We are 183

L09: Week 6 - Monday

Due Soon

- Assignment 3: due Friday

- Exam 1: Next week!
  - Exam Review: Sunday 2/14 at 6pm in CHEM 1800

Last Time… on EECS 183

for loops
Nested loops
Strings
Header files

Syntax of *for* loop

Initialize

Test

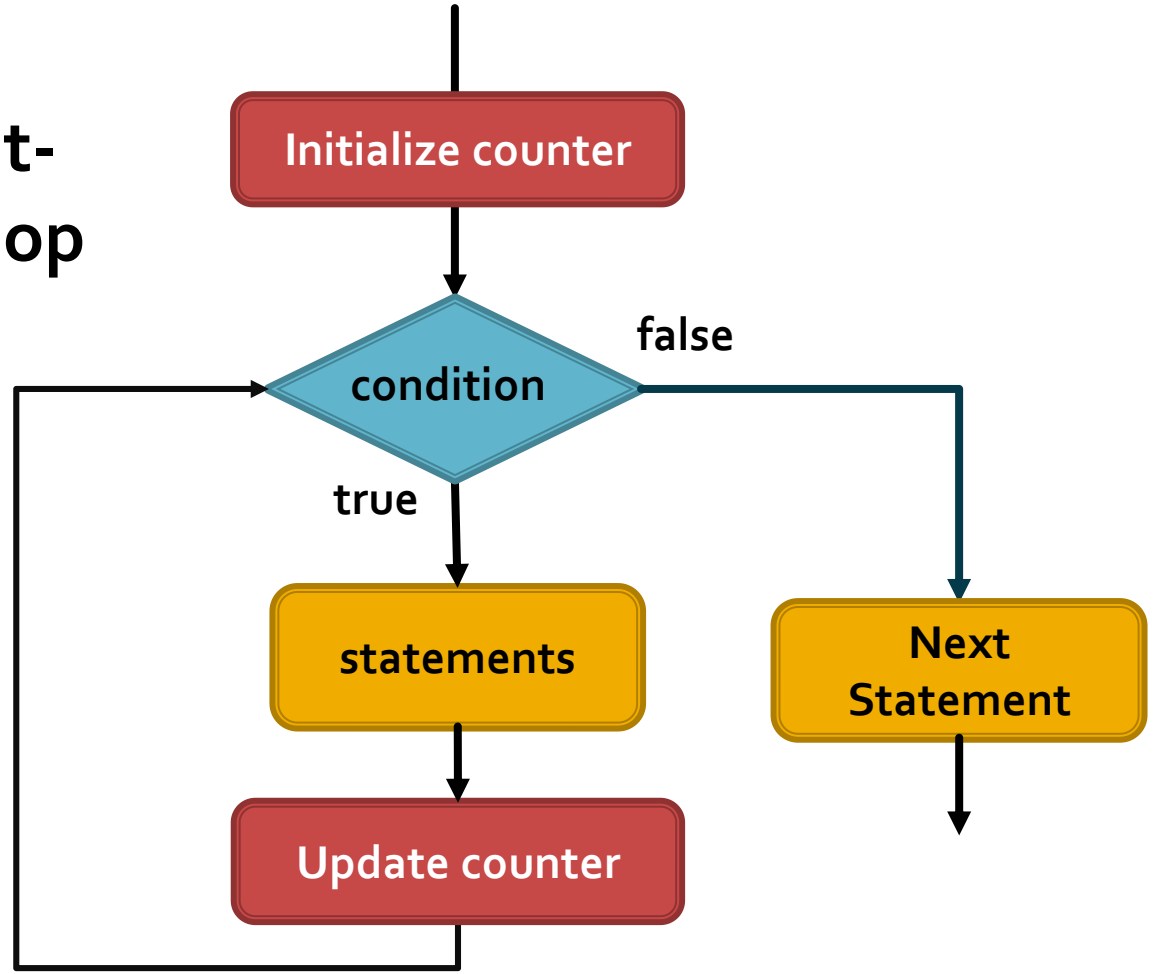
Update

```
for (int i = 0; i < 5; i++) {

    cout << i << endl;

}
```

Loop Body

Count Controlled Loops

Basic `for` loop is equivalent to count-controlled `while` loop



while loop vs. *for* loop

```
count = 1;
while (count <= 5) {
    square = count * count;
    cout << count << "  " << square << endl;
    count++;
}
```

```
for (count = 1; count <= 5; count++) {
    square = count * count;
    cout << count << "  " << square << endl;
}
```

i>Clicker #1

```
int number = 4;
int result = 1;

for (int i = 1; i <= number; i++) {
    result *= i;
}

cout << result;
```

What value is printed?

