Functions & Scope

C++ Lecture 4

Nick Matthews

Adam Kohl

Motivation

- Putting all our code in main is NOT feasible
- We need to separate out code instead of putting EVERYTHING in one place (Modular)
- We need to make things more organized and easily referenceable (Maintainable)
- We need to make scope changes to prevent cluttered memory

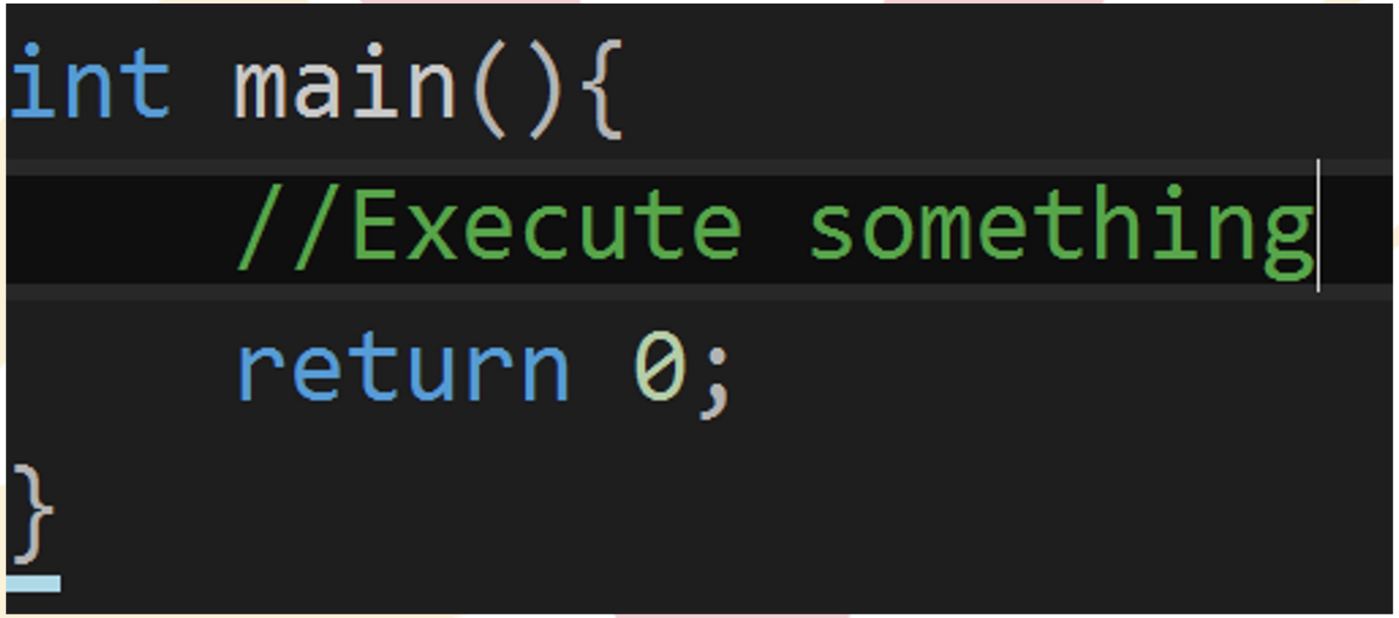
🗣 Lecture Goals ✅

- Functions
- Program scope
- Program stack*

* If we get to it

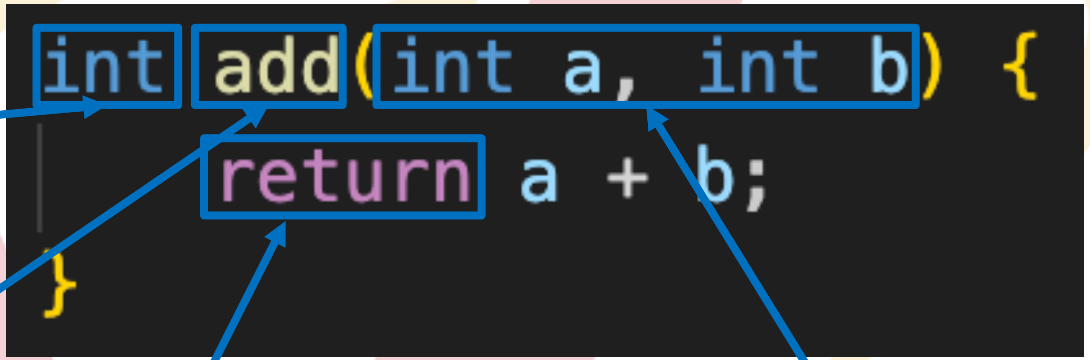
Functions

- A **function** is a group of statements that together perform a task.
- You have already come across at least one ⬇️



```
