Functions and Serial Communication in Arduino

CSE 315

Peripherals & Interfacing

Abdullah Al Omar

Lecturer, CSE, UAP

Function

What is Function?

Functions are the block of code which allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "**called**".

The typical case for creating a function is when one needs to perform the same action multiple times in a program

Advantages of Function:

Functions help the programmer stay organized. Often this helps to conceptualize the program.

Advantages of Function: (contd.)

Functions codify one action in one place so that the function only has to be thought out and debugged once.

Advantages of Function: (contd.)

This also reduces chances for errors in modification, if the code needs to be changed.

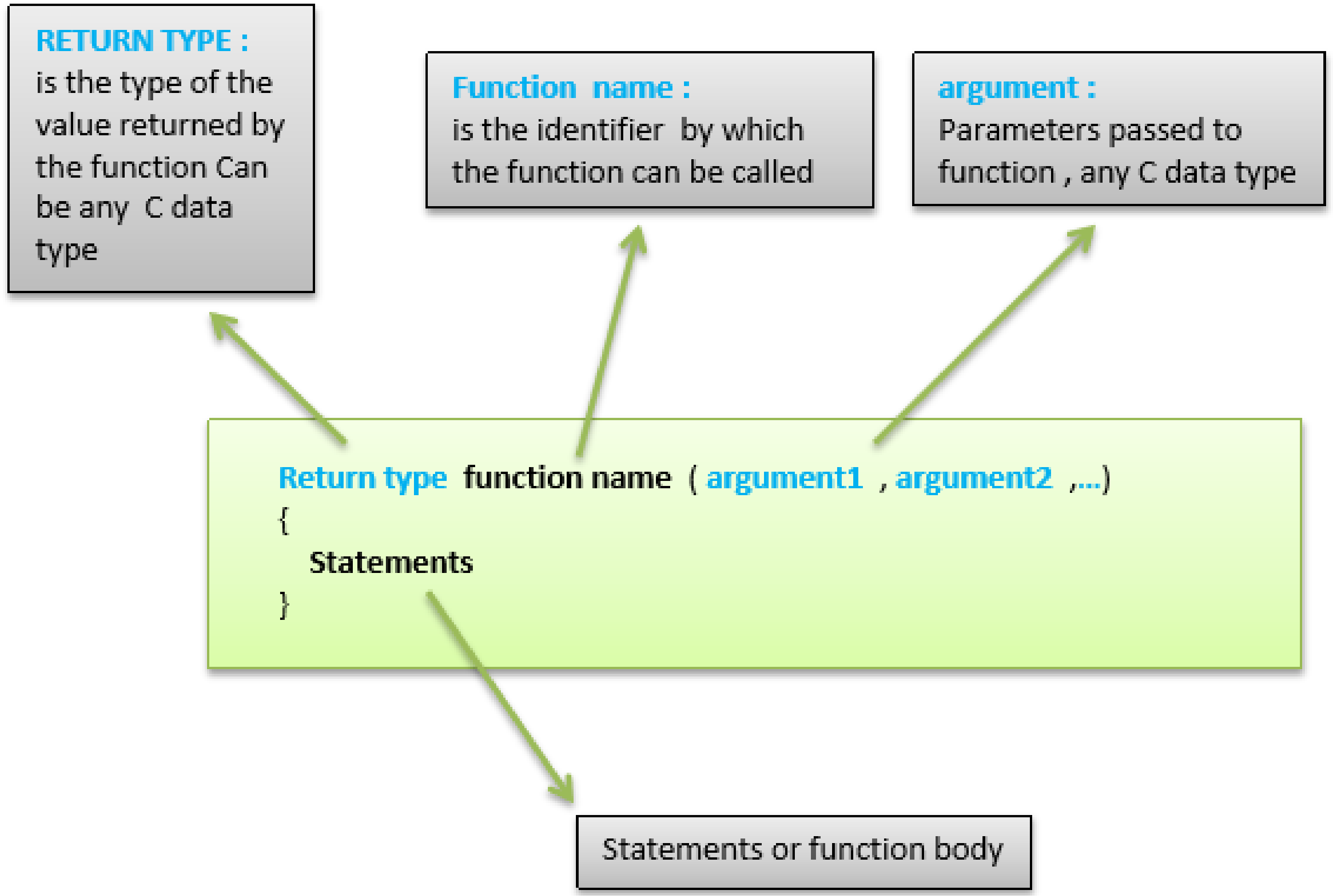
Advantages of Function: (contd.)

