

Functions in C-Language



Hoon Yoo, Ph.D.

Understanding Functions

- Example of C program

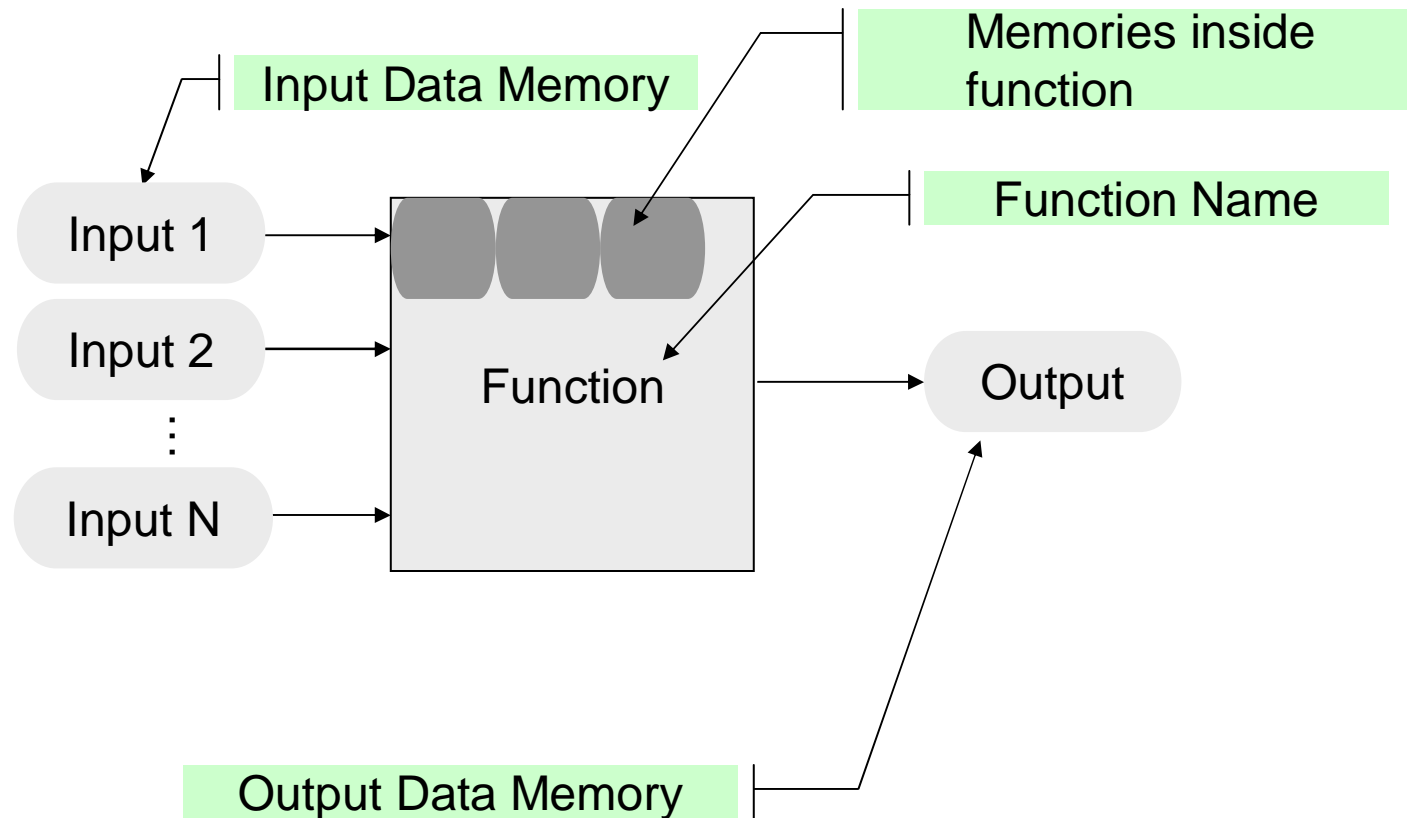
ExampleB.c

```
Function Prototype → myTest();  
main()  
{  
    myTest();  
}  
User Defines → myTest()  
{  
}  
}
```

