1. Strings
   1. Represented as char arrays
      1. Often is larger than necessary to accommodate the string.
   2. End with the **null terminator** (\0)
      1. Without it, the computer would not be able to determine where the string ends.
   3. Declaration
      1. “hello”
      2. char word[20];
      3. word[0] = ‘h’;
      4. word[1] = ‘e’;
      5. word[2] = ‘o’;
      6. word[3] = ‘l’;
      7. word[4] = ‘o’;
      8. word[5] = ‘\o’;
   4. Initializing
      1. Can do
         1. char str1[] = {“Hi y’all!”};
         2. char \* str2 = “By gosh”;
         3. char \* str3;
         4. str3 = “Go Knights!”;
      2. Cannot do
         1. char \* str4;
         2. str4 = {“Go Knights!”};
   5. Reading and printing
      1. scanf(“%s”, word);
         1. String is a pointer already and does not need &.
         2. scanf() stops reading as soon as it hits **whitespace** (i.e., a space).
            1. Whitespace is normally OK within “”s, but not as input to scanf().
      2. printf(“%s”, word);
2. String Functions Found in <string.h>
   1. strcpy(): copies the second string into the first string.
      1. Other than in initialization, you cannot use = to assign values to string variables, because they are arrays.
         1. Must use strcpy() as equivalent of “=”.
      2. Syntax
         1. strcpy(word, “Hello”);
   2. strlen: returns an int indicating string length.
      1. Not the size of the char array.
         1. Does not include NULL
      2. Example
         1. strlen(word);
         2. Returns 6
   3. strcat: concatenates (tacks on the second string to the end of the first string).
      1. += for strings
      2. Can also use variables
      3. Can handle 2 string variables as arguments, so long as the second has a value.
      4. Example 1
         1. strcat(“Hello”, “World”);
         2. Returns “Hello World”
      5. Example 2
         1. char word[] = “Hello”;
         2. strcat(word, “World”);
         3. Returns Hello World
   4. strcmp(): compares 2 strings *lexicographically* (like the dictionary).
      1. Returns 0 if strings are the same
      2. Returns – int if the first comes before the second.
      3. Returns + int if the first comes after the second.
      4. Example
         1. if(strcmp(word,”hello”) == 0)
         2. printf(“You entered \”Hello\”.\n”);

|  |  |
| --- | --- |
| **Integers** | **Strings** |
| a == b | strcmp(a, b) == 0 |
| a != b | strcmp(a, b) != 0 |
| a < b | strcmp(a, b) < 0 |
| a > b | strcmp(a, b) > 0 |
| a <= b | strcmp(a, b) <= 0 |
| a >= b | strcmp(a, b) >= 0 |

1. Arrays of Strings
   1. Declare a 2-dimensional arrays.
      1. First []: number of *rows*
         1. Number of strings
      2. Second []: number of *columns*
         1. Maximum length of each string
   2. Example: print whatever string is in word[i].
      1. printf(“%s\n”, words[i]);
      2. //What string is in *row* i of the array *word*