UNIVERSIDADE FEDERAL DO CEARÁ

DEPARTAMENTO DE COMPUTAÇÃO

Programação

(CK0226 – 2022.2)

Curso: Ciência da Computação
Professor: Lincoln Souza Rocha
E-mail: lincoln@dc.ufc.br

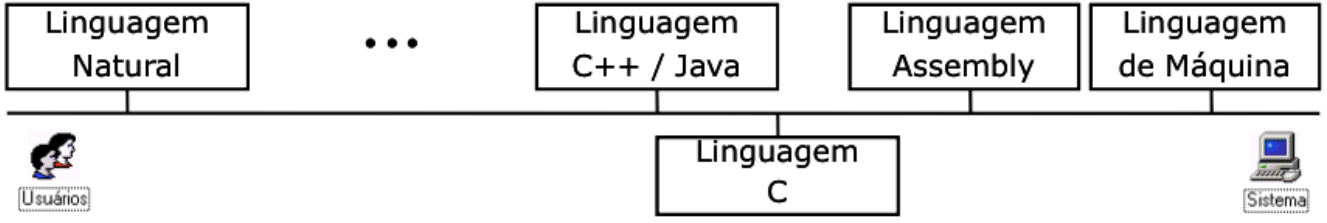
Introdução à Linguagem C – Parte I

Sumário

- Compilação de Programas
- Variáveis e Constantes
- Operadores e Expressões
- Entrada e Saída Padrão

Compilação de Programas

Linguagem Natural vs Linguagem de Máquina



Linguagem Natural:

Calcule a raiz de 9.

Linguagem C/C++:

```
#include <math.h>
  int main(void) {
  double x = 9.0;
  printf("%lf", sqrt(x));
}
```

Linguagem Assembly:

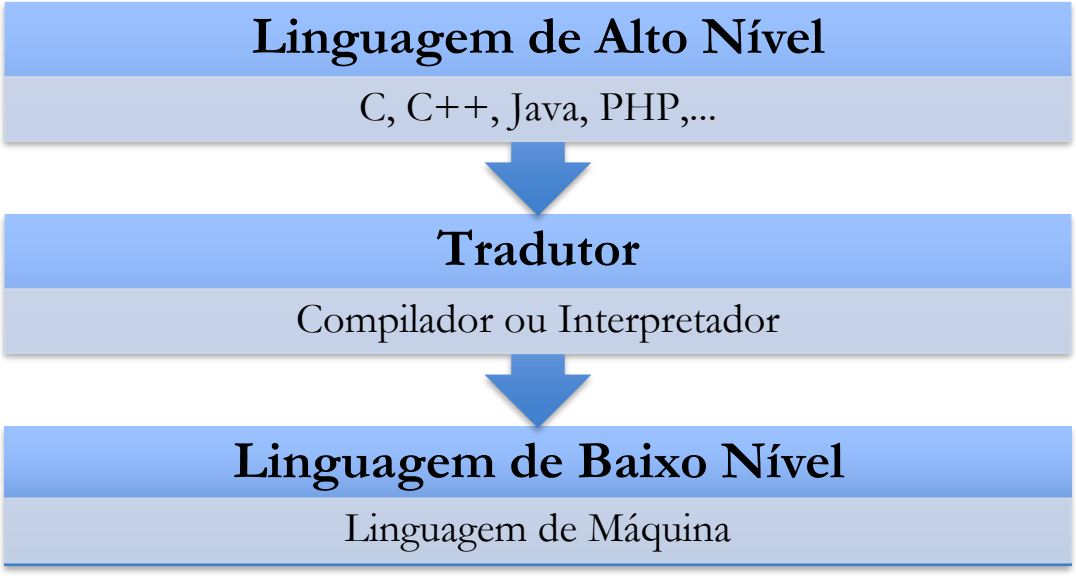
```
0000: mov bx, [FFE6]
0003: and bx, 1
0006: cmp bx, 0
0009: je 0000
000C: mov ax, [FFE4]
000F: nop
0010: mov dx, ax
0013: mov bl, al
0015: cmp al
0017: xor ax
. . .
```

