Engineering Technology (ENGR 101)

Functions

Mohammad Nekooei

Engineering and Computer Science

Victoria University of Wellington

Arduino Functions

- What are functions good for?
  - structuring our thoughts (structured programming)
  - allowing us to re-use code, reducing work and reducing errors

- A C program can be modularised by functions
  - A big program can be broken down into a number of smaller ones

```
void setup() {
   Serial.begin(9600);
   DashedLine();
   Serial.println("| Program Menu |");
   DashedLine();
}
void loop() {
}
void DashedLine() {
   Serial.println("----------------");
}
```

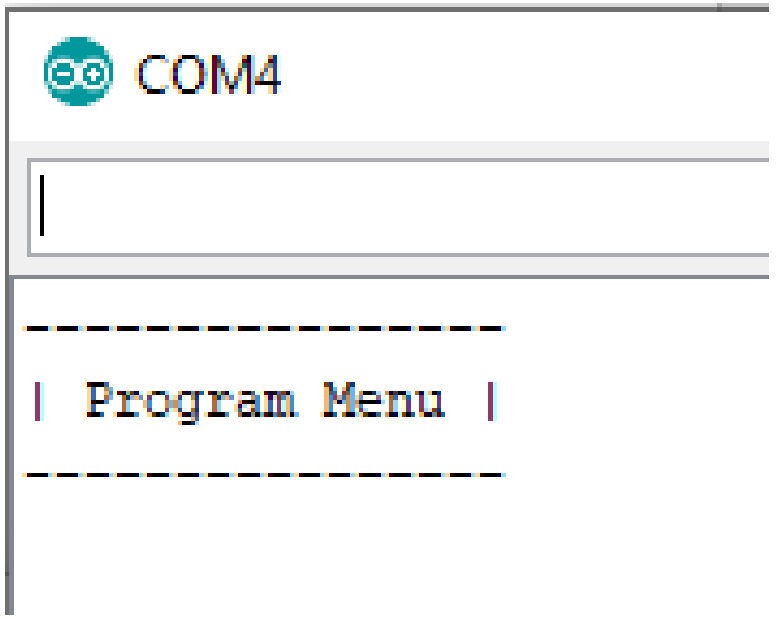
Function is called here

Function is called here

Return type

Function Name

Statement(s) that run when function is called

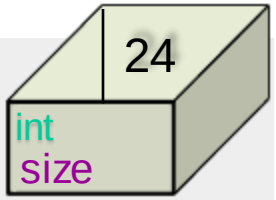


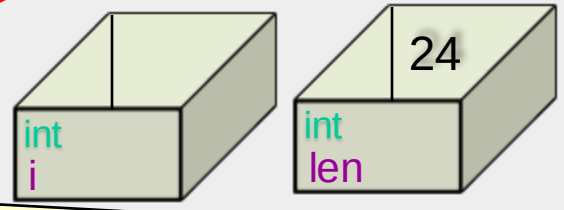
Passing a Value to a Function

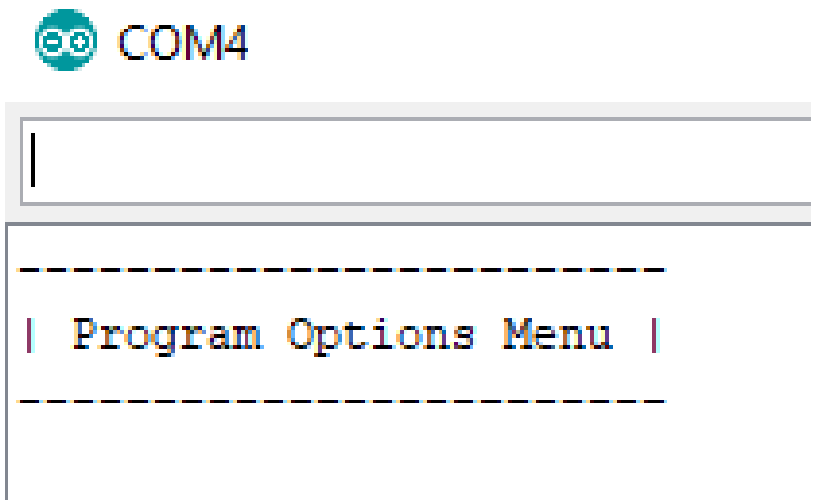
```
void setup() {
   Serial.begin(9600);
   int size = 24;
   DashedLine(size);
   Serial.println("| Program Options Menu |");
   DashedLine(size);
}

void loop() {
}

void DashedLine(int len) {
   // draw the line
   for (int i; i = 0; i < len; i++) {
      Serial.print("-");
   }
   Serial.println("");
}
```

