6. MACROS DO PRÉ-PROCESSADOR

É possível incluir diversas instruções do compilador no código-fonte de um programa C. Elas são chamadas de diretivas do pré-processador e, embora não sejam realmente parte da linguagem de programação C, expandem o escopo do ambiente de programação em C. Algumas diretivas do pré-processador definidas pelo padrão C ANSI são:

- *#define*
- *#include*
- *#undef*
- *#if*
- *#else*
- *#elif*
- *# endif*
- *#ifdef*
- *#ifndef*

Pode-se observar que todas as diretivas do pré-processador começam com #. Além disso, cada diretiva deve estar na sua própria linha.

#define, define um identificador e uma *string* que o substituirá toda vez que for encontrado no arquivo-fonte. O padrão C ANSI refere-se ao identificador como um "nome de macro" e ao processo de substituição como "substituição de macro". Forma geral:

#define <nome_macro> <string>

sem ponto e vírgula no final.

Exemplos:

```
#define VERDADEIRO  1
#define FALSO       0
#define XYZ         Isso eh um teste
#define ABS(a)      (a)<0 ? –(a) : (a)
```

#include, instrui o compilador a ler outro arquivo-fonte adicionado àquele que contém a diretiva *#include*. O nome do arquivo adicional deve estar entre aspas ou símbolo de maior e menor. Exemplo:

```
#include "teste.h"      // procura no diretório de trabalho atual
#include <stdio.h>      // procura no diretório padrão especificado
```

Arquivos de inclusão podem ter diretivas *#include* neles. Isso é denominado "*includes* aninhados". O número de níveis de aninhamento varia entre compiladores, mas o padrão C ANSI estipula que pelo menos oito níveis de inclusões aninhadas estão disponíveis.

Há diversas diretivas que permitem uma compilação seletiva de porções de código-fonte do programa. Esse processo é chamado de compilação condicional e é utilizado amplamente em *software houses* comerciais que fornecem e mantêm muitas versões de um programa.

As diretivas de compilação condicional mais usadas são ***#if***, ***#else***, ***#elif*** **e** ***#endif***. Estas diretivas permitem que sejam incluídas condicionalmente partes do código baseado no resultado de uma expressão constante. A forma geral do *#if* é:

```
#if <expressão_constante>
        <seqüência_de_comandos>
#endif
```

Se a expressão constante que segue o *#if* for verdadeira, o código que está entre ele e o *#endif* é compilado, caso contrário, será saltado. A diretiva *#endif* marca o final de um bloco *#if*. A diretiva *#else* opera de forma semelhante ao *else* que é parte da linguagem C: estabelece uma alternativa se *#if* for falso.

```
// Exemplo simples de #if
#include <stdio.h>
#define MAX 100
main()
{
  #if MAX > 99
        printf("Compilado para matriz maior que 99 \n");
  #else
        printf("Compilado para matriz pequena \n");
  #endif
}
```

A diretiva *#elif* significa "*else if*" e estabelece uma seqüência *if-else-if* para múltiplas opções de compilação. *#elif* é seguido por uma expressão constante. Se a expressão é verdadeira, esse bloco de código é compilado e nenhuma outra expressão *#elif* é testada. Caso contrário, o próximo bloco na série é verificado. O padrão C ANSI define que *#if* e *#elif* podem ser aninhados até pelo menos oito níveis. A forma geral para *#elif* é

```
#if <expressão>
        <seqüência_de_comandos>
#elif <expressão1>
        <seqüência_de_comandos>
#elif <expressão2>
        <seqüência_de_comandos>
#elif <expressão3>
        <seqüência_de_comandos>
:
#elif <expressãoN>
        <seqüência_de_comandos>
#endif
```

Exemplo:

```
#if MAX>100
        #if SERIAL_VERSION
                int port=198;
        #elif
                int port=200;
        #endif
#else
        char out_buffer[100];
#endif
```

Um outro método de compilação condicional usa as diretivas ***#ifdef*** e ***#ifndef***, que significam "se definido" e "se não definido", respectivamente. A forma geral de *#ifdef* é

```
#ifdef <nome_da_macro>
        <seqüência_de_comandos>
#endif
```

Se o <nome_da_macro> tiver sido definido anteriormente em um bloco de código *#define*, o bloco de código será compilado.

A forma geral de *#ifndef* é

```
#ifndef <nome_da_macro>
        <seqüência_de_comandos>
#endif
```

Se o <nome_da_macro> estiver atualmente indefinido, o bloco de código será compilado.

A diretiva ***#undef*** remove a definição anterior do nome de macro que a segue. *#undef* é usado principalmente para permitir que nomes de macros sejam localizados apenas naquelas seções de código que precisam deles. A forma geral de *#undef* é [SCH 96]:
#undef <nome_da_macro>

Por exemplo:

```
#define LEN      100
#define WIDTH 100

char array[LEN][WIDTH];

#undef LEN
#undef WIDTH
// Nesse ponto LEN e WIDTH não estão definidos
```