Functions make the whole sketch smaller and more compact because sections of code are reused many times.

Advantages of Function: (contd.)

They make it easier to reuse code in other programs by making it more modular, and as a nice side effect, using functions also often makes the code more readable.

How Function looks like?

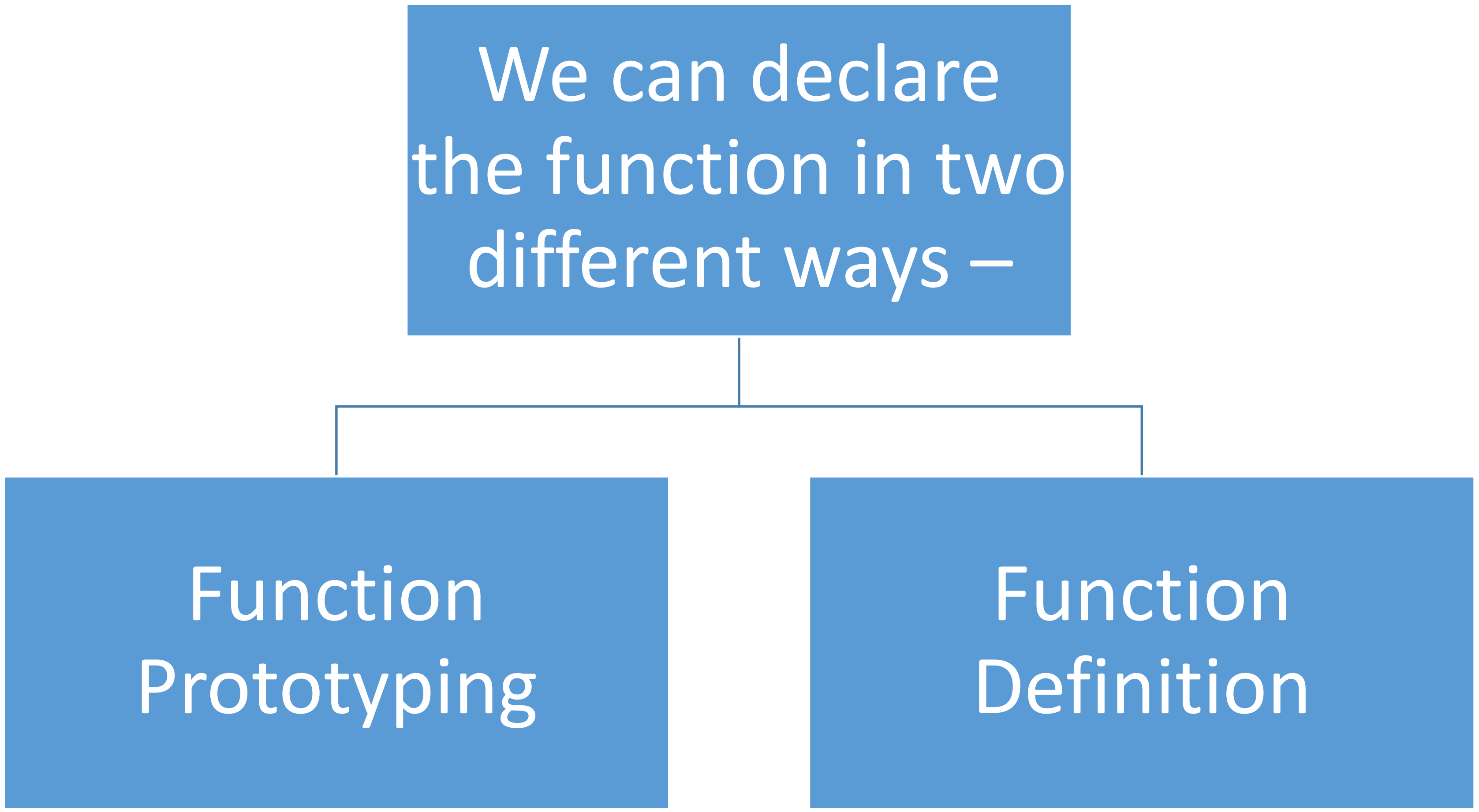

RETURN TYPE :
is the type of the value returned by the function Can be any C data type
Function name :
is the identifier by which the function can be called
argument :
Parameters passed to function , any C data type
Return type function name (argument1 , argument2 ,...)
{
Statements
}
Statements or function body

Function declaration in Arduino:

$f(x)$

A function is declared outside any other functions, above or below the loop function.

