LINGUAGEM DE PROGRAMAÇÃO

Prof. Muriel Mazzetto

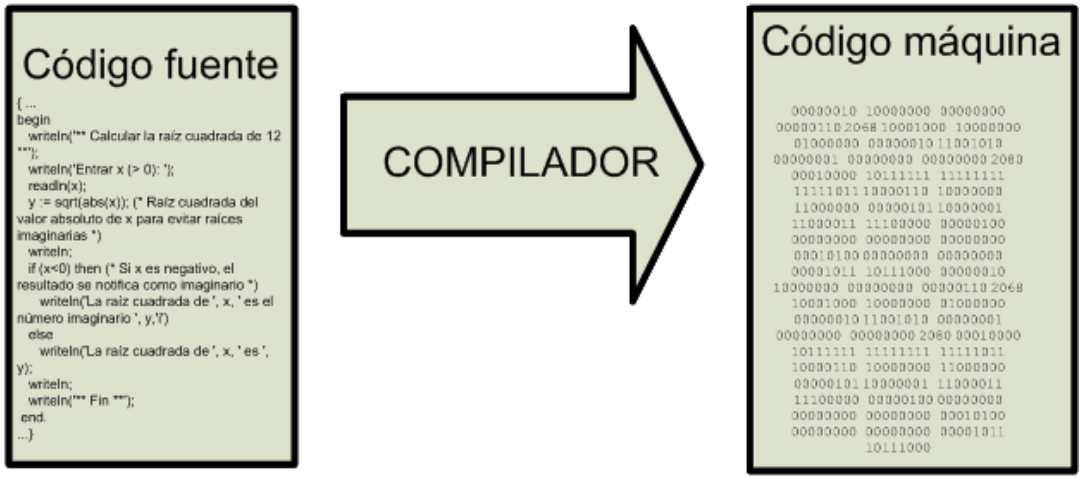
Algoritmos 1

Algoritmo

- **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

- 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

- 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

- 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

- 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

- 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

- 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

- Estrutura básica

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

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

    /*Instruções:
    declaração de variáveis e constantes;
    obtençao 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

- 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

- Declara-se uma variável utilizando:

Type	Size (in bytes)	Range
char	1	-127 to 127 or 0 to 255
unsigned char	1	0 to 255
int	4	-2147483648 to 2147483647
unsigned int	4	0 to 4294967295
short int	2	-32768 to 32767
unsigned short int	2	0 to 65,535
float	4	+/- 3.4e +/- 38 (~7 digits)
double	8	+/- 1.7e +/- 308 (~15 digits)

Tipos de dados e Variáveis

- 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

Tipos de dados e Variáveis

- 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

- 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

- 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

- 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

- 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

- Utiliza-se o símbolo = para atribuir um valor à 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

- 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

- 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:

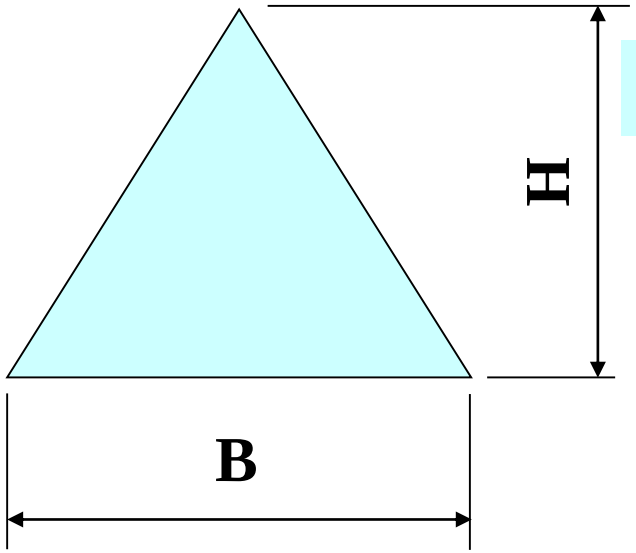
  #define identificador valor //SEM ; AO FINAL

  - **#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

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

Operadores aritméticos



Triângulo de base (b) e altura (h)

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

area = (b * h) / 2; ⟶ computacional

Operadores aritméticos

- 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.
    - $\sqrt{valor} \rightarrow sqrt(valor)$

Precedência de operadores

- Os operadores aritméticos seguem as regras 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

- 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

- 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

- 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

- 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

- Es
  - p
    - ração
  - veis
- Ex
  - p
  - p

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

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

Função de saída de dados

- 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

- Escrita de dados da saída padrão (monitor):
  - p
    - ração
  - veis
- E
  - p
  - p

\n nova linha
\r enter
\t tabulação
\b retrocesso
\" imprimir aspas
\\ imprimir contra barra
%% para imprimir símbolo %

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

Função de saída de dados

- Formatar casas decimais:
  - Após o ponto:

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

Saída: 324.75

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

Saída: 324

Função de saída de dados

- 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

- 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 com espaços vazios, equivalente a quantidade de zeros a esquerda.