int main(){
    //Execute something
    return 0;
}
```

Anatomy of a Function



```
int add(int a, int b) {
    return a + b;
}
```

Return Value
What will come out of the function

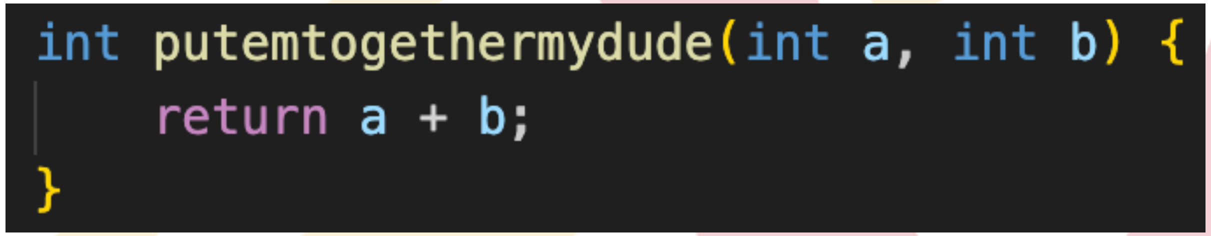
Function Name
Unique identifier

Return Keyword
End of the function!

Input parameters
What YOU put in the function

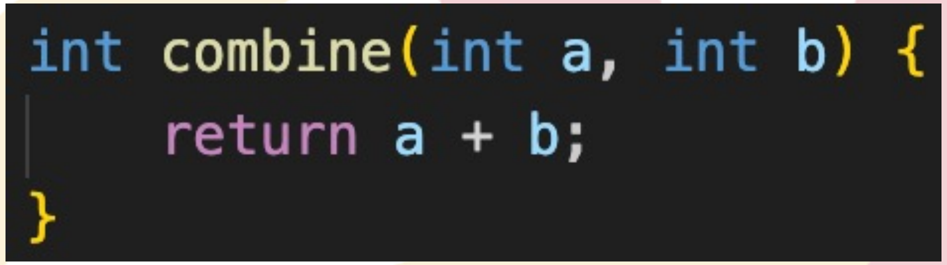
👊 Code Etiquette (Functions)

- If you repeat code → use functions!
- Always use function names that represent what the function does (use verbs!!!!)



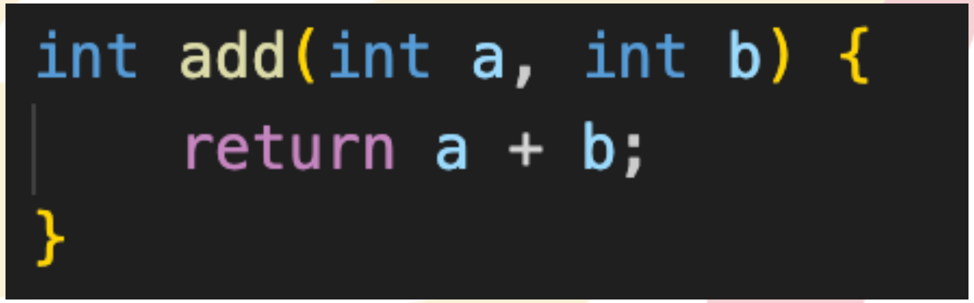
```
int putemtogethermydude(int a, int b) {
    return a + b;
}
```