Function declaration in Arduino: (contd.)



Function Prototyping:

The first way is just writing the part of the function called **a function prototype** above the loop function, which consists of –

- Function return type
- Function name
- Function argument type, no need to write the argument name
- Function prototype must be followed by a semicolon (;).

Function Prototyping: (contd.)

```
int sum_func (int x, int y)    // function declaration
{
            int z = 0;
            z = x+y ;
            return z; // return the value
}
void setup () {
            Statements                    // group of statements…..
}
Void loop () {
            int result = 0 ;
            result = Sum_func (5,6) ;    // function call
}
```

Function Definition:

The second part, which is called the function definition or declaration, must be declared below the loop function, which consists of –

- Function return type
- Function name
- Function argument type, here you must add the argument name
- The function body (statements inside the function executing when the function is called)

Function Definition: (contd.)

```
int sum_func (int , int ) ; // function prototype
void setup () {
        Statements // group of statements
}
Void loop () {
        int result = 0 ;
        result = Sum_func (5,6) ; // function call
}
int sum_func (int x, int y) // function declaration {
         int z = 0;
        z = x+y ;
        return z; // return the value
}
```

Class work:

Write a sketch with function to find the average of two numbers:

Serial Communication

Use of Serial:

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

Serial.begin():

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with Serial Monitor, make sure to use one of the baud rates listed in the menu of Arduino. You can, however, specify other rates - for example,

- to communicate over pins 0 and 1 with a component that requires a particular baud rate.

Serial.begin(): (contd.)

```
void setup() {
   Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {}
```

Serial.print():

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

- Serial.print(78) gives "78"
- Serial.print(1.23456) gives "1.23"
- Serial.print('N') gives "N"
- Serial.print("Hello world.") gives "Hello world."

Serial.print(): [second parameter]

An optional second parameter specifies the base (format) to use; permitted values are BIN(binary, or base 2), OCT(octal, or base 8), DEC(decimal, or base 10), HEX(hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example-

- Serial.print(78, BIN) gives "1001110"
- Serial.print(78, OCT) gives "116"
- Serial.print(78, DEC) gives "78"
- Serial.print(78, HEX) gives "4E"
- Serial.print(1.23456, 0) gives "1"
- Serial.print(1.23456, 2) gives "1.23"
- Serial.print(1.23456, 4) gives "1.2346"

Printing Decimal and Binary of a number:

```
void setup() {
  Serial.begin(9600); // open the serial port at
9600 bps:
}

void loop() {
  // print labels

  Serial.print("DEC");
  Serial.print("\t");

  Serial.print("BIN");
  Serial.println();       // carriage return after
the last label
 for (int x = 0; x < 10; x++) { // only part of the ASCII chart,
change to suit
   // print it out in two formats:

   Serial.print(x, DEC);  // print as an ASCII-encoded decimal
   Serial.print("\t");    // prints a tab


   Serial.print(x, BIN);  // print as an ASCII-encoded binary
   Serial.println();
  delay(200);           // delay 200 milliseconds
 }
 Serial.println();       // prints another carriage return
}
```

Printing Decimal and Binary of a number: (Output)

```
Dec          Bin
0            0000
1            0001
2            0010
3            0011
4            0100
5            0101
.             . . . .
.             . . . .
9            1001
```

Serial.end()

Disables serial communication, allowing the RX and TX pins to be used for general input and output. To re-enable serial communication, call Serial.begin().

Incorporating Function and Serial:

```
int myMultiplyFunction(int x, int y){
                int result;
                result = x * y;
                return result;
        }
void setup(){
        Serial.begin(9600);
}

void loop() {
        int i = 2;
        int j = 3;
         int k;

        k = myMultiplyFunction(i, j);

        Serial.println(k);   // k now contains 6

}
```

Write a sketch with function to find the average of three numbers with serial output (after decimal- "." it will print 4 numbers):

Assignments:

- Serial.write()
- Serial.println()
- Serial.available()
- Serial.read()

Thank you