

# Introduction to Programming (CS 101)

Spring 2024



## Lecture 9:

Functions and References (Part II)

**Instructor:** Preethi Jyothi

Based on material developed by Prof. Abhiram Ranade and Prof. Manoj Prabhakaran

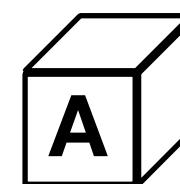
# Recap (IA)

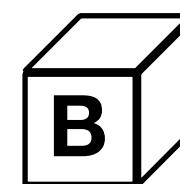
What is the output of the following program?

```
int findPower(int base = 2, int exponent = 3) {  
    int result = 1;  
    for(int i = 0; i < exponent; i++, result *= base);  
    return result;  
}
```

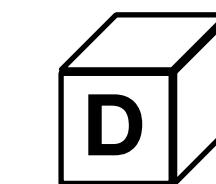
```
main_program {  
    cout << findPower(5);
```

```
}
```

 A 5

 B 25

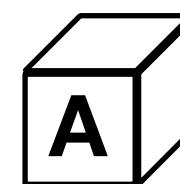
 C 125

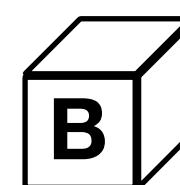
 D 625

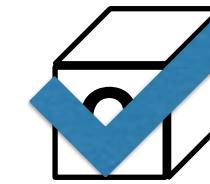
# Recap (IB)

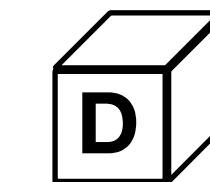
What is the output of the following program?

```
int findPower(int base = 2, int exponent = 3) {  
    int result = 1;  
    for(int i = 0; i < exponent; i++, result *= base);  
    return result;  
}  
  
main_program {  
    cout << findPower(5);  
  
    cout << findPower(findPower(5,1));  
  
}
```

 A 5

 B 25

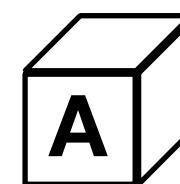
 C 125

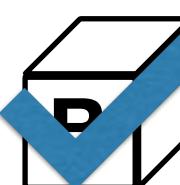
 D 625

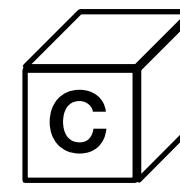
# Recap (IC)

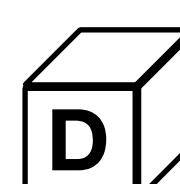
What is the output of the following program?

```
int findPower(int base = 2, int exponent = 3) {  
    int result = 1;  
    for(int i = 0; i < exponent; i++, result *= base);  
    return result;  
}  
  
main_program {  
    cout << findPower(5);  
  
    cout << findPower(findPower(5,1));  
  
    cout << findPower();  
}
```

 A 5

 B 8

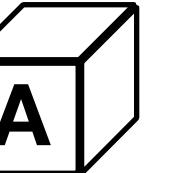
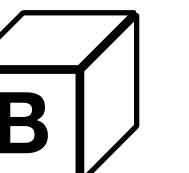
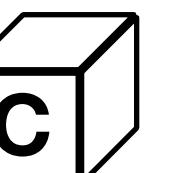
 C 2

 D 125

# Recap (IIA)

What is the output of the following program?

```
int change(int k) {  
    return(k+=2);  
}  
  
main_program {  
    int i = 1;  
    i = change(++i);  
    cout << i;  
}
```

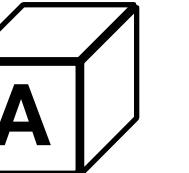
-  1
-  2
-  3
-  4

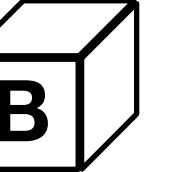
# Recap (IIB)

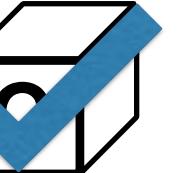
What is the output of the following program?

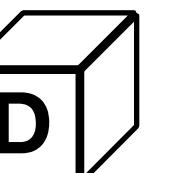
```
int change(int k) {  
    {  
        int k = 3;  
        return(k+=2);  
    }  
}
```

```
main_program {  
    int k = 1;  
    k = change(++k);  
    cout << k;  
}
```

 A 2

 B 4

 C 5

 D 3



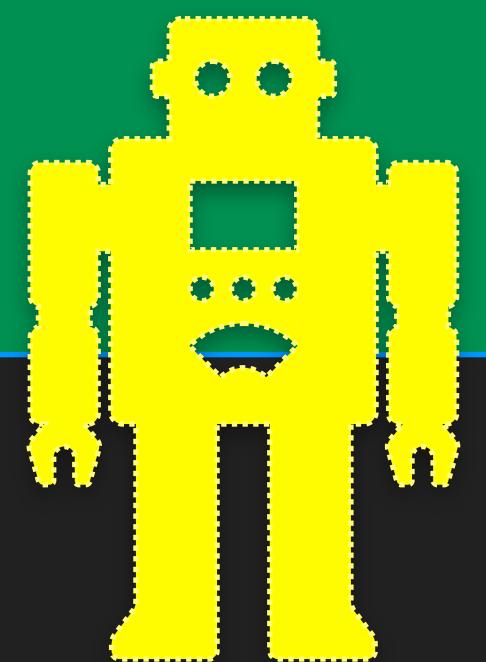
# Function calls: Behaviour on the stack

