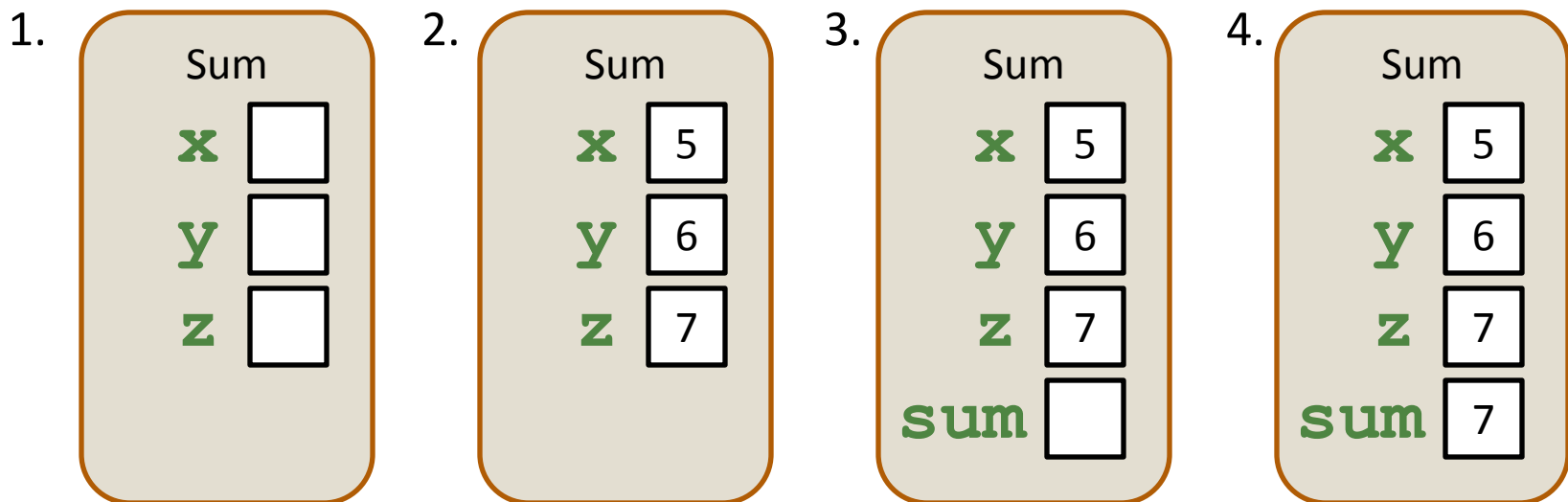# Functions, Variables and Memory

- Each function has its own memory space
  - Including `main`
  - All variables and parameters declared in a function refer to memory *allocated* in that space
  - When a function ends, its variables are deallocated

```
double sum_three( double x, double y, double z )
{
        double sum;
        sum = x + y + z;
        return sum;
}
…
sum = sum_three( 5, 6, 7 );
```

Functions, Variables and Memory

```
sum = sum_three( 5, 6, 7 );
```

1. Allocate memory for formal parameters
2. Assign actual parameter values
3. Allocate memory for declared variable sum
4. Calculate the sum
5. Return the sum (all memory de-allocated)

1.
Sum	
x	
y	
z	

2.
Sum	
x	5
y	6
z	7

3.
Sum	
x	5
y	6
z	7
sum	

4.
Sum	
x	5
y	6
z	7
sum	7

Functions, Variables and Memory

- Functions cannot use variables declared in another function

  – They are *out of scope*

- Variables with the same name in different functions do not refer to the same memory

Pass by Value

- By default, parameters are passed to a function *by value*

  – The value of the actual parameters are copied into the space allocated for the formal parameters

  – Each formal parameter has its own copy of the data in the function memory space

- Inside the function

  – Parameters passed by value are used to manipulate the data stored in the function memory space
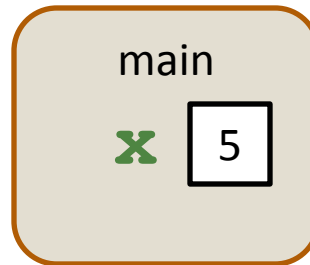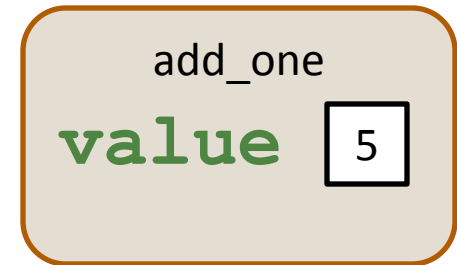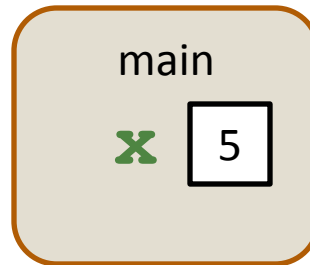
Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
   value = value + 1;
   return value;
}