Linguagem de Máquina:

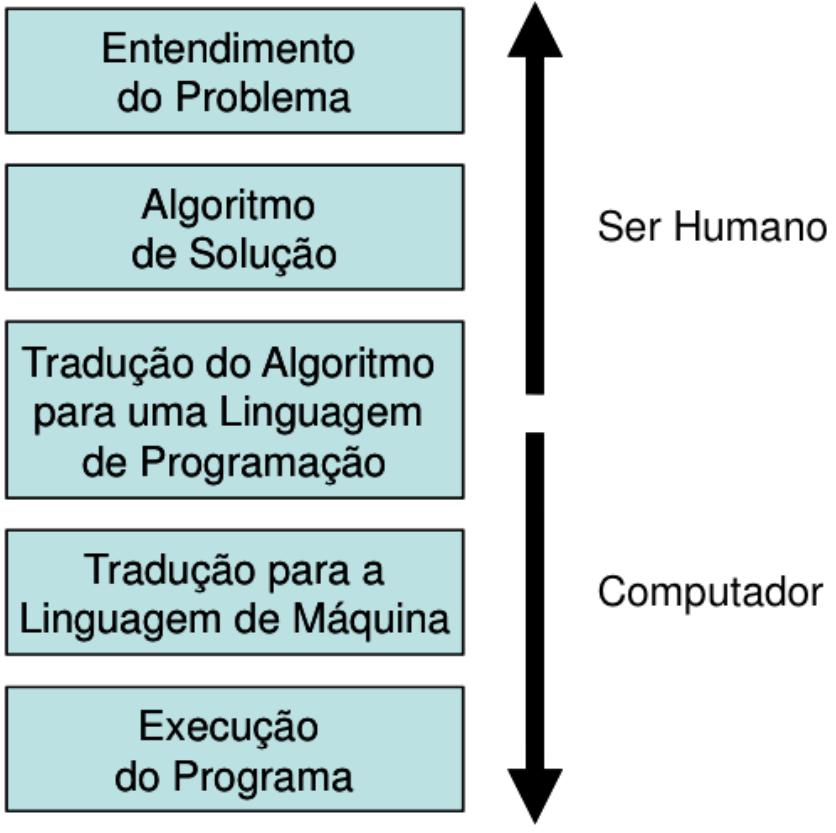
```
0100 0011   1100 1000
1111 0010   0011 1110
0100 0011   1000 0000
1100 1001   1110 1000
0100 0011   1100 1000
1111 0010   0011 1110
0100 0011   1000 0000
1100 1001   1110 1000
0100 0011   1100 1000
1111 0010   0011 1110
. . .
```

Linguagens de Programação

São traduzidas em linguagem de máquina (compostas por 0s e 1s) que podem ser processadas pelo computador. São livres de ambiguidades → única interpretação.



Resolução de Problema

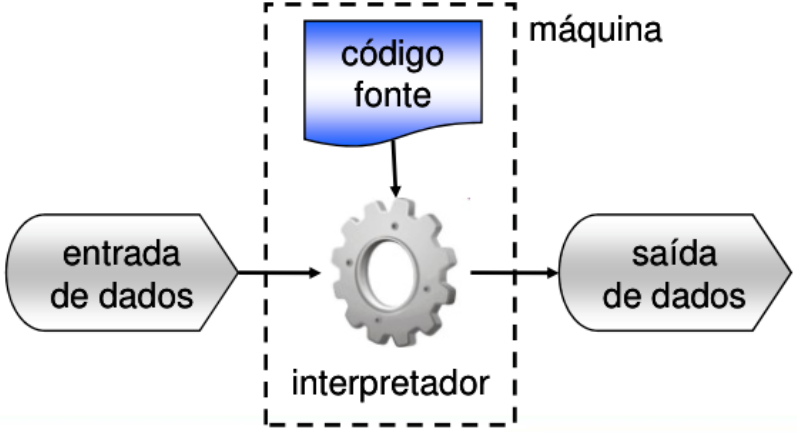


Tradutores

- Processam linguagens de alto nível, traduzindo-as em linguagens de baixo nível
  - Interpretadores
  - Compiladores