Passing a value

Arguments





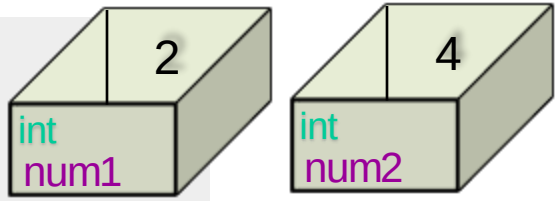


Parameters

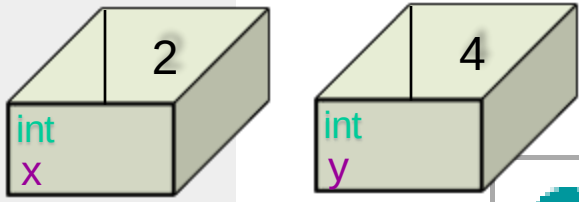
Special variables which are given values each time the function is called. Body of function can use the values in the parameters

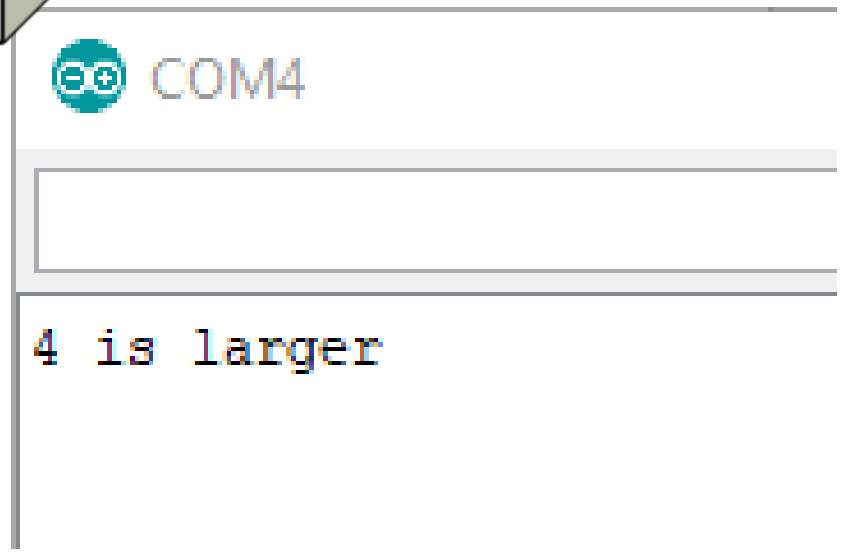
Another Example of a Function

```
void setup() {
   int num1=2, num2=4;
   Serial.begin(9600);
   larger(num1, num2);
}

void loop() {
}

void larger(int x, int y) {
   if (x>y) {
      Serial.print(x);
      Serial.println(" is larger");
   }
   else{
      Serial.print(y);
      Serial.println(" is larger");
   }
}
```



Passing values





Function Calls with parameters

Function Definition: Like a pad of worksheets

```
void larger( int x, int y){
     if (x>y) {
          Serial.print(x);
          Serial.println(" is larger");
     }
     else{
          Serial.print(y);
          Serial.println(" is larger");
     }
}
```

Calling a Function:

larger(2, 4);

⇒ get a "copy" of the function worksheet

⇒ copy the arguments to the parameter places

⇒ perform each action in the body

⇒ throw the worksheet away (losing all the information on it)

