

Function Call

• A function is "called" when you want to use the algorithm that was implemented in the function

• Here is how the main function would call computeFactorial:

int main(void)
{

int fact; //Factorial result

int val = 5;

fact = computeFactorial(val); //function call!

cout << "Fact. of " << val << " is: " << fact << endl;

return (0);
}

Fact. of 5 is: 120

```
Function Prototype

• A function prototype "declares a function"

• Provides user with information about the function.

- Name of function, what values (and types) need to be passed in, what type to expect the function to return

• Here is a function prototype for a factorial function

int computeFactorial(int num);

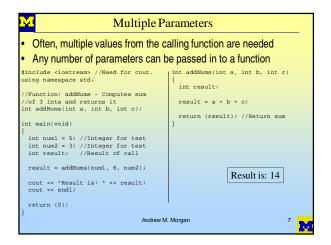
• The function is called "computeFactorial"

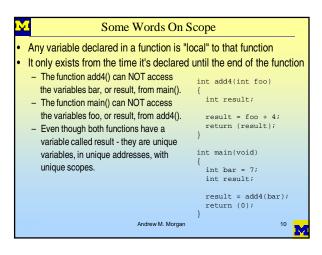
- Takes in one integer value from the calling function as a parameter

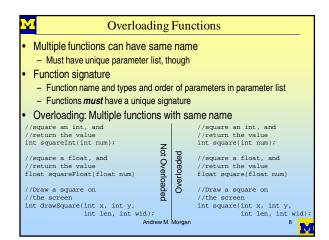
- Returns an integer value to the calling function

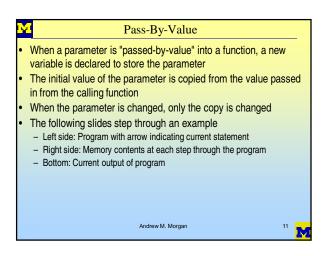
• Note: The function name should be descriptive of its purpose!
```

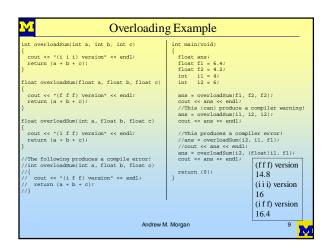
```
Complete Function Program
  Here is the complete program layout
#include <iostream> //Need for cout.
using namespace std;
                                            int computeFactorial(int num) //header
                                              int result = 1; //Value to return..
//Function: computeFacorial - Computes
//factorial of num and returns it.
int computeFactorial(int num); //proto
                                                              //Loop variable
                                             for (i = 1; i <= num; i++)
int main(void)
                                                result *= i;
 int fact; //Factorial result
int val = 5;
                                             return (result); //Return factorial
 fact = computeFactorial(val); //call
 Fact. of 5 is: 120
 return (0);
  Note: The order shown is important
```

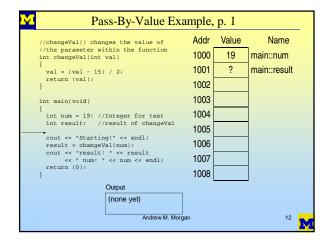


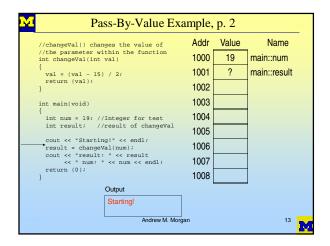


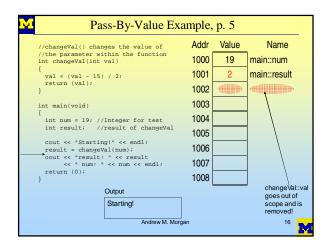


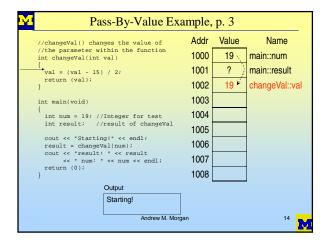


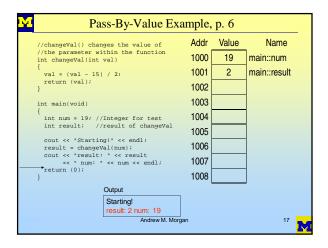


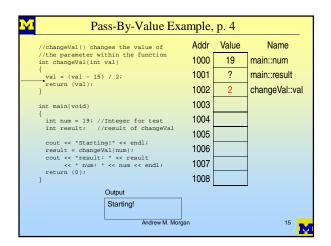


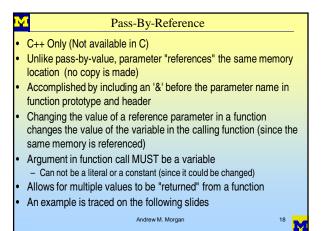


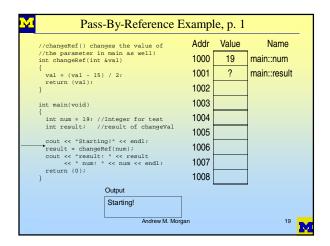


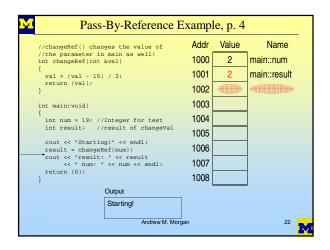


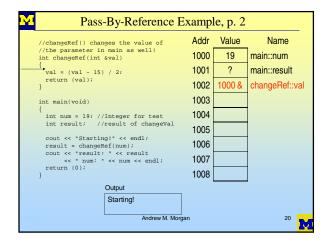


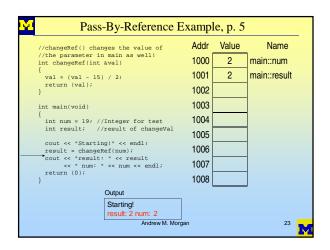












```
Pass-By-Reference Example, p. 3
                                            Addr
                                                     Value
                                                                  Name
 //changeRef() changes the value of
 //the parameter in main as well!
int changeRef(int &val)
                                            1000
                                                             main::num
                                            1001
  val = (val - 15) / 2;
return (val);
                                                             main::result
                                            1002
                                                    1000 &
                                                             changeRef::val
                                             1003
 int main(void)
                                             1004
  int num = 19; //Integer for test
int result; //result of changeVal
1005
                                            1006
                                             1007
                                             1008
                    Starting!
                              Andrew M. Morgan
                                                                      21
```

```
Swap Example, Multiple Reference Params

void swap(int &a, int &b) //Pass-by-reference!
{
  int temp;
  temp = a;
  a = b;
  b = temp;
}

int main(void)
{
  int n1 = 5;
  int n2 = 10;
  cout << "Before swap - n1: " << n1 << " n2: " << n2 << endl;
  swap(n1, n2);
  cout << "After swap - n1: " << n1 << " n2: " << n2 << endl;
  return (0);
}

Before swap - n1: 10 n2: 5

Andrew M. Morgan</pre>
```

