• more data types
• strings
• constants
• arrays of strings

what are strings
• a string in C++ is one of a special kind of complex data type called a class
• we will talk more about classes in detail at the end of the term
• a class is a compound data type, unlike the simple, native data types we've already discussed (e.g., int, char, bool, double and float)
• a class has members:
  it has data fields and functions

strings: declaring and initializing
• strings are declared like this:
  string s;
  where s is a variable whose data type is a string
• you can set the value of the string using the assignment operator and double quotes ("), e.g.:
  s = "hello";
• NOTE that you use single quotes for char values and double quotes for string values:
  char c = 'A';
  string s = "hello";
• ALSO NOTE that when you use the string class, you also need to include the string header file, in addition to the one(s) you've already been using:

strings: output
• we have already used strings for output, e.g.:
  cout << "hello" << endl;
• but we have not yet used variables that are declared as strings
• here’s how that works:

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string s = "hello";
  cout << s << endl;
} // end of main()
```
strings: concatenation operator

• the plus sign (+) is the concatenation operator, e.g.:

```cpp
string s1, s2, s3;
s1 = "david ";
s2 = "ortiz";
s3 = s1 + s2;
```

After the above code fragment, the value of `s3` will be "david ortiz"

strings: indexes

• a string is like an array of `char`

• so you can use the `index` of the individual characters of the string just like you can use the indexes of the individual elements of an array, like the arrays of `int` you created for the last homework assignment

• if you have:
  ```cpp
  string s = "ortiz";
  then: s[0] is assigned the value o (the letter "oh")
  s[1] is assigned the value r
  s[2] is assigned the value t
  s[3] is assigned the value i
  s[4] is assigned the value z
  
  you can also use the member function `at()` to find the value of an individual character of a string
  e.g., instead of using s[3], you can use s.at(3)
  ```

strings: length

• if you have:
  ```cpp
  string s = "ortiz";
  then the length of the string is 5
  
  there are two member functions of the string class that will tell you the length of a string: `length()` and `size()` (they do the same thing)
  you call them like this:
  ```cpp
  string s1;
  int n1, n2;
  s1 = "ortiz";
  n1 = s1.length();
  n2 = s1.size();
  ```
  After this code fragment, the value of `n1` will be 5 and so will the value of `n2`

strings: searching

• the `find()` member function is used to locate a substring within a primary string

• the function returns the value of the index in the primary string at which the substring starts, if the substring exists in the primary string; or else the function returns the constant `string::npos`

• for example:
  ```cpp
  string s1 = "david ortiz";
  int n1, n2;
  s1 = "avid"
  n1 = s1.find( "avid" );
  n2 = s1.find( "ask", 0 );
  ```
  After the above code fragment:
  the value of `n1` will be 1
  the value of `n2` will be `string::npos` or -1

• the first argument to the `find()` function is the substring to search for
• the second argument (which is OPTIONAL) to the `find()` function is the index in the primary string at which to start searching; 0 means to start searching at the beginning of
the string

strings: editing

• there are three editing member functions that are part of the string class:
  – insert()
  – replace()
  – erase()

• the insert() function inserts a substring into the primary string
  
  syntax:
  
  ```cpp
  mystring.insert( <pos1>, <str> );
  ```

  inserts the entire string str into mystring, starting at position pos1 in mystring

  for example:

  if mystring = "hello", then
    mystring.insert( 0, "goodbye" );
  will change the value of mystring to "goodbyehello"

  if mystring = "hello", then
    mystring.insert( 1, "goodbye" );
  will change the value of mystring to "hgoodbyehello"

  if mystring = "hello", then
    mystring.insert( 5, "goodbye" );
  will change the value of mystring to "hellogoodbye"

• the replace() function replaces a substring in one string with another string
  
  syntax:
  
  ```cpp
  mystring.replace( <pos1>, <pos2>, <str> );
  ```

  replaces the section of the string mystring between position <pos1> and <pos2> with string str

  for example:

  if mystring = "hello", then
    mystring.replace( 0, 3, "goodbye" );
  will change the value of mystring to "goodbyelo"
strings: parsing

- the `substr()` member function is used to extract a substring from within a primary string
- example:

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s1 = "D Ortiz";
    string s2;
    cout << "s1= " << s1 << endl;
    s2 = s1.substr(2, 5);
    cout << "s2= " << s2 << endl;
} // end main()
```

The output of the above program will be:
```
s1=D Ortiz
s2=ortiz
```

constants

- **constants** are types of data values that are defined in programs and do NOT change while the program runs
- these are similar to **variables** because they have a name, data type and value
- BUT they are DIFFERENT from variables because the value DOES NOT CHANGE
- some libraries define constants as well as functions
- you can also define your own constants
to define a constant, use the keyword `const`
for example:
```
#include <iostream>
using namespace std;

int main() {
    const int NORTH = 0;
    const int WEST = 1;
    const int SOUTH = 2;
    const int EAST = 3;
    cout << "the sprite is moving " << EAST << "\n";
} // end of main()
```

constants are handy for defining the length of an array
```
#include <iostream>
#include <time.h>
#include <stdlib.h>

int main() {
    // declare constant
    const int MAX = 100;
    // declare variables
    int a[MAX];
    int i;
    // initialize random number generator
    srand( time( NULL ) );
    // set entries in array to random numbers
    for ( i=0; i<MAX; i++ ) {
        a[i] = rand();
    }
    // output array entries
    for ( i=0; i<MAX; i++ ) {
        cout << a[i] << endl;
    }
} // end of main()
```

arrays of strings
because a string is a special kind of data type (called an object), you can also define arrays of strings, for example:
```
#include <iostream>
#include <string>
using namespace std;

int main() {
    const int MAX = 8;
    string myArray[MAX] = { "Last night I had the strangest dream", "I ever dreamed before", "I dreamed the world had all agreed", "To put an end to war", "I dreamed I saw a mighty room", "The room was filled with men", "And the paper they were signing said", "They'd never fight again" };

    cout << "here is your song: ";
    for ( int i=0; i<MAX; i++ ) {
        cout << i << "-th line = " << myArray[i] << endl;
    }
} // end of main()
```