Função de entrada de dados

- 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

- 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

- 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

- Le
  - so
    - os de
  - veis
- Ex
  - so
  - so

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

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

Função de entrada de dados

- 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

Função de entrada de dados

- Le
  - so
    - os de
    - veis
- Ex
  - so
  - so

IMPORTANTE: No scanf() não são utilizadas formatações de números nem de texto, como \n ou %.2f.

Questionário de revisão

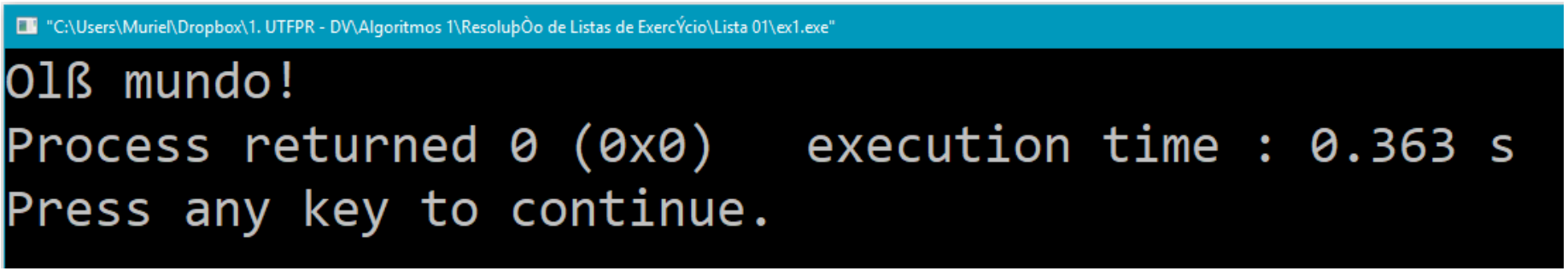
- 1 O que é Algoritmo?
- 2 Qual a estrutura básica de um Algoritmo?
- 3 O que significa uma linguagem ser: compilada, estruturada e imperativa?
- 4 Quais as funções básicas de entrada e saída de um programa em C? Exemplifique.

Exemplos


```c
#include <stdio.h>

int main(void)
{
    printf("Olá mundo!");

    return 0;
}
```

Exemplos


```c
#include <stdio.h>

int main(void)
{
    printf("Olá mundo!");

    return 0;
}
```

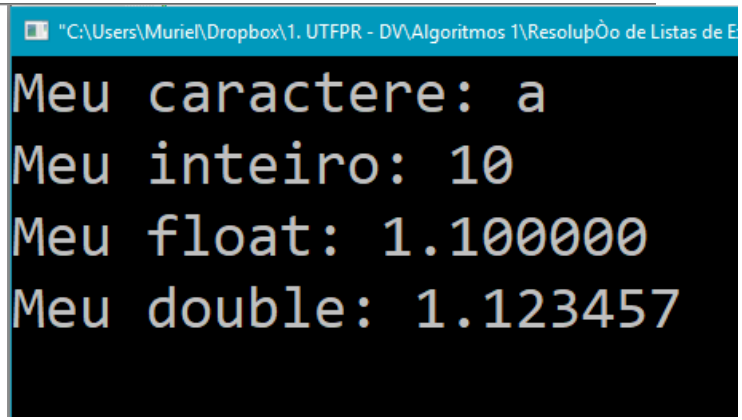


Exemplos


```c
#include <stdio.h>

int main(void)
{
    char meu_caractere = 'a';
    int meu_inteiro = 10;
    float meu_float = 1.1;
    double meu_double = 1.1234567;

    printf("Meu caractere: %c\n", meu_caractere);
    printf("Meu inteiro: %d\n", meu_inteiro);
    printf("Meu float: %f\n", meu_float);
    printf("Meu double: %lf\n", meu_double);

    return 0;
}
```

Exemplos


```c
#include <stdio.h>

int main(void)
{
    char meu_caractere = 'a';
    int meu_inteiro = 10;
    float meu_float = 1.1;
    double meu_double = 1.1234567;

    printf("Meu caractere: %c\n", meu_caractere);
    printf("Meu inteiro: %d\n", meu_inteiro);
    printf("Meu float: %f\n", meu_float);
    printf("Meu double: %lf\n", meu_double);

    return 0;
}
```

