

## Processo de geração de um Programa

- ✓ Análise e Definição do Problema
- ✓ Projeto do Algoritmo
- ✓ Validação do Algoritmo (teste de mesa)
- Tradução do Algoritmo para uma linguagem de programação (codificação e compilação)
- Teste e Depuração
- Implementação

Slide 1

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Processo de solução de problemas

### Enunciado do Problema:

Ler dois valores informados pelo teclado, calcular e informar a soma destes valores

Slide 2

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Análise e Definição do Problema

### Identificar objetivo, entradas e saídas:

**Objetivo:** Calcular a soma de 2 valores  
**Entradas:** 2 valores numéricos  
**Saída:** 1 valor numérico ↔ **SOMA**

atribuição de valor

saída de dados

Slide 3

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Projeto do Algoritmo

### Algoritmo soma2

{ Calcula e informa a soma de 2 valores lidos do teclado  
**entradas:** dois valores  
**saídas:** a soma dos 2 valores lidos }

1. **início**
2. **ler o primeiro valor**
3. **ler o segundo valor**
4. **efetuar a soma**
5. **informar a soma**
6. **fim.**

**Comandos:** claros e precisos  
**Substantivo + verbo???**

**Propriedades:**

- ✓ possui um estado inicial;
- ✓ contém uma sequência lógica e finita de ações (comandos), claras e precisas, com fluxo de execução baseado em:
  - sequência;
  - seleção condicional (seleção de ações);
  - iteração (repetição de ações);
- ✓ possui dados de entrada;
- ✓ produz dados de saída corretos;
- ✓ possui estado final previsível;
- ✓ deve ser eficaz.

- ✓ Identificar o algoritmo (identificador);
- ✓ Incluir no topo finalidade do algoritmo, entradas e saídas;
- ✓ Usar comandos claros e precisos (linguagem algorítmica);
- ✓ Usar 1 comando por linha (substantivo + verbo);
- ✓ Usar indentação (recuo de margens para indicar hierarquia de cada linha);
- ✓ Usar nomes significativos e curtos para dados;
- ✓ Utilizar espaços e linhas em branco para maior legibilidade do algoritmo;
- ✓ Nunca utilizar desvios através de 'vai para' (go to).

Slide 4

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Linguagem algorítmica - refinamento -

### Algoritmo Soma2

{ Calcula e informa a soma de 2 valores lidos  
**entradas:** e1, e2 (valores lidos)  
**saídas:** e3 (soma) }

1. **início**
2. **ler e1**
3. **ler e2**
4. **e3 ← e1 + e2**
5. **escrever e3**
6. **fim.**

**Nomes:** significativos ??  
(mnemônicos)

**Algoritmo Soma2**  
{ Calcula e informa a soma de 2 valores lidos  
**entradas:** dois valores  
**saídas:** a soma dos 2 valores lidos  
1. início  
2. ler o primeiro valor  
3. ler o segundo valor  
4. efetuar a soma  
5. informar a soma  
6. fim.

**Propriedades:**

- ✓ possui um estado inicial;
- ✓ contém uma sequência lógica e finita de ações (comandos), claras e precisas, com fluxo de execução baseado em:
  - sequência;
  - seleção condicional (seleção de ações);
  - iteração (repetição de ações);
- ✓ possui dados de entrada;
- ✓ produz dados de saída corretos;
- ✓ possui estado final previsível;
- ✓ deve ser eficaz.

- ✓ Identificar o algoritmo (identificador);
- ✓ Incluir no topo finalidade do algoritmo, entradas e saídas;
- ✓ Usar comandos claros e precisos (linguagem algorítmica);
- ✓ Usar 1 comando por linha (substantivo + verbo);
- ✓ Usar indentação (recuo de margens para indicar hierarquia de cada linha);
- ✓ Usar nomes significativos e curtos para dados;
- ✓ Utilizar espaços e linhas em branco para maior legibilidade do algoritmo;
- ✓ Nunca utilizar desvios através de 'vai para' (go to).

Slide 5

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Dados (variáveis) - nomes significativos -

### Algoritmo Soma2

{ Calcula e informa a soma de 2 valores lidos do teclado  
**entradas:** val1, val2 (valores lidos)  
**saídas:** soma }

1. **início**
2. **ler val1**
3. **ler val2**
4. **soma ← val1 + val2**
5. **escrever soma**
6. **fim.**

**Algoritmo Soma2**  
{ Calcula e informa a soma de 2 valores lidos  
**entradas:** e1, e2 (valores lidos)  
**saídas:** e3 (soma)  
1. início  
2. ler e1  
3. ler e2  
4. e3 ← e1 + e2  
5. escrever e3  
6. fim.

