INE5231 Computação Científica I

Prof. A. G. Silva

21 de março de 2017

Conteúdo programático

- **O computador - [3 horas-aula]**
- **Representação de algoritmos - [3 horas-aula]:**
- **Linguagens de programação estruturadas [3 horas-aula]**

  • Paradigma Estruturado e Estruturas de Controle • Comandos de Atribuição • Variáveis, Constantes, Funções e Procedimentos • Comandos de Entrada/Saída: Teclado, Vídeo e Arquivos

  • Compilação,Interpretação,Execução,Depuração de Programas
- **Introdução à programação em C [6 horas-aula]**
- **Programas envolvendo processos de repetição e seleção [6 horas-aula]**
- **Variáveis estruturadas unidimensionais homogêneas [9 horas-aula]**
- **Variáveis estruturadas multidimensionais homogêneas [6 horas-aula]**
- **Variáveis estruturadas heterogêneas [6 horas-aula]**
- **Subdivisão de problemas e subprogramação [6 horas-aula]**
- **Programação utilizando uma linguagem de computação técnica numérica [6 horas-aula]**

Algoritmo

- Partes de um algoritmo
  - **Entrada de dados** – informações necessárias à execução, fornecidas em tempo de execução ou embutidas, por interação com o usuário ou por arquivos
  - **Processamento de dados** – avaliação de expressões algébricas, relacionais e lógicas, assim como estruturas de controle (condição e/ou repetição)
  - **Saída de dados** – resultados de processamento enviados a dispositivos de saída (monitor, impressora, ou memória)

Exemplo 1

Algoritmo 1 Pegar um ônibus

```
1: ir até a parada
2: enquanto ônibus não chega faça
3:     esperar
4: fim enquanto
5: subir no ônibus
6: pegar passagem
7: se não há passagem então
8:     pegar dinheiro
9:     pagar cobrador
10:    pegar troco = dinheiro - valor da passagem
11: fim se
12: passar pela catraca
13: enquanto houver banco e banco não vazio faça
14:    ir para o próximo
15: fim enquanto
16: se se banco vazio então
17:    sentar
18: fim se
19: ...
```