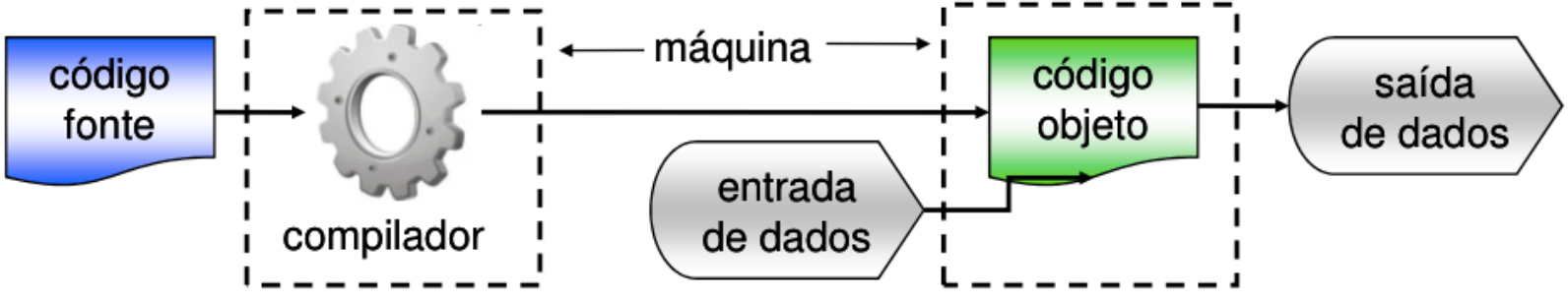
Interpretador

No interpretador, as instruções definidas na linguagem de alto nível (código-fonte) são executadas diretamente. Ele traduz o comando de um programa de cada vez e então chama uma rotina para completar a execução do comando. Mais precisamente, o interpretador é um programa que executa repetidamente a seguinte sequencia: pega a próxima instrução → determina as ações a serem executadas → executa estas ações.



Compilador

Um compilador produz a partir do arquivo de entrada, outro arquivo que é equivalente ao arquivo original, porém numa linguagem que é executável. Este arquivo resultante pode ser em uma linguagem que é diretamente executável, tal como linguagem de máquina, ou indiretamente executável, tal como outra linguagem para a qual já existe um tradutor. O objetivo de um compilador é traduzir um programa escrito em uma linguagem (código fonte) em um programa equivalente expresso em uma linguagem que é executável diretamente pela máquina (código objeto).



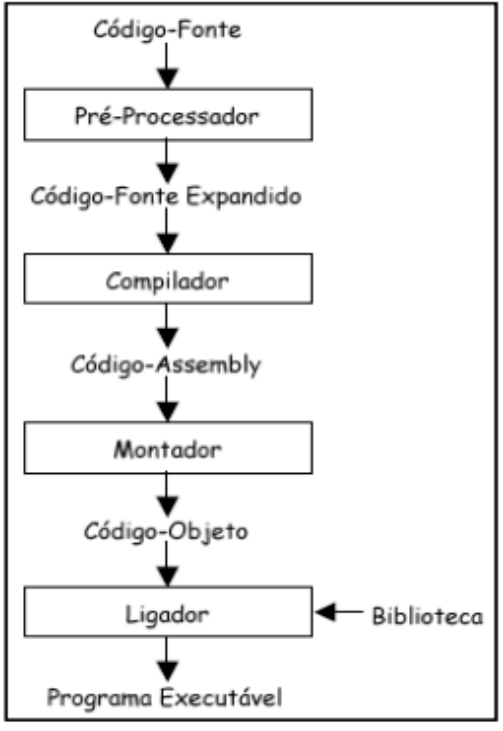
Compilador

Nota: O tempo durante o qual o compilador está "trabalhando", isto é, realizando a conversão do código fonte em código objeto é chamado de tempo de compilação.