Bad 👎



```
int combine(int a, int b) {
    return a + b;
}
```

Ok 😐



```
int add(int a, int b) {
    return a + b;
}
```

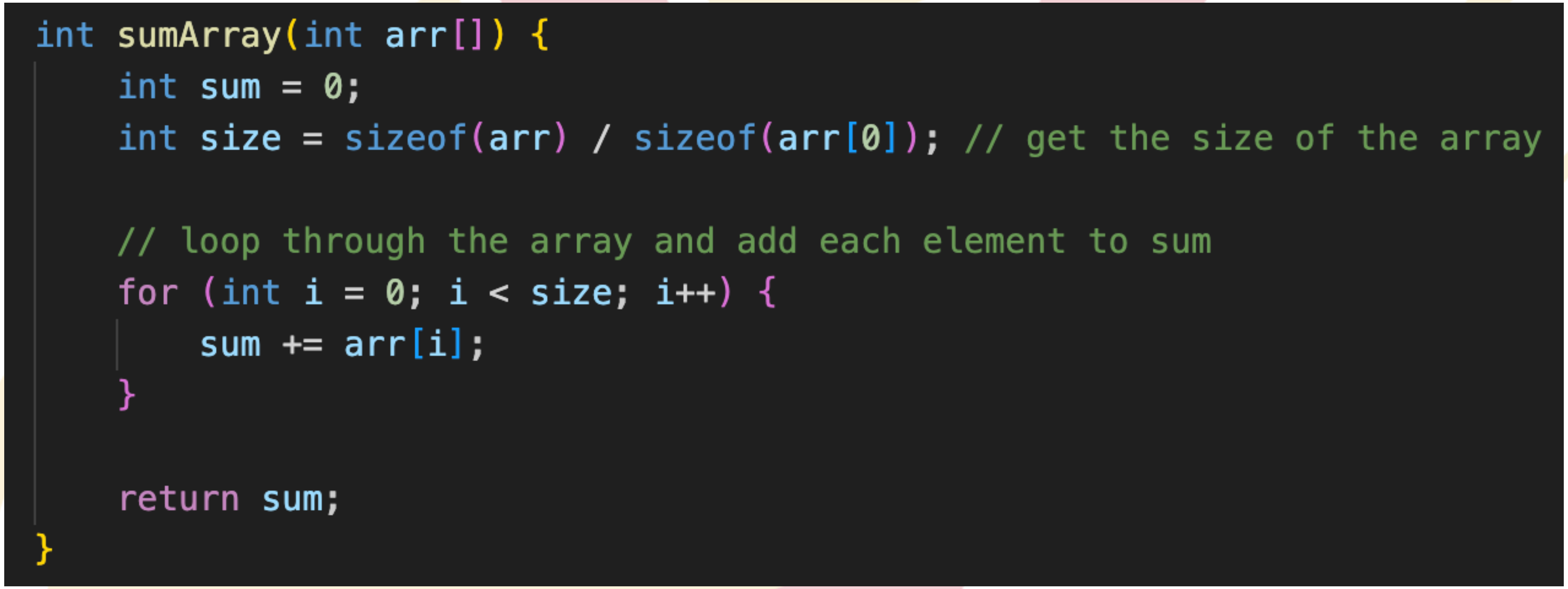
Great! 🥳

🤏 Mini Task ✏️

1. Go to https://github.com/iastate/VRAC_REU_Programming
2. Under **challenges/** read customCalculator.md 📖
3. Make a new project and code 👨‍💻