A. 4
B. 6
C. 24
D. None of the above

i>Clicker #1

```
int number = 4;
int result = 1;

for (int i = 1; i <= number; i++) {
    result *= i;
}

cout << result;
```

What value is printed?

A. 4
B. 6
C. 24
D. None of the above

Declaring Variable in *for* Initialization

```
int count = 1;
while (count <= 5) {
    square = count * count;
    cout << count << "  " << square << endl;
    count++;
}
```

Counter must be declared outside loop

Counter can be declared in for initialization

```
for (int count = 1; count <= 5; count++) {
    square = count * count;
    cout << count << "  " << square << endl;
}
```

Variable Scope

Scope of count

```
...
for (int count = 1; count <= 5; count++) {
    int square = count * count;
    cout << count << " " << square << endl;
}
...
```

i>Clicker #2

```
int number = 0;

for (int i = 1; i <= 3; i++) {
    number += i;
}

cout << i << number;
```

What is printed?

A. 30
B. 36
C. 46
D. None of the above

i>Clicker #2

```
int number = 0;

for (int i = 1; i <= 3; i++) {
    number += i;
}

cout << i << number;
```

i is out of scope

What is printed?

A. 30
B. 36
C. 46
D. None of the above

Nested *for* loops

Execution

```
for (int i = 1; i < 3; i++) {
    for (int j = 0; j < i; j++) {
        cout << '*';
    }

    cout << endl;
}
```

Output

Nested *for* loops

```
for (int i = 1; i < 3; i++) {
    for (int j = 0; j < i; j++) {
        cout << '*';
    }

    cout << endl;
}
```

Execution

Output

```
*

**
```

i>Clicker #3

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= i; j++) {
        cout << "Hello" << endl;
    }
}
```

How many times is "Hello" printed?

A. 1
B. 3
C. 6
D. None of the above

i>Clicker #3

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= i; j++) {
        cout << "Hello" << endl;
    }
}
```

How many times is "Hello" printed?

A. 1
B. 3
C. 6
D. None of the above

What kind of loop is best?

Is the loop count-controlled?

`for` is usually best

Is the loop event-controlled?

`while` should be used

A string is a sequence of chars

```
string str = "hello";
str[0] = 'y';
str[1] = 'o';
str[0] = 'p';
str = 'a' + str;
```

a	p	o	l	l	o
0	1	2	3	4	5

```
cout << str;
```

Console Output

```
apollo
```

Separate Source Files

Benefits of organizing code into separate functions:

- More readable
- More testable
- More reusable
- etc.

Same benefits for organizing groups of related functions into separate source files

stats.h

```
#ifndef STATS_H_
#define STATS_H_

// Statistical Functions

/**
 * Requires: variance >= 0.
 * Effects:  Computes the probability that a random sample
 *           from a normal distribution with the given mean
 *           and variance lies within the given range.
 */
double normalProbabilityInRange(double mean, double variance,
                                double low, double high);
```

Header file

- function declarations

stats.cpp

```
#include "stats.h"
#include <cmath>

// Statistical Functions

double normalProbabilityInRange(double mean, double variance,
                                double low, double high) {
    return 0.0; // TODO: implement
}
```

cpp file

- function definitions

main.cpp file

- main function

Today

Pass by value
Pass by reference

Pass by Value

- What you're used to now
- Variables hold values
- Values are **copied** and passed into functions
- Parameters name **new storage** locations
- **And** the original variable remains unchanged

Pass by Value

- Works well most of the time
- But there are times when it can be a problem
- I need two volunteers…

Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}
```

Execution

```
int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}



int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Execution

Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}

int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Execution

Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}


int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Execution

Function Call – Pass by Value

Execution

```
int square(int a) {
    int sq = a * a;
    return sq;
}


int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Function Call – Pass by Value

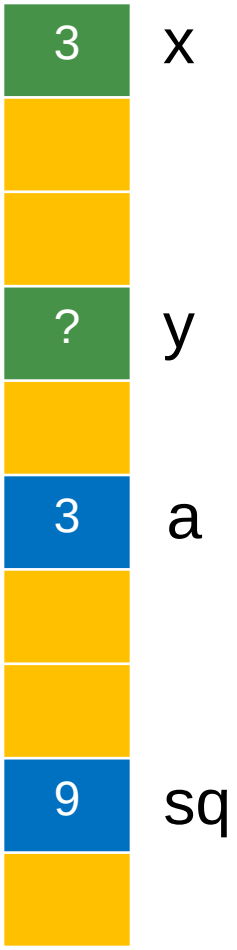
Execution

```
int square(int a) {
    int sq = a * a;
    return sq;
}

int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Function Call – Pass by Value

Execution

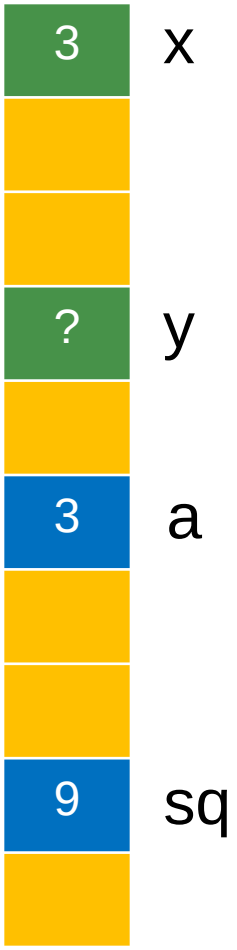
```
int square(int a) {
    int sq = a * a;
    return sq;
}


int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```



Function Call – Pass by Value

Execution