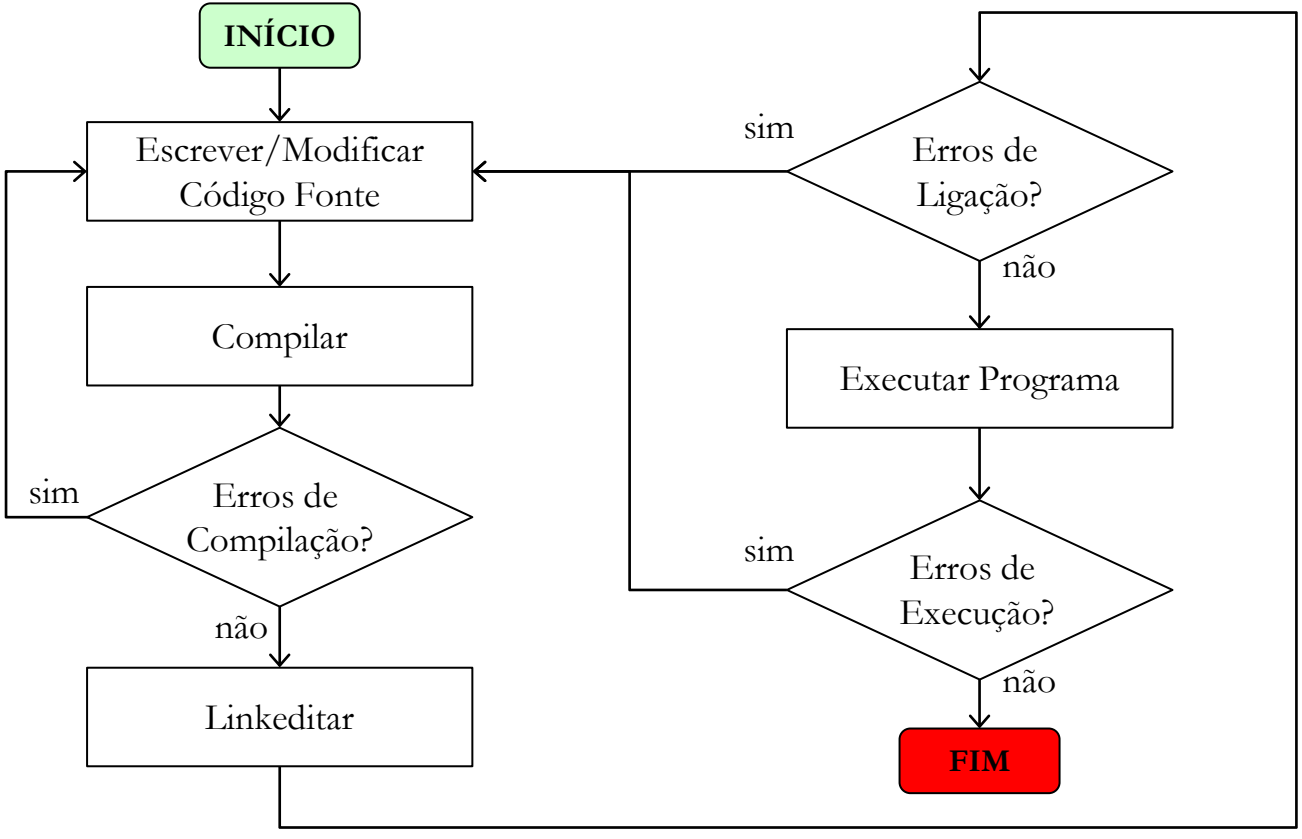
- O pré-processador mapeia instruções escritas numa linguagem de alto nível estendida, para instruções da linguagem de programação original

- O compilador analisa o código fonte e o converte para o código **assembly**

- O ligador (linker) "junta" o código objeto às bibliotecas necessárias para gerar o código executável



Desenvolvimento de um Programa



Variáveis e Constantes

Responda a Questão

- Suponha o seguinte trecho de código em C:

```
a = 3;
b = a/2;
c = b + 3.1;
```

- Qual é o valor da variável c?

a) 4.6
b) 4.1
c) 4
d) Nenhuma das opções acima
e) Não é possível determinar o valor de c

Tipos Básicos

Tipo	Tamanho	Menor Valor	Maior Valor
char	1 byte	-128	+127
unsigned char	1 byte	0	+255
short int (short)	2 bytes	-32.768	+32.767
unsigned short int	2 bytes	0	65.535
int (*)	4 bytes	-2.147.483.648	+2.147.483.647
long int (long)	4 bytes	-2.147.483.648	+2.147.483.647
unsigned long int	4 bytes	0	+4.294.967.295
float	4 bytes	-10^{38}	$+10^{38}$
double	8 bytes	-10^{308}	$+10^{308}$

(*) depende da máquina, sendo de 4 bytes para arquiteturas de 32 bits.