**Propriedades:**

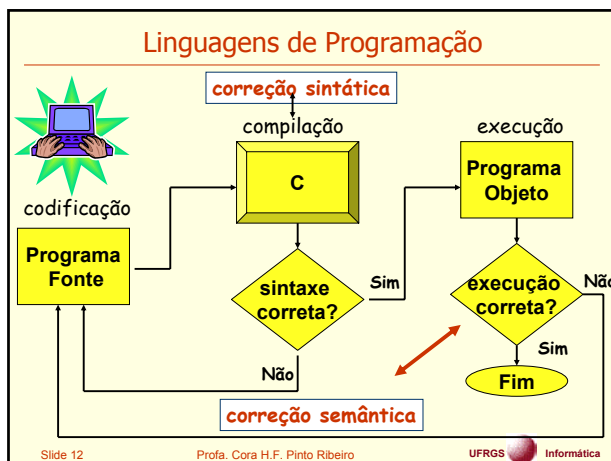
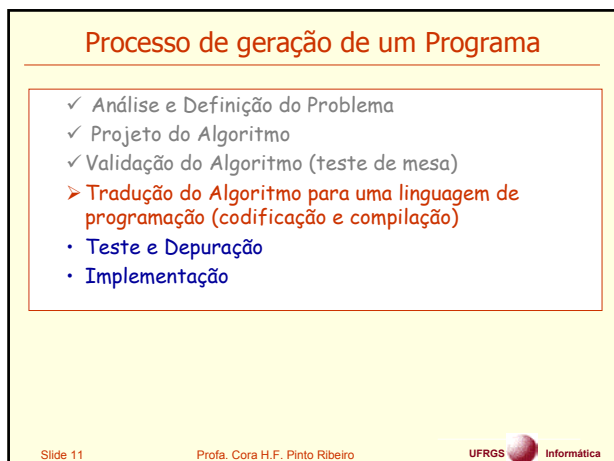
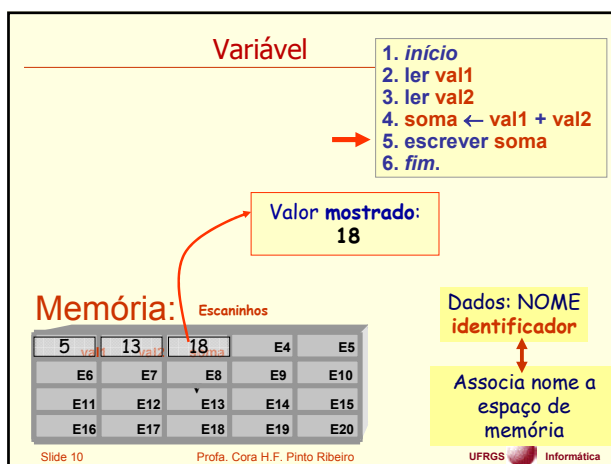
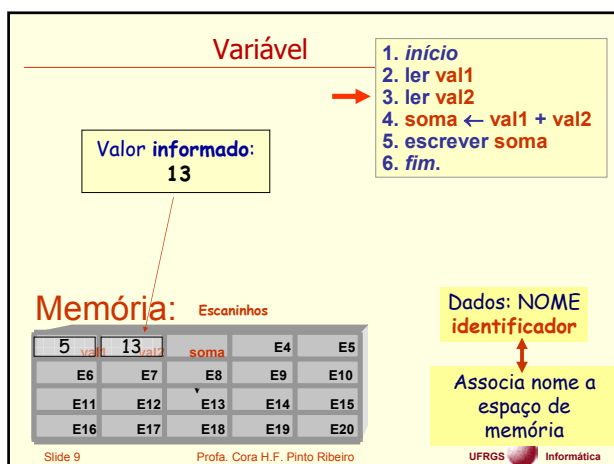
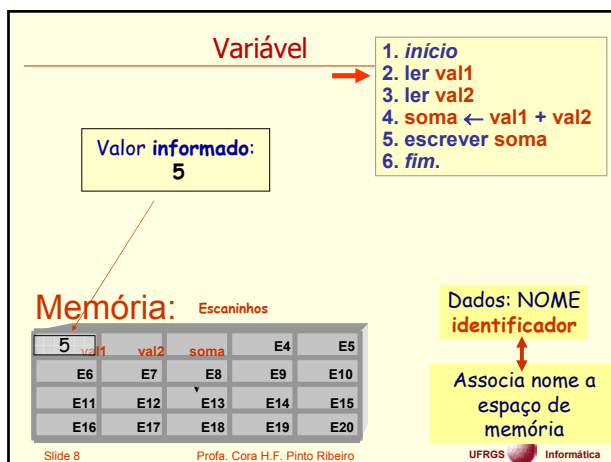
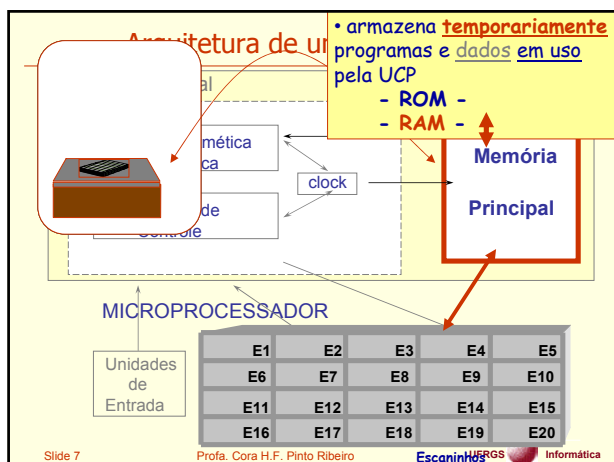
- ✓ possui um estado inicial;
- ✓ contém uma sequência lógica e finita de ações (comandos), claras e precisas, com fluxo de execução baseado em:
  - sequência;
  - seleção condicional (seleção de ações);
  - iteração (repetição de ações);
- ✓ possui dados de entrada;
- ✓ produz dados de saída corretos;
- ✓ possui estado final previsível;
- ✓ deve ser eficaz.

