Functions for data input, Read Marker

- **Read with >>**
  Automatically skipping white spaces: ‘ ‘, ‘\t’, ‘\n’
  Reading integer, float, char, string

  What values are in ch, v1, and f1? where is the reading marker?

  Example: Data

  ```
  char    ch, ch1, ch2;                         (1) 25   \t   A    16.9\n
  int     v1;                                   (2) 25A16.9\n
  float   f1;                                   (3) 16\n
  string  st1, st2;                                 B\n
                                                    2.5\n
                                                (4) 24  34.4  23\n
  cin >> v1 >> ch >> f1;  // or
  myIn >> v1 >> ch >> f1;
  ```

- read character using **get**
  get: read one character from input stream without skipping white spaces

  what values are in v1, f1, ch, ch1, ch2 :

  ```
  (1)  cin >> v1;                    myIn>>v1;
       cin.get(ch);                  myIn.get(ch);
       cin >> str1;                  myIn >> str1

  (2)  cin.get(ch);                  myIn.get(ch);
       cin >> v1;                    myIn >> v1;
       cin >> f1;                    myIn >> f1;
       cin.get(ch);                  myIn.get(ch);
       cin >> str2;

  (3)  cin.get(ch1);                 // change cin to myIn
       cin.get(ch2);
       cin>>ch1;
       cin >>f1;

       data:
       21  C  34.2\n
  ```

- read string : **>> vs. getline**

  ```
  examples:                          data:
  (1) cin >> st1;                    Motel 8 Super
  (2) cin >> st1 >> v1 >> st2;
  (3) getline (cin, st1);
  ```

  getline : read one line of characters (consuming the newline character)
  the reader marker moves to the beginning of the next line

- **using “ignore”** in data input

  the ignore function is used to skip (read and discard) characters in the input steam.

  ```
  cin.ignore(200, ‘\n’)
  ```

  what are the values of the variables, where is the read marker?

  ```
  Examples                                data
  (1)  cin >> v1 >> v2;                   957  34    1235\n
        cin.ignore(100, ‘\n’);            128  96  \n
        cin>> v2;

  (2)   cin >> ch1;                       A 22 B 16 C 19\n
         cin.ignore(100, ‘B’)
         cin >> ch2;

  (4)   cin.ignore(2, ‘\n’);              ABCDEF\n
       cin >> ch;
  ```

// change cin to myIn in each example

- **using “ignore” and “>>” in data input**

  Read the following from a data file, use integer variable “age” and string type variable “address”:

  ```
  34                            ← age of a customer
  2398 Wilder Rd. Atlanta     ← address
  20
  13 Hoston Ave. Nashville
  …
  ```

  ```
   while (myIn >> age)
   {
        cout << “Customer age: “ age << endl;

        myIn.ignore(100, ‘\n’);   // why is it necessary to use ignore here?
       getline(myIn, address);
   }
  ```