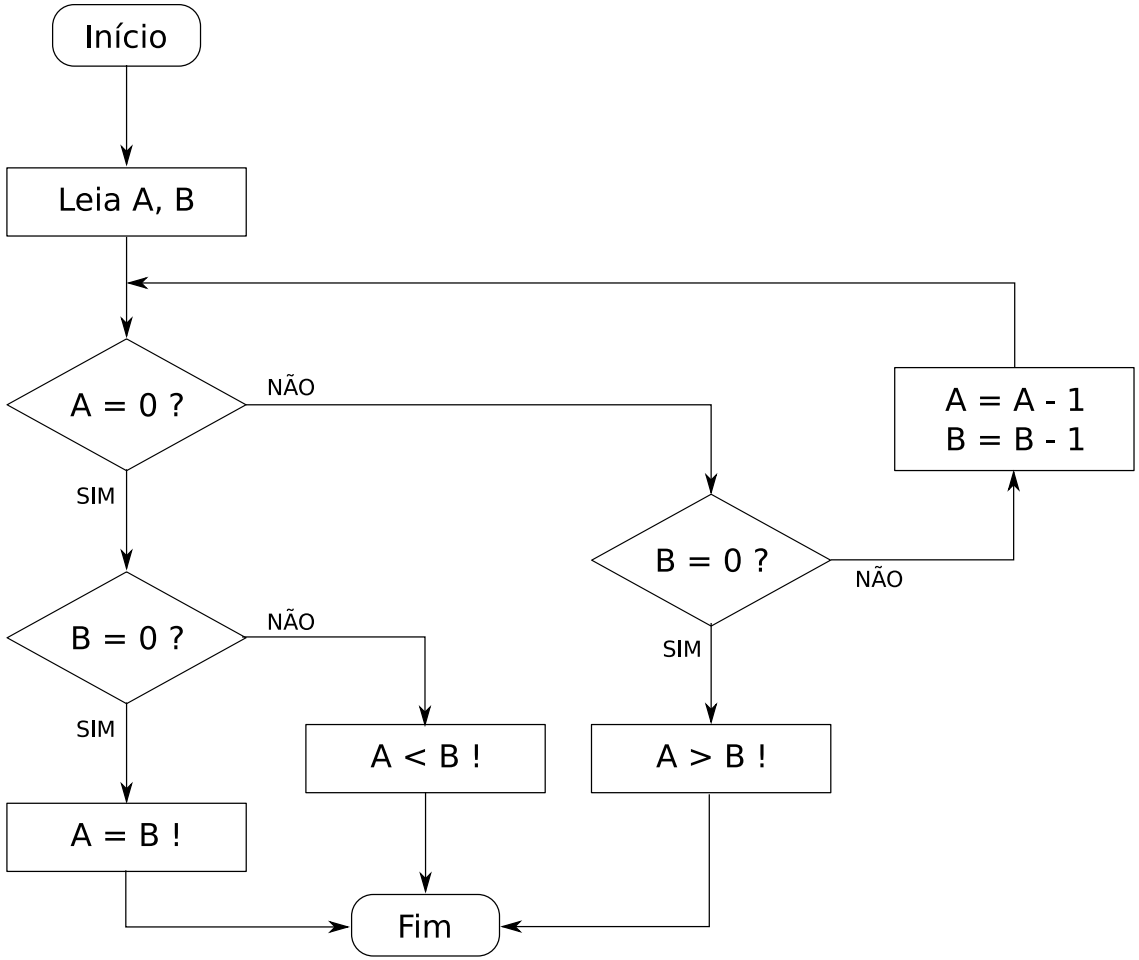
Exemplo 2

Algoritmo 2 Calcular área de uma circunferência

1: $\pi \leftarrow 3.14$ ▷ entrada para o processamento
2: **leia** R ▷ entrada para o processamento
3: $A \leftarrow \pi \cdot R^2$ ▷ processamento
4: **escreva** A ▷ saída

Exemplo 3

Algoritmo 3 Comparar A e B



Exemplo 4

Algoritmo 4 Comparar x e y

```
1: leia x, y
2: se x > y então
3:     escreva "x é maior"
4: senão
5:     se y > x então
6:         escreva "y é maior"
7:     senão
8:         escreva "x e y são iguais"
9:     fim se
10: fim se
```

Exemplo 5 – comparar x e y em C

```c
#include <stdio.h>

int main() {
    int x, y;    // isto e' um comentario de linha

    /* isto e' um comentario
       em bloco */

    printf("\ndigite x: ");
    scanf("%i", &x);
    printf("\ndigite y: ");
    scanf("%i", &y);
    if (x > y) {
        printf("x e' maior\n");
    } else if (x < y) {
        printf("y e' maior\n");
    } else {
        printf("x e' y sao iguais\n");
    }

    return 0;
}
```

Tradução/Compilação

Alto Nível	Baixo Nível	
Linguagem de Alto Nível	**Linguagem Assembly (*Mnemônica*)**	**Linguagem de Máquina**
val2 = val1+val2	LOAD R1, val1	0010 0001 1110
	LOAD R2, val2	0010 0010 1111
	ADD R1, R2	0001 0001 0010
	STORE R1, val2	0011 0001 1111

CÓDIGO FONTE $\Rightarrow$ TRADUTOR $\Rightarrow$ CÓDIGO OBJETO

```
val2 = val1 + val2;
```

(linguagem de programação)

```
00010110111001011001011010 ...
```

("executável")

Introdução e ambientes de desenvolvimento em C

Linguagem C

- Imperativa, procedural, de propósito geral, desenvolvida inicialmente entre 1969 e 1973 Denis Ritchie.
- Da linhagem do ALGOL (criada por comitê de especialistas e primeira linguagem portável).
- Destinada à programação de sistemas Unix, a partir do BCPL (*Basic Combined Programming Language*, 1965) e B (contração de BCPL, 1967) desenvolvidas pela Bell Labs.
- Conceito de blocos, bibliotecas (headers) de funções, array, pointers e casting de tipos.
- Padronizada em 1973 (ANSI C). É a linguagem mais utilizada até hoje.

Estrutura básica de um programa em C

```c
//diretivas para o pre-processador
//declaracao de variaveis globais
int main()
{
        //declaracao de variaveis locais da funcao main
        //comandos da funcao main
}
```

- Para que certas funções (p. ex: entrada e saída) sejam acessíveis, é necessário incluir algumas bilbiotecas (p. ex: `stdio.h`).
- Todo programa em C inicia sua execução pela função `main()`.
- Instruções são finalizadas com ponto-e-vírgula.
- Os blocos de instruções são delimitados por chaves.
- Linhas de comentários são iniciadas por duas barras `//`
- Blocos de comentários são delimitados por `/*` e `*/`

Diretivas para o compilador

- Diretiva `#include` permite incluir uma biblioteca
- Bibliotecas contêm funções pré-definidas, utilizadas nos programas
- Exemplos

`#include <stdio.h>`	Funções de entrada e saída
`#include <stdlib.h>`	Funções padrão
`#include <math.h>`	Funções matemáticas
`#include <string.h>`	Funções de texto

Ambientes de desenvolvimento

- Dev-C++ (Windows)
  - Ambiente integrado com editor, bibliotecas, acionamento do compilador (MingW / GNU GCC)...
  - Página do projeto: http://orwelldevcpp.blogspot.com.br/
  - Download do projeto mais ativo: https://sourceforge.net/projects/orwelldevcpp/files/latest

- Code::Blocks (Windows e Linux)
  - Ambiente integrado alternativo, com suporte a múltiplos compiladores (MingW / GNU GCC, MSVC++, clang, Borland C++ 5.5, ...)
  - Página do projeto: http://www.codeblocks.org/
  - Download do projeto mais ativo: http://www.codeblocks.org/downloads/binaries

Exemplo (em Dev-C++)

