar dados diferentes.



- O tamanho do menor recipiente é de um byte, onde se pode guardar números positivos entre 0 e 255 ou negativos entre -128 a 127.
- Em C, as variáveis de um **byte** são do tipo **char**

TIPOS DE DADOS

- Para armazenar números maiores, os bytes podem ser agrupados formando um recipiente maior.
- Por exemplo, agrupando 2 bytes podem ser guardados números positivos entre 0 e 65535 ($2^{16} - 1$) ou negativos entre -32768 e 32767.
 - Em C, plataforma de 32 bits, as variáveis de 2 bytes são do tipo **short int**.
 - **short** → modificador de tipo.
 - Para representar apenas números positivos utilizar os modificadores `unsigned` e `short` justapostos, como em:
`unsigned short int dias_ano;`

TIPOS INTEIRO

- Em C → `int`
 - O tamanho do dado depende da plataforma/ compilador
 - Se 32 bits, vai de – 2.147.483.648 a 2.147.483.647
 - Usando o modificador **unsigned**, vai de 0 a 4.294.967.295

TIPOS INTEIRO

- Para o compilador gcc 32 bits:
 - long int = int = 4 bytes
 - Para variável de 8 bytes usar long long int:
 - Números de -9223372036854775808 a 9223372036854775807 ($9,233 \times 10^{18}$)
 - Usando o modificador unsigned: 0 a 18446744073709551615 ($1,845 \times 10^{19}$)
- No Scilab as variáveis numéricas são reais por padrão.



ATRIBUIÇÃO DE VALOR

- Em C, se atribui valor com o sinal de = .
- Não confundir com ==, operador relacional de comparação.
- Exemplo :

```
int resposta = 10;
```

TIPO REAL

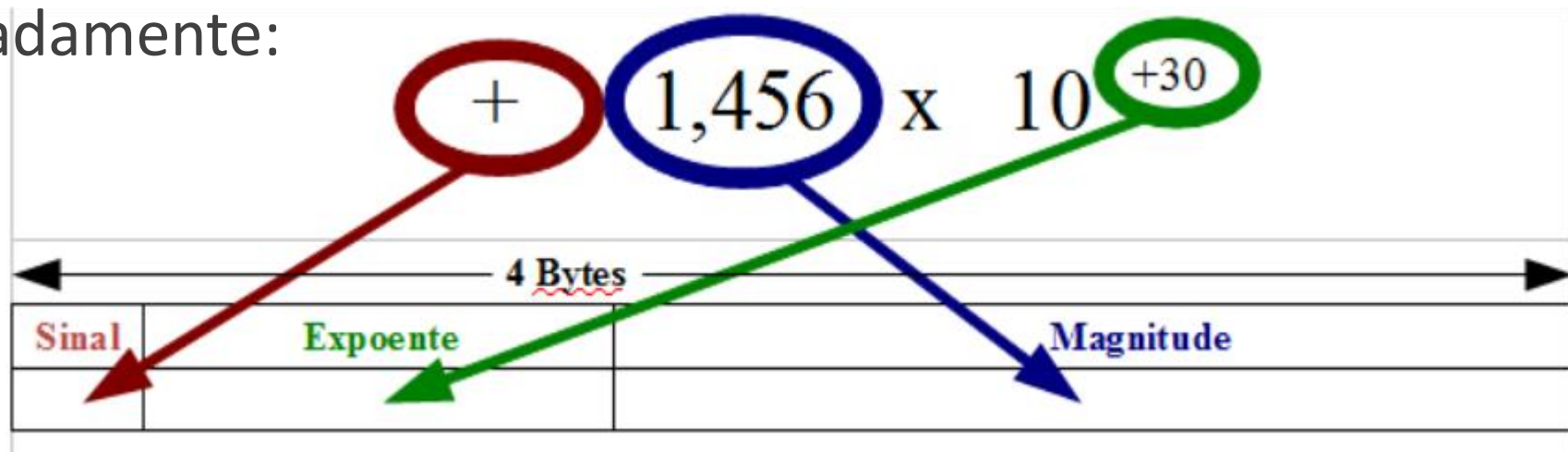
- Precisão simples :

Declaração em C: `float massa_sol = 1.989e30;`

Pode representa um número real entre

$-3,4 \times 10^{38}$ a: $+3,4 \times 10^{38}$, dispostos em 4 bytes.

