

# Strings and arrays: Storing collections of data

**William Hendrix,**

presented by:

**Ali Gok**

*Lecture 8*

# Outline

---

- Strings
- Static arrays

# Strings

---

- Object containing a sequence of `char` values
  - Exclusive to C++
  - Not exactly the same thing as `cout << "message";`
- To use: add `#include <string>` to preamble
- To declare:
  - `string str;`
  - `string str("initial value");`
  - If no value specified initially, stores empty string (`""`)
    - Never has garbage data from memory
- Main operation: concatenation (+)
  - Result of `str1 + str2` puts `str2` immediately after `str1`
    - Does not add a space automatically
  - Can also concatenate with other data types
    - `char, int, double, bool, "literal text", short, etc.`
    - Other types are converted to strings
    - **Pitfall:** cannot concatenate two string literals ("messages")
- Reading in a string: `getline(str, cin);`
  - Reads entire line, not just until first space (unlike `cin >> str;`)

# Extracting values from strings

---

- **String length:** `str.length()`
  - `str` is name of the `string` whose length you want
- **Accessing a single char**
  - Use brackets with an index
    - E.g., `str[0]`
    - Characters are numbered starting with 0, not 1
      - Similar to convention for `for (int i = 0; i < 10; i++)`
    - Will generate an *exception* if index is negative or past end of string
      - Halts program (we will discuss later)
  - Alternate: `str.at(0)`
    - Result is identical to brackets
- **Accessing a substring**
  - `str.substring(x, y)`
    - Returns substring of length `y` starting at index `x`
  - `str.substring(x)`
    - Returns substring starting at index `x` to the end of the string

# String example

- What is printed out by the following code?

```
string cmd("var_1 = 6.283 * r");
```

```
cout << cmd[3] << endl;
```

```
cout << cmd.substr(9) << endl;
```

```
cout << cmd.substr(0, 5) << endl;
```

```
cout << cmd.at(cmd.length() / 2) << endl;
```

<b>v</b>	<b>a</b>	<b>r</b>	<b>_</b>	<b>1</b>		<b>=</b>		<b>6</b>	<b>.</b>	<b>2</b>	<b>8</b>	<b>3</b>		<b>*</b>		<b>r</b>	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

# Sequences of other data types

- **Motivating example:** calculating standard deviation

$$\sigma = \frac{1}{\sqrt{n}} \sum_{i=1}^n (a_i - \mu)^2$$

          ↑          ↑          ↑  
Count  Values  Mean

- We can calculate mean without saving values
- How do we take the difference between each value and the mean?<sup>\*</sup>
  - The user is not going to enter them in again for you
- By saving the sequence of values, we can reuse them
- We can save sequences in *arrays*

<sup>\*</sup>It's possible to calculate standard deviation without storing the values

# Arrays

---

- Two types of array
  - Static: must know max size before program starts
  - Dynamic: can set and reset size as program continues
- Static arrays
  - To declare: `int num[SIZE_OF_ARRAY];`
    - Works with any data type
  - Accessing elements: `num[index]`
    - `index` begins at 0
    - Accessing an out-of-bounds element will usually crash your program
      - Could potentially change other program variables instead
    - Elements can be on LHS of assignment operation
  - Shortcut: array initialization
    - `int num[] = {1, 4, 9, 16, 25};`
      - Size not required (counted by compiler)
      - Braces notation *only* valid when declaring
      - Array cannot be on LHS of assignment, only single elements

# Detour: string literals

---

- String literals (“messages”) technically `const char[]`
  - Cannot be changed or resized
- Always terminated with null character ( `'\0'` )
  - `string` handles `\0` character automatically
- **Pitfall:** Do not forget the terminating null character if you manipulate `char` arrays directly
  - `cout` will continue printing everything in memory until it:
    - Finds a 0 byte ( `'\0'` )
    - Accesses memory that doesn't belong to your program
      - Segmentation fault
- Add 1 to max string length when allocating to account for `\0`



# Tonight

---

**Lab 2** is due Tuesday at noon

**Recommended reading:** Sections 9.1 and 9.3