FUNÇÕES MAIS UTILIZADAS DA LINGUAGEM C PADRÃO CCS

Função	Descrição	Exemplo
output_high()	Ativa um determinado pino do microcontrolador	output_high(PIN_D0); output_high(PIN_C2);
output_low()	Desativa um determinado pino do microcontrolador	output_low(PIN_D0); output_low(PIN_C2);

input()	Busca o estado de um pino	if (input(PIN_A1))
		{
		output_high(PIN_D0);
		x = input(PIN_A4);
		}

output_a()	Envia um byte para o PORT A	output_a(VAR1); // envia VAR1 para PORTA
output_b()	Envia um byte para o PORT B	output_b(0xff); // liga todos bits de PORTB
output_c()	Envia um byte para o PORT C	output_c(VAR1); // envia VAR1 para PORTC
output_d()	Envia um byte para o PORT D	output_d(0x00); // desliga todos os bits de PORTD
output_e()	Envia um byte para o PORT E	output_e(VAR1); // envia VAR1 para PORTE
input_a()	Busca um byte do PORT A	int VAR1; VAR1 = input_a();
input_b()	Busca um byte do PORT B	int VAR1; VAR1 = input_b();
input_c()	Busca um byte do PORT C	int VAR1; VAR1 = input_c();
input_d()	Busca um byte do PORT D	int VAR1; VAR1 = input_d();
input_e()	Busca um byte do PORT E	int VAR1; VAR1 = input_e();

lcd_init()	Inicializa o LCD.	#use delay (clock=4000000)
		#define use_portb_lcd true
		#include <lcd.c>
		...
		void main()
		{
		...
		lcd_init();
		...
		}

lcd_putc()	Envia uma string (seqüência de caracteres) para o LCD	lcd_putc(“ \f TESTE”);
delay_ms()	Causa um atraso em milésimos de segundo	delay_ms(VAR1); delay_ms(100);
delay_us()	Causa um atraso em milionésimos de segundo	delay_us(10); delay_us(VAR1);
printf()	Cria uma saída formatada, geralmente utilizada para exibir dados das variáveis no LCD	float VAR1; int VAR2; long VAR3; printf(lcd_putc, “\f TESTE %f”, VAR1); printf(lcd_putc,”\fTESTE\n %lu %f”, VAR3, VAR1);

DECLARAÇÃO DE VARIÁVEIS

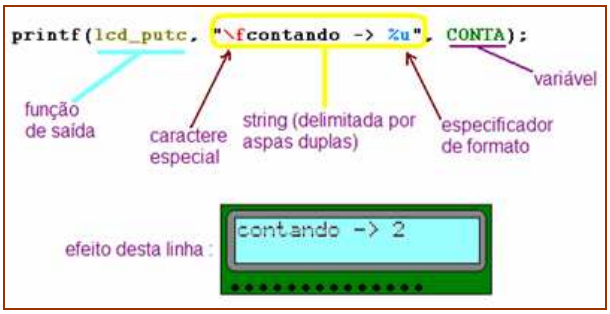
Tipos de dados	Tamanho em bits	Descrição	exemplo
int	**8**	variáveis inteiras sem sinal, capacidade de 0 a 255	int X, Y=1, VAR1=0;
signed int	**8**	variáveis inteiras com sinal, capacidade de –128 a 127	signed int VAR3 = -10;
long	**16**	variáveis inteiras sem sinal, capacidade de 0 a 65535	long A, B = 0, VAR2;
signed long	**16**	variáveis inteiras com sinal, capac. de –32768 a 32767	signed long GRAU=0;
float	**32**	variáveis reais com sinal. Representação aproximada.	float VAR4, VAR5;
short ou **boolean**	**1**	variáveis lógicas, de um bit, podendo valer 0 ou 1	boolean FLAG1=0, SENSOR; short CHAVE;
int32	**32**	variáveis inteiras sem sinal, podendo valer de 0 a 4294967295	int32 CONTADOR;
signed int32	**32**	variáveis inteiras com sinal, podendo armazenar valores de -2147483648 a 2147483647	signed int32 VAR10;
char	**8**	variáveis que armazenam caracteres em forma de bytes.	char C = ‘a’, LETRA = ‘ ‘, H;
OBS: Os tipos de dados int e long podem assumir outras configurações em outros tipos de processadores. Ex: em um processador de 16 bits, o tipo de dado INT assume 65536 possibilidades diferentes.			