- Inicie o Dev-C++ pelo ícone ou pelo menu
- Crie um arquivo, clicando em *File* ou *Arquivo* e *New* ← *Source File* ou *Novo* ← *Arquivo Fonte*, ou simplesmente com Ctrl–N
- Edite o seguinte programa:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf ("Alo mundo!");
    system("PAUSE");
    return 0;
}
```

Exemplo (em Dev-C++ – cont...)

- Salve o programa com o nome `exemplo.c`
- Compile o programa clicando em *Executar* e *Compilar* ou com a tecla Ctrl–F9
- Se houver algum erro de sintaxe, aparece uma ou mais mensagens no rodapé da janela. Neste caso, corrija o programa e repita
- Se não houver erros, execute o programa clicando em *Executar* e *Executar* ou com a tecla Ctrl–F10

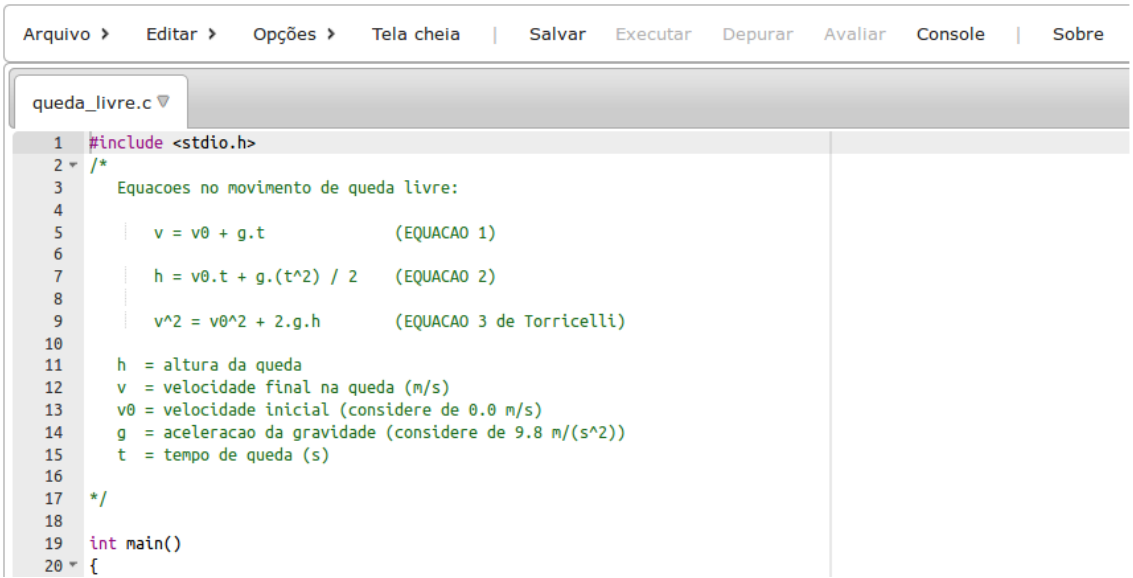
GCC

- Uso do projeto MinGW (*Minimalist GNU for Windows*) diretamente
  - Compilador utilizado pelo Dev-C++ e Code::Blocks
  - Projeto:
    http://www.mingw.org/
  - Dicas:
    https://terminaldeinformacao.com/2015/10/08/como-instalar-e-configurar-o-gcc-no-windows-mingw/

- Usuários Linux (p. ex. Ubuntu) podem instalar o gcc
  - Dicas:
    https://www.quora.com/How-do-I-install-GCC-4-9-0-in-Ubuntu-14-04-LTS-to-compile-C++

Moodle & VPL

- O Moodle tem suporte à programação diretamente pelo navegador
