Programação de sistemas embarcados

Rodrigo Almeida

Universidade Federal de Itajubá

rodrigomax@unifei.edu.br

Conteúdo Programático

Aulas

1. Linguagem C, Hardware utilizado, Arquivos .c e .h, Diretivas de compilação
2. Tipos de dados em linguagem C, Operações aritméticas, Função main(), Rotinas de tempo
3. Operações com bits e dicas para debugar sistemas embarcados
4. Ponteiros e endereços de memória, Arquitetura e configuração do microcontrolador,
5. Programação dos periféricos, acesso às portas do microcontrolador, Barramento de Led's
6. Display de sete segmentos, multiplexação
7. Operação com teclas em arranjo matricial, debounce por software, processo de varredura
8. Display LCD 2x16
9. Protocolos RS232 e SPI. Aplicações de relógio de tempo real (HT1380) e leitura de dados de GPS com protocolo NMEA.
10. Leitura de valores analógicos via conversores AD e utilização de saídas PWM.
11. Utilização de timers para contagem de tempo e reprodução de sons com PWM.
12. Aplicação de interrupções para acesso a dispositivos de hardware. Utilização de watchdog para aumento de confiabilidade.
13. Modelos de arquiteturas de software disponíveis para sistemas embarcados.
14. Programação segura*

Dados Importantes

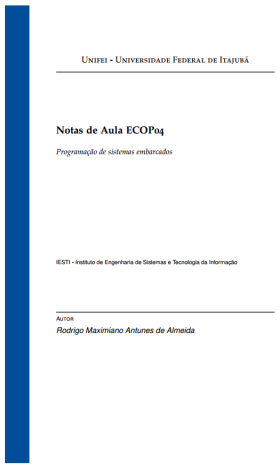
- Teórica:
  - Local ⇒ Sala I.2.1.11
  - Horário ⇒ 4T12
- Laboratório (ELTP14):
  - Local ⇒ Sala LEC II
  - Horário ⇒
    - P1: 5T12
    - P2: 5T34
    - P3: 6T12
    - P4: 6T34
- Datas Importantes *:
  - Prova 01 ⇒ 05/04
  - Prova 02 ⇒ 21/06
  - Sub ⇒ 05/07

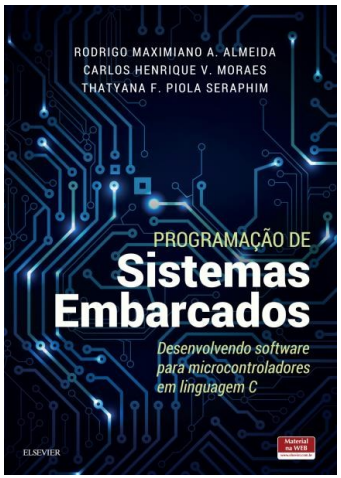
* Datas

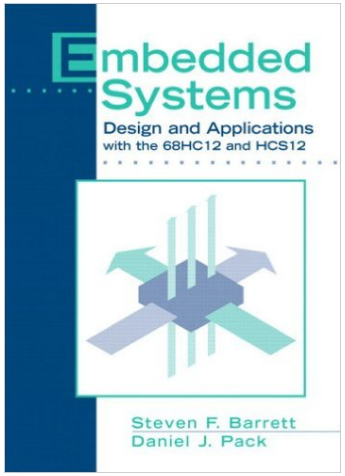
- Todas as datas estão sujeitas a alterações

