

CSCI 200: Foundational Programming Concepts & Design

Lecture 09



Overloading Functions
The Call Stack
Pointers

[Download starter code](#)

Previously in CSCI 200



- *.h – function declarations
- *.cpp – function definitions
- Makefile
 - SRC_FILES : lists ONLY *.cpp files
 - Dependencies: specify *.cpp and *.h that each object file depends upon
 - **make depend** will autopopulate (if available)

Questions?



??

Learning Outcomes For Today



- Explain the concept of local & global scope when functions are used within a program.
- Implement various techniques to trace & debug a program.
- Define an overloaded function and recite common usages for overloaded functions.

On Tap For Today



- Overloading Functions
- Function Scope
- Pointers
- Practice

On Tap For Today



- Overloading Functions
- Function Scope
- Pointers
- Practice

Overloaded Functions (aka Function Overloading)



- Two or more functions can have the same name IFF
 - The number of parameters differ
 - OR
 - The data type of parameters differ
- Functions must have a different function signature

Quick Quiz: What is the Output?



```
int func(double x)      { return 1; }
int func(int x)         { return 2; }
int func(int x, int y) { return 3; }
int func()              { return 4; }

int main() {
    cout << func(1.5) << endl;
    cout << func(1, 5) << endl;
    cout << func() << endl;
    cout << func(1) << endl;
    return 0;
}
```

1
3
4
2

Quick Quiz: What is the Output?



```
int func(char x)          { return 1; }
int func(int x)           { return 2; }
int func(int x, int y) { return 3; }
int func()                { return 4; }
int main() {
    cout << func(1.5) << endl;
    cout << func(1, 5) << endl;
    cout << func() << endl;
    cout << func(1) << endl;
    cout << func('c') << endl;
    return 0;
}
```

Quick Quiz: What is the Output?



```
int func(char x)          { return 1; }
int func(int x)           { return 2; }
int func(int x, int y) { return 3; }
int func()                { return 4; }

int main() {
    cout << func(1.5) << endl;    // compiler error! call is ambiguous
                                   // candidates are funcA(char), funcA(int)

    cout << func(1, 5) << endl;
    cout << func() << endl;
    cout << func(1) << endl;
    cout << func('c') << endl;
    return 0;
}
```

Quick Quiz: What is the Output?



```
int func(int x, double y) { return 1; }  
int func(double x, int y) { return 2; }  
  
int main() {  
    cout << func(1.5, 2) << endl;  
    cout << func(2, 1.5) << endl;  
    return 0;  
}
```

2
1

Quick Quiz: What is the Output?



```
int func(int x, double y) { return 1; }  
char func(int x, double y) { return '1'; }
```

```
int main() {  
    cout << func(1.5, 2) << endl;  
    cout << func(1.5, 2) << endl;  
    return 0;  
}
```

Quick Quiz: What is the Output?



```
int func(int x, double y) { return 1; }

char func(int x, double y) { return '1'; } // compiler error!
// functions cannot differ in return type only

int main() {
    cout << func(1.5, 2) << endl;
    cout << func(1.5, 2) << endl;
    return 0;
}
```

Actual Example



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

int main() {
    cout << max(1, 2) << endl;
    cout << max(1.5, 2.5) << endl;
    return 0;
}
```

2
2.5

Side Note: Ternary Operator



```
int max(int x, int y) { return (x > y ? x : y); }
```

```
double max(double x, double y) { if(x > y) return x; else return y; }
```

```
int main() {  
    cout << max(1, 2) << endl;  
    cout << max(1.5, 2.5) << endl;  
    return 0;  
}
```

2
2.5

Stay Tuned!



- We'll see practical examples of overloaded functions coming up

On Tap For Today



- Overloading Functions
- Function Scope
- Pointers
- Practice

Call Stack



- Each function call appends to the stack:
 - Address of execution (File and line number)
 - Local variables (includes parameters)
- Each appendage is a “stack frame”
 - Frames create part of our scope
- Use debugger to investigate the call stack

Program Entry Point: main()



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```

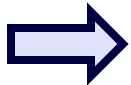
Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c			
0x40960020			
0x40960024			
0x40960028			
0x4096002c			
0x40960030			
0x40960034			
0x40960038			
0x4096003c			

Evaluate main()



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```



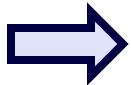
Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024			
0x40960028			
0x4096002c			
0x40960030	a	4	
0x40960034			
0x40960038			b 0x4096001c
0x4096003c			a 0x40960030

Pass by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```



Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c		
0x40960028			
0x4096002c			
0x40960030	a	4	
0x40960034			c 0x40960024
0x40960038			b 0x4096001c
0x4096003c			a 0x40960030

Pass by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	y	3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c		
0x40960028			
0x4096002c			y 0x40960014
0x40960030	a	4	x 0x40960038
0x40960034			c 0x40960024
0x40960038	x	4	b 0x4096001c
0x4096003c			a 0x40960030

Evaluate add()



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	y	3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c		
0x40960028	a		a 0x40960028
0x4096002c			y 0x40960014
0x40960030	a	4	x 0x40960038
0x40960034			c 0x40960024
0x40960038	x	4	b 0x4096001c
0x4096003c			a 0x40960030

Evaluate add()



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	y	3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c		
0x40960028	a	7	a 0x40960028
0x4096002c			y 0x40960014
0x40960030	a	4	x 0x40960038
0x40960034			c 0x40960024
0x40960038	x	4	b 0x4096001c
0x4096003c			a 0x40960030

Return by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	y	3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c		
0x40960028	a	7	a 0x40960028
0x4096002c			y 0x40960014
0x40960030	a	4	x 0x40960038
0x40960034			c 0x40960024
0x40960038	x	4	b 0x4096001c
0x4096003c			a 0x40960030

Return by Value



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```

Address	Identifier	Value	Stack
0x40960014		3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	
0x40960028		7	
0x4096002c			
0x40960030	a	4	
0x40960034			c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c			a 0x40960030

Pass by Value



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```

Address	Identifier	Value	Stack
0x40960014		3	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	
0x40960028	d	7	
0x4096002c			
0x40960030	a	4	d 0x40960028
0x40960034			c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c			a 0x40960030

Pass by Value