## CS 101, 2025

# How function calls work: The stack

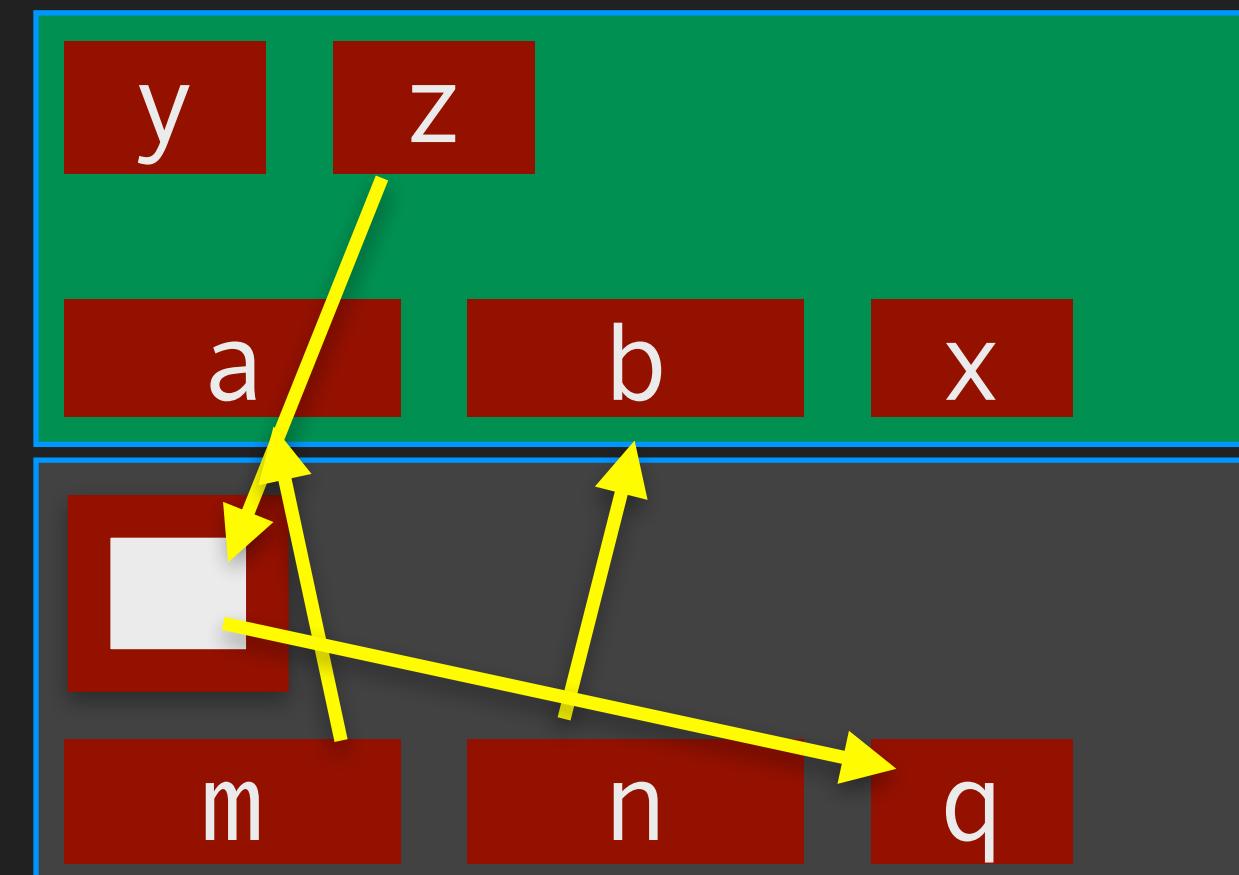
```
main_program {  
    int m, n;  
    bool q;  
    // ...  
    q = PFE(m, n);  
    // ...  
}
```

```
bool PFE(int a, int b) {  
    bool x, y, z;  
    // ...  
    return z;  
}
```



Processor

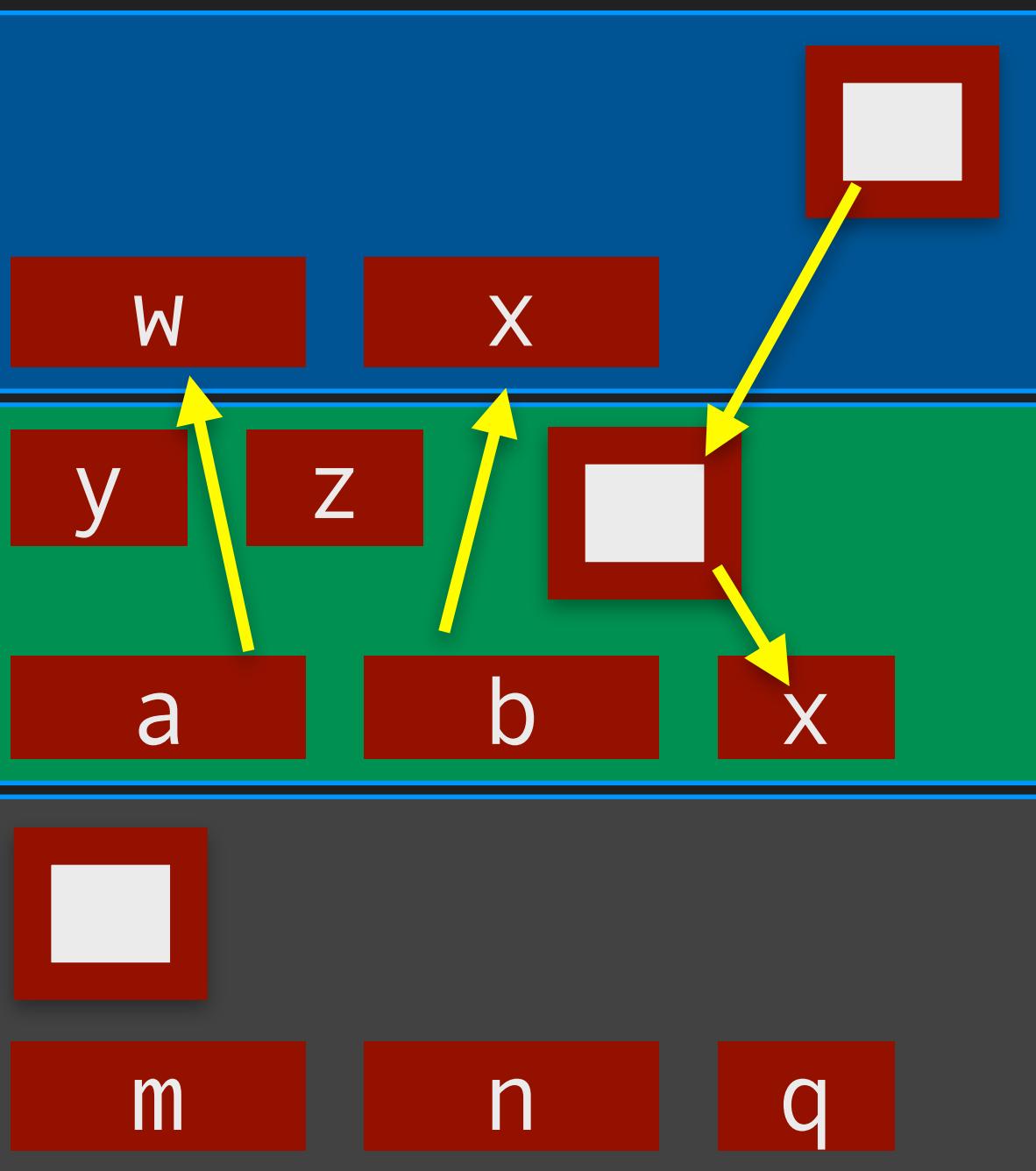
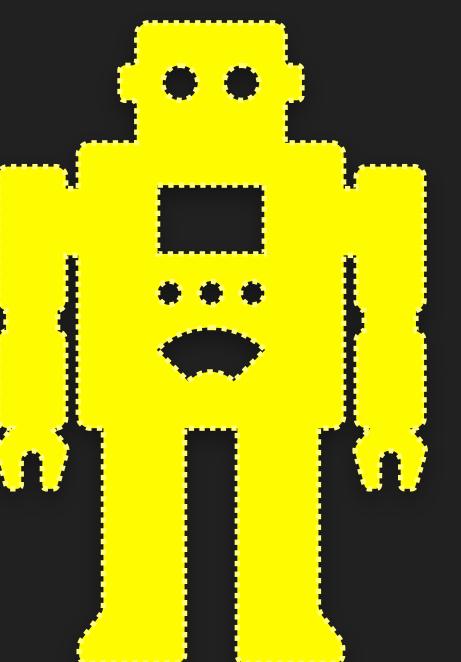
Memory



