

LINGUAGEM DE PROGRAMAÇÃO

Prof. Muriel Mazzetto
Algoritmos 1

Algoritmo

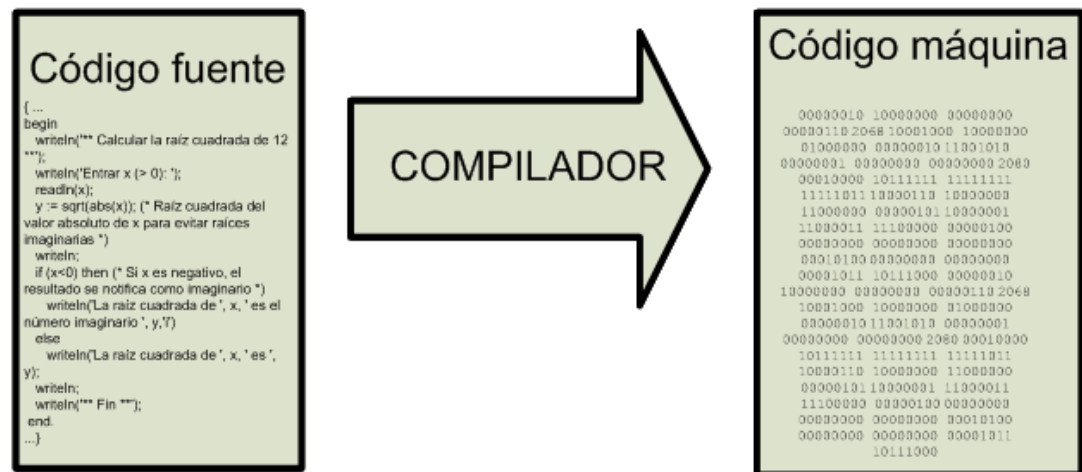
2

- **Algoritmo** é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais devendo ser executadas mecânica ou eletronicamente em um intervalo de tempo finito e com uma quantidade de esforço finita. (Wikipédia)

Programa de computador

3

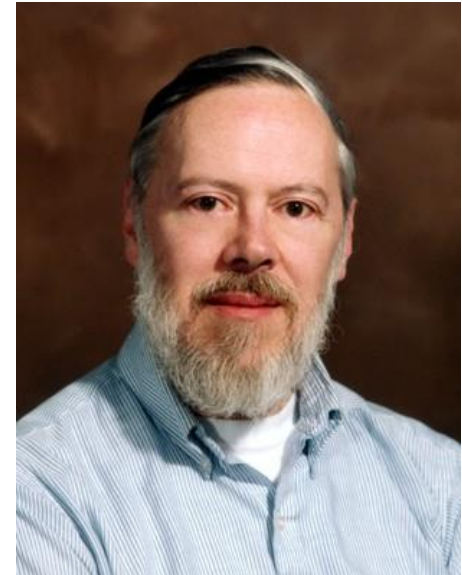
- É um conjunto de instruções que descrevem uma tarefa para ser realizada por um computador.
- Programas são escritos em uma linguagem que será traduzida para a linguagem de máquina.
- O tradutor da linguagem é chamado de *compilador*.



Linguagem C

4

- Linguagem C é uma linguagem de programação, criada em 1972, por Dennis Ritchie.
- É uma das linguagens mais populares e com arquiteturas compatíveis.
- Base de outras linguagens: C++ e Java.
- Linguagem compilada, de propósito geral, estruturada, imperativa, procedural, de alto nível.



Linguagem C

5

- ❑ Compilada: o código é executado diretamente pelo sistema operacional. Não há necessidade de interpretador instalado (*Java*).
- ❑ Propósito geral: desenvolvimento de programas variados (*SQL é só para consulta em banco de dados*).
- ❑ Estruturada: baseada em estruturas sequenciais, de decisão e de repetição.

Linguagem C

6

- ❑ Imperativa: a computação é realizada como ações (instruções/comandos) que alteram o estado (variáveis) de um programa.
- ❑ Procedural: uso de funções (modular).
- ❑ Alto nível: forma de programação mais próxima da linguagem humana, sem necessidade de manipular diretamente registradores (mesmo sendo possível).