- ✓ Identificar o algoritmo (identificador);
- ✓ Incluir no topo finalidade do algoritmo, entradas e saídas;
- ✓ Usar comandos claros e precisos (linguagem algorítmica);
- ✓ Usar 1 comando por linha (substantivo + verbo);
- ✓ Usar indentação (recuo de margens para indicar hierarquia de cada linha);
- ✓ Usar nomes significativos e curtos para dados;
- ✓ Utilizar espaços e linhas em branco para maior legibilidade do algoritmo;
- ✓ Nunca utilizar desvios através de 'vai para' (go to).

Slide 6

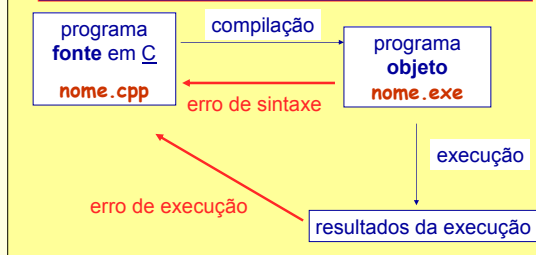
Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática



## Codificação x Execução

**Ambiente de programação C: tradução para linguagem de máquina e execução do programa**



Slide 13

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Linguagem de Programação

- ✓ fornece conjunto de convenções e regras que possibilitam instruir o computador na execução de tarefas;
- ✓ lê programa escrito na linguagem específica (linguagem de programação);
- ✓ verifica sintaxe (compilação): se correta, traduz para linguagem de máquina;
- ✓ executa programa em linguagem de máquina (execução).

Slide 14

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Definição de Linguagem de Programação

- ✓ **Sintática:** Uma linguagem de programação é uma notação utilizada pelo programador para especificar ações a serem executadas por um computador:
  - construções corretas do programa, de acordo com a linguagem.
- ✓ **Semântica:** Uma linguagem de programação compreende um conjunto de conceitos que um programador usa na resolução dos problemas propostos:
  - recursos disponíveis, compatíveis com o modelo da linguagem.

Slide 15

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Elementos léxicos do C - caracteres

- **Letras:** maiúsculas e minúsculas
- **Dígitos:** 0 a 9
- **Sublinhado:** \_
- **Outros símbolos especiais:** # /\* " . , ; .. ( ) { } : = == ^

Slide 16

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Sintaxe: Livre ou Sensível ao Contexto

- C - linguagem sensível ao contexto:**
- maiúsculas e minúsculas são diferentes (no programa, nada a ver com dados!!!)
  - palavras reservadas sempre em minúsculas!
  - variáveis: maiúsculas e minúsculas;
  - prática adotada:
    - programa: **todo em minúsculas**;
    - exceção(adotada): nome de constantes (*define*) em maiúsculas.