# How function calls work: The stack

```
bool covers(int w, int x) {  
    // ...  
    return x==1;  
}  
  
main_program {  
    int m, n;  
    bool q;  
    // ...  
    q = PFE(m, n);  
    // ...  
}
```

```
bool PFE(int a, int b) {  
    bool x, y, z;  
    x = covers(a, b);  
    y = covers(b, a);  
    z = x && y;  
    return z;  
}
```

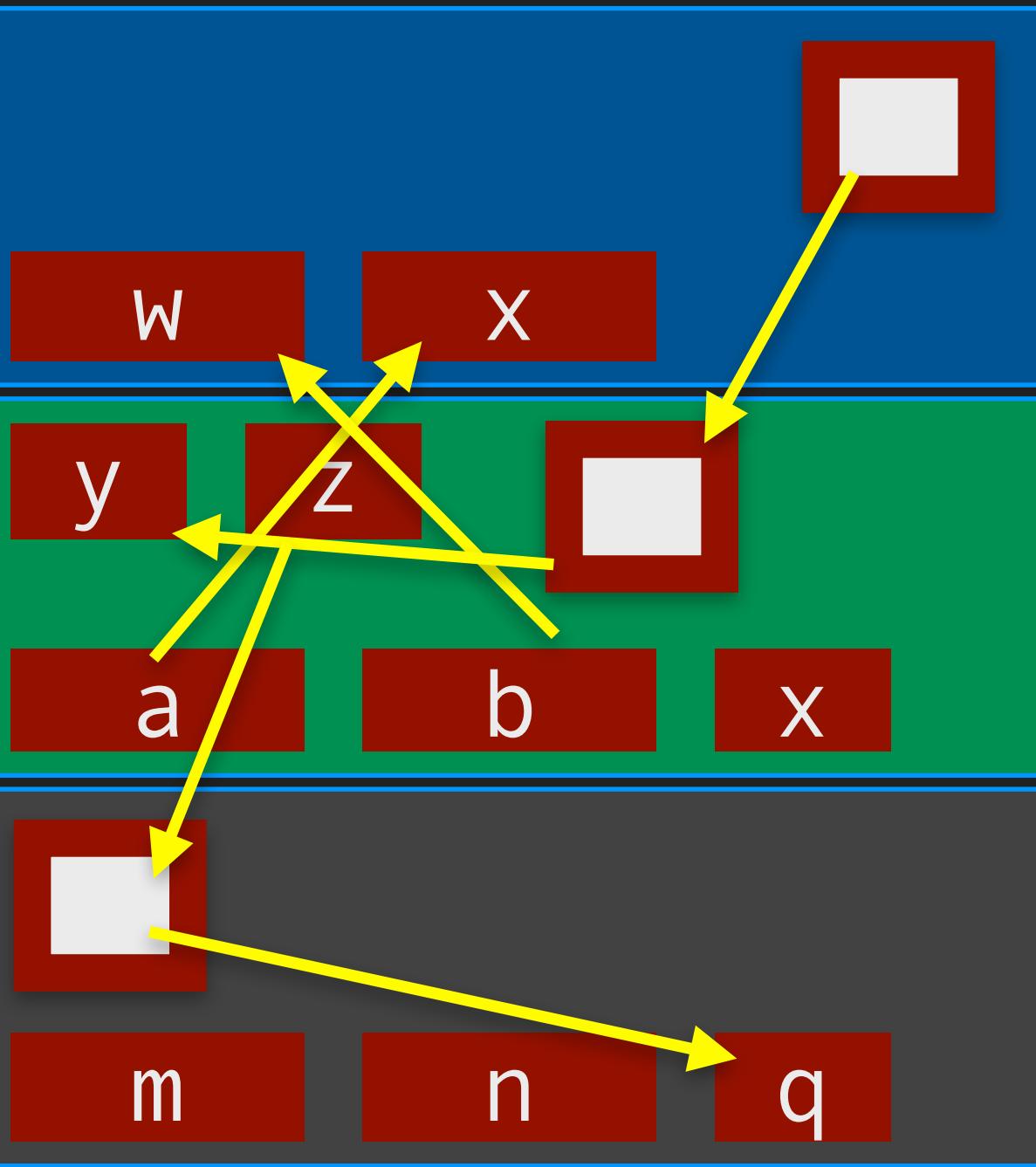
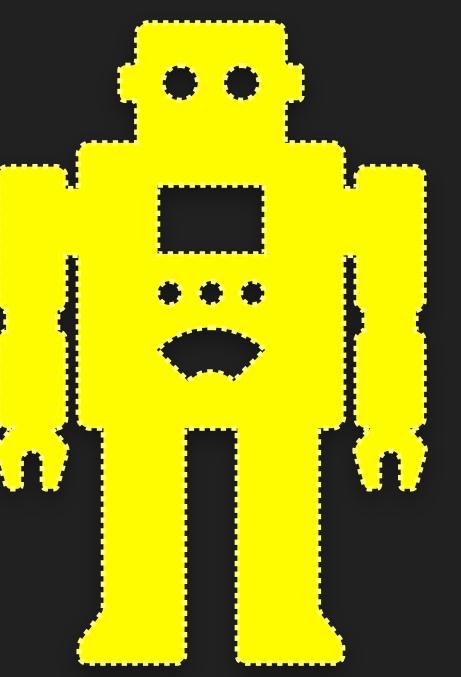