OPERADORES

+ (adição)	**== (igual)**	**<= (menor igual)**	**| (ou binário)**	**= (atribuição)**
- (subtração)	**!= (diferente)**	**&& (e)**	**>> (rotação binária para direita)**	**++ (incremento)**
*** (multiplicação)**	**> (maior que)**	**|| (ou)**	**<< (rotação binária para esquerda)**	**-- (decremento)**
/ (divisão)	**< (menor que)**	**! (negação)**	**~ (negação binária)**	
% (resto divisão)	**>= (maior igual)**	**& (e binário)**	**^ (ou exclusivo)**	

ESPECIFICADORES DE FORMATO PARA O PRINTF

Especificador de formato é o símbolo que indica, em uma string utilizada no comando PRINTF, a posição e o formato onde será impresso o valor da variável.



Caracter especial	Função
\f	Limpar display
\n	Pular linha
\\	Barra invertida
\0	Null

Especificador	Tipo
%u	**int** ou **short**
%i ou %d	**signed int**

%lu	**long** ou **int32**
%li ou %ld	**signed long**
%X	**int** em hexadec.
%f	**Float**
%c	**Caractere**
%s	**String**
%e	**float** (not.cientf.)
%lx	**long hex**
%%	**símbolo %**
%3u	**int (3 casas)**
%03u	**int (3 digitos c/ zeros à esq.)**
%1.2f	**float (2 casas dec.)**

Exemplos de uso :

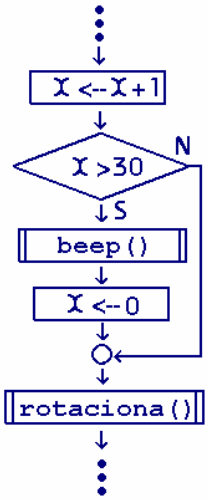
```
int vlr;
....
printf(lcd_putc,"\f Valor: %u", vlr);
```

```
float a, b; int c;
...
printf(lcd_putc,"\fCont: %u \nX:%1.2f Y:%1.2f", c, a, b);
```

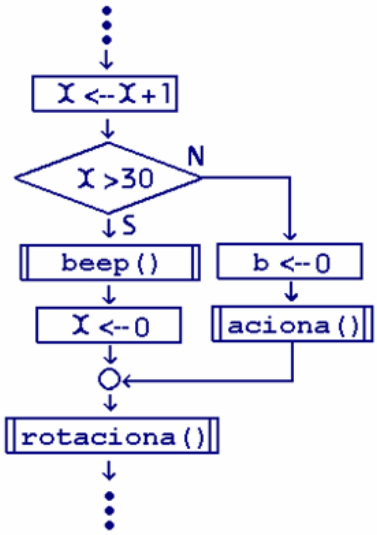
PRINCIPAIS ESTRUTURAS DA LINGUAGEM C

Estruturas de decisão :

```
if (condição)
  {
  instruções;
  }
```



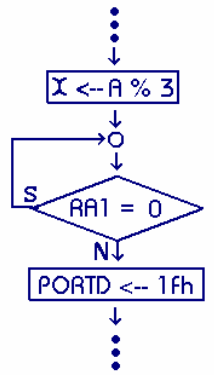

```
...
x ++ ;
if (x > 30)
    {
    beep();
    x = 0;
    }
rotaciona();
...
```




```
...
x ++ ;
if (x > 30)
    {
    beep();
    x = 0;
    }
else
    {
    b = 0;
    aciona();
    }
rotaciona();
...
```

```
if (condição)
  {
  intruções;
  }
else
  {
  instruções;
  }
```

Estruturas de repetição :




```
...
```

while(condição);

```
x = a % 3;
while(input(PIN_A1)==0);
PORTD = 0x1F;
...
```

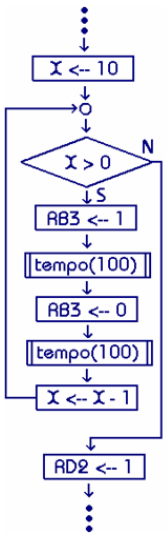
While vazio. Muito útil quando se deseja **"reter"** o programa até que uma condição ocorra.
A condição entre os parênteses indica a condição na qual o sistema ficará retido. Lembre que WHILE significa ENQUANTO. O PONTO e VÍRGULA identifica o laço vazio.