<Source Code>

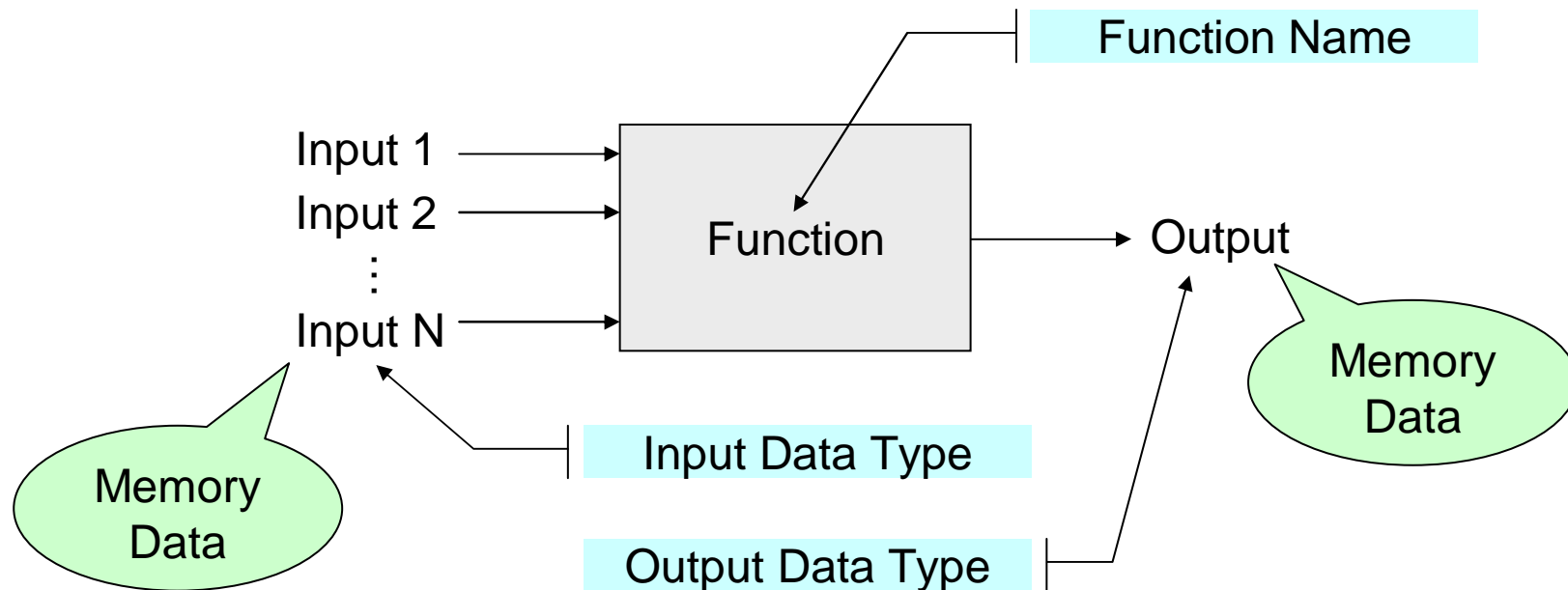
Understanding Functions

- Function Interface
 - Input, output



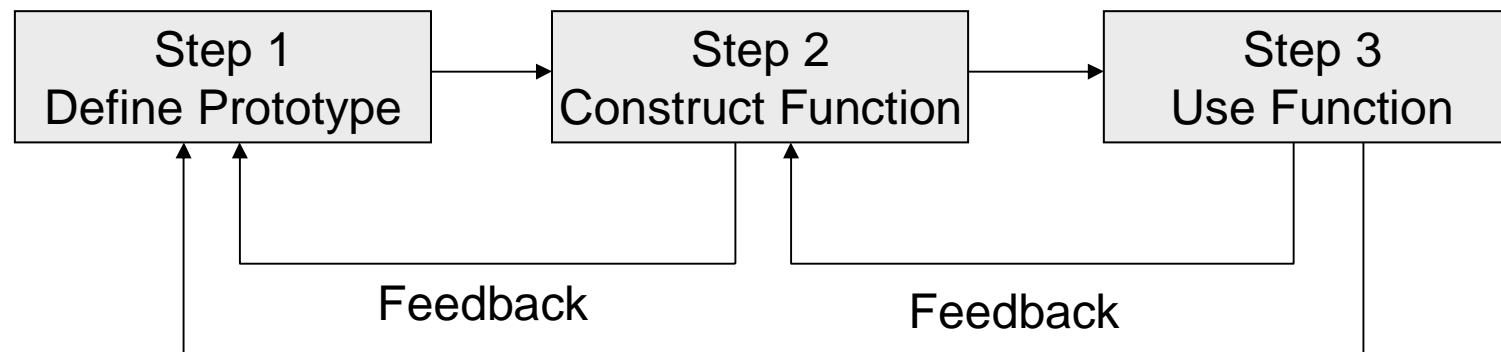
Understanding Functions

- A function has many input parameters and one output parameter
- Key points
 - Function name, input and output data type