# How function calls work: The stack

```
bool covers(int w, int x) {  
    // ...  
    return x==1;  
}
```

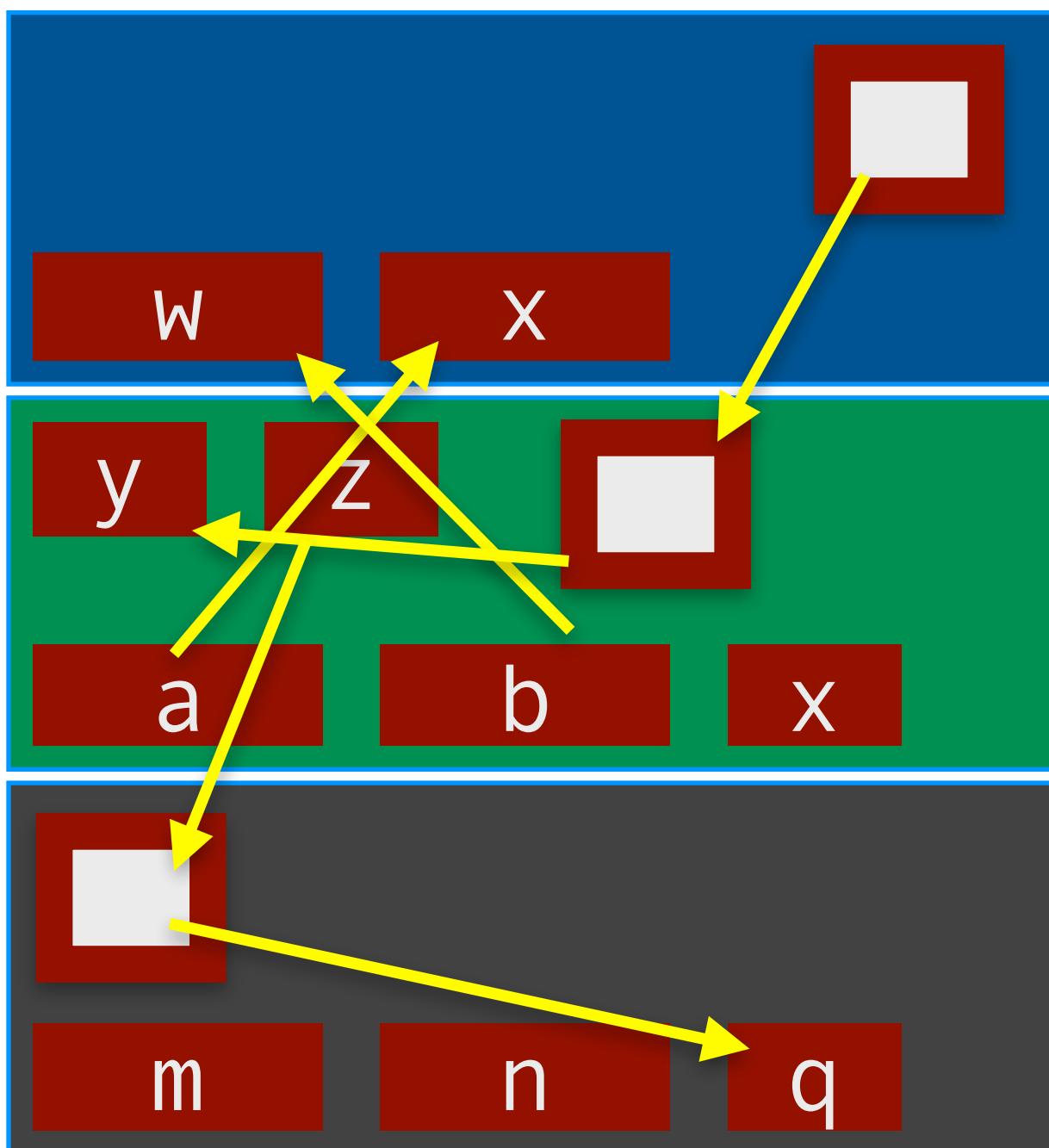
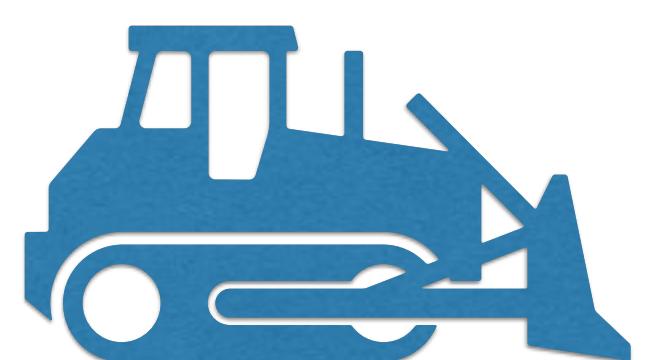
```
main_program {  
    int m, n;  
    bool q;  
    // ...  
    q = PFE(m, n);  
    // ...  
}
```

```
bool PFE(int a, int b) {  
    bool x, y, z;  
    x = covers(a, b);  
    y = covers(b, a);  
    z = x && y;  
    return z;  
}
```



# How function calls work: The stack

- When a function is called, it gets its own piece of memory at the top of the stack: its “frame”
- The inputs to the function (arguments) are copied on to the corresponding variables in its frame (parameters) from the frame below it (frame of the function which called it)
- While executing, the function uses only its own frame (we will cover exceptions like global variables, etc. later, and references next)
- When the function returns, the return value is copied into the frame below, and then its own frame is discarded





# References

## CS 101, 2025

# Swapping

```
void swap(int p, int q) {  
    int tmp;  
    tmp = p; p = q; q = tmp;  
}
```

```
int main() {  
    int a, b, x, y;  
    ...  
    swap(a,b);  
    swap(x,y);  
}
```

- Does this work?
  - No! Only variables local to `swap` (i.e., `p`, `q`) are swapped and are destroyed when `swap` returns

# Swapping

```
void swap(int& p, int& q) {  
    int tmp;  
    tmp = p; p = q; q = tmp;  
}
```

p and q are **references** to the arguments

```
int main() {  
    int a, b, x, y;  
    ...  
    swap(a,b);  
    swap(x,y);  
}
```

- Now swap works as desired. What do references enable?