---

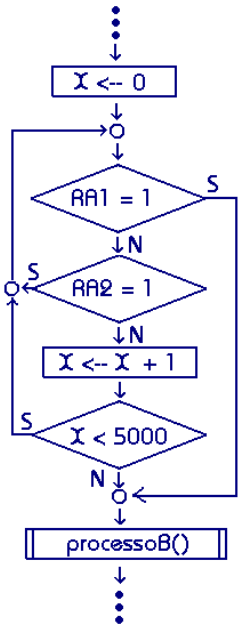
while (condição)

```
{
....;
....;
}
```

// enquanto a condição for
// verdadeira, serão executadas
// as instruções entre as chaves
// ou a instrução seguinte no caso
// de não existirem as chaves.




```
...
x =10;
while (x > 0){
                output_high(PIN_B3);
                delay_ms(100);
                output_low(PIN_B3);
                delay_ms(100);
                x --;
                }
output_high(PIN_D2);
...
```




```
...
x = 0;
do{
  if (input(PIN_A1)) break;
  if (input(PIN_A2)) continue;
  x ++;
  } while(x < 5000);
processoB();
```

```
...      ou ainda ...
x = 0;
while(!input(PIN_A1))
  {
  if (input(PIN_A2)) continue;
  x ++;
  if (x >= 500) break;
  }
processoB();
...
```

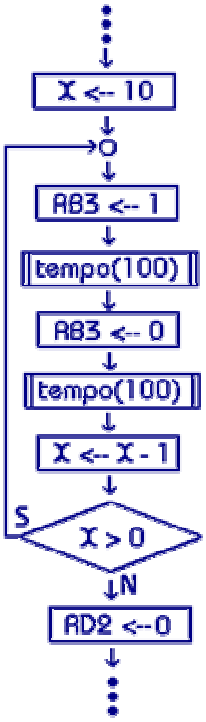
while (condição)

```
{
....;
if (condição)
  continue;   // volta p/ inicio do while
....;
if (condição)
  {
  ...;
  break;      // interrompe o while
  }
}
```

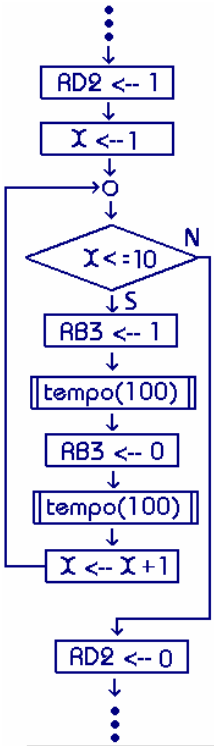
```
do {
   ...;
   } while (condição);
```

do / while singifica faça / enquanto.

Ao contrário do while, o do/while permite que o bloco seja executado ao menos uma vez.




```
...
x =10;
do {
   output_high(PIN_B3);
   delay_ms(100);
   output_low(PIN_B3);
   delay_ms(100);
   x --;
   } while (x > 0);
output_low(PIN_D2);
...
```



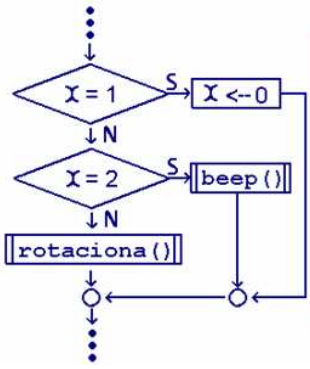

```
...
output_high(PIN_D2);
for (x = 1; x <= 10; x ++)
       {
       output_high(PIN_B3);
       delay_ms(100);
       output_low(PIN_B3);
       delay_ms(100);
       }
output_low(PIN_D2);
...
```

```
for (inicializ ; condição ; incrm)
   ...;
   }
```

O for geralmente é utilizado para se repetir um determinado bloco baseado na contagem (incremetal ou decremental) de uma variável de tipo inteiro (int, long, int32, etc...)

Inicializ : comando a ser executado antes da primeira interação do laço.
Condição : situação para continuar o laço.
Incrm: incremento ou decremento da variável de controle.

```
switch (variável)
  {
  case   VALOR1 : instruções;
                  break;
  case   VALOR2 : instruções;
                  break;
  default       : instruções;
  }
```




```
...
switch(x)
  {
  case 1 : x = 0;
           break;
  case 2 : beep();
           break;
  default : rotaciona();
  }
...
```

DESCRIÇÃO DAS PRINCIPAIS FUNÇÕES EMBUTIDAS NO COMPILADOR PCW (3.4 ou sup.)

