Linguagem de Programação I

Aula 3 – Variáveis, Tipos de Dados e Entrada/Saída

Prof Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Variáveis

Tipos de Dados

Instruções de E/S

Exercícios

Referências

Variáveis

em Linguagem C

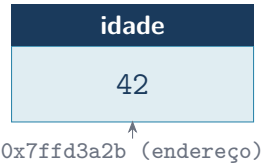
O que é uma Variável?

▪ Definição

Uma **variável** é um espaço de memória de um determinado tipo de dado, associado a um **nome** que permite referenciar seu conteúdo ao longo do programa. Ela pode assumir diversos valores durante a execução.

▪ Analogia

Pense em uma variável como uma **caixa rotulada**: o rótulo é o nome, o tamanho da caixa depende do tipo, e o conteúdo é o valor armazenado.



Regras para Nomes de Variáveis

Permitido

- Até **32 caracteres**
- Começa com **letra** ou **_** (underline)
- Pode conter letras, números e _

Exemplos válidos:

`idade` `nome_aluno` `_aux`

`nota1` `mediaFinal`

Proibido

- Começar com número
- Usar palavras reservadas
- Espaços ou caracteres especiais

Exemplos inválidos:

`1nota` `int` `minha var`

C é Case Sensitive

`peso` $\neq$ `Peso` $\neq$ `pEso`

São três variáveis distintas!

Declaração de Variáveis

Sintaxe

Reserva um espaço de memória de um tipo específico e associa a ele um nome.
[Schildt e Mayer 1997]

```
tipo  nome-da-variável;
```

```
tipo  nome1, nome2, ..., nomeN;
```

Exemplos

```
char   nome;
int    idade, num;
float  salario;
double pi;
```

```
#include <stdio.h>

void main(){
    int idade = 30;
    printf("A idade é: %d\n", idade);
}
```

```
#include <stdio.h>
void main(){
    int idade;
    idade = 30;
    printf("A idade é: %d\n", idade);
}
```

OU pode-se ainda:

```
#include <stdio.h>
void main(){
    int idade;
    printf("Informe a idade:");
    scanf("%d",&idade);
    printf("A idade é: %d\n", idade);
}
```

Palavras Reservadas (lembrete)

Nomes Reservados (42 comandos)

auto	double	if	static
break	else	int	struct
case	entry	long	switch
char	extern	register	typedef
continue	float	return	union
default	for	sizeof	unsigned
do	goto	short	while

Nenhuma variável pode ter o mesmo nome que uma palavra reservada.

Tipos de Dados

da Linguagem C

Tipos de Dados Básicos

Tipo	Tamanho (em bits)	Intervalo
Char	8	-128 a 127
Int	16	-32768 a 32767
Float	32	3,4E-38 a 3,4E+38
double	64	1,7E-308 a 1,7E+308
void	0	sem valor

Resumo dos tipos

- `char` — 1 byte, caractere
- `int` — inteiro (2 ou 4 bytes)
- `float` — ponto flutuante (4 bytes)
- `double` — dupla precisão (8 bytes)
- `void` — sem valor / tipo genérico

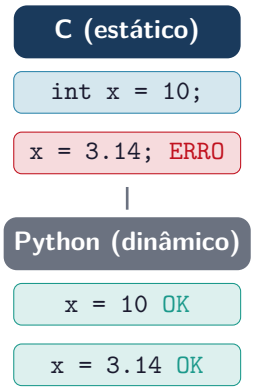
Linguagem Fortemente Tipada

▪ Fortemente Tipada

Uma vez declarada com um tipo, a variável **mantém esse tipo do início ao fim** da execução. Não é possível alterar o tipo de uma variável em tempo de execução.

▪ Tipagem Estática

O tipo das variáveis é definido em **tempo de compilação** e não pode ser alterado depois disso — ao contrário de Python ou JavaScript, que são dinamicamente tipadas.



Modificadores de Tipo

Modificadores

- `long` — aumenta o espaço da variável
- `short` — diminui o espaço da variável
- `unsigned` — apenas positivo (s/ sinal)
- `signed` — com sinal (padrão)

Tipo	Tamanho (em bits)	Intervalo
char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
int	16	-32768 a 32767
unsigned int	16	0 a 65535
signed int	16	-32768 a 32767
short int	16	-32768 a 32767
unsigned short int	16	0 a 65535
signed short int	16	-32768 a 32767
long int	32	-2147483648 a 2147483647
signed long int	32	-2147483648 a 2147483647
unsigned long int	32	0 a 4294967295
float	32	3,4E-38 a 3,4E+38
double	64	1,7E-308 a 1,7E+308
long double	80	3,4E-4932 a 1,1E+4932

Atenção

`float` e `char` **não podem** ter seus tamanhos modificados — os modificadores `long` e `short` não se aplicam a eles.