Functions and arrays

- Passing an array to a function



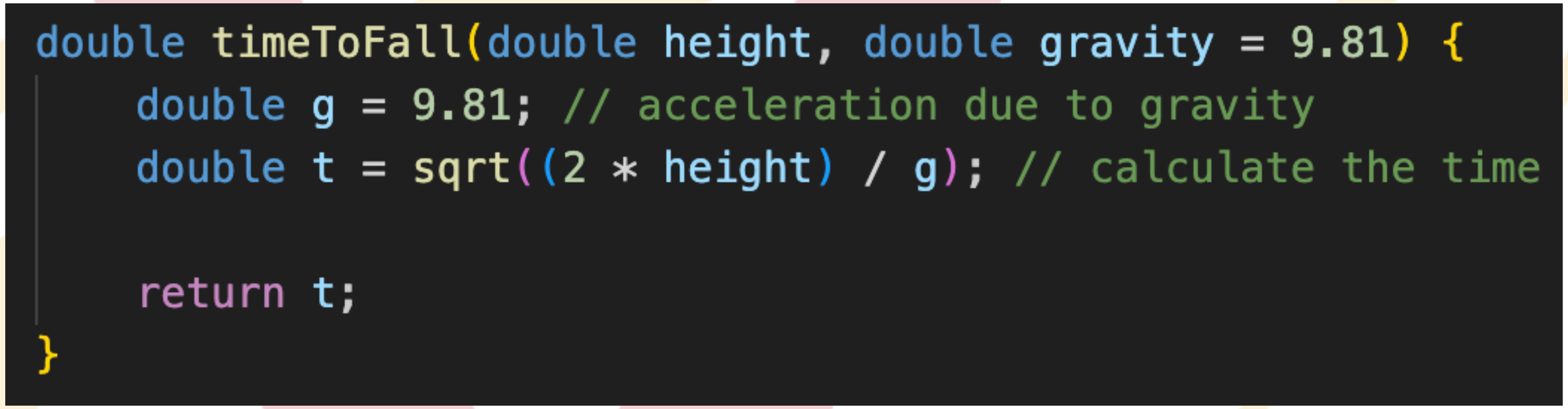
```
int sumArray(int arr[]) {
    int sum = 0;
    int size = sizeof(arr) / sizeof(arr[0]); // get the size of the array

    // loop through the array and add each element to sum
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }

    return sum;
}
```

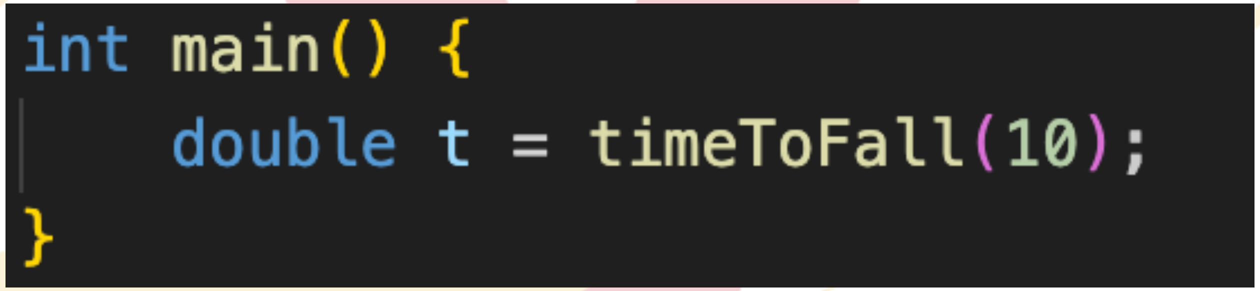
Function defaults

- Sometimes you want the arguments for your functions to have default values



```
double timeToFall(double height, double gravity = 9.81) {
    double g = 9.81; // acceleration due to gravity
    double t = sqrt((2 * height) / g); // calculate the time

    return t;
}
```



```
int main() {
    double t = timeToFall(10);
}
```

🤏 Mini Task ✏️

1. Go to https://github.com/iastate/VRAC_REU_Programming
2. Under **challenges/** read arrayValidation.md 📖
3. Make a new project and code 👨‍💻