Elementos fundamentais

7

- De acordo com o paradigma estruturado:
 - ▣ Tipos de dados;
 - ▣ Variáveis e constantes;
 - ▣ Operadores aritméticos, relacionais e lógicos;
 - ▣ Instruções de entrada e saída;
 - ▣ Estrutura sequencial;
 - ▣ Estrutura de controle e decisão;
 - ▣ Estrutura de controle e repetição;
 - ▣ Estrutura de dados;
 - ▣ Funções;

Linguagem C

8

□ Palavras reservadas / chave:

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

*ANSI

Linguagem C

9

□ Estrutura básica

```
#include <stdio.h> //biblioteca padrão

int main(void) //função principal
{ //marca de início

    /*Instruções:
    declaração de variáveis e constantes;
    obtenção de valores para variáveis;
    operações lógicas e aritméticas;
    estruturas sequenciais, de decisão e de repetição;
    apresentação de resultados;
    */

} //marca de final do bloco de instruções
```

Tipos de dados e Variáveis

10

- Declara-se uma variável utilizando:
tipo identificador;
- ▣ **tipo**: o tipo do dado (int, float, double, char, void):
 - Determina o conjunto de valores aceitos pela variável
 - Determina espaço de memória necessário para armazenar um valor. Exemplo: int – 4 bytes; char – 1 byte.
 - Define as operações que podem ser realizadas com as variáveis Exemplo: % (resto) apenas com valores inteiros (int)

Tipos de dados e Variáveis

11

- Declara-se uma variável utilizando:

Tipo	Tamanho em bits	Faixa
char	8	0 a 255
int	16 (depende da arquitetura de hardware: tamanho da palavra da máquina)	-32768 a 32767
float	32	-3.4E-38 a 3.4E38 seis dígitos de precisão
double	64	-1.7E-308 a 1.7E308 dez dígitos de precisão
void	0	Sem valor

Tipos de dados e Variáveis

12

- Declara-se uma variável utilizando:

Modificador	Aplica-se a	Significa
short	int	Menor dimensão
long	int, double	Maior dimensão
signed	char, int	Com sinal
unsigned	char, int	Sem sinal

Exemplo: **long int** -> 32 bits

Faixa -2.147.483.648 a 2.147.483.647

Tipos de dados e Variáveis

13

- Declara-se uma variável utilizando:
tipo **identificador**;
- ▣ **identificador**: nomes dados às variáveis, funções e elementos definidos pelo programador:
 - Até 32 caracteres
 - O primeiro caractere deve ser letra ou _
 - C é case sensitive: Nome, NOME e NoMe são identificadores distintos
 - O identificador **não pode** ser o nome de uma palavra reservada Exemplo: *int float*;

Tipos de dados e Variáveis

14

- Declara-se uma variável utilizando:

tipo **identificador**;

- Exemplos:

int **valor**;

float **preco** = 0.0;

char **Genero** = 'M';

double **FATORIAL**;

long int **Qtd_itens**;

int **valor**, **quantidade** = 0, **total**, **media**;

Tipos de dados e Variáveis

15

- Declara-se uma variável utilizando:

tipo **identificador**;

- Exemplos:

int **valor**;

float **preco** = 0.0;

char **Genero** = 'M'; // atribuir valor inicial ao declarar.

double **FATORIAL**;

long int **Qtd_itens**;

int **valor**, **quantidade** = 0, **total**, **media**;

Tipos de dados e Variáveis

16

- Declara-se uma variável utilizando:

tipo **identificador**;

- Exemplos:

int **valor**;

float **preco** = 0.0; // usar **ponto** para decimal

char **Genero** = 'M';

double **FATORIAL**;

long int **Qtd_itens**;

int **valor**, **quantidade** = 0, **total**, **media**;

Tipos de dados e Variáveis

17

- Declara-se uma variável utilizando:

tipo **identificador**;

- Exemplos:

int **valor**;

float **preco** = 0.0;

char **Genero** = 'M';

double **FATORIAL**;

long int **Qtd_itens**;

int **valor, quantidade = 0, total, media**; //mais de uma variável, separadas por vírgula.