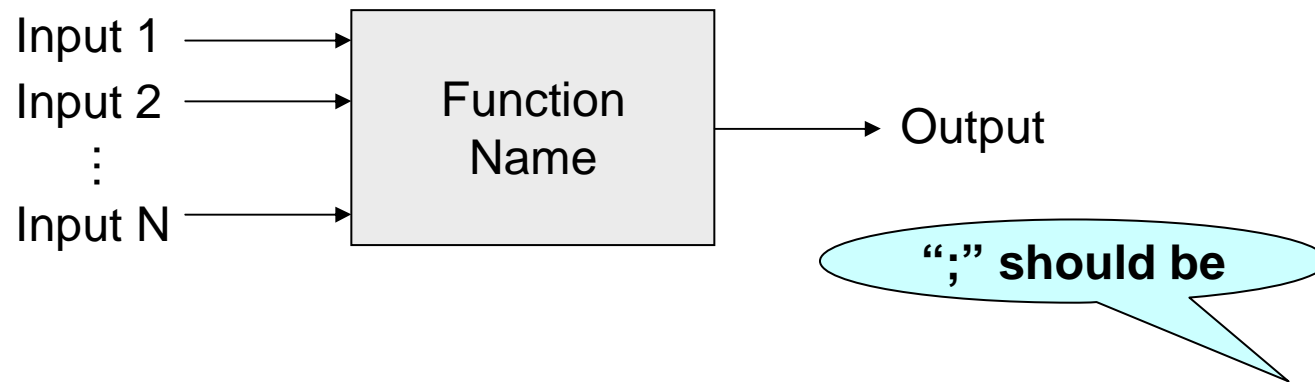
Understanding Functions

- Understand three steps of handling functions in C language
 - Step 1: Define function prototypes (Design I/O)
 - Step 2: Construct functions
 - Step 3: Use functions