- Simplificadamente:



TIPO REAL EM C

- Precisão dupla:

Declaração: `double`

Representa um número real entre $-1,79 \times 10^{308}$ a $+1,79 \times 10^{308}$,
dispostos em 8 bytes.

- Precisão estendida:

Declaração: `long double`

Representa um número real entre $-1,18 \times 10^{4932}$ a $+1,18 \times 10^{4932}$,
dispostos em 10 bytes

TIPO REAL EM C

- Menor número representado:

`float` → $1,17 \times 10^{-38}$;

`double` → $2,22 \times 10^{-308}$;

`long double` → $3,36 \times 10^{-4932}$;

- Precisão:

`float` → 7 algarismos significativos

`double` → 16 algarismos significativos

`long double` → 19 algarismos significativos

TIPO REAL EM NO SCILAB

- No Scilab, as variáveis numéricas são de dupla precisão por padrão.

LENDO VALORES DO TECLADO

Linguagem C

- Uma das funções utilizadas é o scanf.
- Definida em <stdio.h>
- Exemplos de uso:
 - Ler um valor inteiro do teclado e armazenar na variável n:

```
scanf ("%d", &n) ;
```
 - Ler dois valores reais (“com vírgula”) do teclado e armazenar na variável x e y:

```
scanf ("%f %f", &x, &y) ;
```

Especificador	Variável
%hhd ou %hhi	char
%hhu	unsigned char
%hd ou %hi	short int
%hu	unsigned short int
%d ou %i	int
%u	unsigned int
%ld ou %li	long int
%lu	unsigned long int

LENDO VARIÁVEIS INTEIRAS

Especificadores de formato

Especificador	Variável
<code>%lld</code> ou <code>%lli</code>	<code>long long int</code>
<code>%llu</code>	<code>unsigned long long int</code>
<code>%I64d</code> ou <code>%I64i</code>	<code>long long int</code> (mingw gcc)
<code>%I64u</code>	<code>unsigned long long int</code> (mingw gcc)
<code>%hhx</code> , <code>%hx</code> e <code>%x</code>	hexadecimal e suas variantes
<code>%hho</code> , <code>%ho</code> e <code>%o</code>	octal e suas variantes

LENDO VARIÁVEIS INTEIRAS

Especificadores de formato



LENDO VARIÁVEIS Reais

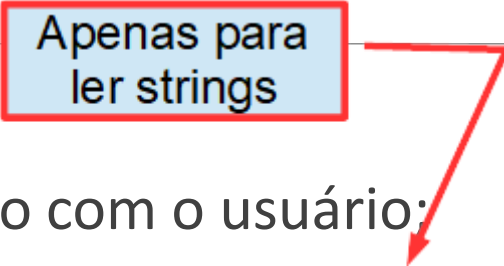
Especificadores de formato

- %f – float
- %lf – double
- %Lf – long double

LENDO VALORES DO TECLADO

Scilab

Apenas para
ler strings



■ Função input

- prompt para interação com o usuário:

```
x = input(mensagem [, "string"]);
```

- Exemplos:

```
/* Cálculo das raízes de uma eq. de seg. grau  $ax^2 + bx + c$  */
```

```
a = input("Digite o valor de a:");
```

```
b = input("Digite o valor de b:");
```

```
c = input("Digite o valor de c:");
```

```
// Lendo uma string(cadeia de caracteres)
```

```
nome = input("Digite seu nome:", "s");
```

APRESENTANDO NA TELA

Linguagem C

- Uma das funções utilizadas é o printf.
- Também definida em <stdio.h>
- Exemplo de uso:
 - Apresenta na tela a mensagem entre aspas:

```
printf("Bem vindo");
```
 - Apresenta na tela a mensagem entre aspas, substituindo o formatador %d pelo valor de n:

```
printf("A variavel n vale %d",n)
```

