Programação de Microcontroladores

Linguagem de Programação (C) - Expressões

CENTEC

INSTITUTO CENTRO DE ENSINO TECNOLÓGICO

Prof. Flávio Murilo de Carvalho Leal
Instituto Centro de Ensino Tecnológico
Faculdade de Tecnologia do Cariri

- **Tipos primitivos de dados:** char, int, float, double e void.

- **Tipos primitivos de dados:** char, int, float, double e void.

Tipo	**Tamanho em bits**	**Faixa**
char	8	-127 a 127
int	16	-32767 a 32767
float	32	$3.4 * 10^{-38}$ a $3.4 * 10^{38}$
double	64	$1.7 * 10^{-308}$ a $1.7 * 10^{308}$

- **Modificadores:** signed, unsigned, long e short.

- **Modificadores:** signed, unsigned, long e short.

Tipo	Tamanho em bits	Faixa
char	8	-127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
int	16	-32767 a 32767
unsigned int	16	0 a 65535
signed int	16	-32767 a 32767
short int	16	-32767 a 32767
unsigned short int	16	0 a 65535
signed short int	16	-32767 a 32767
long int	32	-2147483647 a 2147483647
signed long int	32	-2147483647 a 2147483647
unsigned long int	32	0 a 4294967295
float	32	$3.4 * 10^{-38}$ a $3.4 * 10^{38}$
double	64	$1.7 * 10^{-308}$ a $1.7 * 10^{308}$
long double	80	$3.4 * 10^{-4932}$ a $3.4 * 10^{4932}$

- **Definição:** Posições capazes de guardar valores e que podem ser nomeadas pelo programador seguindo a seguinte forma geral:

- **Definição:** Posições capazes de guardar valores e que podem ser nomeadas pelo programador seguindo a seguinte forma geral:

tipo lista_de_variaveis;

- **Definição:** Posições capazes de guardar valores e que podem ser nomeadas pelo programador seguindo a seguinte forma geral:

  tipo lista_de_variaveis;

- **Exemplo:**

```
1 int i, j, l;
2 short int k;
3 double dist;
4 float temp;
```

- **Definição:** Posições capazes de guardar valores e que podem ser nomeadas pelo programador seguindo a seguinte forma geral:

  tipo lista_de_variaveis;

- **Exemplo:**

```
1 int i, j, l;
2 short int k;
3 double dist;
4 float temp;
```

- **OBS:** Não usar nos nomes das variáveis acentos, espaços (preferível usar _) e números no início do nome (pode ser usado número no final).

- **Definição:** Variáveis declaradas dentro de uma função e que só podem ser referenciadas dentro da função em que foi declarada, ou seja, variáveis locais não são reconhecidas fora do seu bloco de código (bloco delimitado por {}):

- **Definição:** Variáveis declaradas dentro de uma função e que só podem ser referenciadas dentro da função em que foi declarada, ou seja, variáveis locais não são reconhecidas fora do seu bloco de código (bloco delimitado por {}):
- **Exemplo:** As variáveis x são declaradas duas vezes e não têm relação entre si.

```
1  void func1(void)
2  {
3      int x;
4      x=10;
5  }
6
7  void func2(void)
8  {
9      int x;
10     x=-199;
11 }
```

- **Formas adequadas de declaração:** Deve-se declarar todas as variáveis logo após o início do bloco (após “{”)
- **Forma errada:**

- **Formas adequadas de declaração:** Deve-se declarar todas as variáveis logo após o início do bloco (após "{")
- **Forma errada:**

```
1 /*Funcao com erro.*/
2 void funcao(void)
3 {
4       int x;
5       x=10;
6       int y; /*Esta linha ira causar um erro.*/
7       y=20;
8 }
```

- **Formas adequadas de declaração:** Deve-se declarar todas as variáveis logo após o início do bloco (após "{")
- **Forma correta:**

```
1 /*Funcao correta.*/
2 void funcao(void)
3 {
4       int x, y;
5       x=10;
6       y=20;
7 }
```

- **Definição:** Variáveis declaradas fora de qualquer função e que podem ser referenciadas em qualquer lugar do código, desde que declarada em qualquer lugar antes do seu primeiro uso.

- **Definição:** Variáveis declaradas fora de qualquer função e que podem ser referenciadas em qualquer lugar do código, desde que declarada em qualquer lugar antes do seu primeiro uso.
- **Exemplo:**

```
1  #include <stdio.h>
2
3  int x=10, y=20, resultado, dobro;
4
5  void soma(void)
6  {
7      resultado=x+y;
8  }
9
10 void main(void)
11 {
12     soma();
13     dobro=resultado*2;
14 }
```

- São úteis na exibição de textos onde são necessárias modificações como recuo/avanço, pulo de linha e exibição de caracteres especiais:

- São úteis na exibição de textos onde são necessárias modificações como recuo/avanço, pulo de linha e exibição de caracteres especiais:

Código	**Significado**
\b	Retrocesso
\f	Alimentação de formulário
\n	Nova linha
\r	Retorno de carro
\t	Tabulação horizontal
\“	Aspas duplas
\‘	Aspas simples
\0	Nulo
\\	Barra invertida
\v	Tabulação vertical
\a	Alerta (beep)
\N	Constante octal
\xN	Constante hexadecimal

- Operadores de atribuição são utilizados quando se deseja armazenar em uma variável um valor específico ou o resultado de uma expressão matemática. segue a forma geral:

- Operadores de atribuição são utilizados quando se deseja armazenar em uma variável um valor específico ou o resultado de uma expressão matemática. segue a forma geral:

$$nome\_da\_variavel = expressao;$$

- Operadores de atribuição são utilizados quando se deseja armazenar em uma variável um valor específico ou o resultado de uma expressão matemática. segue a forma geral:

$$nome\_da\_variavel = expressao;$$

- É importante saber que uma variável pode ter outra variável como valor atribuído:

- Operadores de atribuição são utilizados quando se deseja armazenar em uma variável um valor específico ou o resultado de uma expressão matemática. segue a forma geral:

$$nome\_da\_variavel = expressao;$$

- É importante saber que uma variável pode ter outra variável como valor atribuído:

```
1 #include <stdio.h>
2
3 int x=10, y=20, z, soma;
4
5 void main(void)
6 {
7       z=y;
8       soma=x+z; /*O valor armazenado na variavel soma sera 30.*/
9 }
```

- É possível atribuir a várias variáveis o mesmo valor ao mesmo tempo utilizando apenas uma linha:

- É possível atribuir a várias variáveis o mesmo valor ao mesmo tempo utilizando apenas uma linha:

```
1 #include <stdio.h>
2
3 int x=10, y, z;
4
5 void main(void)
6 {
7       y=z=x; /*Todas as variaveis terao o mesmo valor de x, 10.*/
8 }
```

- São utilizados para operações matemática conforme os símbolos da tabela:

- São utilizados para operações matemática conforme os símbolos da tabela:

Operador	**Ação**
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Resto de divisão
−−	Decremento (x– equivale a x=x-1)
++	Incremento (x++ equivale a x=x+1)

- **Exemplo:**

```
1 #include <stdio.h>
2
3 int x=10, y=2, subt, soma, mult, divis, resto, dec, inc;
4
5 void main(void)
6 {
7       subt=x-y; /*subt tem valor 8 armazenado.*/
8       soma=x+y; /*soma tem valor 12 armazenado.*/
9       mult=x*y; /*mult tem valor 20 armazenado.*/
10      divi=x/y; /*div tem valor 5 armazenado.*/
11      resto=x%y; /*resto tem valor 0 armazenado.*/
12      dec=x--; /*dex tem valor 9 armazenado.*/
13      inc=x++; /*dex tem valor 11 armazenado.*/
14 }
```

- São utilizados para operações lógicas de igual às operações de eletrônica digital:

- São utilizados para operações lógicas de igual às operações de eletrônica digital:

Operador	**Ação**
&&	E/AND (Pode-se utilizar o termo “and”)
||	OU/OR (Pode-se utilizar o termo “or”)
!	NÃO/NOT/INVERSOR

- Tabela verdade com resultados de operadores lógicos:

- Tabela verdade com resultados de operadores lógicos:

A	**B**	**A&&B**	**A||B**	**!A**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Operadores relacionais

- São utilizados para comparações entre valores e variáveis:

- São utilizados para comparações entre valores e variáveis:

Operador	**Ação**
$>$	Maior que
$>=$	Maior que ou igual
$<$	Menor que
$<=$	Menor que ou igual
$==$	Igual
$!=$	Diferente

- **Exemplo:**

```
1  #include <stdio.h>
2
3  int x=10, y=2, subt, soma, mult, divis, resto, dec, inc;
4
5  void main(void)
6  {
7      subt=x-y; /*subt tem valor 8 armazenado.*/
8      soma=x+y; /*soma tem valor 12 armazenado.*/
9      mult=x*y; /*mult tem valor 20 armazenado.*/
10     divi=x/y; /*div tem valor 5 armazenado.*/
11     resto=x%y; /*resto tem valor 0 armazenado.*/
12     dec=x--; /*dec tem valor 9 armazenado.*/
13     inc=x++; /*dec tem valor 11 armazenado.*/
14 }
```

- São aplicados a bits individuais:

- São aplicados a bits individuais:

Operador	**Ação**
&	AND
|	OR
^	OR Esclusivo (XOR)
~	Complemento de um
>>	Deslocamento para a direita
<<	Deslocamento para a esquerda

- **Exercício:** quais serão os resultados armazenados na variável "valor" em cada linha?

```
1  #include <stdio.h>
2
3  int x=7; /*00000111 em binario*/
4  int y=2; /*00000010 em binario*/
5
6  void main(void)
7  {
8       valor=x\&y;
9       valor=x|y;
10      valor=x^y;
11      valor=x~y;
12      valor=x>>;
13      valor=x<<;
14 }
```