- VPL – *Virtual Programming Lab*
  - Para conhecer mais:
    http://vpl.dis.ulpgc.es/
- Utilizaremos este ambiente para entrega de exercícios
  - Exemplo



Dicas iniciais

- Termine todas as instruções com ponto-e-vírgula
- Sempre salve o programa antes de compilar
- Sempre compile o programa antes de executar
- Quando ocorrer um erro de compilação, dê um duplo clique (se for um ambiente integrado) sobre a mensagem de erro ou simplesmente identifique a posição (linha/coluna) de erro para efetuar sua correção
- Verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ponto-e-vírgula
- Use comentários, iniciados por //, para documentar a implementação facilitando o seu entendimento

Conceitos e instruções básicas em C

Declaração de Variáveis

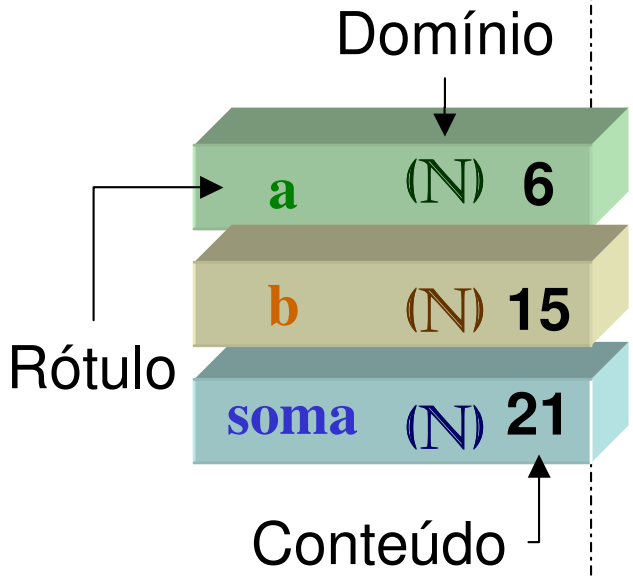
Roteiro:

- Relembrando conceitos
- Tipos de Variáveis
- Declaração
- Identificadores

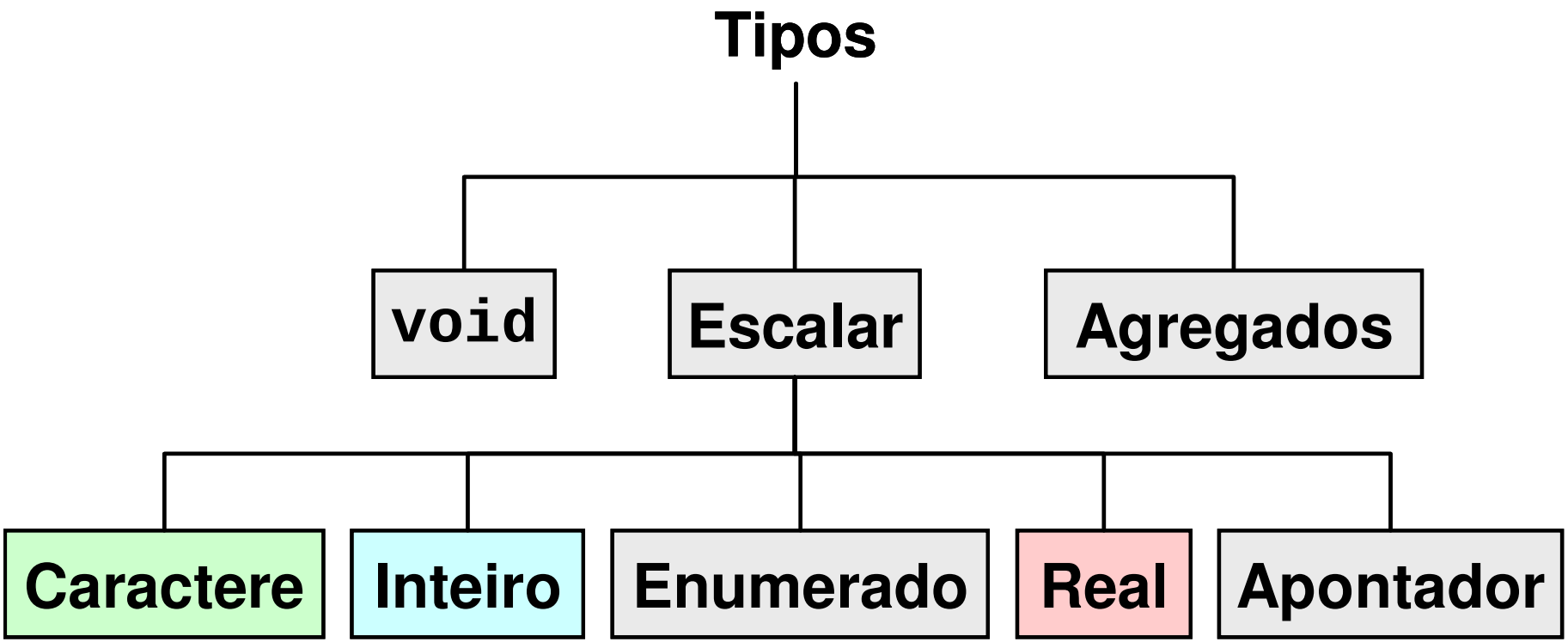
Conceitos

Variável em C:

Variável:
- Nome (rótulo)
- Tipo (domínio)
- Valor (conteúdo)
- Escopo (tempo de vida)



Tipos da Linguagem C:



Declaração de Variável

Declaração:
- *Reservar espaço na memória*
- *Associar com identificador*

Sintaxe: *Valor inicial*

