CSCI 200: Foundational Programming Concepts & Design Lecture 21

Reference

&

Memory

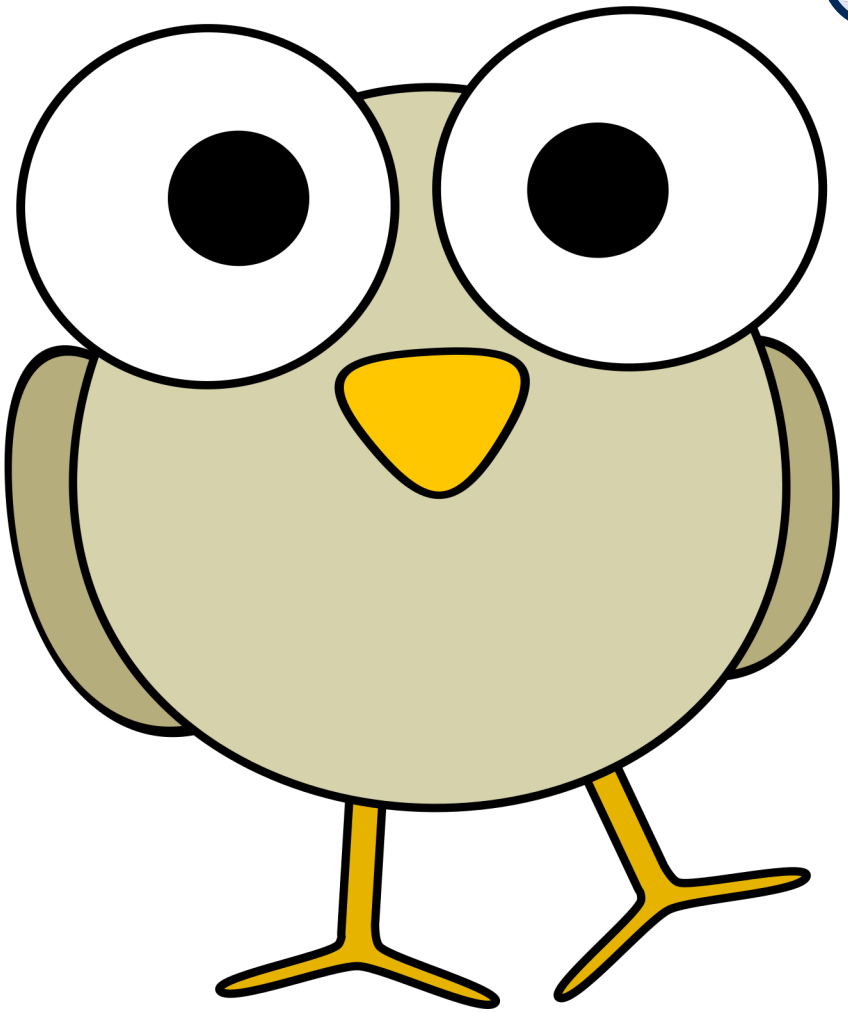
Complete Set3 Survey Now

Access Code: **sneetches**

Previously in CSCI 200

- Four uses of `const`
  - Variable modifier
  - Parameter modifier
  - Pointer modifier
  - Member function modifier

Questions?



Learning Outcomes For Today

- Explain the difference between pass-by-value and pass-by-reference. Draw a diagram of how each stores its parameters in memory.

On Tap For Today

- Reference
- Dynamic Memory Management
- Practice

On Tap For Today

- Reference
- Dynamic Memory Management
- Practice

Binary Operations

- Generally

```
lhs @ rhs
```

- Perform binary operation @ from the right hand side to the left hand side
- Rephrased

```
lvalue @ rvalue
```

`lvalue` vs. `rvalue`

- `lvalue` (formally locator value) is an object that represents some identifiable location in memory (i.e. has an address)

- `rvalue` is an object that does not represent some identifiable location in memory

Examples

```
int var, var2;    // allocate memory on the stack for two integer variables
var = 5;          // ok, place 5 into memory of var
var2 = var;       // ok, implicit lvalue-to-rvalue conversion for var
4 = var;          // error! Cannot assign var to 4.  4 is not an lvalue
(var + 1) = 4;    // error! (var + 1) is not an lvalue
```

Functions Revisited

```
void f(int bar) {          // pass-by-value
  bar = 3;
}
void g(int* pFoo) {        // pass-by-pointer
  *pFoo = 2;
}


int var;          // allocate memory on the stack
var = 5;          // ok, place 5 into memory of var
f(3);             // pass rvalue to f()
g(&var);          // pass lvalue to f()
```

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	

Program Entry Point: main()

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c			
0x40960020			
0x40960024			

Evaluate main()

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Evaluate main()

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Pass by Pointer

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Pass by Pointer

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020	pX	0x4096001c	pX 0x40960020
0x40960024			a 0x4096001c

Evaluate add_five()

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	9	
0x40960020	pX	0x4096001c	pX 0x40960020
0x40960024			a 0x4096001c