Atribuição de valores

18

- Utiliza-se o símbolo $=$ para atribuir um valor em uma variável. Pode-se atribuir:
 - ▣ Um valor (constante);
 - ▣ Uma variável (o valor contido na variável);
 - ▣ Uma expressão aritmética (o resultado da expressão);
 - ▣ Uma chamada de função (resultado retornado);

Constantes

19

- Uma constante é um valor armazenado em uma região de memória que não é alterado durante a execução do programa.

- Define-se a constante em tempo de projeto (codificando), utilizando:

#define **identificador** **valor**

- **#define**: comando para declarar uma constante;
- **identificador**: nome da constante, utilizando no código para acessar o conteúdo;
- **valor**: o conteúdo atribuído, inalterável na execução

Constantes

20

- Uma constante é um valor armazenado em uma região de memória que não é alterado durante a execução do programa.

- Define-se a constante em tempo de projeto (codificando), utilizando:

#define **identificador** **valor** //SEM ; AO FINAL

- **#define**: comando para declarar uma constante;
- **identificador**: nome da constante, utilizando no código para acessar o conteúdo;
- **valor**: o conteúdo atribuído, inalterável na execução

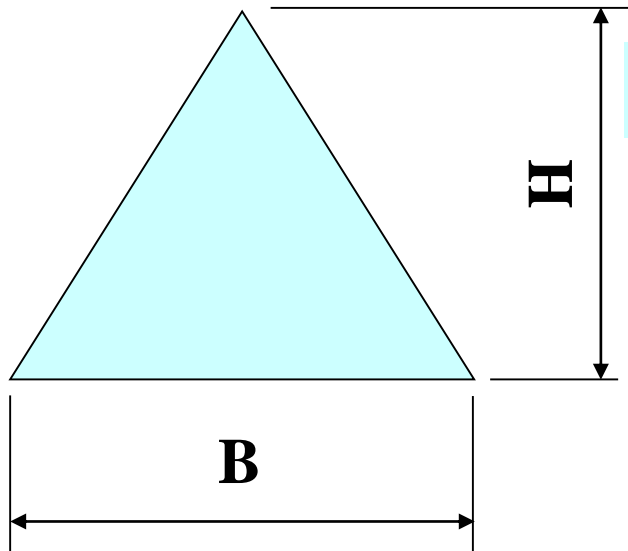
Operadores aritméticos

21

Operador	Ação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto de divisão inteira
-	Subtração (unário)
--	Decremento
++	Incremento

Operadores aritméticos

22



Triângulo de base (b) e altura (h)

$$\text{área} = \frac{b \times h}{2}$$

→ matemática

$$\text{area} = (b * h) / 2;$$

→ computacional

Operadores aritméticos

23

- A linguagem C possibilita a realização de operações matemáticas.
- Algumas operações matemáticas complexas estão disponíveis em bibliotecas externas.
 - ▣ A biblioteca `math.h` contém funções para calcular cosseno, seno, raiz quadrada, potência e outras.
 - $\sqrt{valor} \rightarrow \text{sqrt}(valor)$

Precedência de operadores

24

- Os operadores aritméticos seguem a regra de precedência da matemática.
 - O uso de parênteses altera a precedência.
 - Operadores de mesma precedência são executados da esquerda para a direita.
-
- ▣ $8 * 5 + 3 - 5;$
 - ▣ $8 * (5 + 3) - 5;$
 - ▣ $8 * (5 + 3 - 5);$

Compatibilidade de tipo

25

- As operações aritméticas devem ser realizadas com tipos de dados compatíveis.
- Quando necessário, utilize a **conversão de tipo**.
- A conversão é chamada de *typecasting* ou *cast*.
 - ▣ Basta adicionar o comando (*tipo*) antes da variável que será convertida.
 - `float NUM = 5.5;`
 - `int Var = (int) NUM;`
 - `char Letra = (char) 65;`

Função de saída de dados

26

- Escrita de dados da saída padrão (monitor):
`printf(“string de controle”, lista de argumentos);`
 - ▣ **string de controle:** textos, identificadores e formatação dos tipos de dados que serão escritos;
 - ▣ **lista de argumentos:** os identificadores das variáveis que contem valores para serem escritos.

