Uninitialized Variables

- Recall that variables are declared before they are referenced.
- What happens when executing the following code?

```cpp
int anInt;
Cout << "The integer is " << anInt << endl;
```
**Uninitialized Variables**

- Variables that are referenced before assigning a value to them are *uninitialized variables*.
  - Used on the right hand side of an assignment
  - Used in an output statement
  - Used as a parameter to a function (*getline*)

- Dangerous and unpredictable
  - Usually see a Visual C++ warning
    - Syntactically OK, but semantically bad.

**Initializing Variables**

- In C++, variables can be initialized with an assignment statement just after the declaration.
  ```c++
  int anInt;
  anInt = 0;  // So we know what value anInt has.
  ```

- C++ also provides a shorter mechanism
  ```c++
  int anInt = 0;
  int anInt1 = 0, anInt2 = 0;
  ```
Initializing Variables

- Variables should always be initialized
  - Safety
  - Debugging
    - Use a special value that is known, easy to recognize, but not expected.

Strings

- We’ve seen string variables and constants several times.
  - How do we tell the compiler we want to use strings?
- Strings are sequences of characters
- What other kinds of things might we do with strings?
  - initialize
  - length
  - empty
  - concatenate
String Initialization

- `string` variables can be initialized with literal constants
  ```
  string president = "George Bush";
  ```
- Cannot break lines in the middle of a string literal
  ```
  string opening = "In a hole in the ground there lived a hobbit. Not a nasty, dirty, ...
  ```
- **Unterminated string** error from the compiler.

Escape Sequences

- Strings can contain escape sequences, which are interpreted when printed.
  ```
  string opening = "In a hole in the ground\nthere lived";
  cout << opening << endl;
  ```
  
  In a hole in the ground there lived
String Length

- You can find out how many characters are in a string by using the `length` function for strings.

Syntax

```
StringVariable.length()
```

Semantics

- Evaluates to the number of characters in the string.

String Length (Examples)

```cpp
string name = "Fred Flintstone";
int nameLen = name.length();

cout << "The length of the name is: " << nameLen << endl;

// Can be part of an expression as well
int lineLen;
cout << name;
lineLen = name.length() + lineLen;
```
String Length

- **Problem:** Assuming names are no longer than 15 characters, print an age right-justified at column 20.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary Baker</td>
<td>27</td>
</tr>
</tbody>
</table>

```
const int AGE_COLUMN = 20; // Where to justify the age.
cout << name;
cout << setw(AGE_COLUMN - name.length()) << age;
```

String Length (Exercise)

- Assuming that lines are 80 characters wide and the string to print is less than 80 characters, what C++ statements center a string on a line?
Empty Strings

- There is a special string called the **empty string** 
  
- Can test if a string is empty by using the `empty` function for strings.

**Syntax**

```
StringVariable.empty()
```

**Semantics**

- Evaluates to `true` if the string in `StringVariable` is empty or `false` otherwise.

Empty Strings (Examples)

```c++
string emptyString = "";
string name = "Fred Flintstone";
bool isEmpty;

isEmpty = emptyString.empty(); // true
isEmpty = name.empty(); // false
```

- Uninitialized string variables are not necessarily empty!
Concatenating Strings

- Two strings can be put together or \textit{concatenated} by using the $+$ operator.
  - Can update string variables too

```cpp
string s1 = "Hello ", s2 = "World!";
cout << s1 << s2 << endl; // Prints Hello World!
cout << s1 + s2 << endl; // Also prints Hello World!
string s3 = s1 + s2;
cout << s3 << endl; // Again prints Hello World!
```

Strings and File Names

- You may want to use a string variable or constant to store a file name.
- The open function for file streams require C style strings, not C++ style strings.
- Use the \texttt{c\_str()} function for strings to get a C style string.
- Syntax

```cpp
StringVariable.c\_str()
```
Strings and File Names (Examples)

```cpp
const string IN_FILENAME = "InputData.txt";

// Opens the file named InputData.txt for input
ifstream In;
In.open(IN_FILENAME.c_str());

// alternatively, you can declare and open a file stream
// all at once...
ifstream In(IN_FILENAME.c_str());
```