CS 161
Introduction to CS I
Lecture 17

• Creating static arrays
• How to pass arrays to functions
• Working with C-style strings
Warning

• Incidents involving plagiarism, of English and of code, have occurred in this class
  • Penalties for academic misconduct are severe
• Ensure that you submit your own work!
  • You can discuss the assignment with other students
  • You cannot:
    • Look at another student's code
    • Copy another student's code
    • Submit a modified version of another student's code
    • Show another student your code
    • Include code you found on the Internet in your program
Review

• What is the purpose of the heap?
  • Allow the amount of memory used to dynamically change during runtime
  • E.g. operating systems, web servers, anything with user interaction
Real-life examples of arrays

• Seats in the classroom
• Keys on a keyboard
• Rooms in a dormitory
• Houses in a subdivision
Arrays enable easy iteration

1. `string page[1024]; /* book with 1024 pages */`
2. `cout << page[0] << endl; /* print page 0 */`
3. `cout << page[10] << endl; /* print page 10 */`

4. /* Loop over all pages */
5. `for (int p = 0; p < 1024; p++)`
6. `cout << page[p] << endl; /* print page p */`

Indexing
Arrays in C++

- Multiple items of the same data type
- Stored in contiguous memory locations
- Example:
  - `int grades[5];`
- Questions:
  - Stack or heap?
  - Access an item by its index: `grades[0]`, `grades[1]`, ...
  - Array name = address of first element (`grades[0]`)
  - Initial values?
Static arrays in C++

• Declare but don't initialize

1. `int grades[5];`
2. `for(int i=0; i<5; i++)`
3. `cout << grades[i] << "", ";`
4. `cout << endl;`
Static arrays in C++

• Declare and initialize

1. int grades[5] = {90, 80, 85, 95, 100};
2. for (int i=0; i<5; i++)
3.   cout << grades[i] << ", ";
4. cout << endl;
5. /* {}: initializer; cannot use to assign */
6. //grades = {82, 98, 87, 99, 93};
Static arrays in C++

- Initialization methods

1. `int grades[5] = {0, 0, 0, 0, 0}; /* {} : initializer */`

1. `int grades[5] = {}; /* another way to set all 0s */`

1. `int grades[] = {4,3,1,7,2}; /* can omit size w/init */`
Static arrays in C++

- Declare and initialize with loops
- Which version is better?

A
1. int grades[5];
2. for (int i=0; i<5; i++)
3.   grades[i] = 0;

B
1. int grades[5];
2. int i = 0;
3. while (i < 5) {
4.   grades[i] = 0;
5.   i++;
6. } 

Both work, but version B is less clear to read, and more likely to have bugs.
Static arrays in C++

• Declare and initialize with loops
• Even better (why?):

```c
1. const int n_people = 5;
2. int heights[n_people];
3. for (int i=0; i<n_people; i++)
4.    heights[i] = 0;
```
Your turn: User input to array

• Write a for loop to read values from the user and store them in this array:

1. `const int n_people = 5;`
2. `int heights[n_people];`
3. `for` ...
Your turn: User input to array

• Write a for loop to read values from the user and store them in this array:

```c++
1. const int n_people = 5;
2. int heights[n_people];
3. for (int i=0; i<n_people; i++) {
   4.   cout << "Enter height: ";
   5.   cin >> heights[i];
6. }
```
Arrays use pointers

- Name of array holds the address of the first (zeroth) item

1. `int grades[5] = {90, 80, 85, 95, 100};`
2. `cout << grades << endl;`
3. `cout << grades[0] << endl;`
4. `cout << &grades[0] << endl;`
5. `cout << *grades << endl; /* same as grades[0] */`
"Son of A Dark and Stormy Pointer": A Play

1. `int* captain = NULL;`
2. `int soldier[3];`
3. `soldier[0] = 9;`
4. `soldier[1] = 7;`
5. `soldier[2] = 3;`
6. `soldier[0] = soldier[1] + soldier[2];`
7. `soldier[2]++;`
8. `captain = &soldier[1]; /* address-of */`
9. `captain++;`
10. `*captain = 4;`
11. `soldier[1] = *captain; /* dereference */`
12. `soldier[2]++;`
13. `captain = soldier;`
C-style strings

- Existed before the C++ "string" class we have been using
- C-style string = array of characters ending with '\0' (null)
  - Must allocate space for #chars you want plus 1
- To access C-style string functions, #include <cstring>

1. char name[5] = {}; /* 4 characters plus '\0' */
2. cin.getline(name, 5); /* 5 includes '\0' */
3. cout << name << "", length " << strlen(name) << endl;

Your new best friend
C-style strings

• Initialize with array initializer and null terminator

```c
1. char name[5] = {'L', 'u', 'k', 'e', '\0'};
```

• Easier to read:

```c
1. char name[5] = {"Luke"};  /* adds \0 for you */
```
Assignment 4: Text Surgeon

• Read in a line of text from the user, and perform analysis and manipulation of that string
  • check_vowel_cons()
  • letter_swap()
  • flip_str()
  • count_chars()
  • + your own operation: permute characters? inject random characters? doubledouble stringstring? <creativity opportunity>

• You will use char arrays, not "string" objects
• Design Document is due Feb. 16
More C-style string functions in `<cstring>`

- `strlen()` – length of string up to (not including) null terminator
- `strcpy()` – copy contents of one C-style string into another
  - safer: `strncpy()` – copy a specified number of characters
- `strcmp()` – compare one C-style string with another
  - return 0 if they are the same
- `strcat()` – concatenate one string to another
- `strstr()` – search for one string in another
  - return NULL if not found
- `cin.get()` – take one char from the buffer at a time
- `cin.getline()` – take an entire line of determined size

http://www.cplusplus.com/reference/cstring/
What vocabulary did we learn today?

• Array
• Index
• C-style string (char array with null terminator)
What ideas and skills did we learn today?

• How to declare arrays on the stack
• Array initialization
• How to create C-style strings
  • Character arrays that must be null-terminated (\'0\')
• Useful C-style string functions
Week 6 nearly done!

- **Minute paper**: Please write down on scratch paper (leave in box):
  1. One thing you learned today
  2. One concept you find confusing

- Attend lab (laptop required)
- Read **Rao Lesson 4** (pp. 63-71, 76-79)
  - **Rao Lesson 7** (pp. 165-166)

- **Assignment 4 Design** (due **Sunday, Feb. 16**)

See you Monday!

2/14/2020 CS 161