Valores Constantes

- Armazenados na memória e possui um tipo, indicado pela sintaxe da constante

```
123 /* constante inteira do tipo "int" */
12.45 /* constante real do tipo "double" */
1245e-2 /* constante real do tipo "double" */
12.45F /* constante real do tipo "float" */
```

- Declarando uma constante para reutiliza-la em outro lugar do programa

```
define constante1 123;
define consatnte2 12.45;
define constante3 1245e-2;
define constante4 F 12.45F;
```

Variáveis

- Representam um espaço de memória
- Não são variáveis no sentido matemático
- Possuem um **tipo** e um **nome**
  - O nome identifica o espaço de memória
  - O tipo determina a natureza do dado

Declaração de Variáveis

- Variáveis devem ser explicitamente declaradas
- Variáveis podem ser declaradas em conjunto

```
int a;       /* declara uma variável do tipo int */
int b;       /* declara uma variável do tipo int */
float c;    /* declara uma variável do tipo float */
int d, e;  /* declara duas variáveis do tipo int */
```

Declaração de Variáveis

Variáveis só armazenam valores do mesmo tipo declarado

```
int a;     /* declara uma variável do tipo int */
a = 4.3; /* a armazenará o valor 4 */
```

Uma variável pode receber um valor quando é definida (inicializada), ou através de um operador de atribuição

```
int a = 5, b = 10; /* declara e inicializa duas variáveis do tipo int */
float c = 5.3; /* declara e inicializa uma variável do tipo float */
```

Uma variável deve ter um valor definido quando é utilizada

```
int a, b, c; /* declara e inicializa duas variáveis do tipo int */
a = 2;
c = a + b; /* ERRO: b contém "lixo" */
```

Operadores e Expressões

Operadores Aritméticos

Nome	Símbolo	Exemplo
Adição	+	a = a + 1
Subtração	-	s = s - 1
Multiplicação	*	m = m*2
Divisão	/	d = d/2
Módulo (resto)	%	r = r%2
Incremento	++	i++
Decremento	--	i--

Operadores Aritméticos

Operações são feitas na precisão dos operandos. O operando com tipo de **menor expressividade** é **convertido** para o tipo do operando com tipo de **maior expressividade**. **Divisão entre inteiros trunca a aparte fracionária**.

```
int a
double b, c;
a = 3.5; /* a recebe o valor 3 */
b = a / 2.0; /* b recebe o valor 1.5 */
c = 1/3 + b; /* 1/3 retorna 0 (divisão inteiros) e c recebe o valor de b */
```

Operadores Aritméticos

O operador de módulo, %, aplica-se a inteiros

```
x % 2 /* o resultado será 0, se x for par; caso contrário, será 1 */
```

Precedência de operadores: “*” > “/” > “-” > “+”

```
a + b * c / d /* é equivalente a (a + ((b * c) / d))
```

Operadores de Atribuição

Nome	Símbolo	Exemplo	Equivalente
Atribuição Simples	=	a = 1	-
Atribuição com Adição	+=	s += 1	s = s + 1
Atribuição com Subtração	-=	m -= 2	m = m - 2
Atribuição com Multiplicação	*=	m *= 2	m = m * 2
Atribuição com Divisão	/=	d /= 2	d = d / 2
Atribuição com Módulo	%=	r %= 2	r = r % 2

Operadores de Atribuição

- C trata uma atribuição como expressão
  - A ordem é da direita para esquerda
- C oferece uma notação compacta para atribuições em que a mesma variável aparece dos dois lados
  - `var op= expr` ≈ `var = var op (expr)`

```
i += 2; /* é equivalente a i = i + 2;
x *= y+1; /* é equivalente a x = x * (y+1);
```