Principle of good design

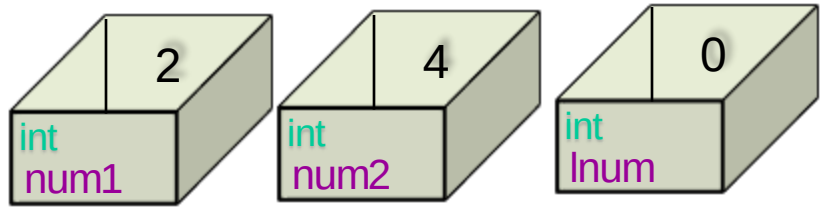
- Parameterising a function makes it more flexible and general
  - Allows us to call the same function with different arguments to do the same thing in different ways
  - Allows us to reuse the same bit of code

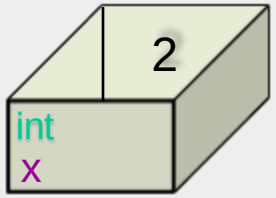
Returning a Value from a Function

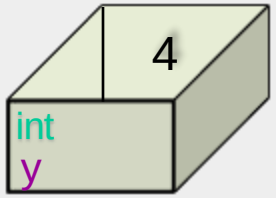
```
void setup() {
   int num1 = 2, num2 = 4, lNum = 0;
   Serial.begin(9600);
   lNum = larger(num1, num2);
   Serial.print(lNum);
   Serial.println(" is larger");
}

void loop() {
}

int larger(int x, int y) {
   if (x>y) {
      return x;
   }
   else{
      return y;
   }
}
```

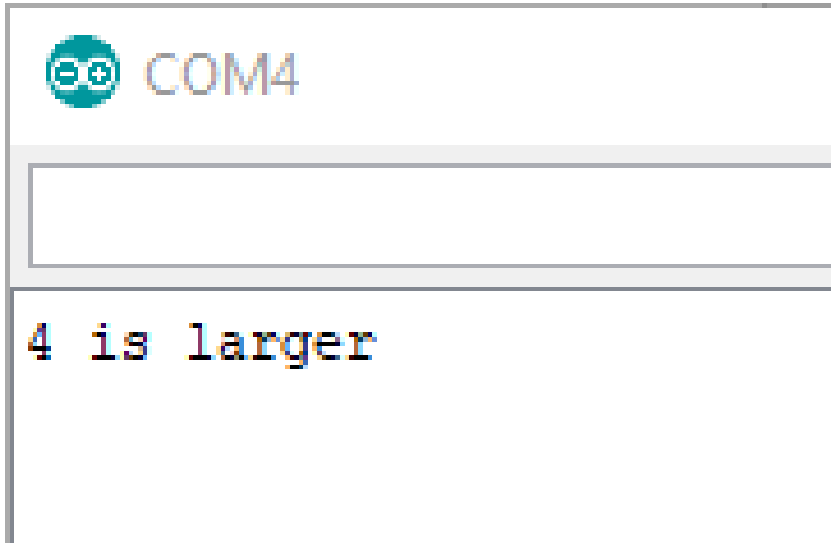
Passing values

Return type







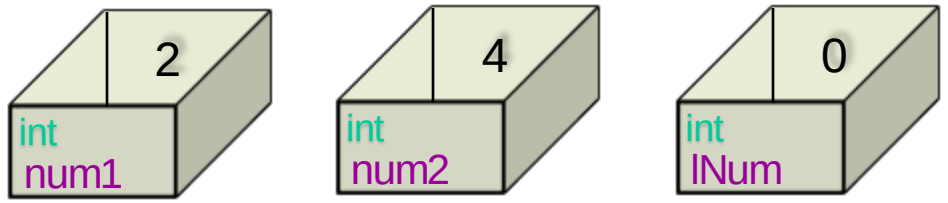


Returning a Value from a Function

- What happens if we call the function:

  larger();

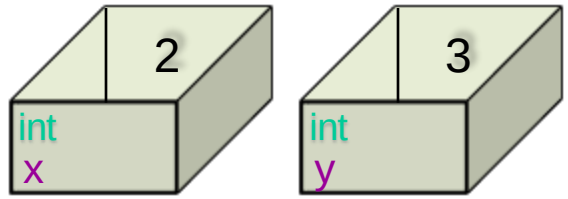
```
void setup() {
 ✓ int num1 = 2, num2 = 4, lNum = 0;

 ✓ Serial.begin(9600);

   lNum = larger(num1, num2);

   Serial.print(lNum);
   Serial.println(" is larger");
}
```