int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
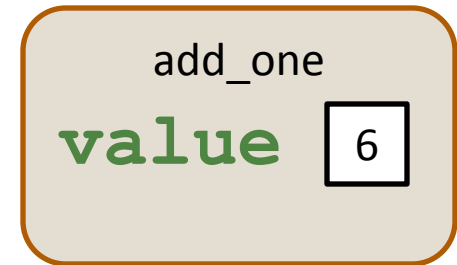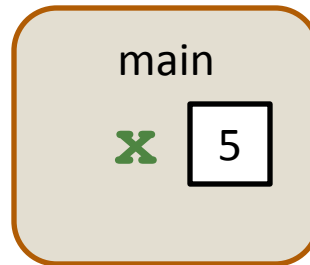
main

x 5

Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```
int add_one( int value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```

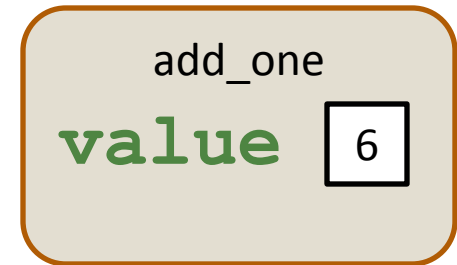main

x 5

add_one

value 5

Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
    value = value + 1;
    return value;
}

int main()
{
    int x = 5;
    add_one( x );
    cout << x << endl;

    return 0;
}
```
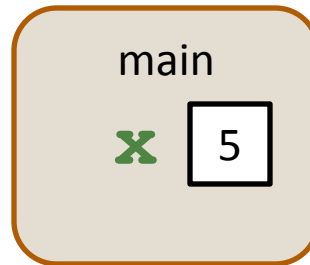
main

x 5

add_one

value 6

Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
    value = value + 1;
    return value;
}

int main()
{
    int x = 5;
    add_one( x );
    cout << x << endl;

    return 0;
}
```
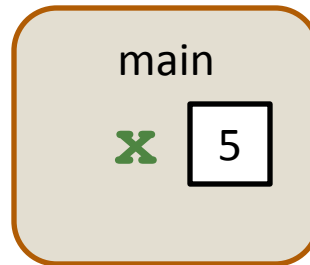
main

x 5

add_one

value 6

Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
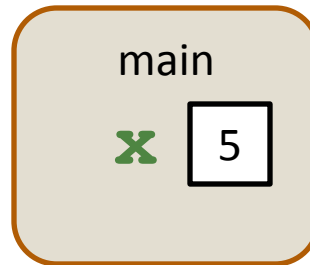
main

x 5

Value Parameters

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
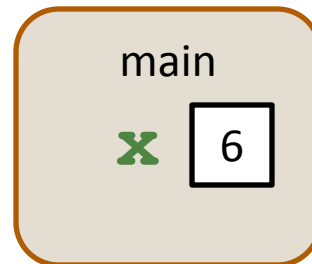
main

x 5

Setting the Return Value

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   x = add_one( x );
   cout << x << endl;

   return 0;
}
```
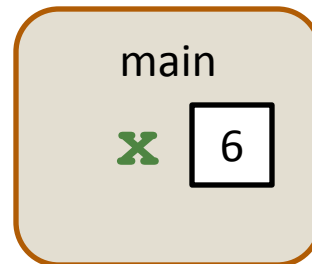
main

x 6

Setting the Return Value

- For value parameters:
  - Copy the value of the corresponding actual parameter
  - Manipulate that copy in the function's memory space

```cpp
int add_one( int value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   x = add_one( x );
   cout << x << endl;