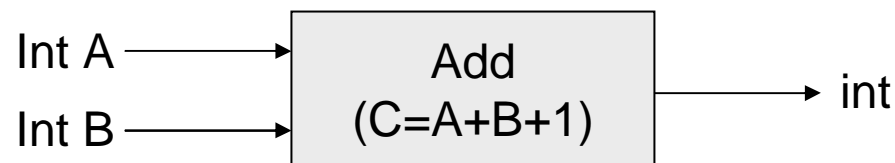


Handling Functions

- Function prototypes



`OutputDataType FunctionName(InputDataType1, ..., InputDataTypeN);`

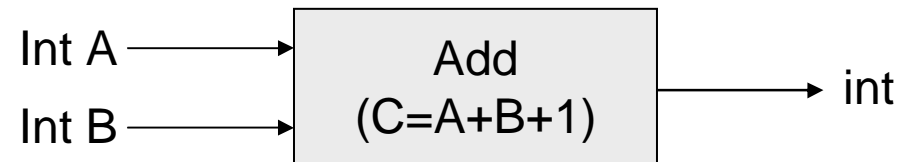


Examples of DataType
1.Int: 4bytes integer
2.float: 4bytes real

`int Add(int, int);`
`int Add(int A, int B);`

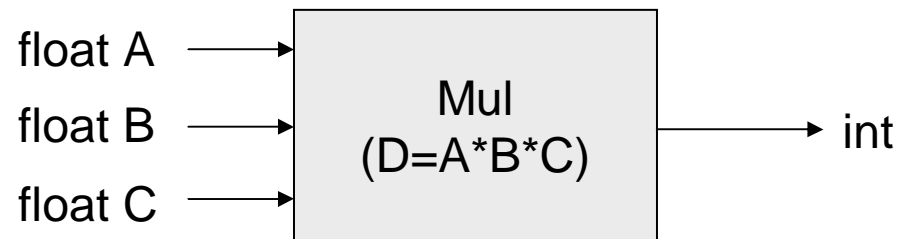
Handling Functions

- Examples for function prototypes



`int Add(int, int);`

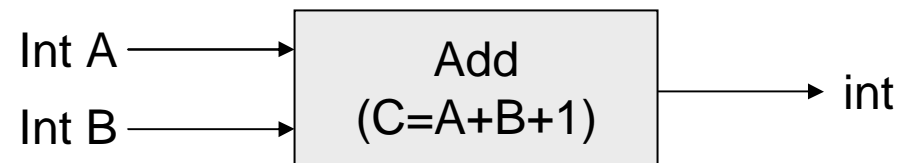
`int Add(int A, int B);`



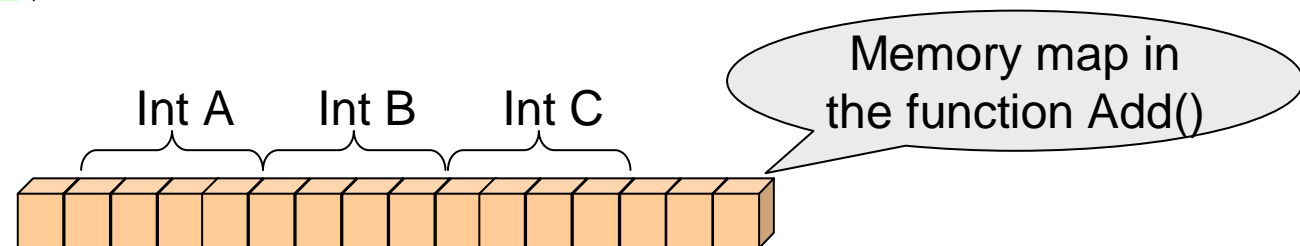
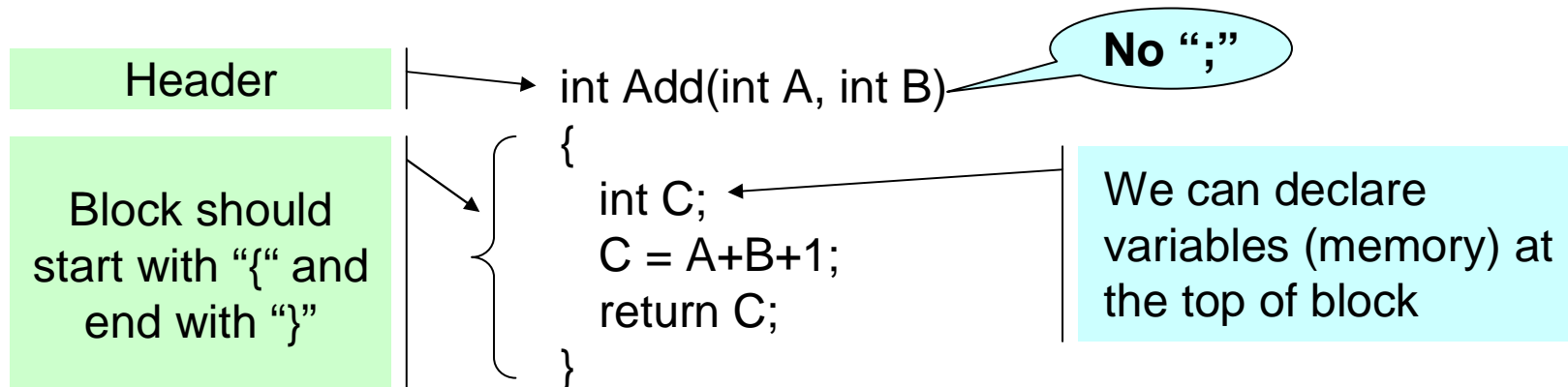
Prototype ?

Handling Functions

- Construct functions

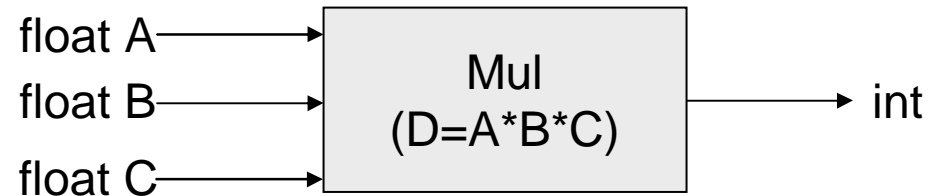


`int Add(int A, int B);`



Handling Functions

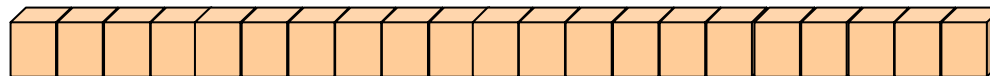
- Construct functions



```
int Mul(float A, float B, float C);
```