Slide 17

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Estrutura de um programa em C

```
/* comentário inicial, descrevendo objetivos do programa */1  
  
//declarações, incluindo bibliotecas, constantes,  
//elementos globais e outras declarações  
  
int main()1 // função principal e obrigatório  
{  
    //declarações locais e comandos//  
}
```

Declarações

Programa principal

<sup>1</sup> Formato obrigatório, para adequação à Programação Estruturada

Slide 18

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Comentário

- ✓ Permitem a inclusão de observações, que visam auxiliar a compreensão do que está sendo executado pelo programa.
- ✓ Podem adotar 2 formas:
  - `//` identifica início de comentário, reserva até o fim da linha.
  - `/* ..... */` delimita início e fim de comentário, podendo utilizar mais de uma linha entre eles.
- ✓ **Programação Estruturada :**  
→ **SEMPRE** incluir comentário na 1ª linha, contendo **descrição do programa!!!!**

Slide 19

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Identificadores

- ✓ Nomes que servem para identificar, fazer referência e definir o conteúdo de **variáveis**, funções e outros objetos definidos pelo usuário.
- ✓ **Sintaxe de identificador:**
  - 1º caractere: **letra** ou **\_** (sublinhado) (**usar letra**)
  - 2º caractere em diante: **letras**, **dígitos** e/ou **\_**
- ✓ **Programação Estruturada :**  
→ **SEMPRE** utilizar nome significativos e curtos!!!

Slide 20

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Palavras reservadas

- ✓ **Identificadores** associados aos comandos que compõem esta linguagem de programação.
- ✓ As 32 palavras reservadas padrão são:  
`auto break1 case char const continue default do double else enum extern float for goto2 if int long register return short signed sizeof static struct switch typedef union unsigned void volatile while`
- <sup>1</sup> programação estruturada: uso limitado.
- <sup>2</sup> programação estruturada: uso **proibido**.

Slide 21

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Sintaxe da linguagem C

Componentes reconhecidos pela linguagem C (sintaxe da linguagem):

- **tipos** - propriedades dos dados;
- **declarações** - partes do programa, podendo dar significado a um identificador, alocar memória, definir conteúdo inicial e definir funções;
- **funções** - ações executadas pelo programa;
- **expressões** - define e altera valores e chama funções de I/O;
- **comentários** - auxilia a compreensão, sem tradução para o executável.

Slide 22

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Tipos de dados

- ✓ Determina o conteúdo e tamanho de área de memória reservada para o dado.
- ✓ No C, os tipos básicos de dados são:

Tipo	Tamanho	Valores Válidos
<b>char</b>	1 byte	Caracteres do <u>Código ASCII</u>
<b>int</b>	4 bytes *	-32768 a +32767
<b>float</b>	4 bytes	-3.4e38 a +3.4e38
<b>double</b>	8 bytes	-1.7e308 a +1.7e308

\* referências: 2 bytes, mas sizeof é 4.

Slide 23

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Variáveis

- ✓ Locais **identificados** onde **dados** são armazenados e recuperados, **durante** a execução de um programa.
- ✓ Devem ser declaradas **antes** de serem usadas, com a especificação do **tipo de conteúdo válido**.
- ✓ Identificador: **nome descritivo**<sup>1</sup> do uso da variável.
- ✓ **Conteúdos de variáveis são atribuídos** através de:
  - comando de leitura **scanf**
  - comando de atribuição **=**

<sup>1</sup> Obrigatório, para adequação à Programação Estruturada

Slide 24

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Estrutura de um programa em C

```
/* comentário inicial, descrevendo objetivos do
programa */1
```

```
//declarações , incluindo bibliotecas, constantes,
elementos globais e outras declarações
```

```
int main( )1 // função principal e obrigatório
```

```
{
    //declarações locais e comandos//
}
```

Declarações

Programa principal

<sup>1</sup> Formato obrigatório, para adequação à Programação Estruturada

Slide 25

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Diretivas para o processador - Biblioteca

- ✓ Bibliotecas contêm funções pré-definidas, utilizadas nos programas para ações específicas.
- ✓ A biblioteca desejada é inserida pelo pré-processador a partir da diretiva **#include**.
- ✓ Exemplos de bibliotecas disponíveis no C:

#include <stdio.h>	Funções de entrada e saída
#include <stdlib.h>	Funções padrão, como <b>system</b>
#include <math.h>	Funções matemáticas
#include <system.h>	Funções do sistema
#include <string.h>	Funções de texto

Slide 26

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Biblioteca system

- ✓ Biblioteca que implementa o funcionamento das funções `scanf` e `printf`, ou seja, inclui os recursos necessários para as operações padrão de entrada e saída (I/O) de dados.