```
int square(int a) {
    int sq = a * a;
    return sq;
}

int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

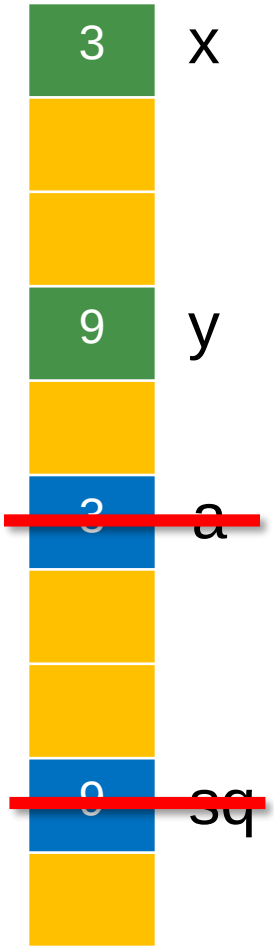


Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}

int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Execution

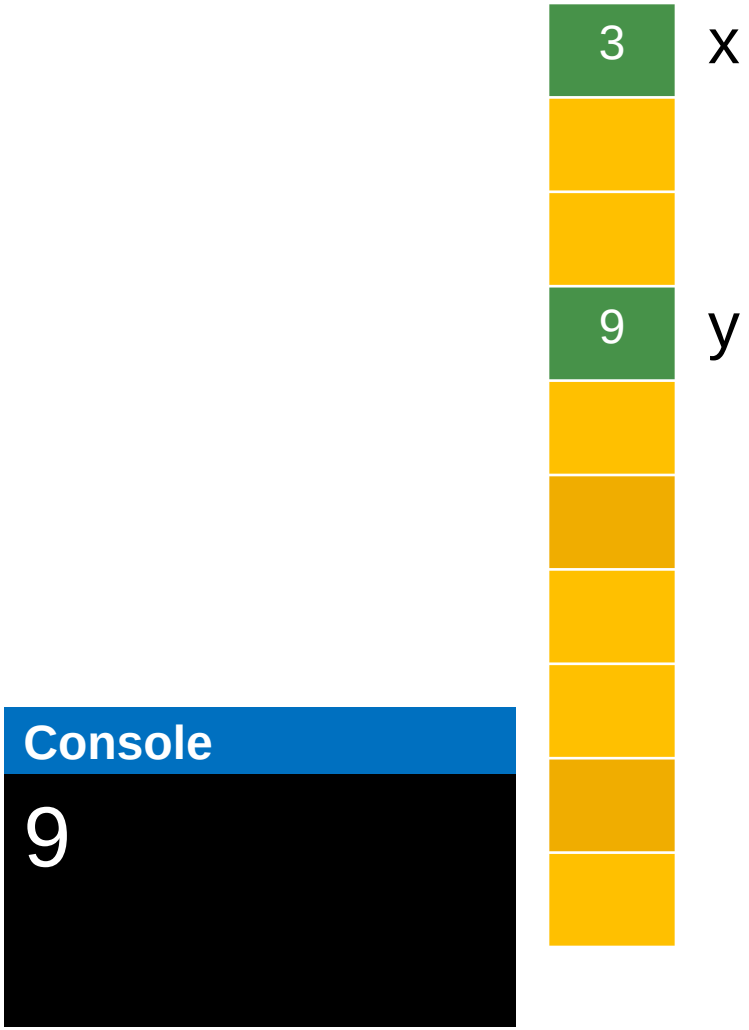
9



Function Call – Pass by Value

```
int square(int a) {
    int sq = a * a;
    return sq;
}

int main(void) {
    int x = 3;
    int y;
    y = square(x);
    cout << y;
}
```

Execution



Pass by Value

- Now, we'll try another function
- This time, we'll use pseudocode
  - Not actual C++
- This function will look up the first word on a specific page of a book

Page Lookup – Pass by Value

Note: This is now *pseudocode*

```
string findFirstWordInPage(int page,
                           book_t book) {
    string firstWord;
    // Look through the book and find
    // the first word of the page
    return firstWord;
}

int main(void) {
    int page = 321;
    book_t book = bookGivenByProfessor();
    string word =
        findFirstWordInPage(page, book);
    cout << word;
}
```

Execution

Page Lookup – Pass by Value

Note: This is now *pseudocode*

```
string findFirstWordInPage(int page,
                           book_t book) {
    string firstWord;
    // Look through the book and find
    // the first word of the page
    return firstWord;
}

int main(void) {
    int page = 321;
    book_t book = bookGivenByProfessor();
    string word =
        findFirstWordInPage(page, book);
    cout << word;
}
```

Execution

Page Lookup – Pass by Value

Note: This is now *pseudocode*

```
string findFirstWordInPage(int page,
                           book_t book) {
    string firstWord;
    // Look through the book and find
    // the first word of the page
    return firstWord;
}

int main(void) {
    int page = 321;
    book_t book = bookGivenByProfessor();
    string word =
        findFirstWordInPage(page, book);
    cout << word;
}
```

Execution

Page Lookup – Pass by Value

Note: This is now *pseudocode*

```
string findFirstWordInPage(int page,
                           book_t book) {
    string firstWord;
    // Look through the book and find
    // the first word of the page
    return firstWord;
}

int main(void) {
    int page = 321;
    book_t book = bookGivenByProfessor();
    string word =
        findFirstWordInPage(page, book);
    cout << word;
}
```

Execution

Pass by Value

- Doesn't work well when:
  - Dealing with something very large in memory, or
  - We **want** to alter the original variable
- In these cases, use **Pass by Reference**

Pass by Value

```
// Pass by Value
int squareByValue(int number) {
    return number * number;
}
```

Pass by Reference

```
// Pass by Value
int squareByValue(int number) {
    return number * number;
}

// Pass by Reference
void squareByReference(int &number) {
    number *= number;
}
```

The **&** indicates *pass by reference*

Pass by Reference

- '&' means ***address*** of variable is passed
  - address is the location in memory
- Pass by value: new storage created for parameter
- Pass by reference: parameter ***references*** the same storage location as the argument variable

Pass by Reference

- '&' means ***address*** of variable is passed
  - address is the location in memory
- Pass by value: new storage created for parameter
- Pass by reference: parameter ***references*** the same storage location as the argument variable

No new storage is created for pass-by-reference parameter

Function Call – Pass by Reference

```
void square(int &a) {
    a = a * a;
}
```

Execution

```
int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Function Call – Pass by Reference