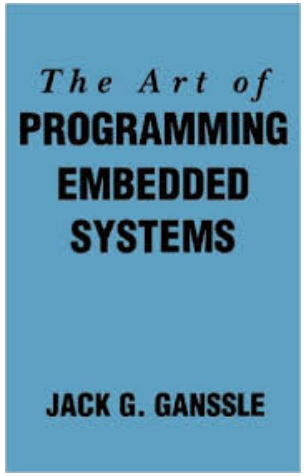
Referências Bibliográficas

- Notas de Aula de Programação de Sistemas Embarcados(https://sites.google.com/site/rmaalmeida/)
- Programação de Sistemas Embarcados. Almeida, R.M.A., Moraes, C.H.V, Seraphim T.F.P., Elsevier 2016
- Embedded systems: design and applications with the 68HC12 and HCS12. BARRETT,Steven F; PACK, Daniel J., Prentice Hall, 2005
- C completo e total. Herbert Schildt. Makron Books. 3ª Edição. 1997.
- The art of Programming Embedded Systems, Ganssle, J. Academic Press, 1991









Sistemas Embarcados - Introdução

Sistemas Embarcados

Hardware Utilizado

Kit de desenvolvimento PIC18F4520

- 1 display LCD 2 linhas por 16 caracteres
- 4 displays de 7 segmentos multiplexados
- 8 leds ligados ao mesmo barramento dos displays
- 16 mini switches organizadas em formato matricial 4x4
- 1 sensor de temperatura LM35C
- 1 resistência de aquecimento ligada a uma saída PWM
- 1 buzzer ligado a uma saída PWM
- 1 motor DC tipo ventilador a uma saída PWM
- 1 canal de comunicação serial padrão RS-232

Ambiente de Programação

Ferramentas a serem utilizadas:

- IDE: MPLAB X - Mista
- Compilador: SDCC 3.4.0 (win32) - GPL
- Linker/Assembler: GPUtils 1.3.0 (win32) - GPL
- Plugin MPLAB: Sdcc Toolchain - GPL

Cuidado

O processo de instalação exige certos cuidados. Referenciem pela apostila.