```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```


```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	x	5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	
0x40960028	d	7	y 0x4096003c
0x4096002c			x 0x40960014
0x40960030	a	4	d 0x40960028
0x40960034			c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c	y	8	a 0x40960030

Evaluate add()



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```



```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```

Address	Identifier	Value	Stack
0x40960014	x	5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	a 0x40960034
0x40960028	d	7	y 0x4096003c
0x4096002c			x 0x40960014
0x40960030	a	4	d 0x40960028
0x40960034	a		c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c	y	8	a 0x40960030

Evaluate add()



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```

Address	Identifier	Value	Stack
0x40960014	x	5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	a 0x40960034
0x40960028	d	7	y 0x4096003c
0x4096002c			x 0x40960014
0x40960030	a	4	d 0x40960028
0x40960034	a	13	c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c	y	8	a 0x40960030

Return by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014	x	5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	a 0x40960034
0x40960028	d	7	y 0x4096003c
0x4096002c			x 0x40960014
0x40960030	a	4	d 0x40960028
0x40960034	a	13	c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c	y	8	a 0x40960030

Return by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014		5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	
0x40960028	d	13	
0x4096002c			
0x40960030	a	4	d 0x40960028
0x40960034		13	c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c		8	a 0x40960030

Return by Value



```
int add( int x, int y ) {
    int a;
    a = x + y;
    return a;
}
```

```
int main() {
    int a(4), b(3);
    int c = add( a, b );
    int d = add( 5, 8 );
    return 0;
}
```

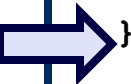
Address	Identifier	Value	Stack
0x40960014		5	
0x40960018			
0x4096001c	b	3	
0x40960020			
0x40960024	c	7	
0x40960028	d	13	
0x4096002c			
0x40960030	a	4	d 0x40960028
0x40960034		13	c 0x40960024
0x40960038		4	b 0x4096001c
0x4096003c		8	a 0x40960030

Program Terminates



```
int add( int x, int y ) {  
    int a;  
    a = x + y;  
    return a;  
}
```

```
int main() {  
    int a(4), b(3);  
    int c = add( a, b );  
    int d = add( 5, 8 );  
    return 0;  
}
```



Address	Identifier	Value	Stack
0x40960014		5	
0x40960018			
0x4096001c		3	
0x40960020			
0x40960024		7	
0x40960028		13	
0x4096002c			
0x40960030		4	
0x40960034		13	
0x40960038		4	
0x4096003c		8	

Running Debugger Part I



- `gdb .\HelloWorld.exe`
 - `b <#>`
 - `run`
 - `print <var>`
 - `info b`
 - `step`
 - `continue`
 - `kill`
 - `q`
- `lldb ./HelloWorld`
 - `b <#>`
 - `run`
 - `print <var>`
 - `br l`
 - `step`
 - `continue`
 - `kill`
 - `q`

Running Debugger Part II



- `gdb .\HelloWorld.exe`
 - `b <file.ext:#>`
 - `bt`
 - `info locals`
 - `info args`
 - `up`
 - `down`
 - `frame`
 - `frame #`
- `lldb ./HelloWorld`
 - `b <file.ext:#>`
 - `bt`
 - `frame variable`
 - `up`
 - `down`
 - `frame info`
 - `frame select #`
 - `target variable`

On Tap For Today



- Overloading Functions
- Function Scope
- Pointers
- Practice

Pointers



- Pointer denoted by _____
- Value of a pointer is _____

Types



- Clockwise winding order resolution
- What is the type of the identifier **x** in each

`int x;`

`double x;`

`int x(int a, int b);`

`float* x;`

`float** x;`

Pointer Values



- Value of a pointer is _____
- Value pointed at is _____

```
float* x;
```

```
float** x;
```

```
int (*x) (int, int);
```


On Tap For Today



- Overloading Functions
- Function Scope
- Pointers
- Practice

To Do For Next Time



- Procedural Programming Quiz Monday
 - Functions
- Exam 1 XC Due Monday
- Exam I in class on Wednesday
 - No Pointers on exam
- Keep working on Set2 – A2 to come next time