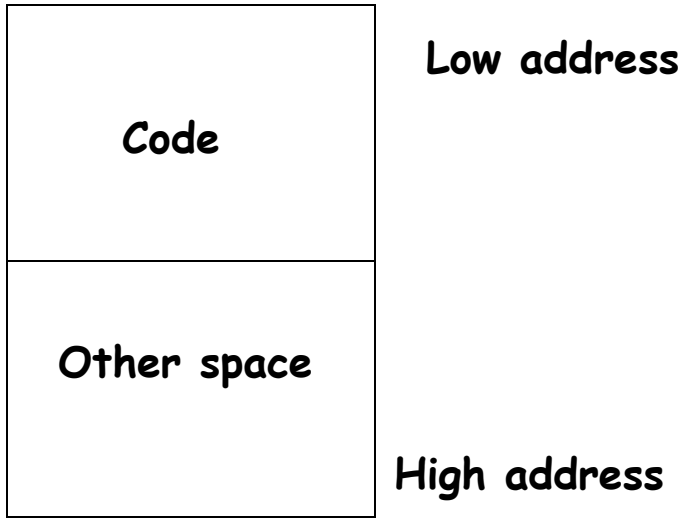
Code Generation & Run time Environments

1. What *information* do we need to execute a program?
2. What does *runtime memory layout* look like?
3. What goes where?
4. How do we implement functions and calls?
5. Object-oriented features?

Executing a program is initially under control of operating system

When a program is invoked:

- the operating system allocates space for the program
- the code is loaded into part of the space
- jump to entry point of the code - the main program



Note: not necessary for all program space to be contiguous

What is the purpose of "other space"?

- **Holds all data that the program needs and creates**

Compiler is responsible for

- generating the code
- orchestrating/managing the use of the data area

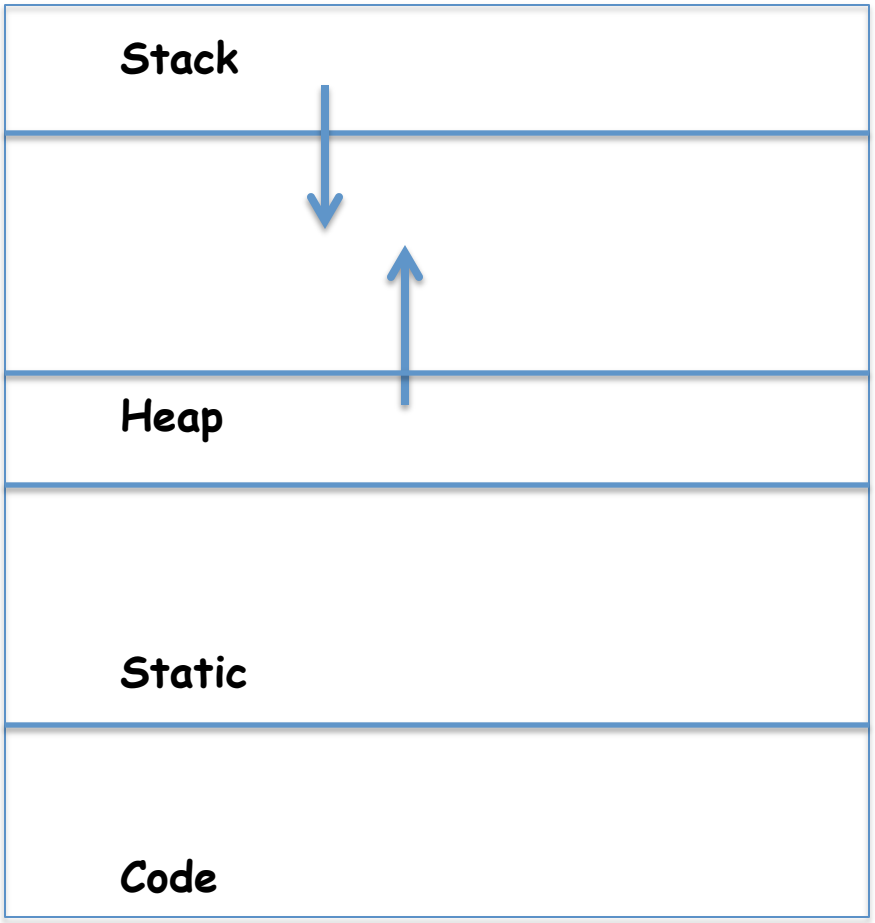
What does data organization depend on? What language features?

Names, Bindings, Scope, Lifetimes

```
Int x;
Int Function y(int a)
  { int x;
    …
  }
Void function z(int c)
    { int x;
    call y(x);
    }
Main()
X = read(); print( y(x));
```

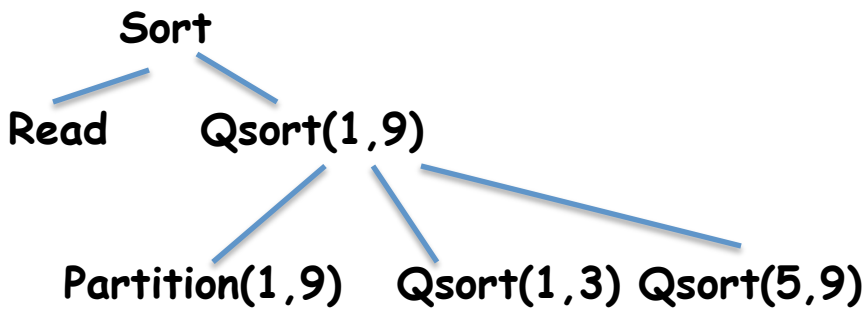
- Declaration versus activation of a function
- Binding name to storage location
- Binding values to storage locations
- Declaration versus binding
- Scope of declaration vs Lifetime of binding

Where do we put the data and code?



Function Activations & Activation Trees

```
Sort(list)
{
Read {}
Qsort (low,high){
    int x
      Partition (low,high){}
    x = Partition(low,high)
    call Qsort(low, x-1 }
    call Qsort(x+1,high)
```



Exercise

```
Class Main {
    g() : Int { 1 };
    f(x:Int):  Int { if x = 0 then g() else f(x - 1) fi};
    main(): Int {{f(3); }};
}
```

What is the activation tree for this example?

Notes

- The activation tree depends on run-time behavior
- The activation tree may be different for every program input
- Need to keep track of procedure activations during execution

What information needs to be in the activation record for a function activation?