# References (I)

- A variable `x` occupies a region in the memory
- Reference is like a tube (with a valve) to the space in memory, that allows information to flow in and out
- References allow the user to read and write to the memory locations

```
int& a = x;  
int& b = x;
```

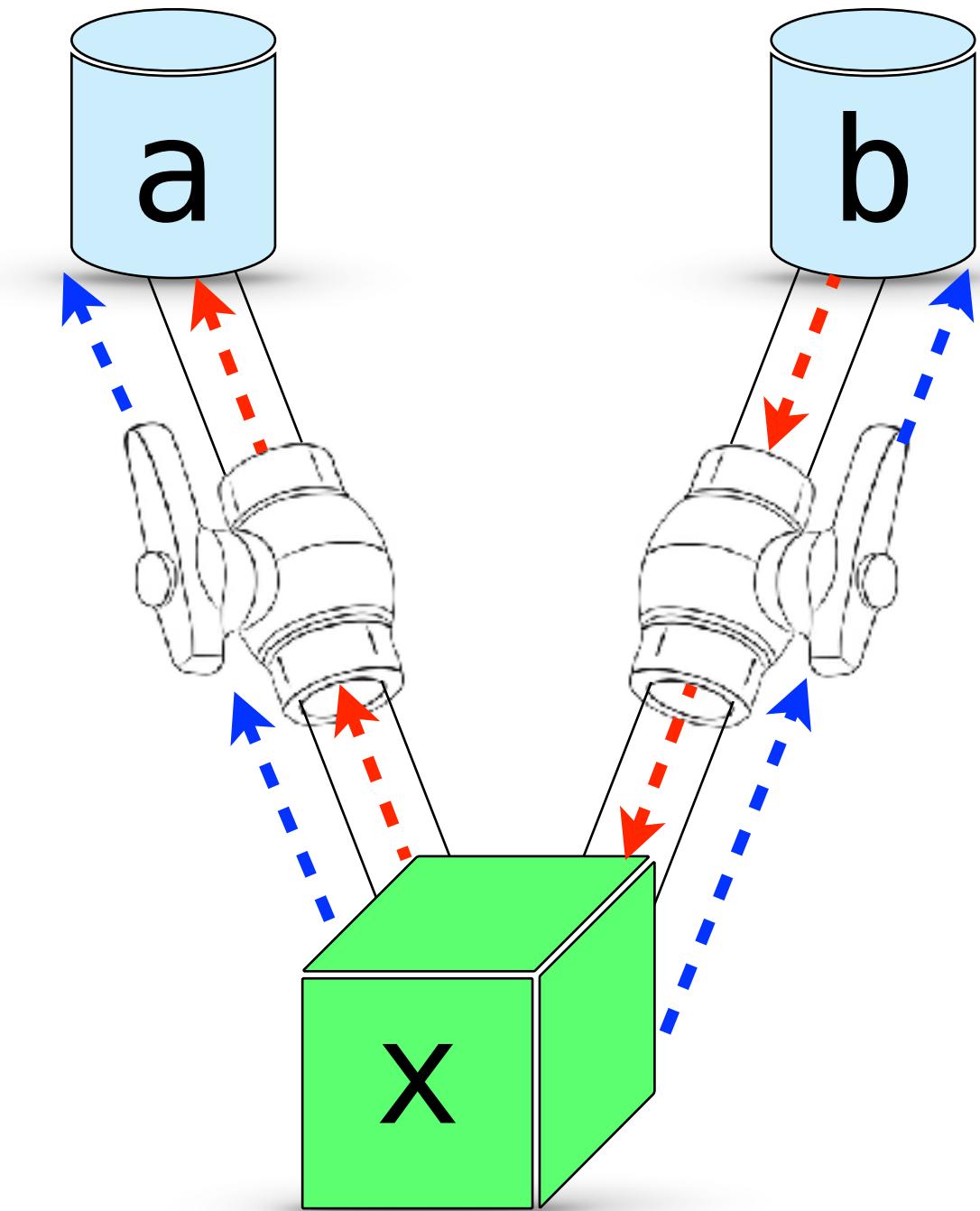
- You can read from and write to these tubes:

```
a = b + 1;
```

```
int x;  
int &a = x;  
int &b = x;  
a = 10;  
x++;  
b == 11;
```

Is this expression true or false?

true



## References (II)

- Cannot create a reference to an intermediate variable

```
int& a = x + 1; // throws a compiler error
```

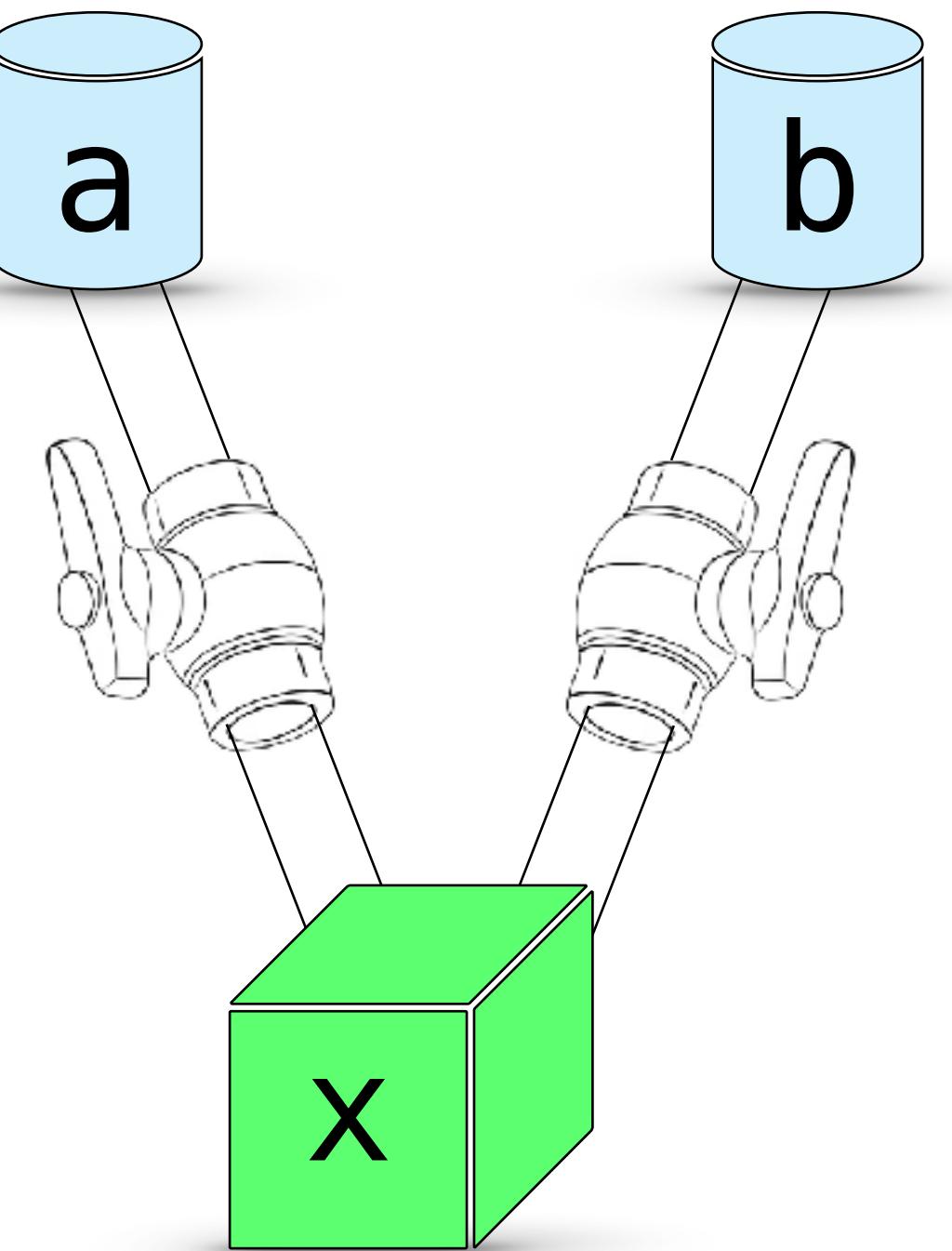
- Intermediate variables cannot be assigned values

```
x+1 = 3; // throws a compiler error
```