Operadores de Incremento

Nome	Símbolo	Exemplo
Incremento	++	i++
Decremento	--	i--

- Incrementa ou decrementa de uma unidade o valor de uma variável
- Os operadores não se aplicam a expressões
- O incremento pode ser antes ou depois da variável ser utilizada
  - n++ incrementa n de uma unidade, depois de ser usado
  - ++n incrementa n de uma unidade, antes de ser usado

Operadores Relacionais

Nome	Símbolo	Exemplo
Igualdade	==	if(a == b) {...}
Desigualdade	!=	if(a != b) {...}
Menor que	<	if(a < b){...}
Maior que	>	if(a > b){...}
Menor ou igual	<=	if(a <= b){...}
Maior ou igual	>=	if(a >= b){...}

O resultado será 0 ou 1 (não há valores booleanos em C)

```
int a, b;
int c = 23;
int d = c + 4;

c < 20 /* retorna 0 */
d > c   /* retorna 1 */
```

Operadores Lógicos

Nome	Símbolo	Exemplo
Negação	!	`if(!(a == b)) {...}`
AND	&&	`if((a == 1) && (b > 1)){...}`
OR	||	`if((a == 1) || (b > 1)){...}`

A avaliação é feita da esquerda para direita e o resultado será 0 ou 1

```
int a, b;
int c = 23;
int d = c + 4;

a = (c < 20) || (d > c); /* retorna 1 e as duas sub-expressões são avaliadas */

b = (c < 20) && (d > c); /* retorna 0 e apenas a primeira sub-expressão é avaliada */
```

Operadores de Tamanho

C oferece um operador que permite saber o tamanho em bytes de ocupado por um tipo

```
int tamanhoFloat = sizeof(float); /* armazena o valor 4 na variável */
```

Conversão de Tipo

Conversão de tipo é automática na avaliação de uma expressão. Em C a conversão de tipo pode ser requisita explicitamente

```
float f;
float f = 3;     /* valor 3 é convertido automaticamente para "float" */
                 /* ou seja, passa a valer 3.0F, antes de ser atribuído a f */
int g, h;

g = (int) 3.5; /* 3.5 é convertido (e arredondado) para "int" */
               /* antes de ser atribuído à variável g */

h = (int) 3.5 % 2 /* e antes de aplicar o operador módulo "%" */
```

Responda a Questão

- Suponha o seguinte trecho de código em C:

```
a = 3;
b = a/2;
c = b + 3.1;
```

- Qual é o valor da variável c?

a) 4.6
b) 4.1
c) 4
d) Nenhuma das opções acima
e) Não é possível determinar o valor de c

Nota! Defina as variáveis a, b e c para obter todas as possíveis respostas da questão.

Entrada e Saída

A função `printf()`

- Possibilita a saída de valores no terminal segundo um determinado formato

```
printf (formato, lista de constantes/variáveis/expressões...);
```

```
printf ("%d %g", 33, 5.3);

tem como resultado a impressão da linha:
33 5.3
```

```
printf ("Inteiro = %d Real = %g", 33, 5.3);

com saída:
Inteiro = 33 Real = 5.3
```

A função `printf()` - Formato

Símbolo	Descrição
%c	Especifica um char
%d	Especifica um int
%u	Especifica um unsigned int
%f	Especifica um double (ou float)
%e	Especifica um double (ou float) no formato científico
%g	Especifica um double (ou float) no formato mais apropriado (%f ou %e)
%s	Especifica uma cadeia de caracteres