FUNÇÕES DE I/O VIA RS232	
getc() ou getchar();	Busca caractere via porta serial
putc() ou putchar();	Envia caractere via porta serial
fgetc();	Busca caractere em um dispositivo
gets();	Envia uma string pela porta serial
puts();	Envia seqüência de caracteres via porta serial
fgets();	Busca uma seqüência de caracteres via porta serial
fputc();	Envia um caractere a um dispositivo
fputs();	Envia uma seqüência de caracteres a um dispositivo
printf();	Imprime uma seqüência formatada de texto em um dispositivo
kbhit();	Verifica se há caractere disponível na entrada serial
fprintf();	Saída formatada para um dispositivo
set_uart_speed();	Determina velocidade da porta serial
perror();	Imprime uma mensagem de erro no dispositivo padrão de saída
assert();	Usado para depuração
SPI (I/O 2 fios)	
setup_spi();	Inicializa SPI
spi_read();	Lê da interface serial
spi_write();	Grava na interface serial
spi_data_is_in();	Retorna "verdadeiro" se existem dados recebidos pela SPI
ENTRADA E SAÍDA DIGITAL	
output_low();	Desativa uma saída
output_high();	Ativa uma saída
output_float();	Habilita o terceiro estado do pino (coletor aberto)
output_bit();	Envia o valor de um bit para um pino
input();	Lê o valor de um pino
output_a();	Envia um byte para o PORTA
output_b();	Envia um byte para o PORTB
output_c();	Envia um byte para o PORTC
output_d();	Envia um byte para o PORTD
output_e();	Envia um byte para o PORTE
input_a();	Lê um byte do PORTA
input_b();	Lê um byte do PORTB
input_c();	Lê um byte do PORTC
input_d();	Lê um byte do PORTD
input_e();	Lê um byte do PORTE
port_b_pullups();	Ativa os PULL-Ups de entrada do portb
set_tris_a();	Define a direção para os pinos do PORTA
set_tris_b();	Define a direção para os pinos do PORTB
set_tris_c();	Define a direção para os pinos do PORTC
set_tris_d();	Define a direção para os pinos do PORTD
set_tris_e();	Define a direção para os pinos do PORTE
PWM	
setup_ccpX();	Define o modo de operação dos pinos de PWM
set_pwmX_duty();	Determina o valor do PWM, de 0 a 1023
INTERFACE DE PORTA PARALELA ESCRAVA (PORTD)	
setup_psp()	Ativa a porta paralela escrava
psp_input_full()	Verifica o funcionamento do recurso de porta paralela escrava
psp_output_full()	Verifica o funcionamento do recurso de porta paralela escrava
psp_overflow()	Verifica o funcionamento do recurso de porta paralela escrava
I2C	
i2c_start()	Inicia interface I2C
i2c_stop()	Para interface I2C
i2c_read()	Lê byte da interface I2C
i2c_write()	Grava byte na interface I2C
i2c_poll()	Verifica buffer da interface
PROCESSOR	
sleep()	Entra em modo SLEEP
reset_cpu()	Reinicia (reseta) o microcontrolador
restart_cause()	Retorna a causa do último reset
disable_interrupts()	Desativa interrupções
ext_int_edge()	Configura comportamento da interrupção por borda
read_bank()	Lê o valor de um registrador em um determinado banco
write_bank()	Grava uma informação em uma posição de memória

