ATMEGA328/ARDUINO – C_LANGUAGE – LOGIC – if – else – ternary
https://github.com/teaksoon/lmaewapm

The C-Language Keywords and Symbols

Keywords

MEMORY	CONTROL
01.void	21.return
02.char	**22.if**
03.int	**23.else**
04.short	24.switch
05.long	25.case
06.float	26.default
07.double	27.while
08.signed	28.do
09.unsigned	29.for
10.struct	30.break
11.union	31.continue
12.enum	32.goto
13.const	
14.volatile	
15.auto	
16.extern	
17.static	
18.register	
19.typedef	
20.sizeof	

Symbols

CONTROL	LOGIC	MATH	BIT OP
#	==	*	|
< >	!=	%	&
//	<	/	^
/* */	>	+	~
( )	<=	-	<<
{ }	>=		>>
;	&&		
,	||		
“			
‘	!		
=			
[ ]			
:			
?			
.			
\			
MEMORY			
&			
*			

Symbols for LOGIC COMPARISON and OPERATION

We use these Symbols to make logic comparison and operations, then based on the results we our PROGRAM can then make various decision

(The foundation for Artificial Intelligence)

We need to use them for some of the C-Language CONTROL Keywords, in this topic we talk about the **if** and **else** Keyword

Also the **? :** Symbol for ternary operations

C-LANGUAGE LOGIC

The C-LANGUAGE LOGIC is simple. It is either TRUE or FALSE

TRUE = 1 and **FALSE = 0**

LOGIC COMPARISON

COMPARISON **between two numbers,** will **return a LOGIC_NUMBER**

LOGIC_NUMBER = 1 for TRUE or **LOGIC_NUMBER = 0 for FALSE**

```
- The COMPARISON Symbol is placed between two numbers
( a == b ) when a EQUAL b,                returns 1, otherwise returns 0
( a != b ) when a NOT EQUAL b,            returns 1, otherwise returns 0
( a < b )  when a LESS THAN b,            returns 1, otherwise returns 0
( a > b )  when a MORE THAN b,            returns 1, otherwise returns 0
( a <= b ) when a LESS THAN or EQUAL b,   returns 1, otherwise returns 0
( a >= b ) when a MORE THAN or EQUAL b,   returns 1, otherwise returns 0
```

NOTE:
a and **b** can be any numbers

LOGIC OPERATION

LOGIC OPERATION, will **return a LOGIC_NUMBER**

LOGIC_NUMBER = 1 for TRUE or **LOGIC_NUMBER = 0 for FALSE**

AND OPERATION (double Ampersand **&&**)
- The **&&** Symbol is placed **between Two LOGIC_NUMBER**

```
( 1 && 1 ) returns 1
( 1 && 0 ) returns 0
( 0 && 1 ) returns 0
( 0 && 0 ) returns 0
```

Easy way to remember the AND Operation: as long as there is a 0, the return is 0

OR OPERATION (double Vertical Bar Symbol **||**)
- The **||** Symbol is placed **between Two LOGIC_NUMBER**

```
( 1 || 1 ) returns 1
( 1 || 0 ) returns 1
( 0 || 1 ) returns 1
( 0 || 0 ) returns 0
```

Easy way to remember the OR Operation: as long as there is a 1, the return is 1

NOT OPERATION (single Exclamation Symbol **!**)
- The **!** Symbol is placed **before One LOGIC_NUMBER**

```
( !1 ) returns 0
( !0 ) returns 1
```

NOTE:
LOGIC_NUMBER is either 1 or 0
- Any **number** that is not 1 or 0 when used in LOGIC OPERATION will be considered as LOGIC_NUMBER = 1

- The Bracket ( ) pair is also used to decide code execution precedence (the most inner bracket is performed first)

- example: `( ( (a && 1) > b ) || c )`

In the example above, the AND operation (a && 1) will be performed first, result return from that operation will be used for comparison with b using the greater than > symbol, then the result from that comparison will be used to perform the OR operation with c using the || symbol

if

The “if” Keyword allows us to run one or more instruction codes based on a **TRUE LOGIC (LOGIC_NUMBER=1)**

“if” Keyword used alone
Part1:“if” Keyword
Part2:LOGIC_NUMBER - Numbers other than 0 and 1 will be considered as 1
Part3:if_body

Part2:LOGIC_NUMBER
- placed within the bracket ( ) pair
- TRUE or FALSE (1 or 0)

Part1:“if” Keyword

Part3:if_body
- multiple “instruction codes” within a Curly Bracket { } pair
- Curly bracket is optional for single “instruction code” in body

```
if(LOGIC_NUMBER)
{


}
```

Arduino IDE|Save PROGRAM as: **c_if**
Enter codes below and upload. Use the Serial Monitor to see results