- References are valid lvalues (other than variables)

- Reference declarations have to be initialized

```
float& f1; // throws a compiler error
```



# References (III)

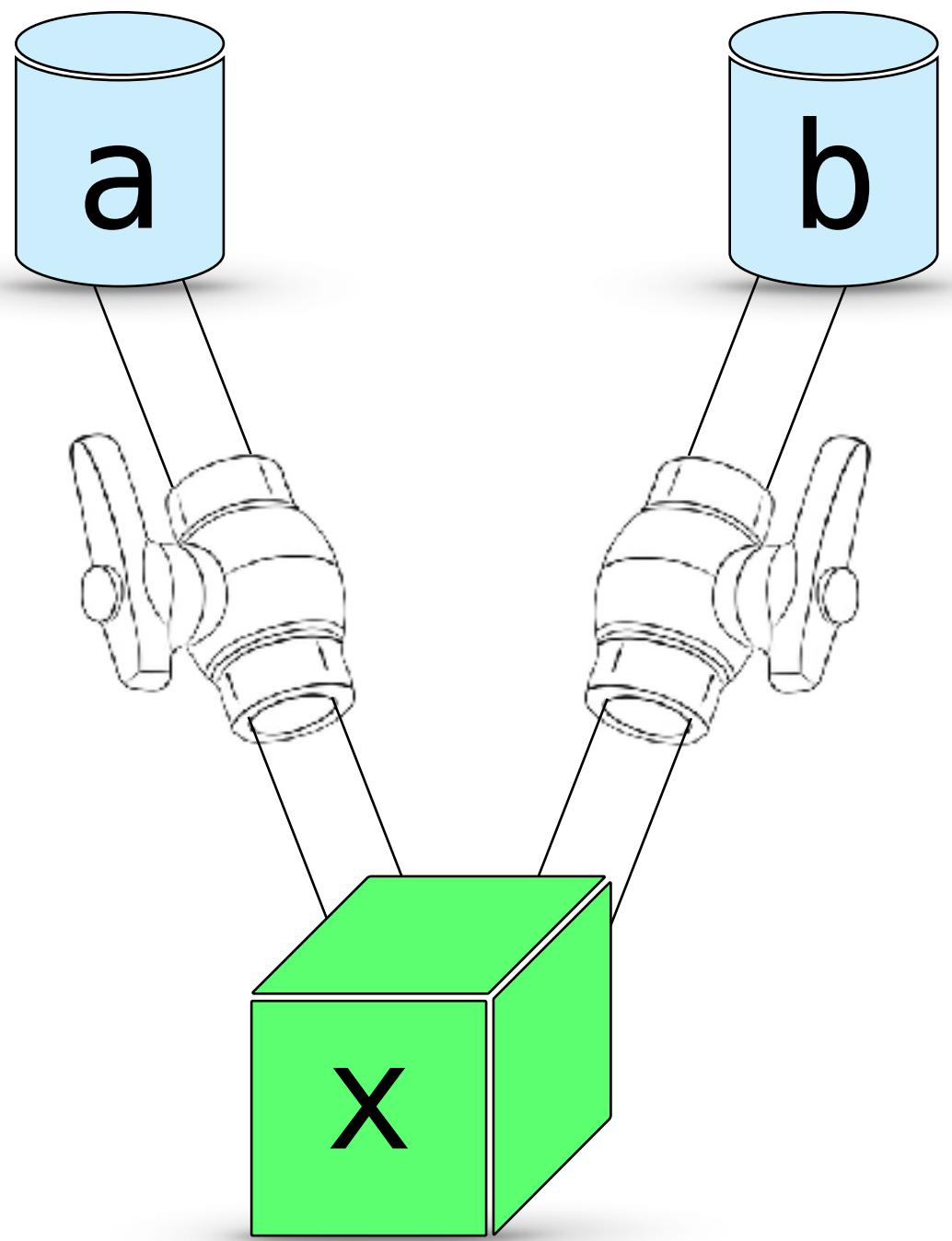
- A reference **has to be** bound to a variable during declaration and cannot be reattached later

```
int x, y;  
int &a = x; // a is bound to x forever  
a = y;      //copies value of y to x
```

- While declaring a reference, instead of specifying a variable to attach to, one can also specify a reference to attach to

- Example:

```
int x; int& a = x;  
int& b = a; //b and a are attached to x
```



# References (IV)

- When declaring multiple references, each one should be prefixed with the & sign