Linguagem C

Linguagem C

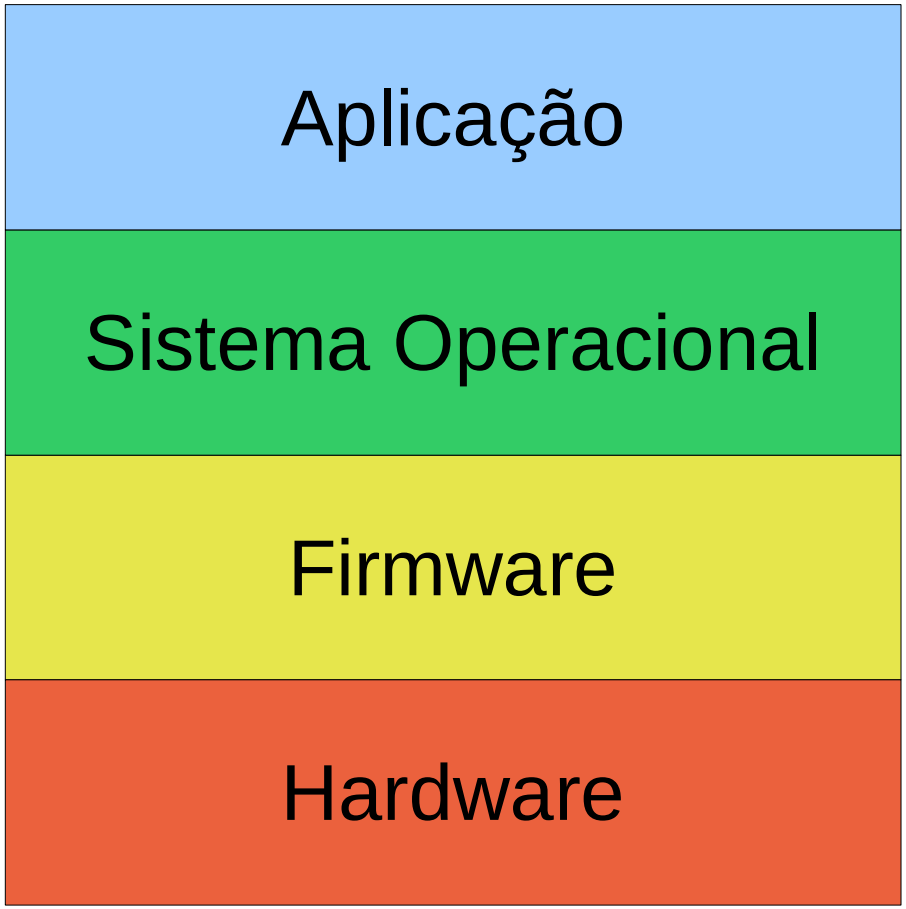
Lógica de Programação

- é necessária para as pessoas que desejam trabalhar com desenvolvimento de programas e sistemas
- permite definir uma sequência natural de atividades com a intenção de atingir um objetivo

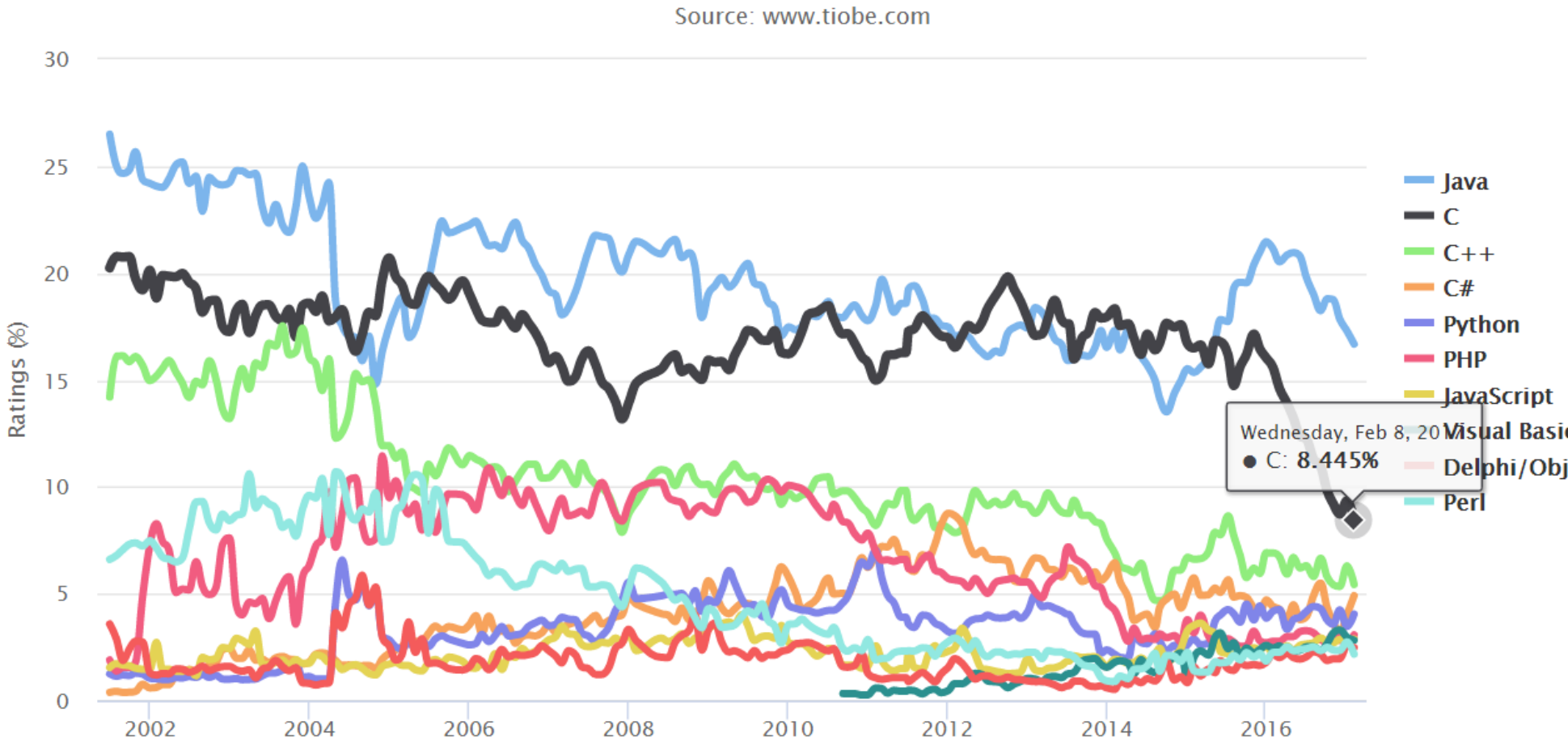
Lógica de Programação

É a técnica de encadear pensamentos em uma sequência lógica para atingir um determinado objetivo.