```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Execution

Function Call – Pass by Reference

```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Execution

Function Call – Pass by Reference

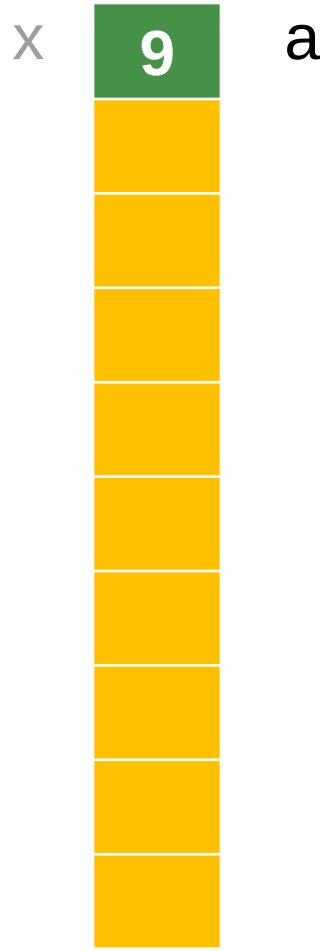
Execution

```
void square(int &a) {
    a = a * a;
}



int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Function Call – Pass by Reference

Execution

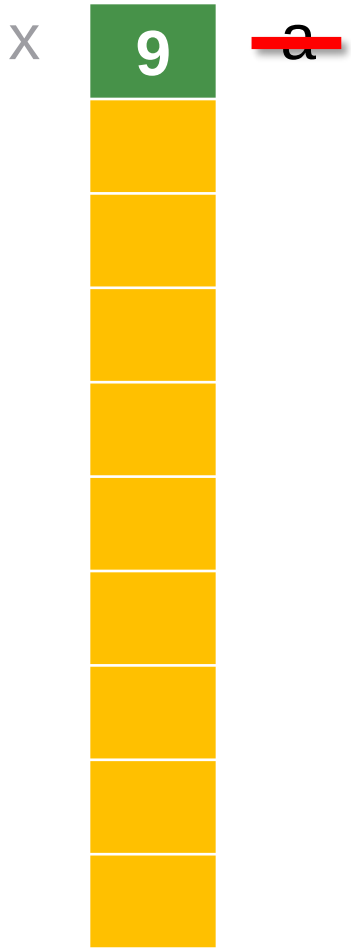
```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```



Function Call – Pass by Reference

Execution

```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```



Function Call – Pass by Reference

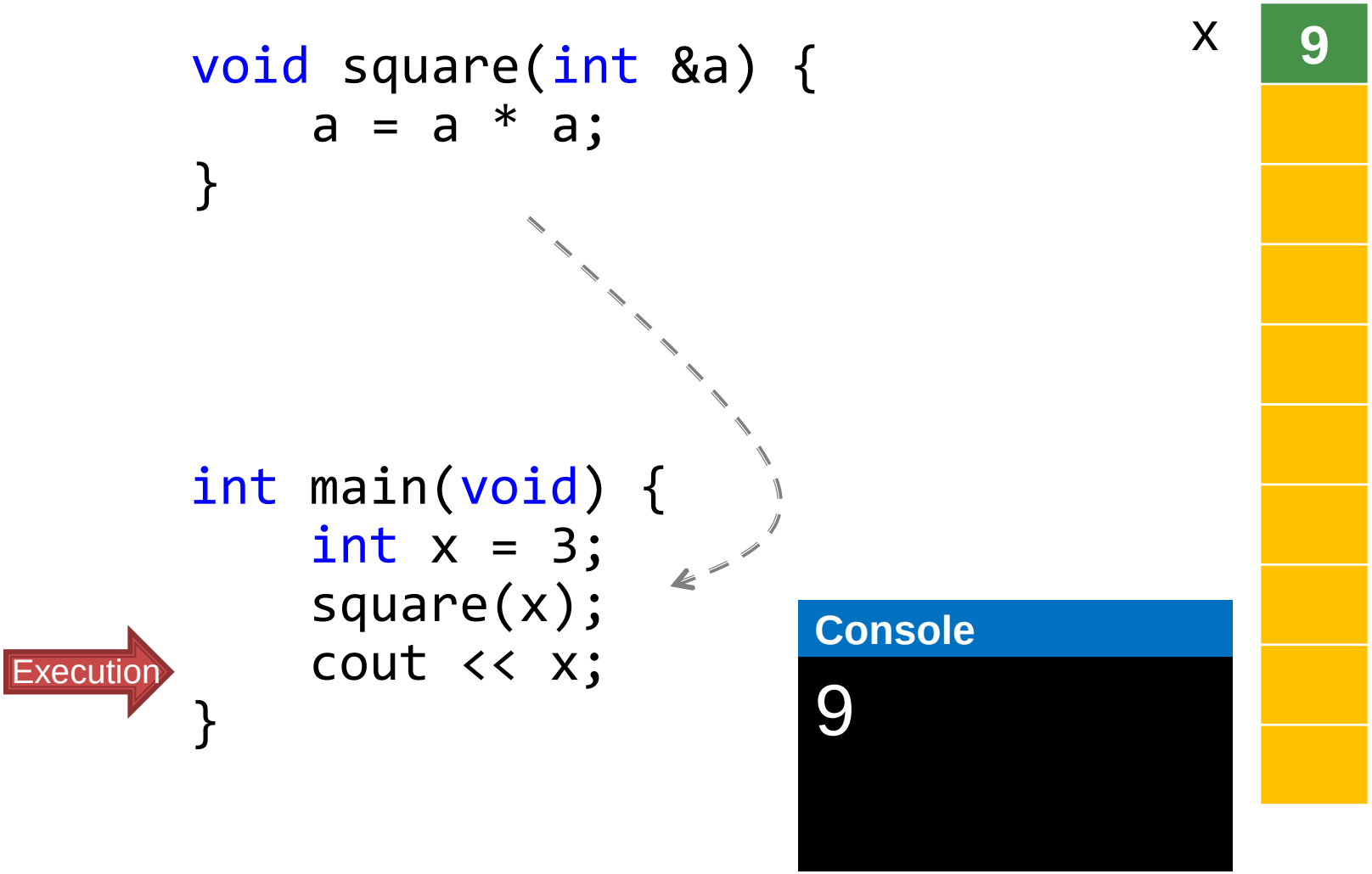
```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Execution

Function Call – Pass by Reference

