2 Variables and assignments

- Variables are like small blackboards
  - We can write a number on them
  - We can change the number
  - We can erase the number
- C++ variables are names for memory locations
  - We can write a value in them
  - We can change the value stored there
  - We cannot erase the memory location
    - Some value is always there

2.1 Identifiers

- Variables names are called identifiers
- Choosing variable names
  - Use meaningful names that represent data to be stored
  - First character must be
    - a letter
    - the underscore character
  - Remaining characters must be
    - letters
    - numbers
    - underscore character, _

2.2 Keywords

- Keywords (also called reserved words)
  - Are used by the C++ language
  - Must be used as they are defined in the programming language
  - Cannot be used as identifiers
  - See `http://en.cppreference.com/w/cpp/keyword`

2.3 Declaring Variables

- Before use, variables must be declared
  - Tells the compiler the type of data to store
    - Examples:

      ```
      int number_of_bars;
      double one_weight, total_weight;
      ```
  - **int** is an abbreviation for integer.
    - could store 3, 102, 3211, -456, etc.
    - number_of_bars is of type integer
  - **double** represents numbers with a fractional component
    - could store 1.34, 4.0, -345.6, etc.

    - one_weight and total_weight are both of type **double**

- Two locations for variable declarations

```
#include <iostream>
int main(int argc, const char * argv[]) {
        ...
        int sum;
        sum = score1 + score 2;
        ...
        return 0;
}
```

```
#include <iostream>
int main(int argc, const char * argv[]) {
        int sum;
        ...
        sum = score1 + score 2;
        ...
        return 0;
}
```

- Declaration syntax:

```
Type_name Variable_1 , Variable_2,...;
```

- Declaration Examples:

```
double average, m_score, total_score;
double moon_distance;
int age, num_students;
int cars_waiting;
```

2.4 Assignment Statements

- An assignment statement changes the value of a variable
  - `total_weight = one_weight + number_of_bars;`
    - total_weight is set to the sum one_weight + number_of_bars
  - Assignment statements end with a semi-colon
  - The single variable to be changed is always on the left of the assignment operator '='
  - On the right of the assignment operator can be
    - Constants – `age = 21;`
    - Variables – `my_cost = your_cost;`
    - Expressions – `circumference = diameter * 3.14159;`

2.5 Assignment Statements and Algebra

- The '=' operator in C++ is not an equal sign
  - The following statement cannot be true in algebra

    ```
    number_of_bars = number_of_bars + 3;
    ```

  - In C++ it means the **new** value of number_of_bars is the **previous** value of number_of_bars plus 3

2.6 Initializing Variables

- Declaring a variable does not give it a value
  - Giving a variable its first value is initializing the variable
- Variables are initialized in assignment statements

```
double mpg; // declare the variable
mpg = 26.3; // initialize the variable
```

- Declaration and initialization can be combined using two methods

- Method 1

  ```
  double mpg = 26.3, area = 0.0 , volume;
  ```

- Method 2

  ```
  double mpg(26.3), area(0.0), volume;
  ```