   return 0;
}
```
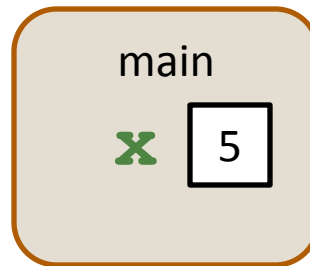
main

x 6

Reference Parameters

- For reference parameters:
  - Copy the *address* of the corresponding actual parameter
  - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
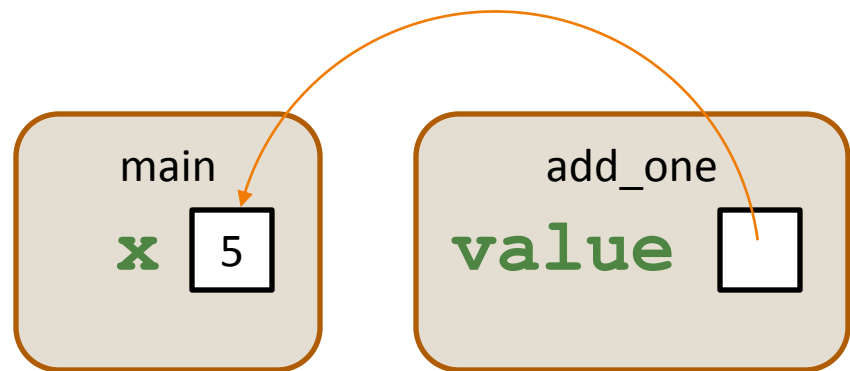
main

x | 5 |

Reference Parameters

- For reference parameters:
  - Copy the *address* of the corresponding actual parameter
  - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
   value = value + 1;
   return value;
}

int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
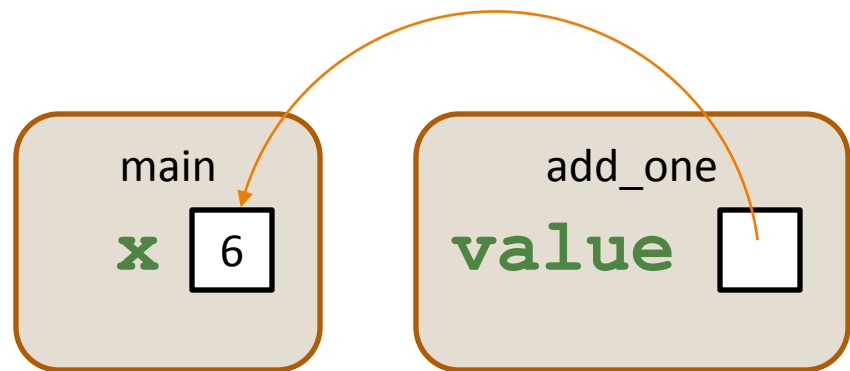
Reference Parameters

- For reference parameters:
    - Copy the *address* of the corresponding actual parameter
    - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
    value = value + 1;
    return value;
}

int main()
{
    int x = 5;
    add_one( x );
    cout << x << endl;

    return 0;
}
```
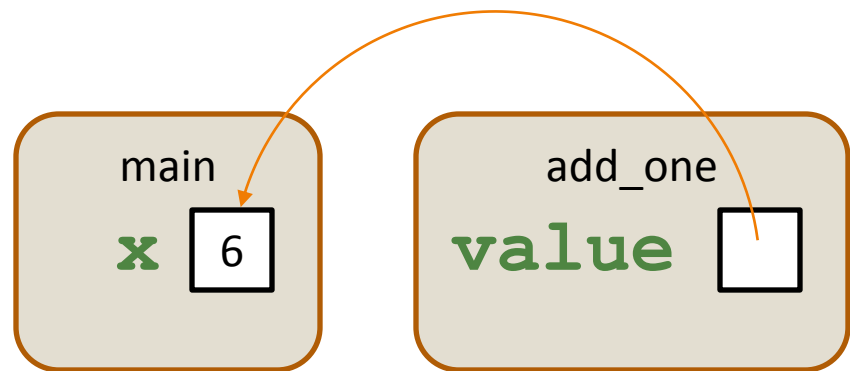
Reference Parameters

- For reference parameters:
  - Copy the *address* of the corresponding actual parameter
  - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
    value = value + 1;
→   return value;

}


int main()
{
    int x = 5;
⇒   add_one( x );
    cout << x << endl;

    return 0;

}
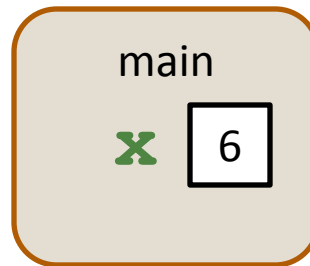```

main
**x** 6

add_one
**value** ☐

# Reference Parameters

- For reference parameters:
  - Copy the *address* of the corresponding actual parameter
  - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```
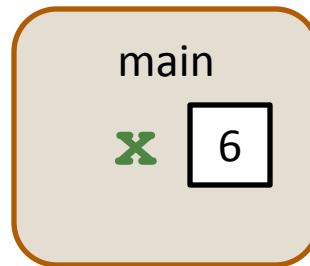
main

x | 6

Reference Parameters

- For reference parameters:
  - Copy the *address* of the corresponding actual parameter
  - Manipulate the data at that address in the calling function's memory space

```cpp
int add_one( int& value )
{
   value = value + 1;
   return value;
}


int main()
{
   int x = 5;
   add_one( x );
   cout << x << endl;

   return 0;
}
```

main

x 6

Exercise

- What are the values of num1 and num2 after this code executes?

```cpp
void aFunction( int a, int &b )
{
    b = a * 2;
    a = b + 1;
    b = a;
}


int main()
{
    int num1 = 3;
    int num2 = 4;
    aFunction( num1, num2 );

    return 0;
}
```