Linguagem C



Linguagem C



Fonte: http://www.tiobe.com/tiobe-index/

Linguagem C

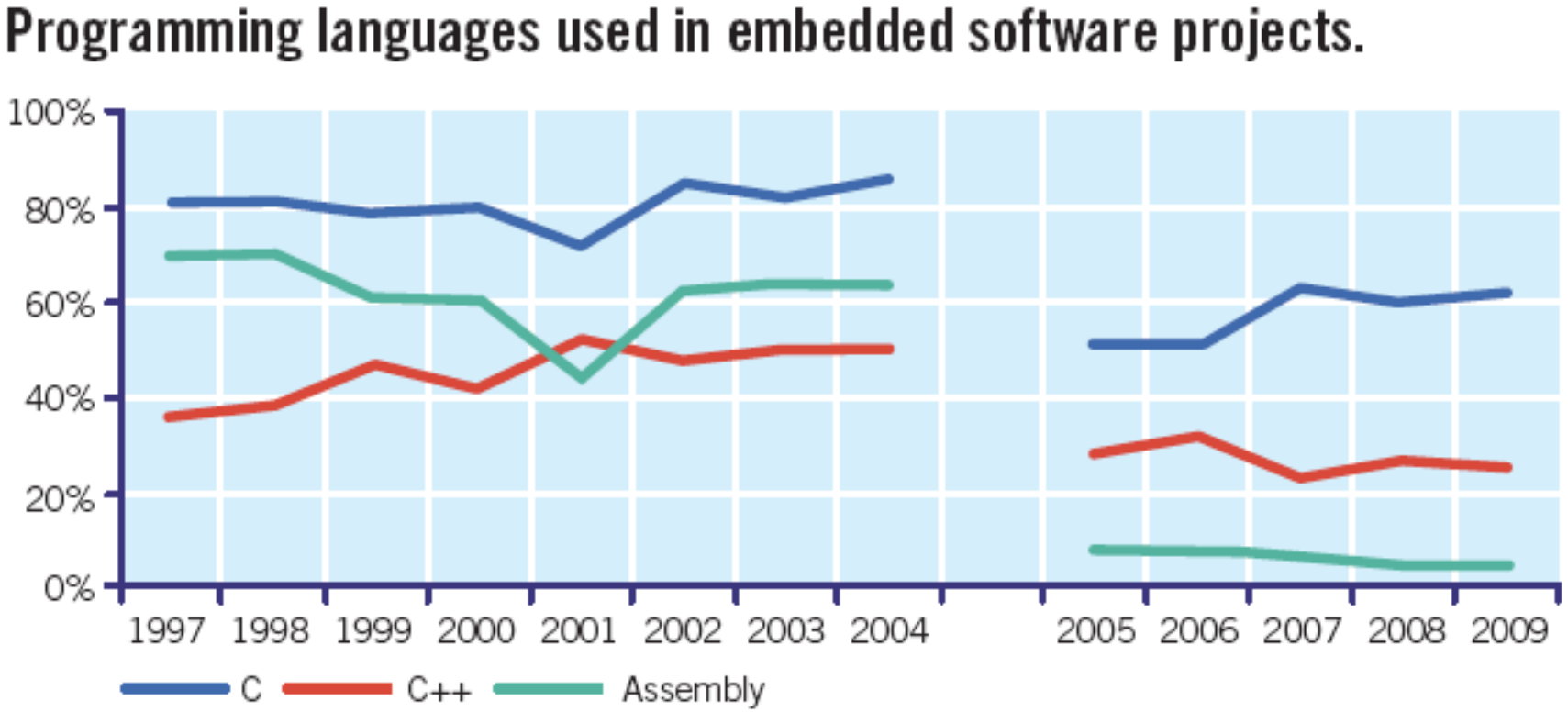


Figure 1

Fonte: http://www.embedded.com/design/218600142

Indentação e padrão de escrita

Atenção

Detalhes do estilo não são importantes. A coerência dentro do projeto sim.