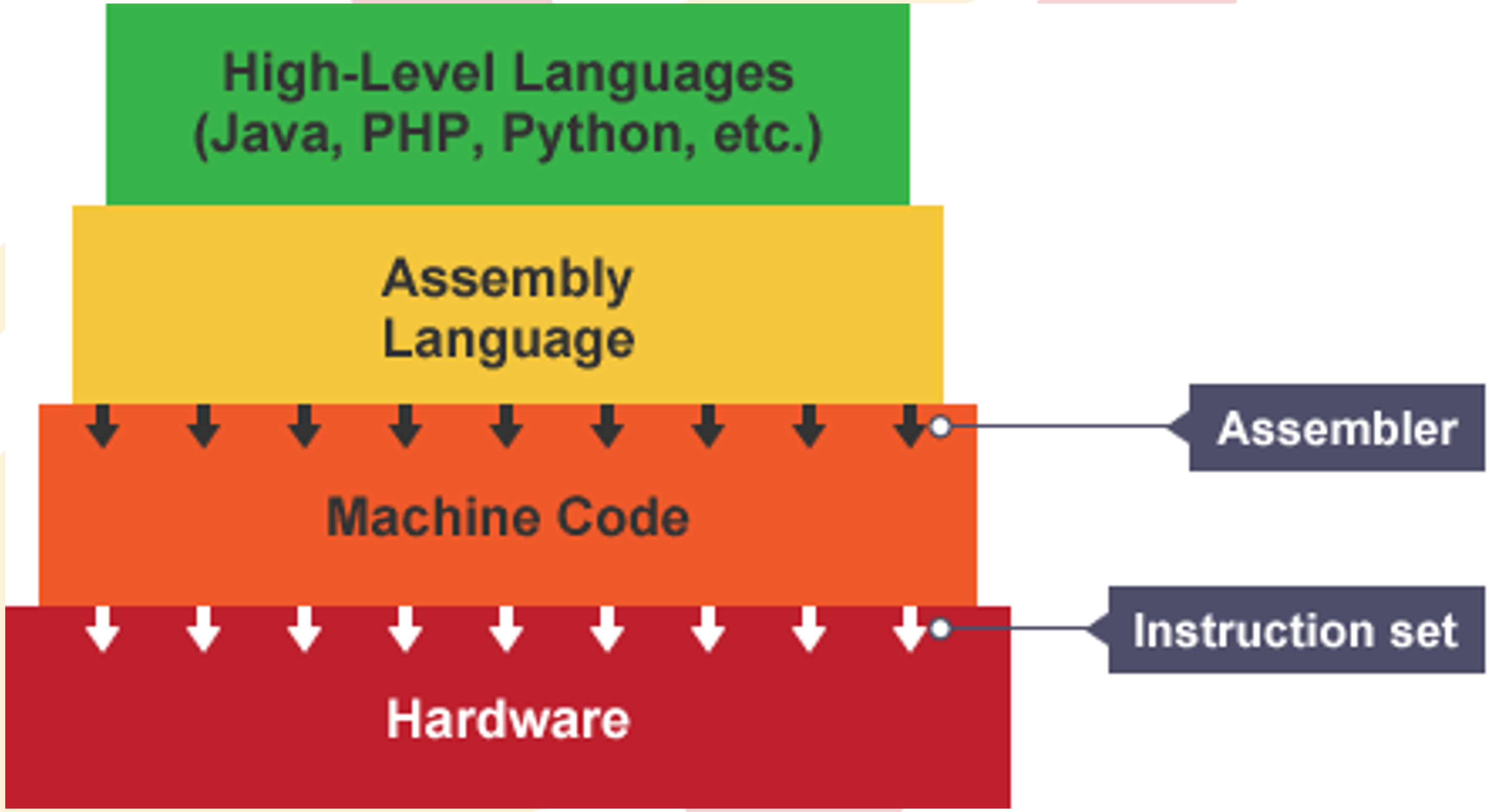
Scope Examples 💻

- Where you create variables determines their accessibility and lifetime

Local Variables: Variables defined between '{' and '}'. They <u>cannot</u> be accessed outside the braces.