```
void square(int &a) {
    a = a * a;
}

int main(void) {
    int x = 3;
    square(x);
    cout << x;
}
```

Execution



i>Clicker #4: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
int square(int &x) {
    x = x * x;
    return x;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

i>Clicker #4: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
int square(int &x) {
    x = x * x;
    return x;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

BAD code – decide which way you want the result back

Choose ONE - don't do both

i>Clicker #5: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
int square(int x) {
    x = x * x;
    return x;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

i>Clicker #5: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
int square(int x) {
    x = x * x;
    return x;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

i>Clicker #6: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
void square(int a) {
    a = a * a;
    return;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

i>Clicker #6: What prints?

```
int main() {
    int x;
    x = 2;
    int y = square(x);
    cout << x << y;
    return 0;
}
```

```
void square(int a) {
    a = a * a;
    return;
}
```

A. 24
B. 23
C. 44
D. 43
E. Compile Error

i>Clicker #7: What prints?

```
int main() {
    int x;
    x = 2;
    square(x);
    cout << x;
    return 0;
}
```

```
void square(int &a) {
    a = a * a;
    return;
}
```

A. 4
B. 3
C. 2
D. 1
E. Compile Error

i>Clicker #7: What prints?

```
int main() {
    int x;
    x = 2;
    square(x);
    cout << x;
    return 0;
}
```

```
void square(int &a) {
    a = a * a;
    return;
}
```

A. 4
B. 3
C. 2
D. 1
E. Compile Error

Example

```
int squareByValue(int number) {
    return number * number;
}

void squareByReference(int &number) {
    number *= number;
}

int main() {
    int x = 2;
    int y = 5;

    cout << "x is: " << x << " before squareByValue" << endl;
    cout << "Returned by squareByValue: " << squareByValue(x) << endl;
    cout << "x is: " << x << " after squareByValue" << endl;
    int z = squareByValue(x);
    cout << "z is: " << z << " after squareByValue" << endl << endl;

    cout << "y is: " << y << " before squareByReference" << endl;
    squareByReference(y);
    cout << "y is: " << y << " after squareByReference" << endl;
}
```

What prints? (don't use iClicker)

```
int main() {
    int x = 1;
    int y = 2;
    square(x + y);
    cout << x;
    return 0;
}
```

```
void square(int& a) {
    a = a * a;
    return;
}
```

A. 4
B. 3
C. 2
D. 1
E. Compile Error

What prints?