```
tipo nome = valor;
```

Domínio — Rótulo — Conteúdo

Sintaxe: *Sem valor inicial*

```
tipo nome;
```

Declaração de Variável

Sintaxe: *Diversas variáveis, mesmo tipo*

```
tipo nome1, nome2, nome3;
```

Sintaxe: *Diversas variáveis, mesmo tipo*

```
tipo nome1 = valor, nome2;
```

Declaração de Variável

Exemplo:

```
float nota_prova_a = 8.0;
float nota_prova_b = 6.0;
float nota_laboratorio = 10.0;
float media;
```

Identificadores

Nome de variável:

- Seqüência de:
  - Letras maiúsculas (A-Z)
  - Letras minúsculas (a-z)
  - Dígitos (0-9)
  - Sublinhado (_)

Não pode:

- Começar com digito
- Ser uma palavra chave

Identificadores

Nome de variável:

Correto:

`contador` `nota1`

`media` `resto_divisao`

Errado:

`2lugares` `&valor`

`média`

Identificadores

Nome de variável:

- Distinção maiúscula/minúscula
- Máximo 31 símbolos
- Palavras chaves (proibidas):

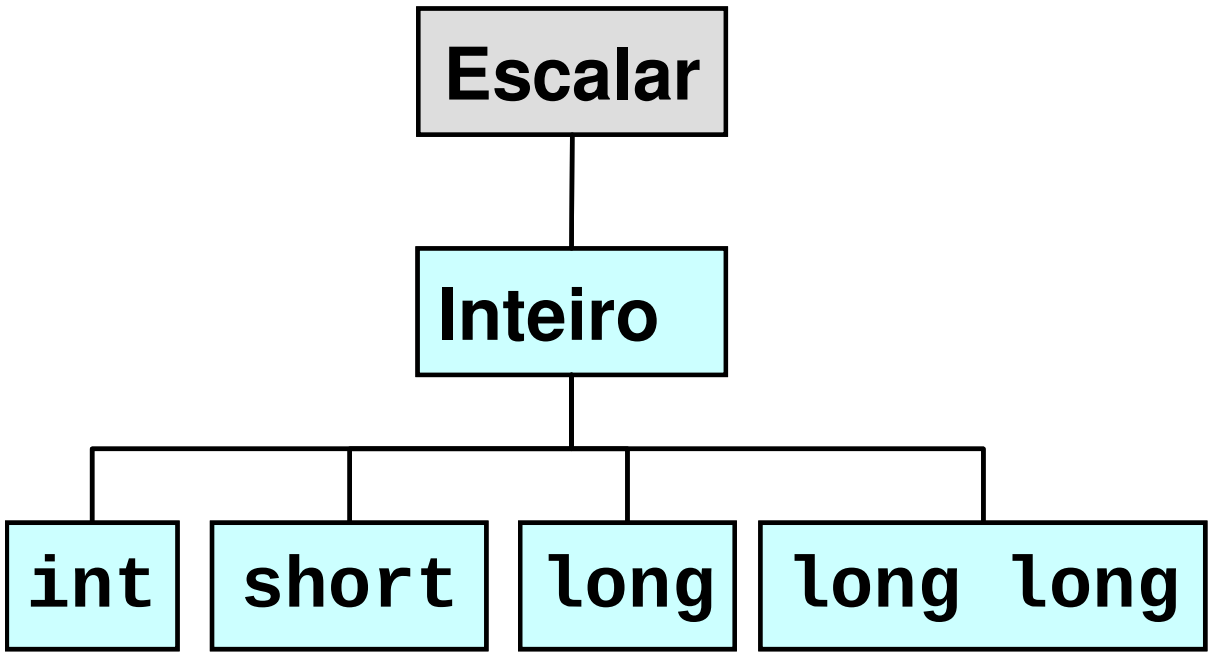
```
auto, break, case, char, const, continue,
default, do, double, else, enum, extern,
float, for, goto, if, inline, int, long,
register, restrict, return, short,
signed, sizeof, static, struct, switch,
typedef, union, unsigned, void, volatile,
while
```

O Tipo Inteiro

Tipos Inteiros:

- Representação de números inteiros
  - Positivos e negativos
- Limitação de valor mínimo e máximo
  - Intervalo válido para números inteiros
- Compromisso:
  - Memória x Amplitude

Tipos Inteiros: *Hierarquia*



O Tipo Inteiro

Opções de Tipos Inteiros:

Declaração

```
tipo  nome = valor;
```

`short int`	Números pequenos
`long int`	Números grandes
`long long int`	Números muito grandes
`int`	Velocidade

O Tipo Inteiro

Opções de Tipos Inteiros:

Exemplos de declaração:

```
int contador;
int limite_tentativas = 100;

  short int numero_pequeno;
  short int contador = 4;

long int quantidade_pecas;
long int numero_repeticoes = 5000000;
```

O Tipo Inteiro

Tipo	Descrição	Memória*	Intervalo*
`int`	Tamanho padrão	4 bytes	- 2.147.483.648 até 2.147.483.647
`short int`	Números pequenos	2 bytes	-32.768 até 32.767
`long int`	Números grandes	4 bytes	- 2.147.483.648 até 2.147.483.647
`long long int`	Números muito grandes	8 bytes	$-9{,}223 \cdot 10^{15}$ até $9{,}223 \cdot 10^{15}$

Escrever texto

Comando `printf`()

Sintaxe: *Mesma linha*

```
printf("mensagem");
```

Sintaxe: *Avançar para próxima linha*

```
printf("mensagem\n");
```

Escrever texto

Exemplo:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
        printf("Primeira linha\n");
        printf("   Segunda linha\n");
        printf("Terceira linha");
        printf("continua terceira linha");

        return 0;
}
```

```
Primeira linha
   Segunda linha
Terceira linhacontinua terceira linha
```

Escrever números inteiros

Indicador de escrita: **%d**

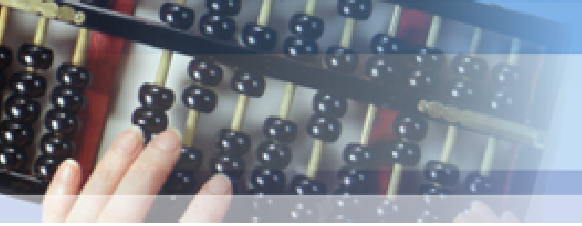
Sintaxe: *Uma variável*

```
printf("texto com %d", variavel);
```

Exemplo:

```
int q = 10;
printf("Quantidade: %d itens", q);
```

```
Quantidade: 10 itens
```



Escrever números inteiros

Indicador de escrita: %d

Sintaxe: *Mais variáveis*

```
printf("mensagem com varios %d", v1, v2 ...);
```

Exemplo:

```
int nota1 = 7;
int nota2 = 8;
printf("Primera nota: %d; segunda: %d.",
        nota1, nota2);
```

```
Primeira nota: 7; segunda: 8.
```