Slide 27

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Biblioteca stdlib

- ✓ Biblioteca que inclui diversas funções de uso geral, incluindo recursos de tela e comunicação com o ambiente, geração de números randômicos, alocação dinâmica de memória, conversões, buscas e classificações.
- ✓ Possibilita entre outras coisas, que a execução/conclusão do programa seja interrompida até que uma tecla qualquer seja pressionada. Isto pode ser feito de duas formas:
  1. `system("pause");` - antes de interromper, imprime na tela a mensagem `Pressione qualquer tecla para continuar...`
  2. `system("pause>>null");` - interrompe, sem imprimir mensagem na tela, aguardando que uma tecla seja pressionada.

Slide 28

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Estrutura de um programa em C - esqueleto

```
/* comentário inicial, descrevendo objetivos do
programa */
#include <stdlib.h> // para usar system
#include <stdio.h> // para entrada e saída de dados
// outras bibliotecas necessárias
// outras declarações
```

```
int main( ) // função principal e obrigatório
```

```
{
    //declarações e comandos//

    system("pause"); /* ou system("pause >> null");
                     - para eliminar mensagem */
    return 0; // para encerramento normal do programa
}
```

Declarações

Programa principal

Slide 29

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Declaração de variáveis

Sintaxe:

<tipo> <nome\_da\_variavel>;

- ✓ declaração das variáveis / conteúdos usadas no programa;
- ✓ nomes: letra inicial + letras (minúsculas) dígitos ou \_ (até 32)
- ✓ tipos principais:
  - int / inteiro
  - float / decimal, precisão simples
  - double / decimal, precisão dupla
  - char / 1 único caractere
- ✓ exemplos:
  - int num;
  - int quant\_valores;
  - float val1, val2, somavalores;
  - char sexo;

Slide 30

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Entrada formatada de dados: função scanf

### Sintaxe:

`scanf ("formatos", &var1, &var2,...);`

#### Exemplos:

```
int i, j;
float x;
char c;
scanf("%d", &i);
scanf("%d %f", &j, &x);
scanf("%c", &c);
scanf("%c", &c);
```

%d	1 número inteiro
%i	mesmo que %d
%f	1 número float
%lf	1 número double
%c	1 caractere char
%x	1 número hexadecimal

Slide 31

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Comando (operador) de atribuição

### Sintaxe:

`identificador = expressão;`

- ✓ Atribui o valor da direita à variável da esquerda;
- ✓ O valor pode ser uma constante, uma variável ou uma expressão;

#### Exemplos:

```
x = 4; // significa que a variável X recebe conteúdo 4
y = x + 2;
y = y + 4;
valor = 2.5; //casa decimal indicada pelo ponto
sexo = 'F'; /*atribuição de 1 única caractere para uma
variável char é entre apóstrofes*/
```

Slide 32

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Saída formatada de dados: função printf

### Sintaxe:

`printf ("formatos", var1, var2,...);`

#### Exemplos:

```
int i, j;
float x;
char c;
printf("Resultados: i=%d \n j=%d\n", i, j);
printf("%d", i);
printf("%d, %f", j, x);
printf("%c", c);
```

%c	char
%d	inteiro
%i	inteiro
%f	float ou double com notação comum
%e	float ou double com notação científica
%g	dependendo do conteúdo, usa %f ou %e

\	Função
\b	BackSpace
\n	New Line (mudança de Linha)
\t	Tabulação Horizontal
\v	Tabulação Vertical
\	imprime o próprio caractere \
'	imprime o caractere '
"	imprime o caractere "
?	imprime o caractere ?

Slide 33

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Exercícios

### 1) Faça um algoritmo e programa em C que:

- leia dois valores informados pelo teclado;
- calcule a soma destes valores;
- mostre a soma calculada na tela.

#### Algoritmo Soma2

```
{ Calcula e informa a soma de 2 valores lidos do teclado
entradas: val1, val2 (valores lidos)
saídas: soma
1. início
2. ler val1
3. ler val2
4. soma ← val1 + val2
5. escrever soma
6. fim.
```

Slide 34

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Exercícios

### 1) Faça um algoritmo e programa em C que:

- leia dois valores informados pelo teclado;
- calcule a soma destes valores;
- mostre a soma calculada na tela.