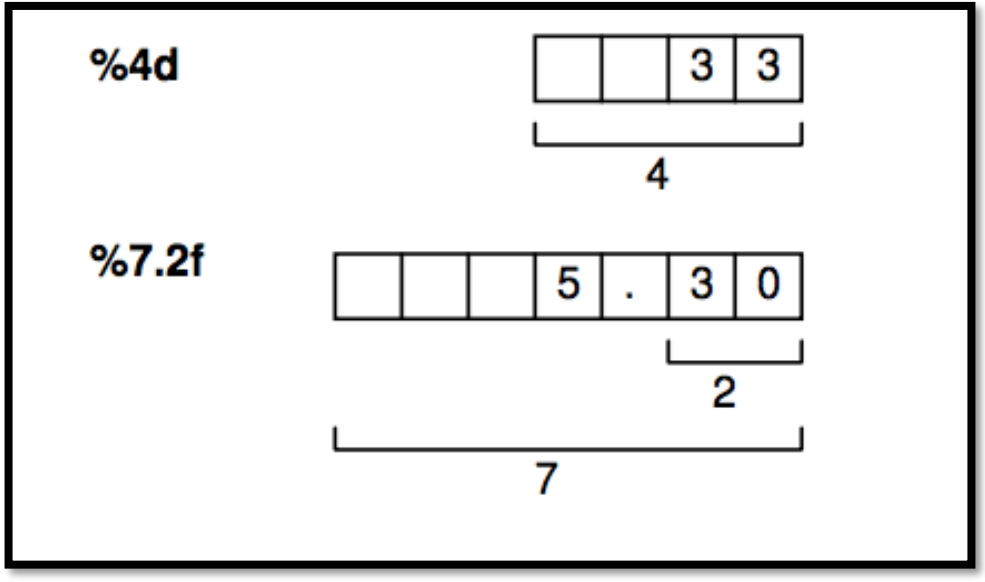
```
printf("Curso de Programação\n");

exibe na tela a mensagem:
Curso de Programação
```

A função `printf()` - Scape

Símbolo	Descrição
\n	Caractere de nova linha
\t	Caractere de tabulação
\v	Caractere de tabulação vertical
\r	Caractere de retrocesso (*carriage return*)
\b	Caractere de backspace
\a	Faz soar um bip de alerta
\"	Imprime o caractere "
\\	Imprime o caractere \

A função `printf()` - Tamano



A função `scanf()`

- Captura valores fornecidos via entrada padrão (i.e., o teclado)

```
scanf (formato, lista de endereços das variáveis...);
```

```
int n;
scanf ("%d", &n);

valor inteiro digitado pelo usuário é armazenado na variável n
```

OBS: o uso do caractere **&** indica que estamos referenciando o endereço de memória da variável **n** e não a variável propriamente dita.

A função `scanf()` - Formato

Símbolo	Descrição
`%c`	Especifica um char
`%d`	Especifica um int
`%u`	Especifica um unsigned int
`%f, %e e %g`	Especifica um float
`%lf, %le e %lg`	Especifica um double
`%s`	Especifica uma cadeia de caracteres

- Caracteres diferentes dos especificadores no formato servem para cercar a entrada
- Espaço em branco dentro do formato faz com que sejam "pulados" eventuais brancos da entrada
- Os caracteres especiais %d, %f, %e e %g automaticamente pulam os brancos que precederem os valores numéricos a serem capturados

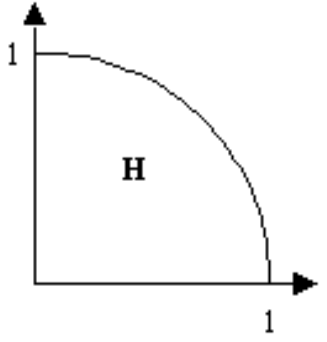
```
scanf ("%d:%d", &h, &m);

valores (inteiros) fornecidos devem ser separados pelo caractere dois pontos (:)
```


hands
on

Dojo de Programação

Os pontos (x, y) que pertencem à figura H (abaixo) são tais que $x \geq 0$, $y \geq 0$ e $x^2 + y^2 \leq 1$. Escreva um programa que receba um par (x, y) e imprima se o ponto representado pelo par pertence ou não a H.



Nota: gcc <nome do programa>.c –o <nome do executável>

UNIVERSIDADE FEDERAL DO CEARÁ

DEPARTAMENTO DE COMPUTAÇÃO

Programação

(CK0226 – 2022.2)

Curso: Ciência da Computação
Professor: Lincoln Souza Rocha
E-mail: lincoln@dc.ufc.br