Ler números inteiros

Comando `scanf()` com %d

Sintaxe: *Um número por comando*

```
scanf("formato com %d", &variavel);
```

Exemplo:

```
int quantidade;
printf("Digite a quantidade: ");
scanf("%d", &quantidade);
```

Ler números inteiros

Comando `scanf()` com %d

Sintaxe: *Vários números por comando*

```
scanf("formato com %d", &v1, &v1, ...);
```

Exemplo:

```
int nota1, nota2;
printf("Digite as duas notas: ");
scanf("%d %d", &nota1, &nota2);
```

Particularidades da leitura

Comando `scanf()` com `%d`

- Programa bloqueia até o usuário:
  - escrever todos os valores pendentes
  - pressionar ENTER.

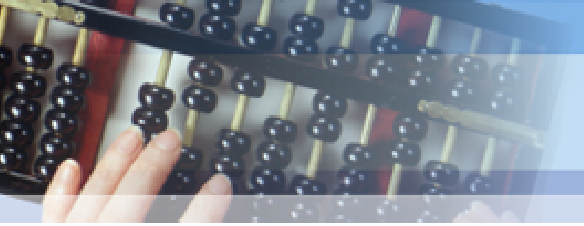
```
int a, b, c;
scanf("%d %d %d", &a, &b, &c);
```

O usuário poderá escrever:

```
3 4 6 (enter)
```

```
3 (enter)
4 6 (enter)
```

```
3 (enter)
4 (enter)
6 (enter)
```



Particularidades da leitura

Comando `scanf()` **com** %d

- Números digitados em excesso:
  - Ficam em uma fila para próximos `scanf`

```
int a, b, c, d, e;
scanf("%d %d %d", &a, &b, &c);
scanf("%d %d", &d, &e);
```

O usuário poderá escrever:

```
3 4 6 (enter)
7 8 (enter)
```

```
3 4 6 7 8(enter)
```

Introdução aos Operadores

Roteiro:

- Atribuição
- Matemática
- Exemplo

Atribuição

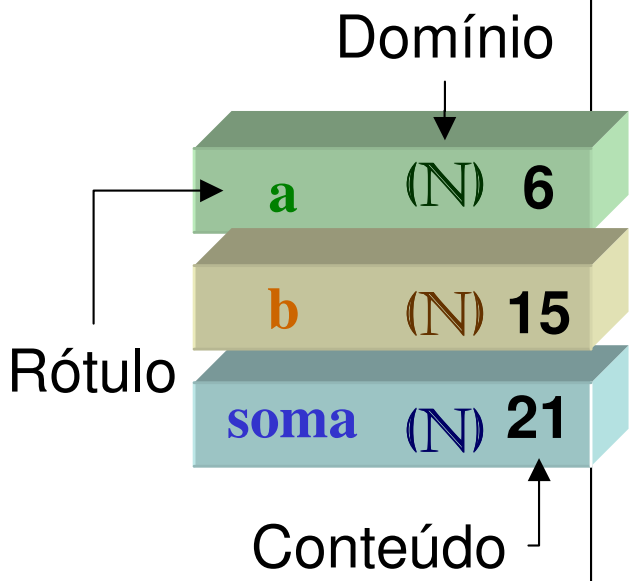
Atribuição: *Substitui o valor da variável*

Sintaxe:

```
variavel = valor;
```

Sintaxe:

```
variavel = expressão;
```



Atribuição

Atribuição: *Substitui o valor da variável*

Atribuir um novo valor:

```
quantidade = 10;
```

Armazenar resultado de uma conta:

```
soma = valor_a + valor_b;
```

Atualizar um contador:

```
contador = contador + 1;
```

Matemática

Matemática:

- Operadores:
  - Soma
  - Subtração
  - Multiplicação
  - Divisão
  - Módulo (resto)
- Expressões

Matemática

Soma:

```
int parcela1 = 10, parcela2 = 16;
int soma;
soma = parcela1 + parcela2;
printf("Soma: %d mais %d é %d",
       parcela1, parcela2, soma);
```

```
Soma: 10 mais 16 é 26
```

Matemática

Subtração:

```
int parcela1 = 10, parcela2 = 16;
int subtracao;
subtracao = parcela1 - parcela2;
printf("Subtração: %d menos %d é %d",
        parcela1, parcela2, subtracao);
```

```
Subtração: 10 menos 16 é -6
```

Matemática

Multiplicação:

```
int fator_a = 4, fator_b = 6;
int produto;
produto = fator_a * fator_b;
printf("Multipliação: %d vezes %d é %d",
        fator_a, fator_b, produto);
```

```
Multiplicação: 4 vezes 6 é 24
```

Divisão inteira:

```
int dividendo = 46, divisor = 6;
int quociente;


quociente = dividendo / divisor;
printf("Divisão: %d por %d é %d",
        dividendo, divisor, quociente);
```

Divisão Inteira!
Sem parte fracionária

```
Divisão: 46 por 6 é 7
```

Resto:

```
int dividendo = 46, divisor = 6;
int quociente, resto;
quociente = dividendo / divisor;
resto = dividendo % divisor;
printf("Divisão: %d por %d é %d, resto %d",
        dividendo, divisor, quociente, resto);
```

```
Divisão: 46 por 6 é 7, resto 4
```