#### Algoritmo Soma2

```
{ Calcula e informa a soma de 2 valores lidos do teclado
entradas: val1, val2 (valores lidos)
saídas: soma
1. início
2. ler val1
3. ler val2
4. soma ← val1 + val2
5. escrever soma
6. fim.
```

#### Linguagem de Programação C:

- entrada de dados: função `scanf()`
- atribuição: operador `=`
- saída de dados: função `printf()`

Slide 35

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Codificação

**Algoritmo Soma2**  
{ Calcula e informa a soma de 2 valores lidos do teclado  
entradas: val1, val2  
saídas: soma  
1. início  
2. ler val1  
3. ler val2  
4. soma ← val1 + val2  
5. escrever soma  
6. fim.

/\*calcula a soma de 2 valores inteiros lidos no teclado\*/

```
#include <stdio.h> // biblioteca com operações de I/O
#include <stdlib.h> // para usar o system(pause)
```

```
int main()
```

```
{
    int val1, val2; // declaração das variáveis informadas
    int soma; // declaração da variável resultante
    printf("Valor 1:\n"); // mensagem impressa na tela
    scanf("%d", &val1); // leitura da informação digitada
```

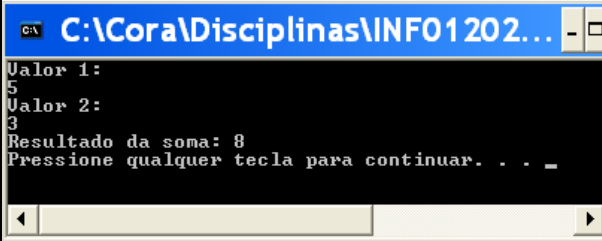
```
    printf("Valor 2:\n");
    scanf("%d", &val2);
    soma = val1 + val2; // armazena resultado em soma
    printf("Resultado da soma: %d\n", soma); // e mostra na tela
    system("pause"); // segura o encerramento do programa
    return 0; // encerra o programa de forma correta
}
```

Slide 36

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Execução



Slide 37

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Biblioteca stdlib

- ✓ Biblioteca que inclui diversas funções de uso geral, incluindo recursos de tela, como cores de fundo e do texto.
- ✓ Possibilita:
  - Limpar o conteúdo da tela: `system("cls");`
  - Controlar as cores da tela: `system("color xy");`  
onde x e y correspondem a 2 números hexadecimais (de 0 à F), significando x - cor do fundo da tela; y - cor do 1º plano.  
Se o símbolo hexadecimal for letra, pode ser informado como maiúscula ou minúscula.  
A correspondência valor-cor é:

0 - preto	4 - vermelho	8 - cinza	C - vermelho claro
1 - azul	5 - roxo	9 - azul claro	D - lilás
2 - verde	6 - amarelo	a - verde claro	e - amarelo claro
3 - verde-água	7 - branco sujo	b - verde-água claro	f - branco brilhante

Slide 38

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Codificação

```
/*calcula a soma de 2 valores inteiros lidos no teclado*/
#include <stdio.h> // biblioteca com operações de I/O
#include <stdlib.h> // para usar o system(pause)

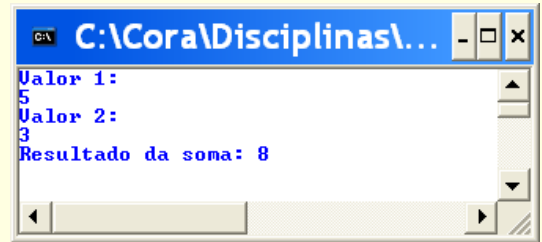
int main()
{
    int val1, val2; // declaração das variáveis informadas
    int soma; // declaração da variável resultante
    system("color f9"); // modifica cor de fundo e do texto
    printf("Valor 1:\n"); // mensagem impressa na tela
    scanf("%d", &val1); // leitura da informação digitada
    printf("Valor 2:\n");
    scanf("%d", &val2);
    soma = val1 + val2; // armazena resultado em soma
    printf("Resultado da soma: %d\n", soma); // e mostra na tela
    system("pause>>null"); // segura o encerramento do programa, sem mensagem
    return 0; // encerra o programa de forma correta
}
```

Slide 39

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática

## Execução



Slide 40

Profa. Cora H.F. Pinto Ribeiro

UFRGS Informática