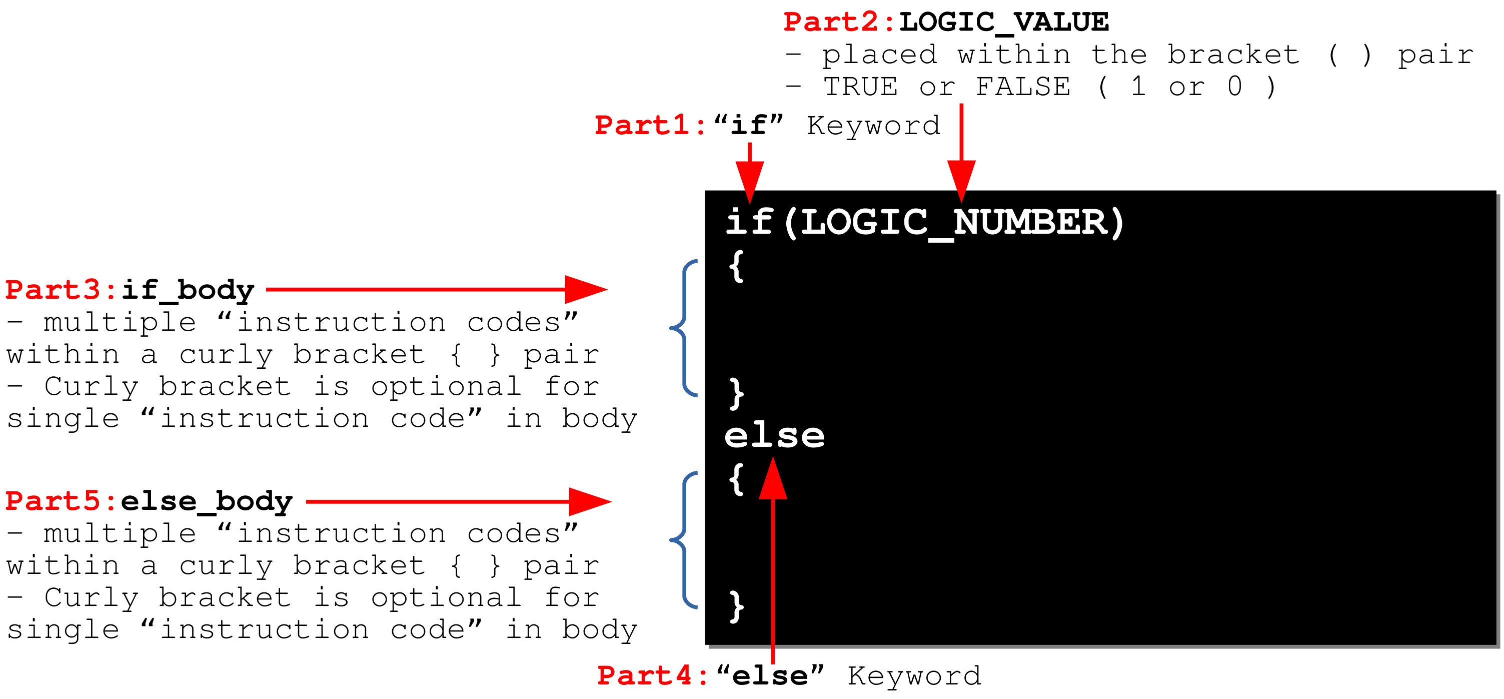
```
void setup() {
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...");

  Serial.print("\n\nif( LOGIC_NUMBER )");
  if( 1 ) {
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  }
  if( 0 ) {
    Serial.print("\nLOGIC_NUMBER=0, if_body will NOT be executed");
  }

  int a = 0;
  int b = 6;
  Serial.print("\n\nCOMPARISON if( a < b ), when a=0 and b=6");
  Serial.print("\nif(a < b) will become if(0 < 6)");
  Serial.print("\n(0 < 6) returns ( 1 ), our code becomes if( 1 )");
  if(a < b) {
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  }
  Serial.print("\n\nOPERATION  if( a || b ), when a=0 and b=6");
  Serial.print("\nif(a || b) will become if(0 || 1)");
  Serial.print("\n(0 || 1) returns ( 1 ), our code becomes if( 1 )");
  if(a || b) {
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  }
  Serial.print("\n\nOPERATION  if( a && b ), when a=0 and b=6");
  Serial.print("\nif(a && b) will become if(0 && 1)");
  Serial.print("\n(0 && 1) returns ( 0 ), our code becomes if( 0 )");
  if(a && b) {
    Serial.print("\nLOGIC_NUMBER=0, if_body will NOT be executed");
  }
}
void loop(){}
```

else
The “else” Keyword allows us to run one or more instruction codes based on a **FALSE LOGIC (LOGIC_NUMBER=0)**
The “else” keyword must be used together with the “if” Keyword

“else” Keyword used with **“if”** Keyword
Part1:“if” Keyword
Part2:LOGIC_NUMBER – Numbers other than 0 and 1 will be considered as 1
Part3:if_body
Part4:“else” Keyword
Part5:else_body