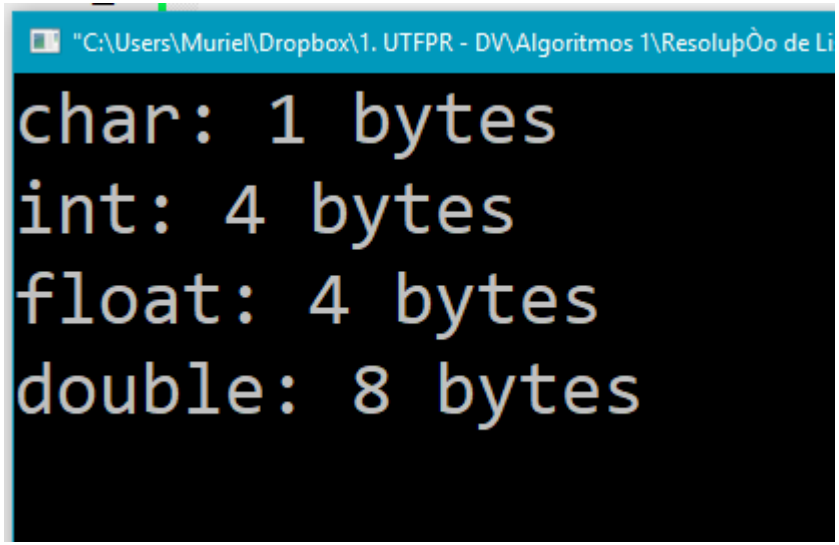


Exemplos


```c
#include <stdio.h>

int main(void)
{
    printf("char: %d bytes\n", sizeof(char));
    printf("int: %d bytes\n", sizeof(int));
    printf("float: %d bytes\n", sizeof(float));
    printf("double: %d bytes\n", sizeof(double));

    return 0;
}
```

Exemplos


```c
#include <stdio.h>

int main(void)
{
    printf("char: %d bytes\n", sizeof(char));
    printf("int: %d bytes\n", sizeof(int));
    printf("float: %d bytes\n", sizeof(float));
    printf("double: %d bytes\n", sizeof(double));

    return 0;
}
```

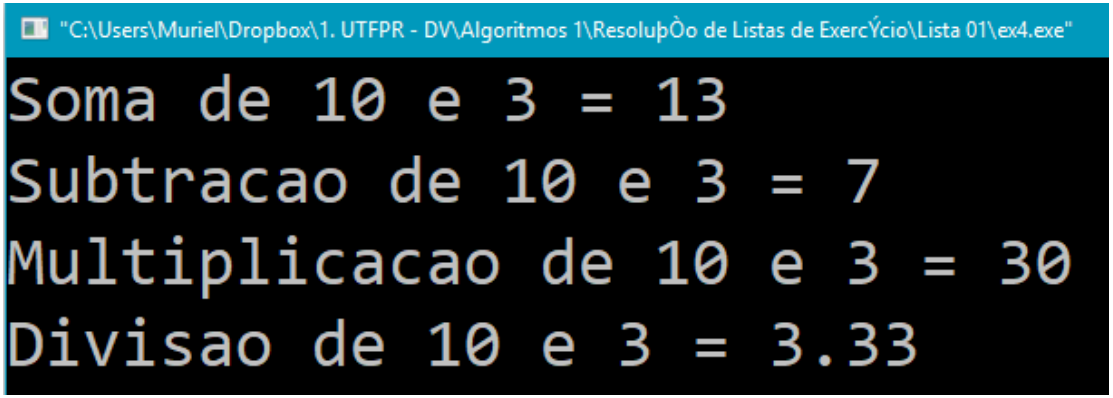


Exemplos


```c
#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


```c
#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;
}
```

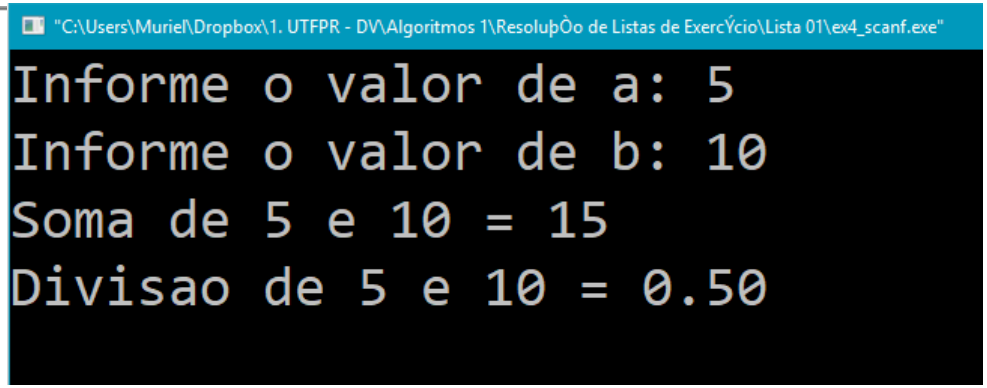


Exemplos


```c
#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


```c
#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;
}
```



Exercício

- 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.