PIC 10A 2B

TA: Bumsu Kim

UCLA

Today…

- Arrays
- Exercises on Arrays
- HW5 Hints

UCLA

Vectors, **Arrays**, and Pointers

Basic Syntax

// Comparisons Between Vector and Arrays, and Arrays and Pointers

// Pointer Arithmetic

Arrays

- Consider a string variable. It consists of a *sequence of characters*
- Likewise, we can also think of a *sequence of "some other data type"*
  - For instance, a sequence of 6 integers

123	456	789	10	11	12

- In fact, the most "basic" object in C++ for a sequence of data is "Array"
  - Unlike a string, it doesn't have additional features (e.g. member functions) like length(), substr(), etc.
  - In fact, strings are indeed a "class" defined using C++ arrays (to be covered later)

UCLA

Arrays

- Array declaration/definition

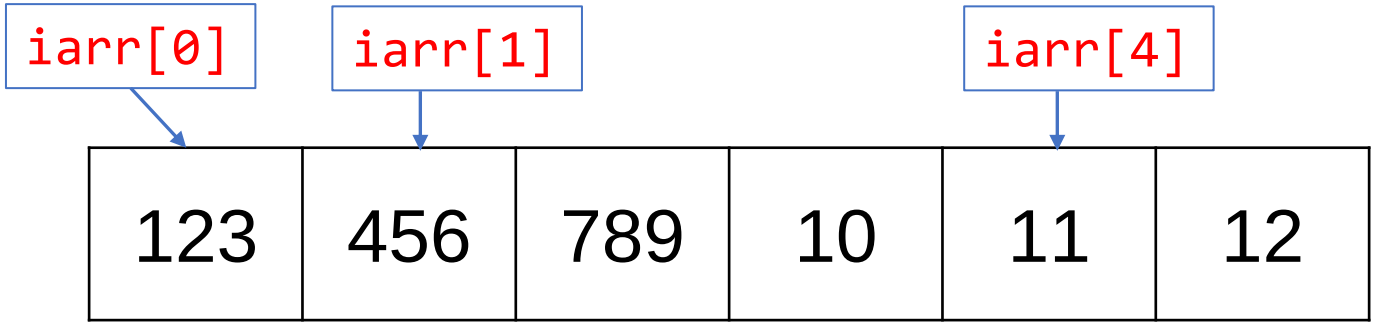
```
int arr[7] =
       { 0, 1, 2, 3, 4, 5, 6 }; // initialization
double darr[10] = { 0.1, 0.2 }; // legal?
char    uninitialized_carr[10]; // legal?
```

- Does not need to be initialized (*legal* in terms of syntax)
- However, ALWAYS initialize your variable (*better* coding practice!)
- Brace initialization
  - #elts in the braces can be smaller or equal to the size of the array
  - (larger → compilation error)
  - If #elts is strictly smaller, it fills the array from the front, and the rest will be *"empty-initialized"*
    - All numeric types becomes 0 (i.e. null character for `char`, false for `bool`, etc.)

UCLA

Arrays

- An int array of size 6 may look like (in the memory):



- "`iarr`" is the name of the array object
- This "`iarr`" actually "points to" the address of the part of the memory that stores the value 123 (or, `iarr`[0])
- Accessing the k-th element of `iarr`: `iarr`[k] (subscript operator)

`sizeof` Operator

- `sizeof` operator measures the size of the type in bytes
  - e.g. What is the `sizeof(char)`?

    A. 1 B. 2 C. 4 D. 8
  - c.f. 1 byte = 8 bits, and can express $2^8 = 256$ different values (i.e. characters)
- Sizes of fundamental types
  - C++ standard does NOT specify the size of integral types in bytes, but it specifies minimum ranges (e.g. [-32767, 32767] for int)
  - In VS 2022, we have:

```
size of int: 4
size of unsigned int: 4
size of short: 2
size of long: 4
size of long long: 8
size of char: 1
size of bool: 1
size of float: 4
size of double: 8
```

UCLA

Arrays

- Arrays cannot change its size "dynamically"
  - Its size should be set in compile-time, and cannot be changed

- For instance, you can't do:
  - The int variable **`sz`** is not a compile-time constant

```
int sz = 5;
int iarr[sz] = {};
```

- On the other hand, you can do:
  - **`sz`** is now a compile-time constant

```
const int sz = 5;
int iarr[sz] = {};
```

- However, it doesn't mean that `const int` is always a compile-time constant
  - You can't do:
  - To make it clear, use the `constexpr` keyword

```
int get_num() {
    int n;
    cin >> n;
    return n;
}
int main() {
    const int sz = get_num();
    int iarr[sz] = {};
```

UCLA

Array Exercises

- Week7_1_Array_Exercises.cpp on Github
  - `PrintArr`
  - `maxArr`
  - `sumRange`

UCLA

HW5 Hints

- Swapping two strings is easy (recall how we did it TWO weeks ago)
- For combining, construct a new string and add characters to it
  - 1. Figure out which string is “longer” (say `size1` >= `size2`)
  - 2. Run a for loop for `i` = 0, 1, …, size1 (larger!)
  - 3. In the loop body, first check if `i` exceeds the size of the string, and add it if not
    - Be careful about the order of adding characters
    - e.g. If you swapped the strings already, then which one should come first?

UCLA

Your Feedback is welcome

- Don’t hesitate to give a feedback on the discussion
- Use the link on my Github repo, or the link below:
  - https://forms.gle/erZj1iSgHNrHQuXk6

My Github repo on the web looks like:

code	Week2 Tu
LICENSE	Initial commit
README.md	Update README.md

README.md

PIC10A

PIC10A discussion 2B, UCLA for Fall 2022

Google form link for feedbacks: https://forms.gle/erZj1iSgHNrHQuXk6

Click this link

UCLA