Part2:LOGIC_VALUE
– placed within the bracket ( ) pair
– TRUE or FALSE (1 or 0)

Part1:“if” Keyword

Part3:if_body
– multiple “instruction codes” within a curly bracket { } pair
– Curly bracket is optional for single “instruction code” in body

Part5:else_body
– multiple “instruction codes” within a curly bracket { } pair
– Curly bracket is optional for single “instruction code” in body

Part4:“else” Keyword

```
if(LOGIC_NUMBER)
{


}
else
{


}
```

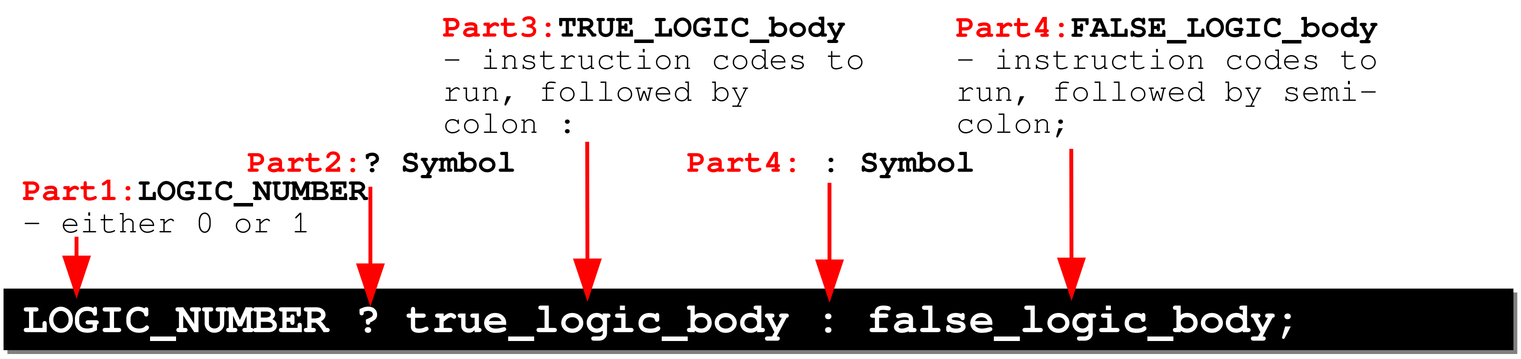
Arduino IDE|Save PROGRAM as: **c_if_else**
Enter codes below and upload. Use the Serial Monitor to see results

```
void setup() {
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...");

  Serial.print("\n\nif( LOGIC_NUMBER )");
  if( 1 ) { // LOGIC_NUMBER = 1, will run if body
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  } else {
    Serial.print("\nLOGIC_NUMBER=0, else_body will be executed");
  }
  if( 0 ) { // LOGIC_NUMBER = 0, will run else body
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  } else {
    Serial.print("\nLOGIC_NUMBER=0, else_body will be executed");
  }
  int a = 0;
  int b = 6;
  Serial.print("\n\nCOMPARISON if( a < b ), when a=0 and b=6");
  Serial.print("\nif(a < b) will become if(0 < 6)");
  Serial.print("\n(0 < 6) returns ( 1 ), our code becomes if( 1 )");
  if(a < b) { // try change these
    Serial.print("\nLOGIC_NUMBER=1, if_body will be executed");
  } else {
    Serial.print("\nLOGIC_NUMBER=0, else_body will be executed");
  }
}
void loop(){}
```

ternary operation
The is similar to the "if-else" operation, used as a shortcut "if-else" shortcut coding style

"**? :**" Ternary Operation Symbols
Part1:LOGIC_NUMBER
Part2:? Question Mark Symbol
Part3:TRUE_LOGIC_body
Part4:: Colon Symbol
Part5:FALSE_LOGIC_body



```
LOGIC_NUMBER ? true_logic_body : false_logic_body;
```

```
result = LOGIC_NUMBER ? true_logic_body : false_logic_body;
```

Arduino IDE|Save PROGRAM as: **c_ternary**
Enter codes below and upload. Use the Serial Monitor to see results

```
void setup() {
int tmp;
int toggle_var;
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...\n");

  Serial.print("\n\nPlain ternary operation");
  1 ? true_logic() : false_logic();

  Serial.print("\n\nTernary operation with results");
  tmp = 0 ? true_logic():false_logic();

  Serial.print("\nResult from ternary operation = ");
  Serial.print(tmp);

  // toggle
  toggle_var = 0; // try change this between 0 and 1 to test
  Serial.print("\n\ntoggle var before ternary = ");
  Serial.print(toggle_var);

  toggle_var = toggle_var ? 0 : 1;

  Serial.print("\ntoggle var after ternary = ");
  Serial.print(toggle_var);
}
void loop(){}

int true_logic() {
  Serial.print("\nCode Inside true_logic function");
  return 2;
}
int false_logic() {
  Serial.print("\nCode Inside false_logic function");
  return 3;
}
```