Função de saída de dados

27

- Escrita de dados da saída padrão (monitor):
`printf(“string de controle”, lista de argumentos);`
 - ▣ **string de controle**: textos, identificadores e formatação dos tipos de dados que serão escritos;
 - ▣ **lista de argumentos**: os identificadores das variáveis que contem valores para serem escritos.
- Exemplos:
`printf(“escrever inteiro: %d \t”, idade);`
`printf(“\nEscrever char: %c\n”, genero);`

Função de saída de dados

28

- Escrita de dados da saída padrão (monitor):
`printf("string de controle", lista de argumentos);`
 - ▣ **string de controle:** textos, identificadores e formatação dos tipos de dados que serão escritos;
 - ▣ **lista de argumentos:** os identificadores das variáveis que contem valores para serem escritos.

- Exemplos:

```
printf("escrever inteiro: %d \t", idade);  
printf("\nEscrever char: %c\n", genero);
```

%X, onde **X** depende do tipo de dado a ser lido

Função de saída de dados

29

□ Escrever dados da saída padrão (monitor)

p	%c	→ caractere	
□	%d ou %i	→ inteiro	ação
	%ld ou %li	→ inteiro longo	
□	%e	→ número ou notação científica	veis
	%f	→ ponto flutuante	
	%o	→ octal	
□	%x	→ hexadecimal	
Ex	%s	→ string (cadeia de caracteres)	
p	%lf	→ double	
p	%lu	→ endereço de memória	

%X, onde **X** depende do tipo de dado a ser lido

Função de saída de dados

30

- Escrita de dados da saída padrão (monitor):
`printf("string de controle", lista de argumentos);`
 - ▣ **string de controle:** textos, identificadores e formatação dos tipos de dados que serão escritos;
 - ▣ **lista de argumentos:** os identificadores das variáveis que contem valores para serem escritos.

- Exemplos:

```
printf("escrever inteiro: %d \t", idade);
```

```
printf("\nEscrever char: %c\n", genero);
```

`\t` define a tabulação e `\n` define a quebra de linha.

Função de saída de dados

31

- Escrita de dados da saída padrão (monitor):

`\n` nova linha

`\r` enter

`\t` tabulação

`\b` retrocesso

`\"` imprimir aspas

`\\` imprimir barra

`%%` para imprimir símbolo %

`\t` define a tabulação e `\n` define a quebra de linha.

Função de saída de dados

32

- Formatar casas decimais:
 - ▣ Após o ponto:

```
printf("%.2f", 324.749);
```

Saída: 324.75

```
printf("%.0f", 324.749);
```

Saída: 324

Função de saída de dados

33

- Formatar casas decimais:
 - ▣ Antes do ponto

```
printf("V = %5d\n", 3);  
printf("V = %5d\n", 33);  
printf("V = %5d\n", 333);
```

Saída:

```
V =      3  
V =     33  
V =    333
```

Função de saída de dados

34

- Formatar casas decimais:
 - ▣ Antes do ponto

```
printf("V = %5d\n", 3);  
printf("V = %5d\n", 33);  
printf("V = %5d\n", 333);
```

Saída:

V = 00003

V = 00033

V = 00333

Preenche as casas decimais com espaços vazios,
equivalente a quantidade de zeros a esquerda.

Função de entrada de dados

35

- Leitura de dados da entrada padrão (teclado):
scanf(**“string de controle”**, **lista de argumentos**);
 - ▣ **string de controle**: textos e identificadores dos tipos de dados que serão lidos;
 - ▣ **lista de argumentos**: os identificadores das variáveis que armazenarão os valores lidos.