- O estilo adotado é uma variação do "K&R-1TBS".
  - Só muda a chave de abertura da função, que fica na mesma linha
- Todo bloco deve ser aberto e fechado por chaves, mesmo que com apenas uma linha.

Porque escolher esse estilo?

1. Porque sim.
2. Economizar espaço no slide.
3. Porque sim.

Arquivos .c e .h

Arquivos .c e .h

- Arquivo de código (code)
  - terminado com a extensão .c
  - contém a implementação do código
  - é compilado gerando um arquivo .o
- Arquivo de cabeçalho (header)
  - terminado com a extensão .h
  - contém apenas defines e protótipos
  - não é compilado

Arquivos .c e .h

```c
//exemplo de código usando o estilo adotado

//variável usada apenas dentro deste arquivo
static char temp;
//variável que será usada também fora do arquivo
static char valor;
//funções usadas dentro e fora do arquivo
void MudaDigito(char val){
    valor = val;
}
char LerDigito(void){
    return valor;
}
void InicializaDisplays(void){
    //código da função
}
//função usada apenas dentro deste arquivo
void AtualizaDisplay(void){
    //código da função
}
```

Arquivos .c e .h

```
1 #ifndef VAR_H
2     #define VAR_H
3     void MudaDigito(char val);
4     char LerDigito(void);
5     void InicializaDisplays(void);
6 #endif //VAR_H
```

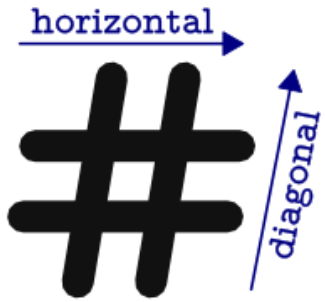
Atenção

- Não existe a função AtualizaDisplay()
- A variável “digito” só pode ser lida ou gravada pelas funções MudaDigito() e LerDigito()
- Cuidado com o overhead de funções

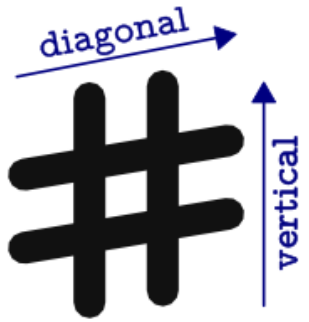
Diretivas de compilação

Diretivas de compilação

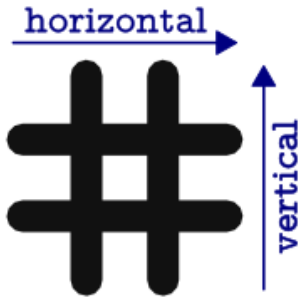
Number sign (pound sign, hash)