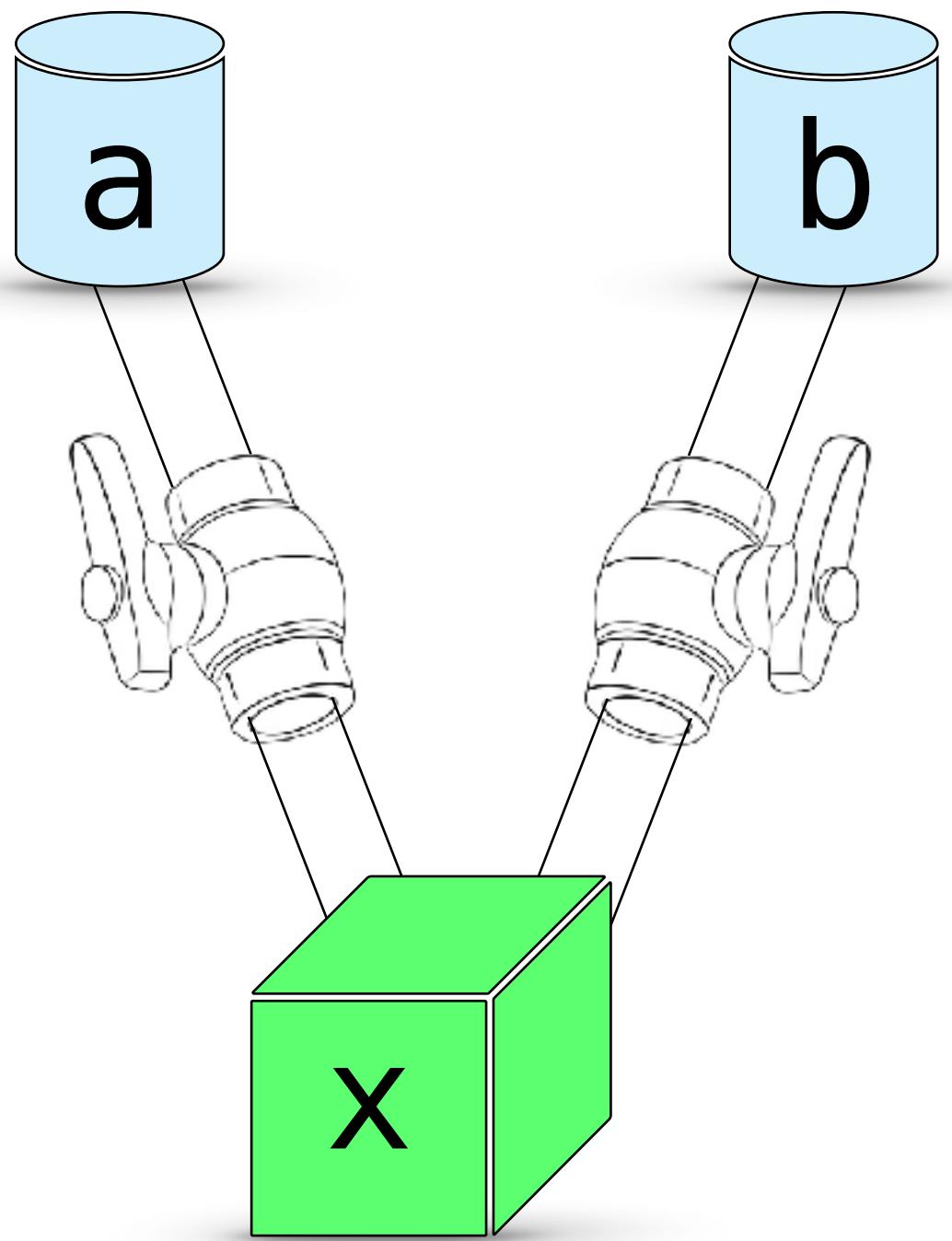
```
int &a = x, &b = x;
```

- Spaces around & are optional

```
int &a = x;  
int& a = x; } all are the same  
int&a = x;
```

- You can mix non-reference variables and references in a declaration:

```
int x, &a = x;
```





# Functions and References

## CS 101, 2025

# Swapping

```
void swap(int& p, int& q) {  
    int tmp;  
    tmp = p; p = q; q = tmp;  
}
```

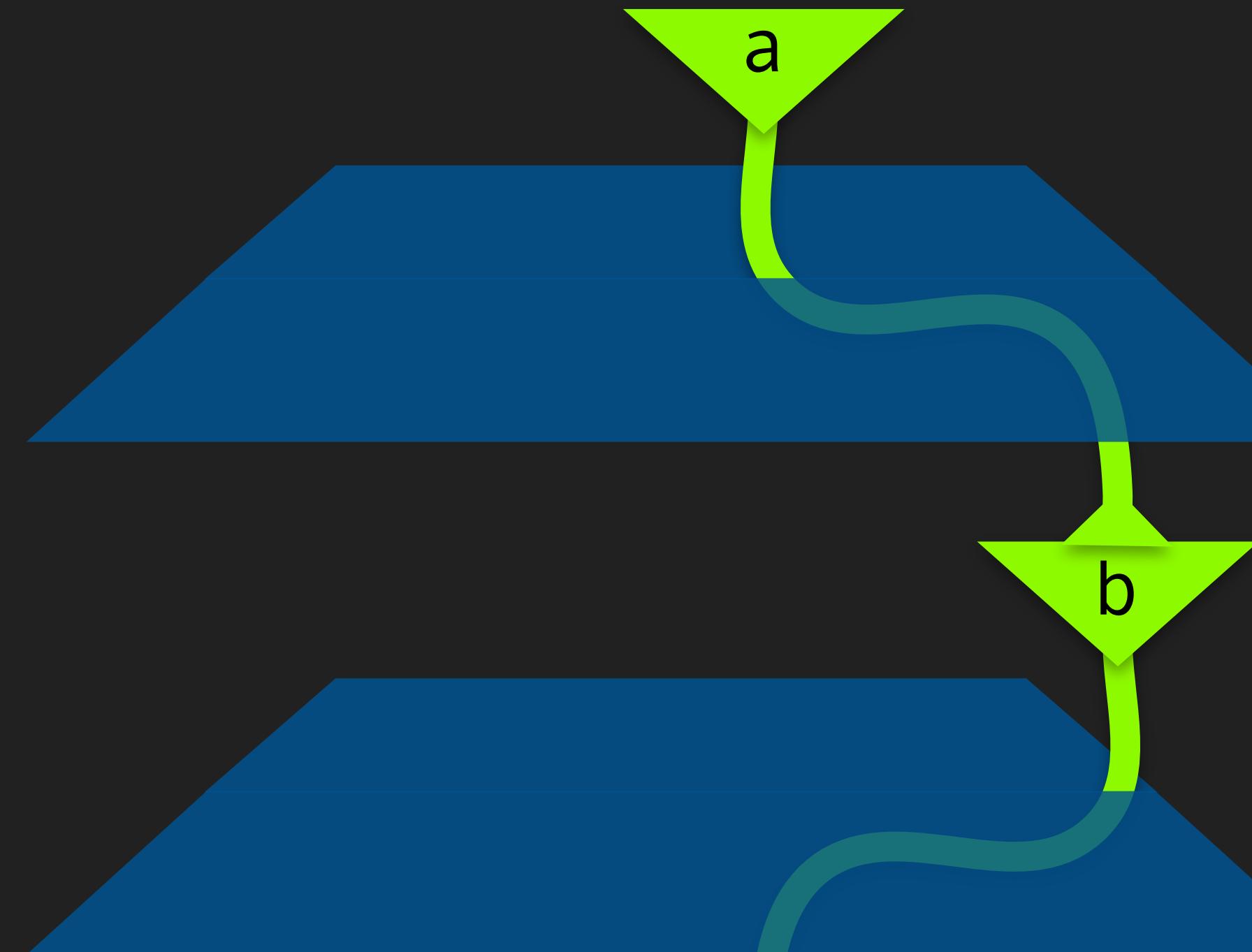
p and q are **references** to the arguments

```
int main() {  
    int a, b, x, y;  
    ...  
    swap(a,b);  
    swap(x,y);  
}
```

- Swap works as desired since p and q, being references to a, b and x, y will change the values in these respective variables.