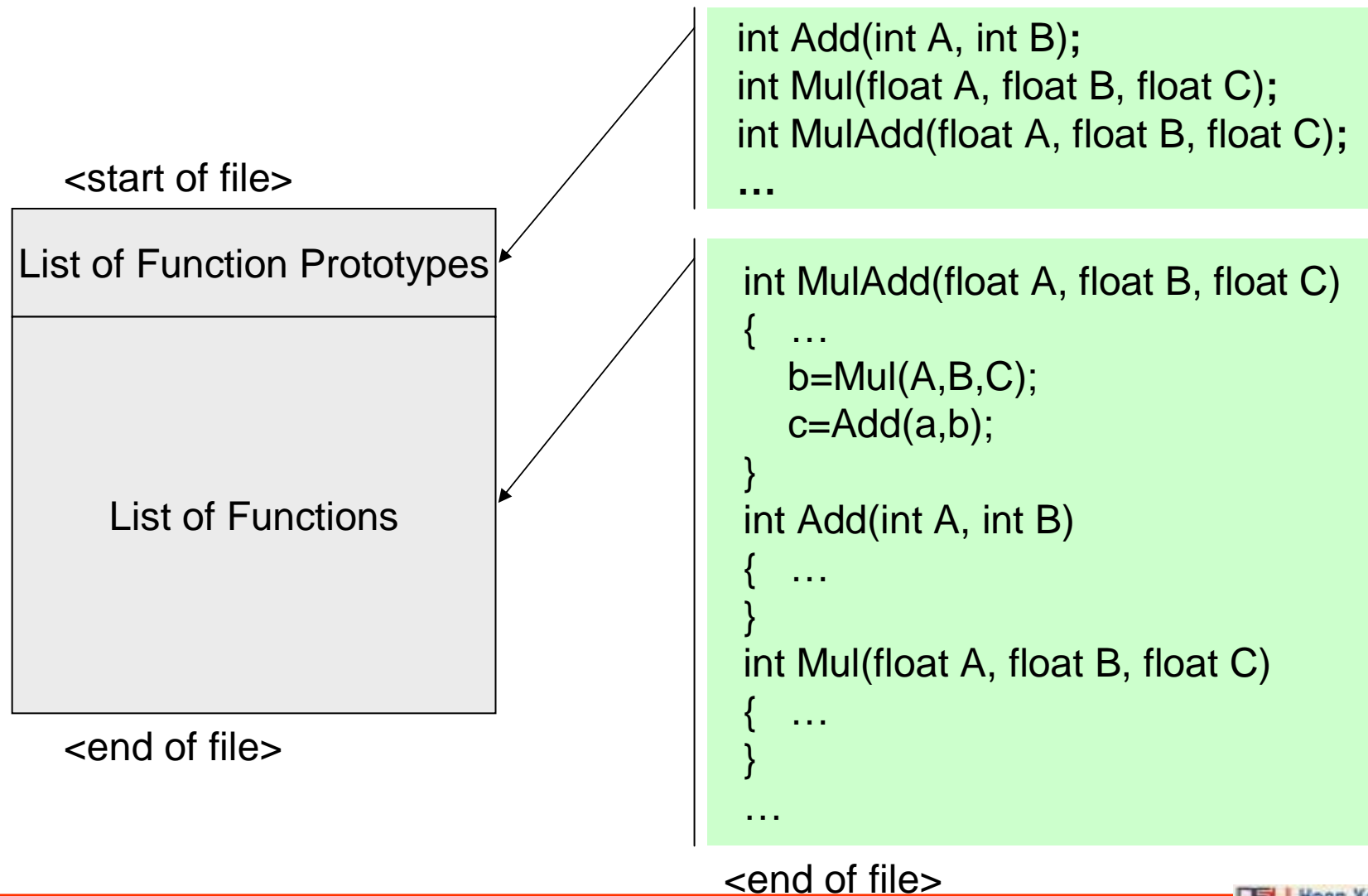
```
int Mul(float A, float B, float C)
{
    ?
}
```

Memory map in the
function Mul() ?



Using Functions

- Layout of a C source file



Summary

- Functions are key in C-language
- Three steps when functions are used