Sharp

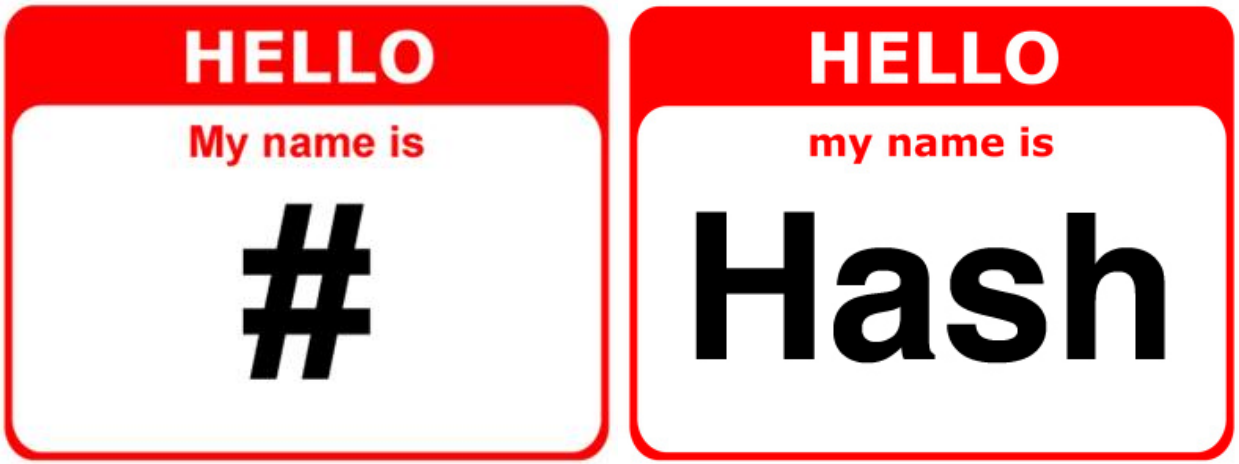


Tic-tac-toe



Diretivas de compilação

- As diretivas de compilação são instruções dadas ao compilador.
- Elas não são executadas.
- As diretivas de compilação começam com um sinal #, conhecido como jogo da velha ou hash.



#define

A diretiva #define é utilizada para que o código fonte seja modificado *antes* de ser compilado.

```
1 #define CONST 15
2 void main(void){
3     printf("%d", CONST * 3);
4 }
5
6 //depois de compilado
7 void main(void){
8     printf("%d", 15 * 3);
9     //é possível: printf("%d", 45);
10 }
```

#define

Atenção!

A diretiva #define NÃO cria uma constante.

Compilação condicional

#define

```
1 //função para apresentar dispositivo selecionado
2
3 void MostraSaidaPadrao(void){
4
5 #ifdef PADRAO Serial
6     char * msg = "SERIAL";
7 #else
8     char * msg = "LCD";
9 #endif
10    printf(msg);
11 }
```

#define

Opções de uso com o #define	Resultado na Tela
``1 #include <stdio.h>``` ```2 #define PADRAO Serial``` ```3``` ```4 void main(void) {``` ```5 MostraSaidaPadrao();``` ```6 }```	SERIAL
``1 #include <stdio.h>``` ```2 #define PADRAO LCD``` ```3``` ```4 void main(void){``` ```5 MostraSaidaPadrao();``` ```6 }```	LCD

#ifdef, #ifndef, #else e #endif

```
1  void ImprimirTemp(char valor){
2      #ifdef LCD
3          Imprime_LCD(valor);
4      #else
5          if (valor > 30){
6              led = 1;
7          }else{
8              led = 0;
9          }
10     #endif //LCD
11 }
```

Funcionamento

No momento da compilação o pré-compilador irá verificar se a "tag" LCD foi definida em algum lugar. Em caso positivo o pré-compilador irá deixar tudo que estiver entre o #ifdef e o #else e retirará tudo que está entre o #else e o #endif.