```
int main() {
    int x = 1;
    int y = 2;
    square(x + y);
    cout << x;
    return 0;
}
```

```
void square(int& a) {
    a = a * a;
    return;
}
```

Must be a variable
can't be an expression

A. 4
B. 3
C. 2
D. 1
E. Compile Error

No memory location for 'a' to connect to

What prints? (don't use iClicker)

```
int main() {
    double x = 1;

    square(x);
    cout << x;
    return 0;
}
```

```
void square(int& a) {
    a = a * a;
    return;
}
```

A. 4
B. 3
C. 2
D. 1
E. Compile Error

What prints?

```
int main() {
    double x = 1;

    square(x);
    cout << x;
    return 0;
}
```

```
void square(int& a) {
    a = a * a;
    return;
}
```

A. 4
B. 3
C. 2
D. 1
E. Compile Error

No casting happens when passing by reference. Type MUST exactly match

Function Parameters - Summary

Value Parameters

- Receives a copy
- Single value, or nothing returned
- Type coercion is allowed
- Can be a variable, constant, or any other expression

Function Parameters - Summary

Value Parameters	Reference Parameters
Receives a copy	Receives a **reference** to the memory location
Single value, or nothing returned	**Multiple values** may be passed back *in effect*
Type coercion is allowed	**Types must match** parameter declaration
Can be a variable, constant, or any other expression	Must have a named storage location (e.g. a **variable**)

Intermission

Two Minute Break

Problem: swap

You want to swap two values

`Let x = -2 and y = 5`

Say we want to swap x and y so that
`x = 5` and `y = -2`

How do we do it? …

swap

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

How would you code swap?

No return value!

swap

```
/**
 * Requires: nothing
 * Modifies: a and b?????
 * Effects:  swaps values of a and b
 */
void swap(int a, int b);
```

swap: Can't Pass by Value


```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value


```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value


```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
-2	a
5	b

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
-2	a
5	b
?	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
-2	a
5	b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
5	a
5	b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
5	a
-2	b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
~~5~~	~~a~~
~~-2~~	~~b~~
~~-2~~	~~temp~~

deallocated

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
~~5~~	~~a~~
~~-2~~	~~b~~
~~-2~~	~~temp~~

deallocated

Execution

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Can't Pass by Value

-2	x
5	y
~~5~~	~~a~~
~~-2~~	~~b~~
~~-2~~	~~temp~~

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y
    return 0;
}
```

Execution

```
void swap(int a,
          int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

Console

```
x=-2 y=5
```

swap: Must Pass by Reference

```
/**
 * Requires: nothing
 * Modifies: a and b
 * Effects:  swaps values of a and b
 */
void swap(int &a, int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

-2	x
5	y

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

-2	x renamed a
5	y renamed b

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

-2	x a
5	y b

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

-2	x a
5	y b
?	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

-2	x a
5	y b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

5	x a
5	y b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

5	x a
-2	y b
-2	temp

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

5	x ~~a~~
-2	y ~~b~~
~~2~~	~~temp~~

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y  << endl;
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

swap: Must Pass by Reference

5	x
-2	y

```
int main() {
    int x = -2;
    int y = 5;

    swap(x, y);
    cout << "x = " << x
         << "y = " << y
    return 0;
}
```

Execution

```
void swap(int &a,
          int &b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

Console

```
x=5 y=-2
```

Example 2: getValue

Problem

- Read length & width
- Print area of corresponding rectangle

Area = ???

- Sample Run:

```
> Enter the length: 13
> Enter the width: 25
> Area is: 325
```

- There are two ways to read length & width

1st Method: Through Return Statement

```
// calls:
length = getValue("Enter the length: ");
width  = getValue("Enter the width: ");


int getValue(string prompt) {
    int value;
    cout << prompt;
    cin >> value;
    return value;
}
```

2nd method: Pass by Reference

```
// calls:
getValue("Enter the length: ", length);
getValue("Enter the width: ", width);


void getValue(string prompt, int &val) {
    cout << prompt;
    cin >> val;
}
```

2nd method: Pass by Reference

```
// calls:
getValue("Enter the length: ", length);
getValue("Enter the width: ", width);


void getValue(string prompt, int &val) {
    cout << prompt;
    cin >> val;
}
```

We can add validation as well

2nd method: Pass by Reference

```
// calls:
while (!getValue("Enter the length: ", length)) {
    cout << "Length must be an integer" << endl;
    cin.clear();
    string garbage;
    getline(cin, garbage);
}

bool getValue(string prompt, int &val) {
    cout << prompt;
    cin >> val;
    return !cin.fail();
}
```

