Definitions & Data Types

A definition statement allocates memory for a variable's value. You will normally combine both declaration and defintion into a <mark>defining declaration</mark> like this:

```
int counter;        // counter is declared an defined
char letters[10];   // letters is an array of 10 chars
string verb;        // verb is declared and defined
Star rigel;         // Star is a user-defined type
```

Each name is associated with a particular **kind of data** (ints <mark>type</mark>), and the compiler **allocates space** in memory to hold a value for each one. A variable may be **defined exactly once** in a program, but may be **declared** any number of times.

In your homework, you must **define** the `STUDENT` varible, the `ASSIGNMENT` is only **declared**, <mark>not</mark> **defined**. This means it must be defined elsewhere (in this case, in the procompiled <mark>library</mark> file `libh01.a`.)

Data Types

Variables in C++ are <mark>strongly</mark> typed. **Strong typing** means that each variable has a particular **data type** which does not change as the program runs. If you think of a variable as a box, you can think of a variable's data type as the <mark>kind</mark> of data which it can store. As an analogy, think of your local convenience store, where different product containers are specialized for a particular kind of beverage or snack.

In some languages, (such as Python), it is possible for the same variable to store a number at one point, and a string at another.

C++ is also <mark>statically</mark> typed. **Static typing** means that variable **types** are **explicitly specified** in your source code, unlike Python or JavaScript where they are not.

We categorize the C++ types as.

- <mark>Built-in value types</mark> are part of the language; also called **fundamental**, **primitive** types. In the previous section, `counter` is a built-in, primitive type

- <mark>Derived</mark> **(or compound) types** are part of the language, but are built upon one of the other types; this includes **pointers**, **arrays** and **references**. The array `letters` in example at the top of this page is a derived or compound type.

- <mark>Library types</mark>, such as `string` and `vector`, are class types supplied as part of the Standard C++ library; they are not part of the C++ language. In the example from the previous section, `verb` is a `string`, one of the library types.

- <mark>User-defined types</mark> are designed and implemented by programmers. These are **enumerated types**, **classes** and **structures**. The variable `rigel` in the example is a variable of a user-defined type named `Star`.