# Passing arguments by reference

- If a function's parameter is a reference (a tube), it will be attached to a memory location (a box), when the function is called
- The box is (typically) in the frame of the calling function
  - Note: the called function gets access to variables not in its frame!
  - The box can be further down in the stack too!



```
void h(int &a) {...}
```

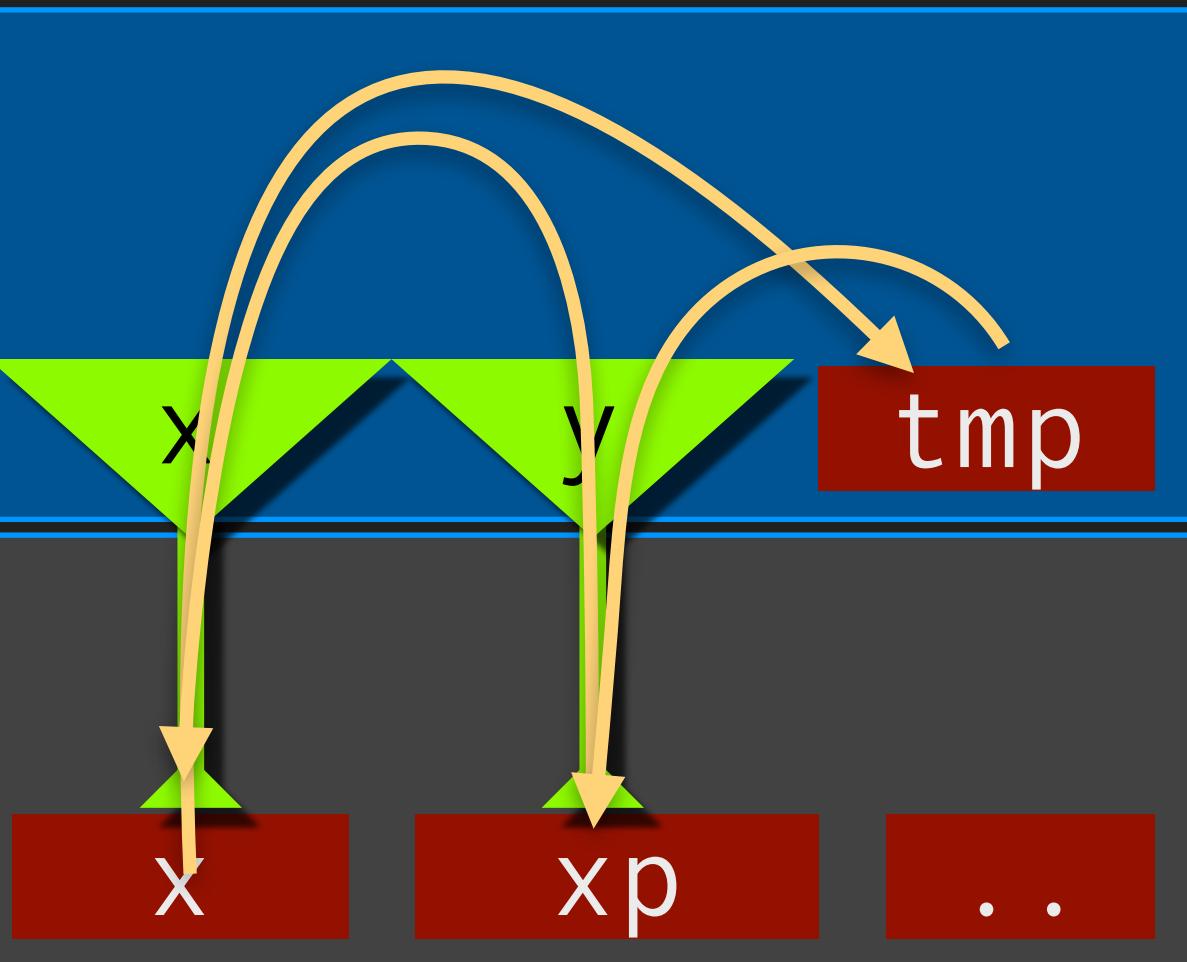
```
void f() {  
    int x; ... g(x); ...  
}
```

```
void g(int &b) {  
    ... h(b); ...  
}
```

# Example: Swapping

```
void swp(int& x, int& y) {  
    int tmp; tmp=x; x=y; y=tmp;  
}
```

```
int main() {  
    int x, xp, y, yp, deg, degp;  
    ...  
    swp(x, xp); swp(y, yp); swp(deg, degp);  
}
```



# Implement post increment using references

- Demo code in class

# Implement post increment using references

```
int postincr(int& m) {  
    int old_m = m;  
    m = m + 1;  
    return old_m;  
}
```

Recall: assigning an **int** reference to an **int** variable will appropriately assign the underlying value

```
int main() {  
    int x = 1;  
    cout << postincr(x) << endl;  
    cout << x << endl;  
}
```

Returns the old value stored in **x**

- `postincr(x)` works like the post-increment operator `x++`

# Implement post increment using references

- What is the output of the following program?

```
int maxvalue(int& a, int& b) {  
    return (a>b) ? a : b;  
}
```

```
main_program {  
    int i = 1;  
    cout << maxvalue(1, 2);
```



Compiler Error

```
}
```

# Implement post increment using references

- What is the output of the following program?

```
int maxvalue(int& a, int& b) {  
    return (a>b) ? a : b;  
}
```

```
main_program {  
    int i = 1;  
    cout << maxvalue(1, 2);
```

```
    cout << maxvalue(i++, i);
```

```
}
```



Compiler Error

# Implement post increment using references

- What is the output of the following program?

```
int maxvalue(int& a, int& b) {  
    return (a>b) ? a : b;  
}
```

```
main_program {  
    int i = 1;  
    cout << maxvalue(1, 2);  
  
    cout << maxvalue(i++, i);  
  
    cout << maxvalue(++i, i);  
}
```

2

output



# Next class: More about References and Recursion

CS 101, 2025