Parameter passing methods

- Most programming languages provides **two common** way to pass parameters to a function/method.

  Java however, provides only one: pass-by-value....

  So we have to resort to C++ to illustrate this concept.

- If a programming language does provide more than 1 way to pass parameters, then the language will specify a way to allow the programmer to specify his **choice**

  In some language (like in C++), the spefication can be using **a very minor symbol**...

  But the **meaning** is significantly changed and you will see a **dramatic** change in the **assembler program**

- The **two most common** way to pass parameters are:

  - **Pass-by-value**: the **value** of the actual parameter is copied to the parameter variable.
  - **Pass-by-reference**: the **reference** (**address**) of the actual parameter is copied to the parameter variable.

- NOTE:

  - The mechanism for passing parameters **has not** change.
  - In other words: the caller and callee functions will still have to agree on the **location** to pass the parameter
  - In addition, the caller and callee function must also agree on **how** the parameter is passed - pass-by-value or pass-by-reference

Pass-by-value

- The caller must copy the **value** of the actual parameter to the agreed location for the parameter
- The callee must use the parameter **assuming** that the parameter contains a **value**
- Example of pass-by-value in C++: click here

  NOTE: get a copy, compile it and **run** it to see the effect !
  The variable **i** in main() **does NOT** get incremented !

- Here is what happens in pass-by-value exposed in Assembler code:

```
Assume the following agreement: parameter passed in D0
                                parameter passed by value

C++ code:                            Assembler code:
---------                            --------------

main()                          main:
```

```
{                                       MOVE.L  i, D0     // pass value
   int i;                                                 // of var i

   func(i);  // pass value              BSR     func ------+
             // of var i                ....               |
}                                                         |
                                                          |
void func( int x )                 func: ADD.L   #1, D0 <---+
{                                                        // Assume value
   x = x + 1;                                            // is pass
}                                       RTS
```

- You can clearly see in the assembler code why the variable **i** in main() did not get incremented.

  (My personal experience has been that after I learned assembler programming, I understood the concepts in high level programming languages much better....)

- When passing parameters, always pass a parameter by value, unless you want the function to **update** the variable...

- Example of a useful pass-by-value case: sum of squares

```
// func(x,y) computes x^2 + y^2
int func(int x, int y)
{
    x = x + 1;
    y = y + 1;
    return(x*x + y*y);
}

main()
{
   int a, b, c;
   int i, j, k;

   c = func(a,b);
   k = func(i,j);
}
```

  - First agree on where to pass the parameters:
    - param 1: in D0
    - param 2: in D1
  - Next agree on where to pass the return value back:
    - return value: in D7
  - Now we can write the program...

```
main:
      MOVE.L a, D0      // Pass param 1
      MOVE.L b, D1      // Pass param 2
      BSR    func

      MOVE.L D7, c      // c = func(a,b)

      MOVE.L i, D0      // Pass param 1
      MOVE.L j, D1      // Pass param 2
      BSR    func
```

```
        MOVE.L D7, k      // k = func(i,j)
        ....
        ....

func:
        ADD.L  #1, D0     // x = x + 1 (because D0 contains the value of x)
        ADD.L  #1, D1     // y = y + 1 (because D1 contains the value of y)

        MULS   D0, D0     // x*x
        MULS   D1, D1     // y*y
        ADD.L  D1, D0     // x*x + y*y

        MOVE.L D0, D7     // Put return value
                          // in the agreed location
        RTS
```

  - Here is the code that you can run: click here

- Here is my CS170 lecture notes on **Pass-by-value** if you want to review: click here

---