Local Variables Do Not Exist Outside of their Blocks

Usually, **local variables** are
- **created and destroyed**
- **at the start and end**
- **of the enclosing function**.

```
{
    int i, j;
    // i and j can be accessed here
}
```

space allocated before execution

space reclaimed after execution

Variables Can be Used without Naming Them

A **variable's scope**
- **defines** the **part of the program**
- in which the **variable can be used by name**.

Using a variable **does not require** its name.
- You have already seen an example: **scanf**
- If variable exists,
  - its address can be used
  - to read or write the variable.

C Provides Three Storage Classes for Variables

A variable's **storage class** defines
- **when the variable is created and destroyed**
- and **where in memory** the variable is stored.

There are **three storage classes** in **C**:
- **static**: exists for the whole program
- **automatic**: exists for a single block of code (such as a function)
- **dynamic**: created and destroyed on demand

A Memory Map Illustrates Use of Memory

But where are the storage classes stored?

How do high-level languages (such as C) make use of LC-3 memory and registers?

Let's take a look, starting with a **memory map**.