We can add validation as well

That's all folks

On your own:

See following slides for examples

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j;


void setup(int i, int j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j;


void setup(int i, int j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j;


void setup(int i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j;


void setup(int i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j ;


void setup(int &i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j ;


void setup(int &i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j ;


void setup(int &i, int &j)
{
  i = i * j;
  j = 12;
}
```

A) 15 12
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3;
setup(i, j);
cout << i << " " << j ;


void setup(int &i, int &j)
{
  i = i * j;
  j = 12;
}
```

A) **15 12**
B) 5 3
C) 5 12
D) 15 3
E) Compile Error

What prints?

```
int i = 5, j = 3, k = 0;
setup(i, j);
cout << k ;


void setup(int &i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15
B) 0
C) 12
D) 3
E) Compile Error

What prints?

```
int i = 5, j = 3, k = 0;
setup(i, j);
cout << k ;


void setup(int &i, int &j)
{
  int k = i * j;
  j = 12;
}
```

A) 15
B) 0
C) 12
D) 3
E) Compile Error

What prints?

```
int i = 5, j = 3, k = 0;
setup(i, j);
cout << k ;


void setup(int &i, int &j)
{
  k = i * j;
  j = 12;
}
```

A) 15
B) 0
C) 12
D) 3
E) Compile Error

What prints?

```
int i = 5, j = 3, k = 0;
setup(i, j);
cout << k;


void setup(int &i, int &j)
{
  k = i * j;
  j = 12;
}
```

A) 15
B) 0
C) 12
D) 3
E) Compile Error

What prints?

```
int i = 5, j = 3, k = 0;
setup(i, j);
cout << k;
```

doesn't compile

```
void setup(int &i, int &j)
{
  k = i * j;
  j = 12;
}
```

k not declared

swap: What prints?

```
int main()
{
      int  x  =  -2,
           y  =  5;

      swap(x, x + y);
      cout << x << ", "
           << y << endl;
      return (0);
}
```

```
void swap(int& a,
          int& b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}
```

A) -2, 5
B) 5, -2
C) Compile Error

swap: What prints?

```
int main()
{
      int  x  =  -2,
           y  =  5;

      swap(x, x + y);
      cout << x << ", "
           << y << endl;
      return (0);
}
```

```
void swap(int& a,
          int& b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}
```

A) -2, 5
B) 5, -2
C) Compile Error

swap: What prints?

```
int main()
{
      int  x,
           y;

      swap(-2, 5);
      cout << x << ", "
           << y << endl;
      return (0);
}
```

```
void swap(int& a,
          int& b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}
```

A) -2, 5
B) 5, -2
C) Compile Error

swap: What prints?

```
int main()
{
      int  x,
           y;

      swap(-2, 5);
      cout << x << ", "
           << y << endl;
      return (0);
}
```

```
void swap(int& a,
          int& b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}
```

A) -2, 5
B) 5, -2
C) Compile Error

```
#include <iostream>
using namespace std;

int f(int y);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << y << endl;
  return 0;
}

int f(int y){
  y = y + 1;
  return y;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 3 \n
D) Compile Error

```
#include <iostream>
using namespace std;

int f(int y);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << y << endl;
  return 0;
}

int f(int y){
  y = y + 1;
  return y;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 3 \n
D) Compile Error

```
#include <iostream>
using namespace std;

int f(int &x);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << y << endl;
  return 0;
}

int f(int &x){
  x = x + 1;
  return x;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 3 \n
D) Compile Error

```
#include <iostream>
using namespace std;

int f(int &x);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << y << endl;
  return 0;
}

int f(int &x){
  x = x + 1;
  return x;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 3 \n
D) Compile Error

```
#include <iostream>
using namespace std;

int f(int &x);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << f(y) << endl;
  return 0;
}

int f(int &x){
  x = x + 1;
  return x;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 5 \n
D) Compile Error

```
#include <iostream>
using namespace std;

int f(int &x);

int main() {
  int y = 3;
  cout << f(y) << endl;
  cout << f(y) << endl;
  return 0;
}

int f(int &x){
  x = x + 1;
  return x;
}
```

A) 4 \n 4 \n
B) 3 \n 4 \n
C) 4 \n 5 \n
D) Compile Error