APRESENTANDO NA TELA

Linguagem C

■ Exemplo de uso:

- Apresenta na tela a mensagem entre aspas, substituindo os formatadores %d e %f pelos valores de n e x:

```
printf("n = %d e x = %f ", n, x);
```



Seção de formato

Lista de variáveis

```
printf ("n = %d e x = %f ', n, x) ;
```

The diagram shows the format string "n = %d e x = %f '" and the variable list n, x. The format specifiers %d and %f are circled in red and purple respectively. A red arrow points from the red circle to the variable n, and a purple arrow points from the purple circle to the variable x.

APRESENTANDO O TIPO REAL EM C

- Código de formato além de %f para float:

- %e, %g

```
const float PI = 3.141592;  
printf ("Reais: %4.2f %e %g \n", PI, PI, PI);  
printf ("Reais: %g \n", PI*10e8);
```

Resulta:

```
Reais: 3.14 3.141592e+000 3.14159
```

```
Reais: 3.14159e+009
```

APRESENTANDO O TIPO REAL EM C

- `%lf`, `%le` ou `%lg` - `double`
- `%Lf`, `%Le` ou `%Lg` - `long double` (sem suporte no mingw gcc)

APRESENTANDO NA TELA

Scilab

- Para exibir as variáveis na tela, utiliza-se a função disp.

```
--> a = [1 2]; b = 5 ; c = "abc";
```

```
--> disp(a,b,c)
```

```
1.    2.
```

```
5.
```

```
"abc"
```

- Pode-se, ainda, utilizar-se a função printf do C, emulada pelo Scilab

EXEMPLO

- O programa a seguir lê dois valores do teclado e apresenta a somas na tela:

```
#include <stdio.h>
int main( )
{
    int x, y;
    printf(" Primeiro valor inteiro : ");
    scanf("%d", &x);
    printf(" Segundo valor inteiro : ");
    scanf("%d", &y);
    printf("Soma = %d", x + y);
    return 0;
}
```

EXEMPLO

- Uma segunda versão do mesmo programa:

```
#include <stdio .h>
int main ( )
{
    int x, y, soma;
    printf("Entre com dois valores\n");
    printf("inteiros separados por espaço\n");
    scanf("%d %d", &x, &y);
    soma = x + y;
    printf("Soma = %d", soma);
    return 0;
}
```

Caracteres que
representam
nova linha

TIPO REAL

OBSERVAÇÕES

- O resultado da divisão de duas constantes inteiras resulta em um valor inteiro:

...

```
float x;
```

```
x = 4/3;
```

```
printf("O valor de x eh: %f", x);
```

...

- O valor apresentado na tela será **1.000000**, a parte após o ponto decimal não será apresentada corretamente

TIPO REAL

OBSERVAÇÕES

- Para resolver o problema basta transformar uma das constantes inteiras em um valor real:

...

```
float x;
```

```
x = 4.0/3;
```

```
printf("O valor de x eh: %f", x);
```

...

- O valor apresentado na tela será **1.333333**, pois a apresentação padrão é com seis casas decimais

TIPO REAL

OBSERVAÇÕES

- Com **variáveis**, resolve-se o problema usando um molde ou “**cast**” na frente da expressão, transformando a variável inteira em real:

...

```
float x; int n = 4;
```

```
x = (float)n/3;
```

```
printf("O valor de x eh: %f", x);
```

...

- O valor apresentado na tela será **1.333333**, pois a apresentação padrão é com seis casas decimais

TIPO REAL

OBSERVAÇÕES

- É possível limitar o número de casas decimais apresentadas:

...

```
float x; int n = 4;
```

```
x = (float)n/3;
```

```
printf("O valor de x eh: %.2f", x);
```

...