label_address()	Endereço ROM representado por um rótulo
goto_address()	Desvia a execução para um endereço ROM
getenv()	Retorna o valor de uma variável de ambiente
BIT/BYTE	
shift_right()	Rola dados para direita.
shift_left()	Rola dados para esquerda.
rotate_right()	Rotaciona dados para direita.
rotate_left()	Rotaciona dados para esquerda.
bit_clear()	Limpa um bit de uma variável
bit_set()	Ativa um bit de uma variável
bit_test()	Testa um bit de uma variável
swap()	Troca os nibbles de uma variável de 8 bits
make8()	Extrai um byte de uma variável
make16()	Extrai uma Word de uma variável
make32()	Extrai um valor de 32 bits de uma variável
ANALOG	
setup_comparator()	Configura o comparador
setup_adc_ports()	Configura portas usadas pelo conversor AD
setup_adc()	Configura o AD
set_adc_channel()	Determina o canal a ser utilizado
read_adc()	Lê valor do canal AD ativado
MATEMÁTICAS	
abs()	Retorna valor absoluto
acos()	Arco cosseno
asin()	Arco seno
atan()	Arco tangente
ceil()	Arredonda acima um float para número inteiro.
cos()	Cosseno
exp()	Calcula função E de um número.
floor()	Arredodna abaixo um float para número inteiro.
labs()	Calcula o valor absoluto de um long
sinh()	Seno hiperbólico
log()	Logaritmo natural
log10()	Logaritmo base 10
pow()	Potência
sin()	Seno
cosh()	Cosseno hiperbólico
tanh()	Tangente hiperbólica
fabs()	Valor absoluto para um float
fmod()	Resto da divisão de ponto flutuante
atan2()	Arco tangente
frexp()	Quebra um float
ldexp()	
modf()	Quebra um float em inteiro e decimal
sqrt()	Raiz quadrada
tan()	Tangente
div()	Divisão retornando quociente e resto
ldiv()	Divisão de um long retornando quociente e resto
VOLTAGE REF	
setup_vref()	Estabelece tensão de refer. dos comparadores
STANDARD	
atoi()	Transforma ASCII em int
atoi32()	Transforma ASCII em int32
atol()	Transforma ASCII em long
atof()	Transforma ASCII em float
tolower()	Transforma letras maiúsculas em minúsculas
toupper()	Transforma letras minúsculas em maiúsculas
isalnum()	Verifica se uma string é numérica
isalpha()	Verifica se uma string é alfabética
isamoung()	Verifica se um caractere pertence a uma string
isdigit()	Verifica se é número
islower()	Verifica se é letra minúscula
isspace()	Verifica se é espaço
isupper()	Verifica se é letra maiúscula
isxdigit()	Verifica se é dígito hexadecimal
strlen()	Retorna comprimento de uma string
strcpy()	Copia uma string
strncpy()	Copia com limite de caracteres
strcmp()	Compara strings
stricmp()	Compara strings ignorando maiúscula/minúscula
strncmp()	Compara com limite de caracteres
strcat()	Concatena strings
strstr()	Procura por uma string dentro de outra
strchr()	Procura caracteres em uma string
strrchr()	Procura caracteres em uma string, de traz para frente.
strtok()	Aponta para próximo caractere após separador em uma string
strspn()	Conta caracteres iniciais em strings
strcspn()	Conta caracteres iniciais em strings
strpbrk()	Procura primeiro caracter comum em strings
strlwr()	Converte uma string em minúsculas
sprintf()	Imprime (printf) em uma string
isgraph()	Testa se é caractere gráfico
iscntrl()	Testa se é caractere de controle

isprint()	Testa se é imprimível
strtod()	Extrai um float de uma string
strtol()	Extrai um inteiro de uma string
strtoul()	Idem
strncat()	Concatena com limite de caracteres
strcoll()	Compara caracteres em uma string
strxfrm()	Compara caracteres em uma string

TIMERS

setup_timer_x()	Configura funcionamento dos TIMERS
set_timer_x()	Inicializa o TIMER
get_timer_x()	Busca valor de um TIMER
setup_counters()	Configura contador
setup_wdt()	Configura o Watch Dog Timer
restart_wdt()	Reinicia o Watch Dog Timer

DELAYS

delay_us()	Causa um atraso (tempo) em milhonésimos de segundos
delay_ms()	Causa um atraso (tempo) em milésimos de segundos
delay_cycles()	Causa um atraso em número de ciclos (clock / 4)

STANDARD C

memset()	Copiar um conjunto de dados na memória
memcpy()	Copiar um conjunto de dados na memória
offsetof()	Retorna valor de deslocamento de dados na memória
offsetofbit()	Retorna valor de deslocamento de dados na memóira

malloc()	Aloca dinamicamente uma área de memória
calloc()	Aloca dinamicamente uma área de memória
free()	Libera memória alocada por malloc ou calloc
realloc()	Realoca memória
memmove()	Copiar um conjunto de dados na memória
memcmp()	
memchr()	

EEPROM

read_eeprom()	Ler um byte da EEPROM
write_eeprom()	Gravar um byte na EEPROM
read_program_eeprom()	Ler área da ROM de programa
write_program_eeprom()	Gravar algo na área de ROM de programa (flash)
read_calibration()	Função exclusiva para PIC14000 – lê dado da memória de calibração
write_program_memory()	Grava uma seqüência de bytes na memória de programa
read_program_memory()	Lê uma seqüência de bytes da memória de programa
write_external_memory()	Grava em uma memória externa. Pode depender de implementação.
erase_program_memory()	Apaga uma área da memória flashROM
setup_external_memory()	Configura forma de utilização de memória externa.