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {

        int horas, minutos, segundos;
        int total_segundos;

        printf("Digite o intervalo de tempo (segundos): ");
        scanf("%d", &total_segundos);

        horas    = (total_segundos / 60) / 60;
        minutos  = (total_segundos / 60) % 60;
        segundos = total_segundos % 60;

        printf("\n");
        printf("Total de segundos: %d \n", total_segundos);
        printf("Tempo: %d:%d:%d\n", horas, minutos, segundos);

        return 0;
}
```

Horario01/Horario02

Tipo Caractere

Roteiro:

- O tipo caractere
- Escrever caracteres na tela
- Ler caracteres do teclado

O Tipo Caractere

Única opção de Tipo Caractere:

Declaração

```
tipo  nome = valor;
```

`char`	Caractere/Letra

Caractere vs Código ASCII:

Exemplos de declaração:

```
char letra = 'A';
```

```
char letra = 65;
```

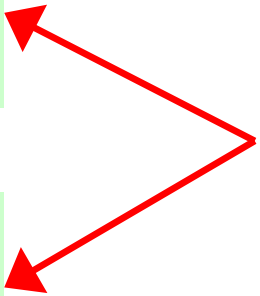


Tabela ASCII
'A' equivale a 65

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	**!**
34	22	100010	42	**"**
35	23	100011	43	**#**
36	24	100100	44	**$**
37	25	100101	45	**%**
38	26	100110	46	**&**
39	27	100111	47	**'**
40	28	101000	50	**(**
41	29	101001	51	**)**
42	2A	101010	52	*****
43	2B	101011	53	**+**
44	2C	101100	54	**,**
45	2D	101101	55	**-**
46	2E	101110	56	**.**
47	2F	101111	57	**/**
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	**:**
59	3B	111011	73	**;**
60	3C	111100	74	**<**
61	3D	111101	75	**=**
62	3E	111110	76	**>**
63	3F	111111	77	**?**
64	40	1000000	100	**@**
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	**[**
92	5C	1011100	134	****
93	5D	1011101	135	**]**
94	5E	1011110	136	**^**
95	5F	1011111	137	**_**
96	60	1100000	140	**`**
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	**{**
124	7C	1111100	174	**|**
125	7D	1111101	175	**}**
126	7E	1111110	176	**~**
127	7F	1111111	177	*[DEL]*

Escrever caracteres

Indicador de escrita: **%c**

Sintaxe: *Uma variável*

```
printf("mensagem", variavel);
```

Exemplo:

```
char l = 'A';
printf("Letra: %c", l);
```

```
Letra: A
```

Ler caracteres

Comando `scanf()` **com** %c

Sintaxe: *Um número por comando*

```
scanf("formato", &variavel);
```

Exemplo:

```
char letra;
printf("Digite a letra: ");
scanf("%c", &letra);
```

Ler caracteres

Dica: Ler próxima letra

```
int numero;
char letra;
printf("Digite um número e uma letra: ");
scanf("%d %c", &numero, &letra);
```

Ou:

```
scanf("%d", &numero);
scanf(" %c", &letra);
```

Espaço!

Outros Tipos Inteiros

Roteiro:

- Tipos com Sinal
- Tipos sem Sinal
- Escrever Inteiros sem Sinal
- Ler Inteiros sem Sinal

Outros Tipos Inteiros

Tipos Modificados:

Declaração:

```
int  variavel;
```

- Tamanho padrão
- 4 bytes
- - 2.147.483.648 até 2.147.483.647

Positivo e negativo

Intervalo simétrico de números

Outros Tipos Inteiros

Tipos com sinal:

Tipos inteiros conhecidos: (**com sinal**)

`char`	`signed char`
`int`	`signed int`
`short int`	`signed short int`
`long int`	`signed long int`
`long long int`	`signed long long int`

Declaração equivalente

Outros Tipos Inteiros

Tipos sem sinal:

Declaração:

```
unsigned int variavel;
```

- Tamanho padrão
- 4 bytes
- 0 até 4.294.967.295
  - Apenas Positivo
  - Intevalo não simétrico

Outros Tipos Inteiros

Tipos sem sinal:

Novos Tipos Inteiros: (**sem sinal**)

```
unsigned char
unsigned int
unsigned short int
unsigned long int
unsigned long long int
```

Outros Tipos Inteiros

Tipo	Tamanho	Domínio
(signed) ***char***	1 byte	- **128** até **127**
unsigned ***char***	1 byte	**0** até **255**
(signed) ***int***	4 bytes	- **2.147.483.648** até **2.147.483.647**
unsigned ***int***	4 bytes	**0** até **4.294.967.296**
(signed) ***short int***	2 bytes	- **32.768** até **32.767**
unsigned ***short int***	2 bytes	**0** até **65.536**
(signed) ***long int***	4 bytes	- **2.147.483.648** até **2.147.483.647**
unsigned ***long int***	4 bytes	**0** até **4.294.967.296**
(signed) ***long long int***	8 bytes	- **$9{,}223 \cdot 10^{15}$** até **$9{,}223 \cdot 10^{15}$**
unsigned ***long long int***	8 bytes	**0** até **$18{,}446 \cdot 10^{15}$**

Escrever Inteiros sem Sinal

Indicador de escrita: **%u**

Sintaxe: *Uma variável*