Exemplos de Uso de Tipos

```
#include <stdio.h>
void main(){

    int quantidade=10;
    double pi=3.14159265;
    float valor=1.25;
    char letra = 'A';

    printf("Foram compradas %d maçãs a R$ %5.2f cada", quantidade,valor);
    printf("%c constante pi = %5.4lf",letra,pi);

}
```

Instruções de E/S

scanf / printf / getchar / putchar

scanf() — Leitura de Dados

```
Leitura de dados tipados via teclado
Scanf (“string de controle”, lista de argumentos);

Exemplo:
scanf("%d",&idade);
```

▪ Sintaxe

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

▪ Atenção

Para **strings** (`%s`), o `&` **não** deve ser usado, pois o nome do array já é um ponteiro.

printf() — Saída de Dados

```
Apresentação de dados no monitor
printf("string de controle", lista de argumentos);

Exemplo:
printf ("Digite a sua idade:\n");
scanf ("%d", &idade);
printf("Sua idade é: %d", idade);
```

Strings de Controle (Especificadores de Formato)

Tipos numéricos e char

- `%c` — caractere
- `%d` — inteiro decimal
- `%f` — ponto flutuante
- `%lf` — double
- `%e` — notação científica
- `%o` — octal
- `%x` — hexadecimal

Outros

- `%s` — string (cadeia de caracteres)
- `%lu` — endereço de memória
- `%%` — imprime o caractere %

Exemplo

```
printf("Cresceu %d%%", 60);
→ Cresceu 60%
```

Caracteres Especiais (Sequências de Escape)

Sequências de escape

- `\n` — nova linha
- `\r` — retorno de carro (enter)
- `\t` — tabulação (tab)
- `\b` — retrocesso (backspace)
- `\"` — aspas duplas
- `\\` — barra invertida

Uso prático

```
printf("Nome:\t%s\n", nome);
printf("Nota:\t%.2f\n", nota);
```

Saída formatada

```
Nome:    Ana
Nota:  9.50
```

Exemplos de Strings de Controle

```c
#include <stdio.h>

void main (){
    printf("Esse é o meu primeiro programa:\n");
    printf("%c é uma letra\n", 'j');
    printf("%d é um número inteiro\n", 30);
    printf("%f é um número de ponto flutuante\n", 12.2);
    printf("%5.2f é um número de ponto flutuante formatado\n", 12.2);
    printf("%s é legal\n", "linguagem C");
    printf("%d + %d = %d\n", 10, 21, 10+21);
}
```

Exibição em Decimal, Octal e Hexadecimal

```c
#include <stdio.h>
void main ( ){
    char a ;
    printf ("digite um caracter: ");
    scanf ( "%c", &a );
    printf (" \n %c = %d em decimal, ",  a, a);
    printf ("%o  em octal, e %x em hexadecimal",  a,  a);
}
```

Leitura de Strings com Espaços — scanf

Problema

O `scanf` para ao encontrar um **espaço em branco** — apenas os caracteres antes do espaço são lidos.

Solução 1:

```c
#include <stdio.h>
void main() {
    char str[20];
    printf("Digite uma string qualquer: ");
    scanf("%s",str);
    printf("A string digitada foi: %s\n", str);
}
```

Solução 2:

```c
#include <stdio.h>
void main(){
    char str[20];
    printf("Digite uma string qualquer: ");
    scanf("%[A-Z a-z]",str);
    printf("A string digitada foi: %s\n", str);
 }
```

Formatando Saídas — printf

```
#include <stdio.h>
void main ( )
{
   printf ("os alunos são %2d \n", 350);
   printf ("os alunos são %4d \n", 350);
   printf ("os alunos são %5d \n", 350);
}
```

```
Saída:   os alunos são 350
        os alunos são  350
         os alunos são   350
```

Formatando Saídas — printf (cont.)

```
#include <stdio.h>
void main ( ){
      printf (" %3.1f \n", 3456.78);
      printf (" %10.3f \n", 3456.78);
}
```

```
Saída:     3456.8
          3456.780
```

getchar() e putchar()

```c
#include <stdio.h>
void main(){
    int exemplo;

    printf("Digite uma letra qualquer entre a-z: ");
    exemplo = getchar();

    printf("A letra digitada foi : ");
    putchar(exemplo);
}
```

▪ getchar()

Lê **um único caractere** da entrada padrão (teclado). Aguarda o Enter para confirmar.

▪ putchar()

Imprime **um único caractere** na saída padrão (tela). Equivalente a `printf("%c", c)`.

Exercícios

Pratique o que aprendeu!

Exercícios

1. Escreva um programa em C que **pergunte, leia e exiba** na tela o seu **nome** e **idade**.

2. Escreva um programa em C que **pergunte, leia e exiba** na tela as seguintes informações de uma pessoa:

- Nome
- Endereço
- Peso
- Altura
- Telefone
- Idade

Referências I

SCHILDT, H.; MAYER, R. *C completo e total*. Pearson Universidades, 1997. ISBN 9788534605953. Disponível em: <https://books.google.com.br/books?id=PbI0AAAACAAJ>.