STANDARD C SPECIAL

rand()	Geração de número aleatório
srand()	Define o valor máximo para geração de número aleatório

```
#include <16f877.h>
#device ADC=10             // para conversor analógico-digital de 10 bits
#use delay (clock=4000000) // mudar conforme velocidade do cristal usado
#define use_portb_lcd true // força biblioteca de lcd a usar o PORTB
#include <lcd.c>           // inclui biblioteca do lcd
int VG1;                   // declara variável global. Vale em todo o programa

void rotina()              // declara uma subrotina (função) chamada "rotina"
   {
   long X;                 // variável local. Só vale dentro da função "rotina"
   while(!input(PIN_E2))   // enquanto não houver sinal em E2 ...
      {
      if (input(PIN_A1) || X == 50)   // se A1 = 1 ou X = 50 …
         {
         output_high(PIN_B3);            // liga B3
         X = 0;                          // X <- 0
         }
      else                            // senão
         {
         output_low(PIN_B3);             // desliga B3
         }
      X ++;                           // incrementa X
      delay_ms(100);                  // tempo de 100 milisegundos
      }                       // fim do while
   }                          // fim da rotina()

void main() // programa principal
   {
   int var;      // variável local. Só pode ser usado dentro da função MAIN
   lcd_init();   // inicializa o display
   while(1)            // laço infinito, ou "forever loop"
       {
       if (input(PIN_A1))     // se A1 = 1 …
          {
          rotina();               // executa "rotina" anteriormente declarada
          }
       if (input(PIN_A2))     // se A2 = 1 …
          {
          for (var=0; var<10; var ++)   // repete 10 vezes (usando var p/ contar)
            {
            output_high(PIN_D0);        // liga D0
            delay_ms(100);              // tempo 100 ms
            output_low(PIN_D0);         // desliga D0
            delay_ms(900);              // tempo 900 ms
            }
          }
       }
   }
```