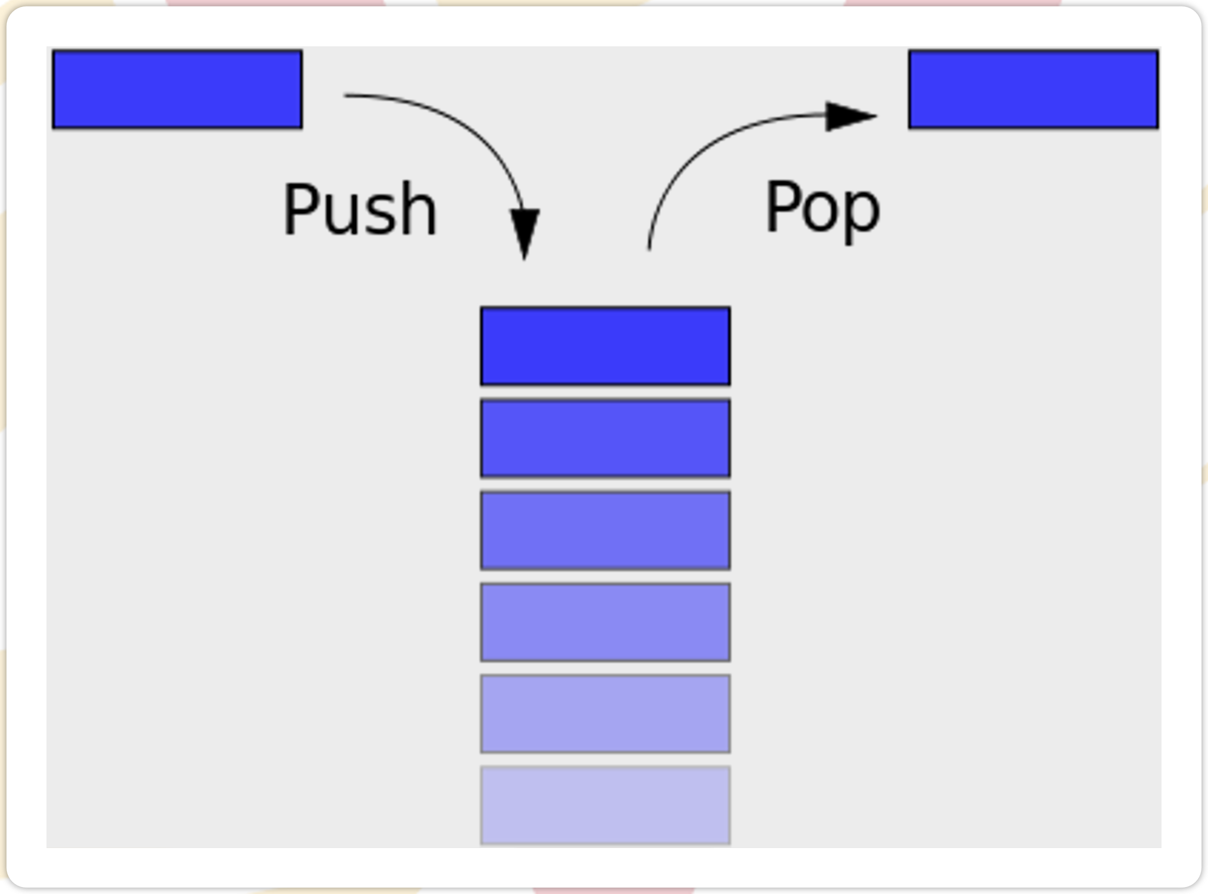
Global Variables: Declared outside all functions and blocks. They can be accessed <u>anytime</u> during the lifetime of the program

How does any program run?



Program Stack

- A dynamic structure in memory where variables are stored and accessed during the runtime of your programs.



🤏 Mini Task ✏️

1. Choose a previous Mini Task or Assignment
2. Use the debugger to step through the program. Watch how the variables and program stack change.

Questions?

📝 Assignment

1. Go to https://github.com/iastate/VRAC_REU_Programming
2. Under **challenges/** read arrayValidation.md 📖
3. Make a new project and code 👨‍💻