Returning values

return value:

```
int larger(int x, int y){
---------------------------------------------------------------------------
      if (x>y) {
         return x;
      }
      else{
         return y;
      }
}
```

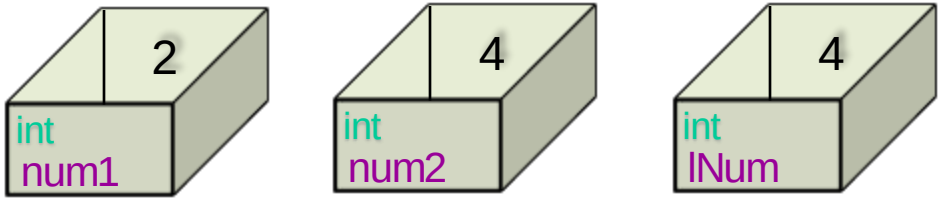


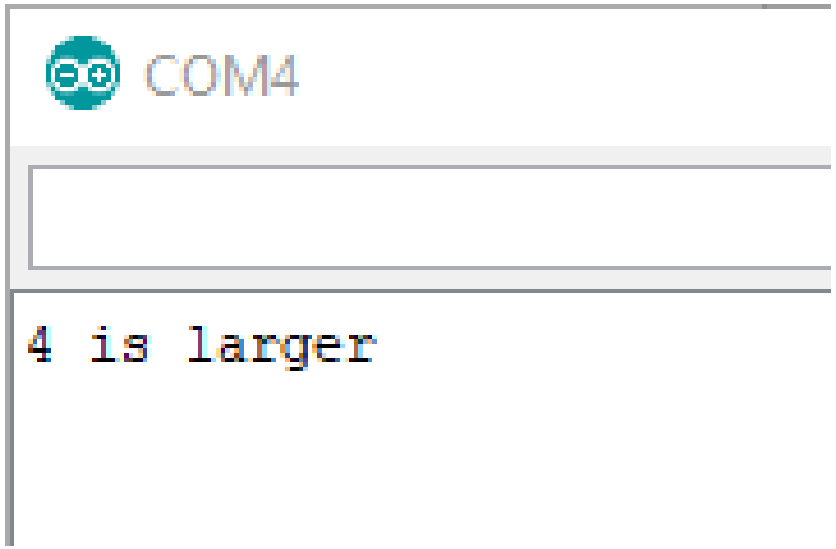
Returning a Value from a Function

- What happens if we call the function:

  larger();

```
void setup() {
 ✓ int num1 =2, num2 = 4, lNum = 0;

 ✓ Serial.begin(9600);

 ✓ lNum = larger(num1, num2);

 ✓ Serial.print(lNum);
 ✓ Serial.println(" is larger");
}
```