```
printf("mensagem com %u", variavel);
```

Exemplo:

```
unsigned int n = 5000;
printf("Quantidade: %u itens", n);
```

```
Quantidade: 5000 itens
```

Ler Inteiros sem Sinal

Comando `scanf()` com %u

Sintaxe: *Um número por comando*

```
scanf("formato com %u", &variavel);
```

Exemplo:

```
unsined int repeticoes;
printf("Número de repetições: ");
scanf("%u", &repeticoes);
```

Tipos de Ponto Flutuante

Roteiro:

- O tipo ponto flutuante
- Escrever número em ponto flutuante
- Ler número em ponto flutuante
- Exemplo

O Tipo Ponto Flutuante

Declaração de tipos ponto flutuante:

Declaração

```
tipo  nome = valor;
```

- `float` — Pouca precisão, baixa magnitude
- `double` — Muita precisão, alta magnitude
- `long double` — Precisão maior, altíssima magnitude

O Tipo Ponto Flutuante

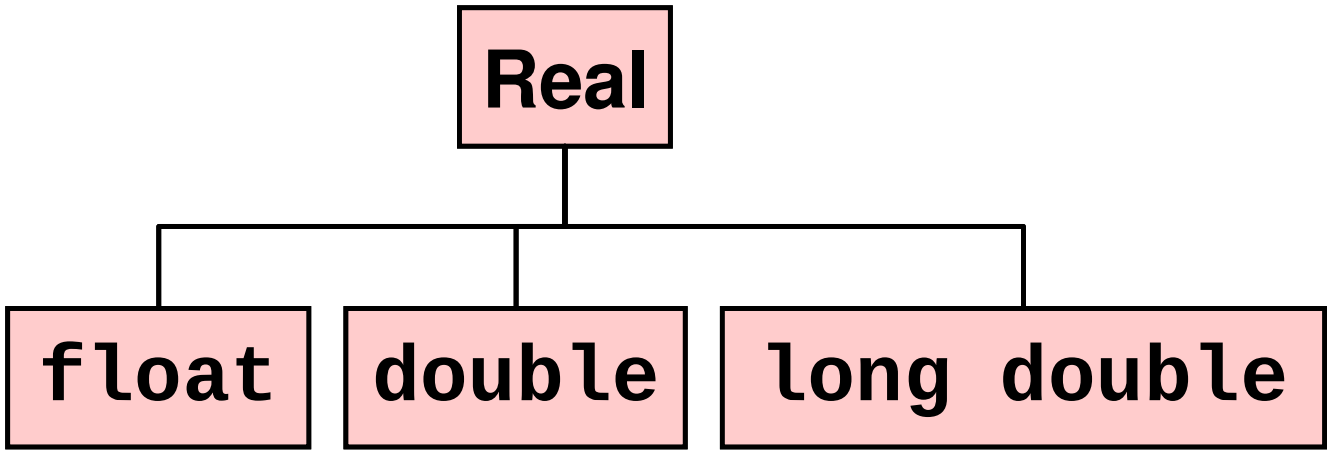
Exemplo:

Exemplos de declaração:

```
float raio = 5.4;
float area = 50040.22;

double velocidade = 5.333222567854;
```

Tipos Ponto Flutuante: *Hierarquia*



O Tipo Ponto Flutuante

Tipo	Tamanho*	Precisão*	Intervalo*
`float`	4 bytes	7 dígitos	$-3{,}4 \cdot 10^{38}$ até $3{,}4 \cdot 10^{38}$
`double`	8 bytes	15 dígitos	$-1{,}7 \cdot 10^{308}$ até $1{,}7 \cdot 10^{308}$
`long double`	10 bytes	19 dígitos	$-1{,}2 \cdot 10^{4932}$ até $1{,}2 \cdot 10^{4932}$

Escrever números reais

Indicadores de substituição: **%f**

Sintaxe: *Uma variável*

```
printf("mensagem com %f", variavel);
```

Exemplo:

```
float v = 10.1;
printf("Velocidade: %fkm/h", v);
```

```
Velocidade: 10.1km/h
```

Ler números reais

Comando `scanf()` com **%f**

Sintaxe: *Um número por comando*

```
scanf("formato com %f", &variavel);
```

Exemplo:

```
float nota;
printf("Digite a nota da prova: ");
scanf("%f", &nota);
```

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[]) {

        double pi = 3.141592;
        double raio, area, perimetro;

        printf("Digite o raio: ");
        scanf("%lf", &raio);

        area = pi * (raio * raio);
        perimetro = 2.0 * pi * raio;

        printf("\n");
        printf("Raio: %f \n", raio);
        printf("Área: %f \n", area);
        printf("Perímetro: %f \n", perimetro);

return 0;
}
```

Circulo01

Referências

- Livro "Introdução a Algoritmos e Programação" de Fabricio Ferrari e Cristian Cechinel
- Notas de aula das professoras Vania Bogorny, Patrícia Jaques, Mônica Py e Deise Saccol
- Notas do Prof. Arnaldo V. Moura e Daniel F. Ferber – Curso C – IC/Unicamp