- O valor apresentado na tela será **1.33**, pois limitou-se em duas as casas decimais

Lembrando:

Operadores aritméticos no C e no Scilab

- Multiplicação: *
- Divisão: /
- Adição: +
- Subtração: –

Lembrando:

Operadores aritméticos no C e no Scilab

- Em C, existe o operador %
 - Resto após divisão inteira: $5 \% 2 \rightarrow 1$
- No Scilab, a operação de potenciação é realizado por
 - \wedge ou $**$.
- A potenciação em C é implementada pela função `pow` da biblioteca `math`
 - $3^3 \rightarrow \text{pow}(3,3)$

1. Elabore um programa que leia um número inteiro e imprima a sua raiz cúbica arredondada para o valor inteiro mais próximo.

Opção de compilador on line:

<https://repl.it/languages/c>



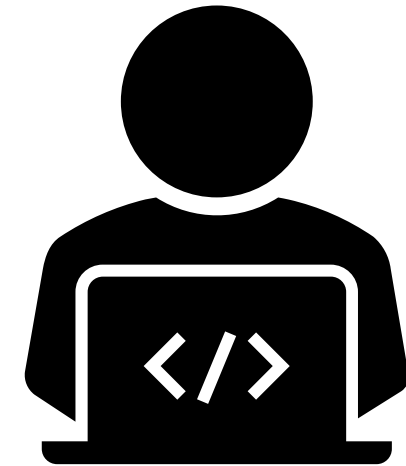
Elabore um programa para calcular e apresentar na tela as raízes de uma equação de segundo grau

$$ax^2 + bx + c$$

Receber pelo teclado o valor de a, b e c.



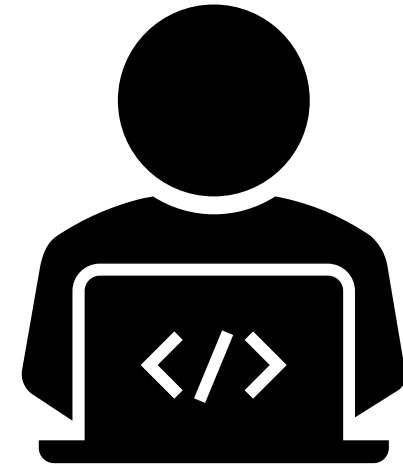
Elabore um programa que solicite pelo teclado o valor dos lados de um retângulo e apresente na tela o valor da sua diagonal.



Exercício 1

Elabore um programa que solicite pelo teclado o valor de dois resistores e calcule o resultado da associação em série e em paralelo dos mesmos.

- Use variáveis reais.

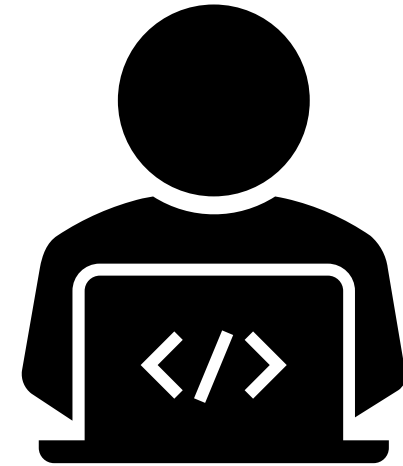


Exercício 2

Elabore um programa que transforme um número complexo na forma retangular para a fórmula polar.

A parte real e a parte imaginária devem solicitadas pelo teclado.

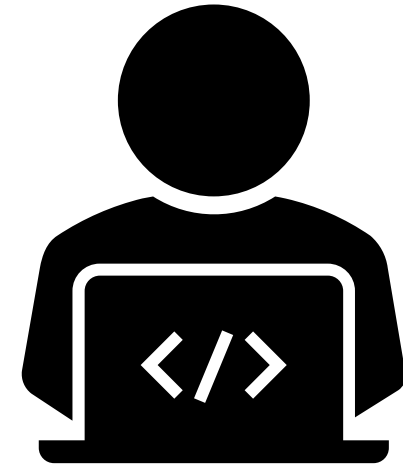
- Use variáveis reais.



Exercício 3

Elabore um programa que calcule o **acrécimo de uma prestação** em atraso, sabendo que são cobrados 2% de multa e 0,1% por dia de atraso (juros simples).

- Entrar com o valor da prestação pelo teclado.



Exercício 4

Referências



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>