Função de entrada de dados

36

- Leitura de dados da entrada padrão (teclado):
scanf(**“string de controle”**, **lista de argumentos**);
 - ▣ **string de controle**: textos e identificadores dos tipos de dados que serão lidos;
 - ▣ **lista de argumentos**: os identificadores das variáveis que armazenarão os valores lidos.
- Exemplos:
scanf(**“%d”**, &idade);
scanf(**“%c”**, &genero);

Função de entrada de dados

37

- Leitura de dados da entrada padrão (teclado):
`scanf("string de controle", lista de argumentos);`
 - ▣ **string de controle:** textos e identificadores dos tipos de dados que serão lidos;
 - ▣ **lista de argumentos:** os identificadores das variáveis que armazenarão os valores lidos.
- Exemplos:
`scanf("%d", &idade);`
`scanf("%c", &genero);`

`%X`, onde **X** depende do tipo de dado a ser lido

Função de entrada de dados

38

□ Leitura de dados da entrada padrão (teclado);

sc	<code>%c</code>	→ caractere	
□	<code>%d</code> ou <code>%i</code>	→ inteiro	os de
	<code>%ld</code> ou <code>%li</code>	→ inteiro longo	
	<code>%e</code>	→ número ou notação científica	veis
□	<code>%f</code>	→ ponto flutuante	
	<code>%o</code>	→ octal	
	<code>%x</code>	→ hexadecimal	
□ Ex	<code>%s</code>	→ string (cadeia de caracteres)	
sc	<code>%lf</code>	→ double	
sc	<code>%lu</code>	→ endereço de memória	

`%X`, onde **X** depende do tipo de dado a ser lido

Função de entrada de dados

39

- Leitura de dados da entrada padrão (teclado):
scanf(**“string de controle”**, **lista de argumentos**);
 - ▣ **string de controle**: textos e identificadores dos tipos de dados que serão lidos;
 - ▣ **lista de argumentos**: os identificadores das variáveis que armazenarão os valores lidos.
- Exemplos:
scanf(**“%d”**, **&idade**);
scanf(**“%c”**, **&genero**);

& define o endereço de memória, p/ armazenar o valor

Questionário de revisão

40

- 1 O que é Algoritmo?
- 2 Qual a estrutura básica de um Algoritmo?
- 3 O que significa uma linguagem ser: compilada, estruturada e imperativa?
- 4 Quais as funções básicas de entrada e saída de um programa em C? Exemplifique.

Exemplos

41

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Olá mundo!");
6
7      return 0;
8  }
```

Exemplos

42

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Olá mundo!");
6
7      return 0;
8  }
```

"C:\Users\MurIEL\Dropbox\1. UTFPR - DV\Algoritmos 1\Resolução de Listas de Exercício\Lista 01\ex1.exe"

Olá mundo!

Process returned 0 (0x0) execution time : 0.363 s

Press any key to continue.

Exemplos

43

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char meu_caractere = 'a';
6      int meu_inteiro = 10;
7      float meu_float = 1.1;
8      double meu_double = 1.1234567;
9
10     printf("Meu caractere: %c\n", meu_caractere);
11     printf("Meu inteiro: %d\n", meu_inteiro);
12     printf("Meu float: %f\n", meu_float);
13     printf("Meu double: %lf\n", meu_double);
14
15     return 0;
16 }
```

Exemplos

44

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char meu_caractere = 'a';
6      int meu_inteiro = 10;
7      float meu_float = 1.1;
8      double meu_double = 1.1234567;
9
10     printf("Meu caractere: %c\n", meu_caractere);
11     printf("Meu inteiro: %d\n", meu_inteiro);
12     printf("Meu float: %f\n", meu_float);
13     printf("Meu double: %lf\n", meu_double);
14
15     return 0;
16 }
```

"C:\Users\Murie\Dropbox\1. UTFPR - DV\Algoritmos 1\Resolu o de Listas de E

```
Meu caractere: a
Meu inteiro: 10
Meu float: 1.100000
Meu double: 1.123457
```