Principais erros do compilador CCS

A #DEVICE required before this line	Falta a definição do microcontrolador utilizado através da diretiva DEVICE. Esta diretiva já está declarada nos arquivos de include relativos ao microcontrolador utilizado.
A numeric expression must appear here	O ponto do programa pede um código executável. Verifique se não está sendo feita a declaração de uma variável após uma linha de código executável. Se isso ocorrer, inverta a ordem, fixando a declaração de todas as variáveis no início da função.
Bad expression syntax	Mensagem de erro genérica para alerta de erro de sintaxe.
Cannot change device type this far into the code	Após uma linha de geração de código não é mais permitida a definição do dispositivo (diretiva DEVICE). Veja se não há linhas de programa ates do primeiro include. Este erro também é comum caso um arquivo tipo HEADER (como o 16f877.h) tenha seu conteúdo alterado inadequadamente. Tente mudar o arquivo de include ou declarar a linha de DEVICE em um ponto mais próximo do início do arquivo de programa.
Constant out of the valid range	O valor da constante está além da capacidade de seu destino. Por exemplo, está se atribuindo o valor 500 a uma variável do tipo INT de 8 bits, que só suporta dados até 255.
Duplicate case value	Há uma dupla ocorrência de um mesmo CASE em uma estrutura SWITCH.
Duplicate DEFAULT statements	Foi encontrada uma segunda ocorrência da cláusula DEFAULT dentro de um mesmo SWITCH.
Duplicate #define	#define duplicado ou já declarado
Duplicate function	Dupla implementação de uma função. Cada função deve ter um nome único.
ELSE with no corresponding IF	Encontrado um ELSE sem um IF correspondente. Veja se não foi adicionado acidentalmente um ponto e vírgula na linha do IF deste ELSE. Lembre que um IF não pode ter ponto e vírgula.
Expect ;	Verifique a falta de um ponto e vírgula. Geralmente o erro é apontado após a linha que está com o erro de pontuação.
Expect }	Verifique a falta de um fechar chaves. Lembre que cada abrir chaves deve possuir um fechar chaves correspondente. Recomenda-se o correto alinhamento para facilitar visualização.
Expect comma	Verifique a falta de uma vírgula
Expect WHILE	Verifique a falta de um WHILE após o uso de um DO {}
Expecting :	Verifique a falta de dois pontos
Expecting =	Verifique a falta de um sinal de atribuição
Expecting a (	Verifique a falta de um abrir parênteses
Expecting a , or)	Verifique a falta de uma vírgula ou fechar parênteses
Expecting a , or }	Verifique a falta de uma vírgula ou de um fechar chaves
Expecting a .	Verifique a falta de um ponto
Expecting a ; or ,	Verifique a falta de um ponto e vírgula ou de uma vírgula
Expecting a ; or {	Verifique a falta de um ponto e vírgula ou de um fechar chaves
Expecting a close paren	Verifique a falta de um fechar parênteses
Expecting a declaration	Esperando uma declaração. Geralmente pode ocorrer por algum erro de pontuação anterior.
Expecting a variable	Esperando uma variável.
Expecting a]	Esperando um fechar colchetes
Expecting a {	Esperando um abrir chaves
Expecting an =	Esperando um sinal de atribuição
File cannot be opened	Verifique o nome e o caminho do arquivo. (se o include estiver certo)
Filename must start with " or <	Verifique a sintaxe do INCLUDE.
Identifier is already used in this scope	É um aviso que você está tentando usar um nome para a função ou variável que já existe ou já foi utilizado neste programa.
No MAIN() function found	Verifique se você criou a função principal: void main()
Not enough RAM for all variables	Significa que existem muitas variáveis para pouca memória RAM. Use tipos de dados mais econômicos, menos variáveis globais e mais variáveis locais.
Out of ROM, A segment or the program is too large	O programa ficou muito grande para a memória ROM. Isso pode ser gerado também por uma string muito longa, ou pelo esquecimento de um fechar aspas. Caso realmente esteja com problemas de falta de ROM, use a diretiva #separate antes de algumas subrotinas para usar páginas de ROM diferentes em cada etapa do programa.
Printf format type is invalid	Veja se você está usando o especificador de formato correto. Para uma variável do tipo SHORT ou INT, use %u ou %i (%i para valores com sinal). Para variáveis do tipo LONG ou INT32, use %lu ou %li. Para float, use %f... para outros tipos ou especificadores, consulte a tabela.
Printf format (%) invalid	Você usou um especificador de formato inválido. (leia o item anterior desta tabela)
Printf variable count (%) does not match actual count	No printf devem existir menos especificadores de formado do que as variáveis que devem ser exibidas.
Recursion not permitted	Devido a escassez de recursos de pilha, o compilador impede o uso de recursão para o PIC Verifique se você não está chamando a função de dentro dela mesma, ou gerando ciclos de chamadas recorrentes.
Recursively defined structures not permitted	
String too long	Uma seqüência de caracteres é muito longa, ou você esqueceu de fechar aspas duplas em alguma linha.
Undefined identifier	Identificador não declarado. Veja se você declarou a variável que está usando.
Undefined label that was used in a GOTO	GOTO para um rótulo não declarado ou inválido.
Unknown device type	Não conhece o dispositivo usado pela diretiva DEVICE

Principais problemas quanto à execução do programa:

1) Não estou achando o arquivo HEX:
    a. Ao abrir vários arquivos no compilador, verifique no canto inferior esquerdo o nome do arquivo a ser compilado. Lembre que o compilador irá compilar somente um dos programas abertos, e isto está identificado na linha inferior esquerda da janela do compilador. Caso estejas compilado o arquivo errado, feche todos os arquivos abertos (CLOSE ALL FILES) e abra novamente só o seu arquivo a ser compilado, repetindo o processo de compilação. Se a compilação ocorrer sem erros e o problema persistir, veja se a pasta onde você está trabalhando é a mesma onde você está procurando o arquivo HEX.
2) Arquivo HEX transfere mas não roda:
    a. Veja se não há erro no seu programa. Tente transferir outro programa que funcione para testar a estação.
    b. Veja se os parâmetros (FUSES) estão configurados adequadamente. Lembre da configuração no caso de usar o EPIC (View->Configuration, selecione HS, e o restante tudo desligado (off). Clique também em View->Options e desligue a opção update configuration (caso estiver ligado).
    c. Veja se seu programa principal não possui erros (como um ponto e vírgula na linha do While(1)...