Evaluate main()

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	9	
0x40960020			
0x40960024			a 0x4096001c

Return by Value

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	9	
0x40960020			
0x40960024			a 0x4096001c

Program Terminates

```
void add_five( int* pX ) {
    *pX += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( &a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c		9	
0x40960020			
0x40960024			

Create Explicit `lvalue`

```
int var;
var = 5;
int& var2 = var;          // now assign lvalue to var2
                          // var2 backed by same memory as var
var2 = 6;                 // changes var as well!
```

Reference

- & - reference operator

```
int main() {

    int x = 4;
    int y = x;              // assign the value of x
    int& z = x;             // assign the reference of x

    cout << x << endl;     // print value of x - 4
    cout << &x << endl;    // print address of x - 0x4ab338cc

    cout << y << endl;     // print value of y - 4
    cout << &y << endl;    // print address of y - 0x5a23bbdf

    cout << z << endl;     // print value of z - 4
    cout << &z << endl;    // print address of z - 0x4ab338cc

    z = 5;
    cout << x << " "       // prints 5
         << y << " "       // prints 4
         << z << endl;     // prints 5

    return 0;

}
```

Precedence Table

Precedence	Operator	Associativity
1	Parenthesis: ( )	Innermost First
2	Postfix Unary Operators: a++ a-- a. p-> f()	Left to Right
3	Prefix Unary Operators: ++a --a +a -a !a (type)a &a *p new delete	Right to Left
4	Binary Operators: a*b a/b a%b	Left to Right
5	Binary Operators: a+b a-b	
6	Relational Operators: a<b a>b a<=b a>=b	
7	Relational Operators: a==b a!=b	
8	Logical Operators: a&&b	
9	Logical Operators: a||b	
10	Assignment Operators: a=b a+=b a-=b a*=b a/=b a%=b	Right to Left

Pass by Reference with &

- Instead of passing the value of an argument to the function, pass the argument's memory address to the function

```
void add_five( int& x ) { // int &x
    x += 5;
}
int main() {
    int a(4);
    cout << a << endl;    // 4
    add_five( a );
    cout << a << endl;    // 9
    return 0;
}
```

Program Entry Point: main()

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c			
0x40960020			
0x40960024			

Evaluate main()

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Evaluate main()

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Pass by Reference

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	4	
0x40960020			
0x40960024			a 0x4096001c

Pass by Reference

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a x	4	
0x40960020			x 0x4096001c
0x40960024			a 0x4096001c

Evaluate add_five()

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a x	9	
0x40960020			x 0x4096001c
0x40960024			a 0x4096001c

Evaluate main()

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	9	
0x40960020			
0x40960024			a 0x4096001c

Return by Value

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c	a	9	
0x40960020			
0x40960024			a 0x4096001c

Program Terminates

```
void add_five( int& x ) {
    x += 5;
}

int main() {
    int a(4);
    cout << a << endl;
    add_five( a );
    cout << a << endl;
    return 0;
}
```

Address	Identifier	Value	Stack
0x40960014			
0x40960018			
0x4096001c		9	
0x40960020			
0x40960024			

On Tap For Today

- Reference
- Dynamic Memory Management
- Practice

Storing Objects on the Free Store

- Use a pointer!

```
int *pNumCars = new int;
```

- * - indirection operator

- **new** – "Computer, allocate enough memory in the free store for one object and tell me the starting address where the object will be stored."

new

- **new** returns a pointer

```
int *pNumCars = new int;
```

- **pNumCars** is a pointer to an integer variable on the free store

`new` and `delete`

- `new`: allocates memory on the free store
- `delete`: returns used memory to the free store

```
int *pNumCars = new int;

delete pNumCars;
```

- Why do we need to return memory to the free store?

PBV / PBR / PBP

- What's the difference?

```
void f1(int x)   { x   = 3; }
void f2(int& y)  { y   = 3; }
void f3(int* pZ) { *pZ = 3; }


...
int x = 1, y = 1, z = 1;
f1(x);   // what is x?
f2(y);   // what is y?
f3(&z);  // what is z?
```

PBV / PBR / PBP

- What's the difference?

```
void g1(int* pY)  { pY = new int; }
void g2(int*& pZ) { pZ = new int; }

...
int* p1 = nullptr;
int* p2 = nullptr;
g1(p1);   // what does p1 point to?
g2(p2);   // what does p2 point to?
```

- What also happens with each of these?

Other Concerns

- What's happens?

```
void h1(int* pY)  { delete pY; pY = nullptr; }
void h2(int*& pZ) { delete pZ; pZ = nullptr; }

...
int* p1 = new int;
int* p2 = new int;
h1( p1 );
h2( p2 );
cout << p1 << endl;        // value?
cout << p2 << endl;        // value?
```

On Tap For Today

- Reference
- Dynamic Memory Management
- Practice

To Do For Next Time

- Final Project Proposals due this evening
- Get jump on next set of zyBooks