Exemplos

45

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("char: %d bytes\n", sizeof(char));
6      printf("int: %d bytes\n", sizeof(int));
7      printf("float: %d bytes\n", sizeof(float));
8      printf("double: %d bytes\n", sizeof(double));
9
10     return 0;
11 }
```

Exemplos

46

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("char: %d bytes\n", sizeof(char));
6      printf("int: %d bytes\n", sizeof(int));
7      printf("float: %d bytes\n", sizeof(float));
8      printf("double: %d bytes\n", sizeof(double));
9
10     return 0;
11 }
```

"C:\Users\Muriel\Dropbox\1. UTFPR - DV\Algoritmos 1\ResoluçÕo de Li

```
char: 1 bytes
int: 4 bytes
float: 4 bytes
double: 8 bytes
```

Exemplos

47

```
#include <stdio.h>

int main(void)
{
    int a, b;
    int r1, r2, r3;
    float r4;

    a = 10;
    b = 3;

    r1 = a + b;
    printf("Soma de %d e %d = %d\n", a, b, r1);

    r2 = a - b;
    printf("Subtracao de %d e %d = %d\n", a, b, r2);

    r3 = a * b;
    printf("Multiplicacao de %d e %d = %d\n", a, b, r3);

    r4 = ((float) a) / ((float) b);
    printf("Divisao de %d e %d = %.2f\n", a, b, r4);

    return 0;
}
```

Exemplos

48

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

```
int main(void)
{
```

```
    int a, b;
    int r1, r2, r3;
    float r4;
```

```
    a = 10;
    b = 3;
```

```
    r1 = a + b;
    printf("Soma de %d e %d = %d\n", a, b, r1);
```

```
    r2 = a - b;
    printf("Subtracao de %d e %d = %d\n", a, b, r2);
```

```
    r3 = a * b;
    printf("Multiplicacao de %d e %d = %d\n", a, b, r3);
```

```
    r4 = ((float) a) / ((float) b);
    printf("Divisao de %d e %d = %.2f\n", a, b, r4);
```

```
    return 0;
```

```
}
```

"C:\Users\Muriele\Dropbox\1. UTFPR - DV\Algoritmos 1\Resolu o de Listas de Exerc cio\Lista 01\ex4.exe"

Soma de 10 e 3 = 13

Subtracao de 10 e 3 = 7

Multiplicacao de 10 e 3 = 30

Divisao de 10 e 3 = 3.33

Exemplos

49

```
#include <stdio.h>

int main(void)
{
    int a, b;
    int r1;
    float r4;

    printf("Informe o valor de a: ");
    scanf("%d", &a);
    printf("Informe o valor de b: ");
    scanf("%d", &b);

    r1 = a + b;
    printf("Soma de %d e %d = %d\n", a, b, r1);

    r4 = ((float) a) / ((float) b);
    printf("Divisao de %d e %d = %.2f\n", a, b, r4);

    return 0;
}
```

Exemplos

50

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

```
int main(void)  
{
```

```
    int a, b;  
    int r1;  
    float r4;
```

```
    printf("Informe o valor de a: ");  
    scanf("%d", &a);  
    printf("Informe o valor de b: ");  
    scanf("%d", &b);
```

```
    r1 = a + b;  
    printf("Soma de %d e %d = %d\n", a, b, r1);
```

```
    r4 = ((float) a) / ((float) b);  
    printf("Divisao de %d e %d = %.2f\n", a, b, r4);
```

```
    return 0;
```

```
}
```

"C:\Users\Muriel\Dropbox\1. UTFPR - DV\Algoritmos 1\Resolu o de Listas de Exerc cio\Lista 01\ex4_scanf.exe"

```
Informe o valor de a: 5  
Informe o valor de b: 10  
Soma de 5 e 10 = 15  
Divisao de 5 e 10 = 0.50
```

Exercício

51

- ❑ Faça um código que leia a largura, o comprimento e a altura de um bloco.
- ❑ Seu código deve calcular o volume, baseado nas medidas lidas.
- ❑ O volume deve ser informado ao usuário.