3) Arquivo HEX dá erro ao transferir
    a. Veja se o microcontrolador selecionado é correto
    b. Veja se o cabo está devidamente instalado
    c. Reinicie o PIC
    d. Veja se a fonte é adequada e está com tensão suficiente para gravação
    e. Veja se você está transferindo o arquivo correto (ARQUIVO HEX)

Situações mais comuns na automação industrial:

1) O programa aciona uma determinada saída por um período de tempo:

```
...
output_high(PIN_xx); // ativa saída
delay_ms(xxxx);  // tempo em milisegundos
output_low(PIN_xx); // desativa saída
...
```

2) O programa aciona uma determinada saída, aguardando um sinal externo para prosseguir:

```
...
output_high(PIN_xx);  // liga saída
while(!input(PIN_xx)); /// enquanto não houver sinal, aguarda...
output_low(PIN_xx); // desliga saída
...
```

3) Aguardar por uma determinada saída, considerando um tempo máximo para resposta:

```
...
long tempomaximo; // declara variável de 16 bits
...
tempomaximo = 10000; // atribui valor 10000 para variável
while(tempomaximo > 0)  // enquanto variável for maior que zero
     {
     delay_ms(1);  // causa atraso de 1 milisegundo
     tempomaximo --; // decrementa variável
     if (input(PIN_xx)) break;  // se houver sinal, interrompe.
     }
...
```

4) Aguardar por uma saída pulsar (ligar e desligar)

```
...
while(!input(PIN_XX));  // aguarda entrada ativar …
while(input(PIN_XX)); // aguarda entrada desligar …
…
```

5) Repetir um evento uma quantidade determinada de vezes.

```
...
int CONTA;
...
for (CONTA = 0; CONTA < 100; CONTA ++)
     {
     output_high(PIN_xx);
     delay_ms(xxxxx);
     output_low(PIN_xx);
     delay_ms(xxxxx);
     }
...
```

6) Aguardar por dois sinais digitais para seguir o programa....

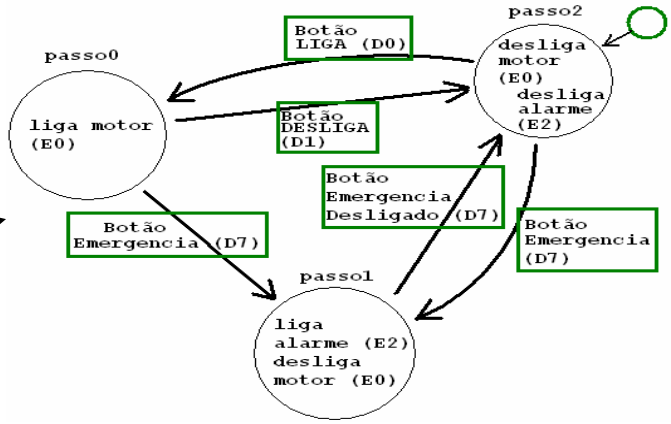
```
...
while(!input(PIN_xx) || !input(PIN_xx));
   // equanto não houver sinal em uma ou outra entrada,
   // fica aguradando ...
...
```

7) Aguardar por um sinal digital (dentre várias opções) para seguir o programa...

```
...
while(1)
    {
    if (input(PIN_xx)) break;
    if (input(PIN_xx)) break;
    if (input(PIN_xx)) break;
    ...
    }
...
```

8) Enquanto determinada entrada digital não for acionada, repetir um determinado processo.

```
...
while(!input(PIN_xx))
       {
       output_high(PIN_xx);
       delay_ms(xxxx);
       output_low(PIN_xx);
       delay_ms(xxxx);
       ...
       }
...
```



9) Máquina de estados

```
...
int PASSO = 2;
...
while(1)
    {
    switch (PASSO)
               {
               case 0 : output_high(PIN_E0);
                        if (input(PIN_D7)) PASSO = 1;
                        if (input(PIN_D1)) PASSO = 2;
                       break;

               case 1 : output_high(PIN_E2);
                       output_low(PIN_E0);
                        if (!input(PIN_D7)) PASSO = 2;
                       break;

               case 2 : output_low(PIN_E0);
                        output_high(PIN_E2);
                        if (input(PIN_D0)) PASSO = 0;
                        if (input(PIN_D7)) PASSO = 1;
                       break;
                  }
      }
```

10) Criando uma função para delay de segundos

```
...
void delay_s(long tempo_s)
   {
   while (tempo_s > 0)
         {
         tempo_s --; // diminui variável
         delay_ms(1000); // tempo de um segundo
         }
   }
…
main( )
    {
    …
    delay_s(10);  // causa atraso de 10 segundos….
    ...
    }
```