Referência circular

Referência circular

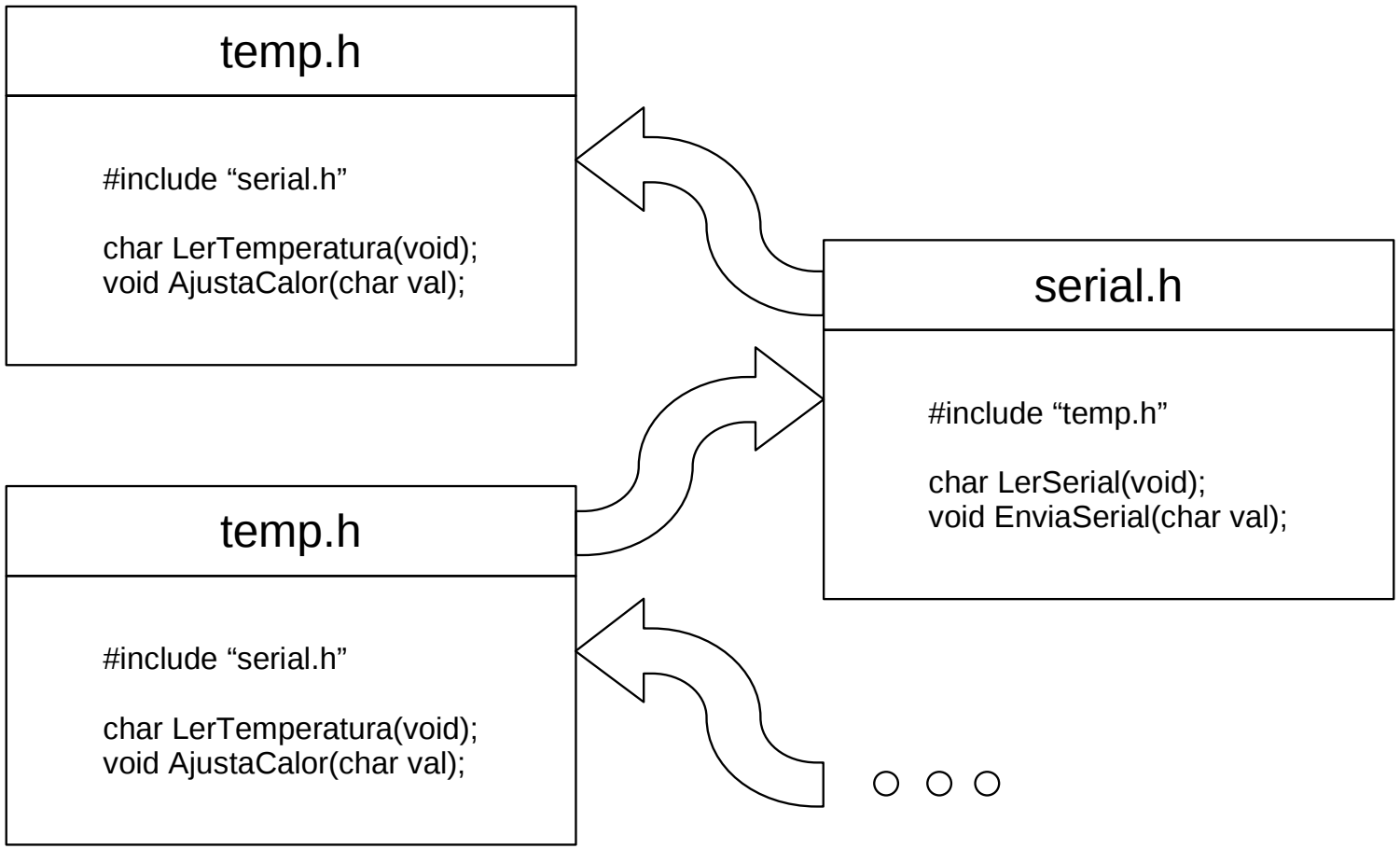
A função LerTemperatura() faz um teste: se o valor for maior que um patamar chama a função EnviaSerial() com o código 0x30.

```
1  #include "serial.h"
2  char LerTemperatura(void);
3  void AjustaCalor(char val);
```

A função LerSerial() recebe um valor e repassa para a função AjustaCalor().

```
1  #include "temp.h"
2  char LerSerial(void);
3  void EnviaSerial(char val);
```

Referência circular



Referência circular

Solução: criar uma estrutura de controle para pré compilação.

```
1  #ifndef TAG_CONTROLE
2      #define TAG_CONTROLE
3      //todo o conteúdo do arquivo vem aqui.
4
5  #endif //TAG_CONTROLE
```

Referência circular

Solução: criar uma estrutura de controle para pré compilação.