Arduino Functions Definitions

returnType functionName(type parameter1, type parameter2, …){

    Function body

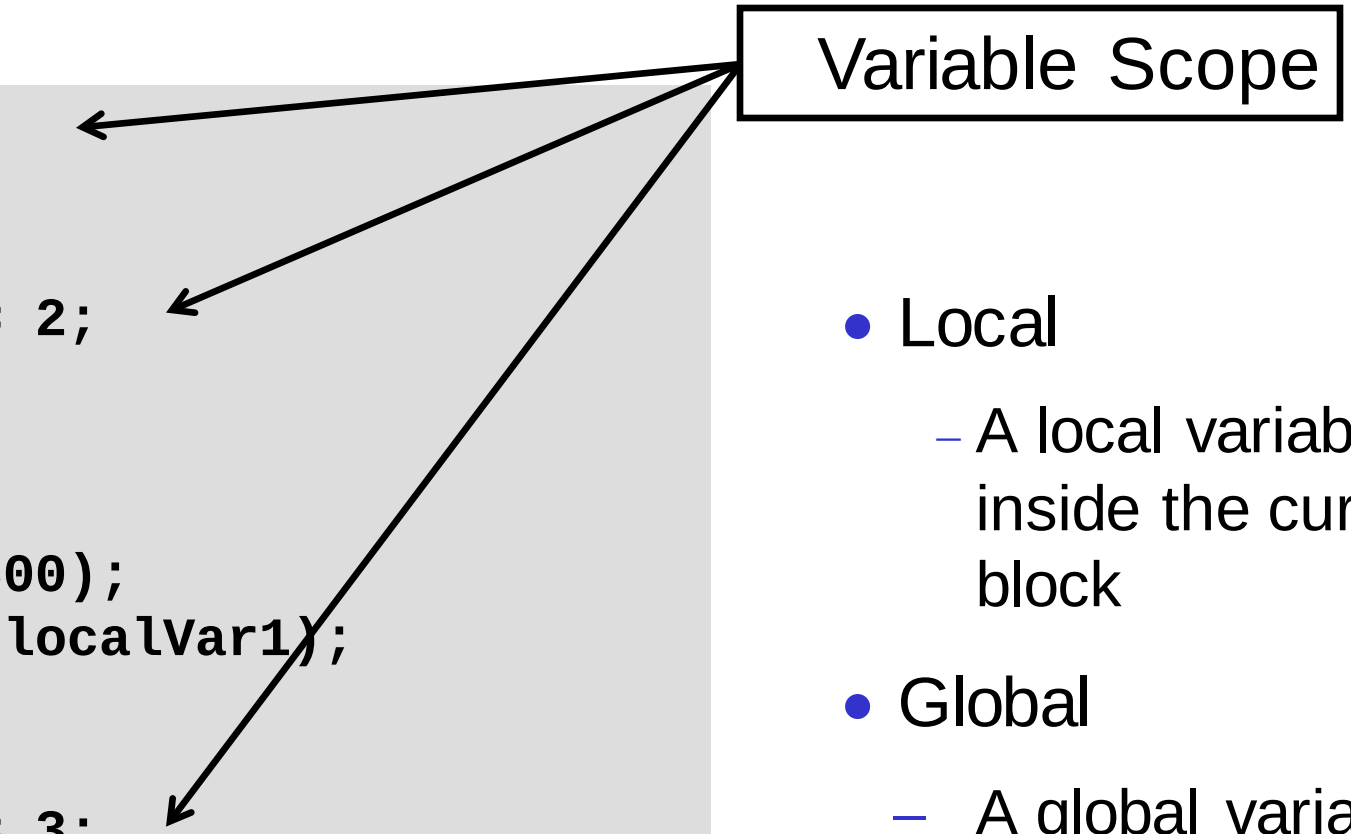
    Optionally return value

}

```
void setup() {
   Serial.begin(9600);
   DashedLine();
   Serial.println("| Program Menu |");
   DashedLine();
}
void loop() {
}
void DashedLine() {
   Serial.println("----------------");
}
```

Variable scope

```
int globalVar = 3;

void setup() {
   int localVar1 = 2;

   globalVar =4;

   Serial.begin(9600);
   Serial.println(localVar1);
}
void loop() {
   int localVar2 = 3;

   Serial.println(localVar2);
   Serial.println(globalVar);
}
```



- Local
  - A local variable is only visible inside the current, innermost block
- Global
  - A global variable is visible in the *whole* compilation unit, from the line of declaration to the end of file

Variable scope

```
int i = 3;

void setup() {
   Serial.begin(9600);
   Serial.println(i);
}
void loop() {
   int a = 0;
   a++;
   Serial.println(a);
}
```

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a = 0;
   int b = 2;

   for( a = 0; a < 3; a++){
      Serial.println(b);
   }
}
```

Variable scope

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a = 0;
   int b = 2;

   for( a = 0; a < 3; a++){
      Serial.println(b);
   }
}
```

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a = 0;

   for( a = 0; a < 3; a++){
      int b = 2;
      Serial.println(b);
   }
   Serial.println(b);
}
```

Variable scope

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a = 0;
   int b = 2;
   Serial.println( add(a, b) );
}

int add(int x, int y){
   Serial.print(x);
   Serial.print("+");
   Serial.print(y);

   return (x + y);
}
```

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a = 0;
   int b = 2;
   Serial.println( add(a, b) );
}

int add(int x, int y){

   Serial.print(a);

   Serial.print(x);
   Serial.print("+");
   Serial.print(y);

   return (x + y);
}
```