CS 161
Introduction to CS I
Lecture 22

• Review for Midterm 2
Midterm 2: Friday 2/28 in LINC 100

- Midterm 2: content through week 7 (but no structs)
- You cannot use cell phones, calculators, tablets, laptops, or other devices, notes, books, Internet access, friends, etc.
- You will be required to sign a Statement of Academic Integrity on the exam for it to be graded
- If you need scratch paper, raise your hand (it will be collected)
- Thursday 2/27: Evening review – 6-7 p.m. in LINC 228
- Friday 2/28: Midterm – 2-2:50 p.m., LINC 100
- Format: true/false, multiple choice, one page short answer
  - Scantron sheet: fill in bubbles with #2 pencil
  - Bring to midterm: student ID and #2 pencil(s)
Midterm 2: From Midterm 1

• Data types and min/max ranges
  • base types: \texttt{bool, char, short, int, long, float, double}
  • signed vs. unsigned
• Expressions
  • Parentheses: $12 / (3 + 1)$
  • Integer vs. floating point math:
    
    \[
    (17-4) / 2 \quad \text{vs.} \quad (17-4) / 2.0
    \]
Midterm 2: From Midterm 1

• Operators
  • Arithmetic: + - * / % ++ --
  • Relational: < <= > >= == !=
  • Logical: && || !
• Indexing: []
• Memory: &(address-of) *(deref) .(member) ->(deref+member)
• Precedence


Operator precedence

<table>
<thead>
<tr>
<th>a++</th>
<th>a--</th>
<th>[]</th>
<th>.</th>
<th>-&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>++a</td>
<td>--a</td>
<td>*p</td>
<td>&amp;a</td>
</tr>
<tr>
<td>*</td>
<td>/</td>
<td>%</td>
<td>*</td>
<td>/</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;=</td>
<td>&gt;</td>
<td>&gt;=</td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>!=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=+=</td>
<td>-=</td>
<td>*=</td>
<td>/=</td>
<td>%=</td>
</tr>
</tbody>
</table>
Midterm 2: From Midterm 1

• Conditional statements
  • if-then
  • switch
  • break

• Loops
  • for
  • while
  • do-while
  • break
  • When to use each?
Midterm 2: From Midterm 1

• Random numbers
  • Generate random numbers between 20 and 25 (inclusive)
  • Generate random numbers between -3 and 5 (inclusive)

• Variable scope (visibility)
  and shadowing
Midterm 2: From Midterm 1

- Random numbers
  - Generate random numbers between 20 and 25 (inclusive)
    \[ \text{rand()} \% 6 + 20 \]
  - Generate random numbers between -3 and 5 (inclusive)
    \[ \text{rand()} \% 9 - 3 \]
- Variable scope (visibility) and shadowing

```c++
int m = 3;
if (m > 0) {
    int m = 43;
    cout << m++ << endl;
}
cout << m << endl;
```
Midterm 2: Functions

• Function declaration vs. definition?

• Parts of a function declaration/definition?

• How to call a function?

• Pass by value vs. pass by reference
Midterm 2: Functions

• Function declaration vs. definition?
  Declaration has return type, name, parameters; definition has code body

• Parts of a function declaration/definition?
  Return type, name, names and types of parameters

• How to call a function?
  \texttt{retval = fn\_name(argument1, argument2, ...);}

• Pass by value vs. pass by reference
  \texttt{Value: make a copy; reference: pass the address (can modify value)
Midterm 2: Functions

• What is function overloading?

• What is a case where function overloading is ambiguous?

• What are default arguments?

• Where must they appear in the function parameter list?
Midterm 2: Functions

• What is function overloading?
  Same function name but different number or type of parameters

• What is a case where function overloading is ambiguous?
  Different return types but same parameter types

• What are default arguments?
  Placeholder values that will be used if the caller does not specify a value

• Where must they appear in the function parameter list?
  At the end of the parameter list
Midterm 2: References and Pointers

• How do you declare a reference to another variable (char d)?

• How do you declare a pointer?

• How do you assign a pointer to point to an existing variable (d)?

• What are 2 ways to print the value in d?

• How do you print the value p points to?
Midterm 2: References and Pointers

• How do you declare a reference to another variable (char d)?
  ```
  char& z = d;
  ```

• How do you declare a pointer?
  ```
  char* p = NULL;
  ```

• How do you assign a pointer to point to an existing variable (d)?
  ```
  p = &d;
  ```

• What are 2 ways to print the value in d?
  ```
  cout << d << endl;  cout << z << endl;
  ```

• How do you print the value p points to?
  ```
  cout << *p << endl;
  ```
Midterm 2: References versus Pointers

• Do not confuse "reference" (a data type) with "pass by reference" (something that happens when you call a function)

• **Reference**: an alias to some variable (permanent)
  - `int& r = s;`
  - Can assign new values to `r` (which is `s`), but cannot make `r` be an alias to another variable later
  - Must be initialized when declared

• **Pointer**: stores the address of some variable
  - `int* p = &s;`
  - Can change what address `p` contains (where it points to) anytime
  - Can be declared, then initialized later
Midterm 2: 1-dimensional arrays

- How do you declare a static array (e.g., of shorts)?

- How do you print item at index 3 in an array?

- If you print the name of the array (cout << arr), what is displayed?

- If you dereference the array (*arr), what do you get?

- How do you pass an array to a function?
Midterm 2: 1-dimensional arrays

- How do you declare a static array (e.g., of shorts)?
  ```
  short array[4];
  ```
- How do you print item at index 3 in an array?
  ```
  cout << array[3] << endl;
  ```
- If you print the name of the array (cout << arr), what is displayed?
  Memory location (address) of first item (array[0])
- If you dereference the array (*arr), what do you get?
  Value of first item (array[0])
- How do you pass an array to a function?
  ```
  fn(array);
  ```
Midterm 2: Dynamic memory allocation

• What is the difference between the stack and the heap?

• When would you use the heap?

• How do you allocate memory (e.g., an integer) from the heap?

• How do you free the memory for an integer on the heap?
Midterm 2: Dynamic memory allocation

- What is the difference between the stack and the heap?
  Stack is statically allocated (in advance); heap is dynamically allocated.

- When would you use the heap?
  To allow memory consumption to grow and shrink as needed; sizes (or numbers of items) are not known in advance.

- How do you allocate memory (e.g., an integer) from the heap?
  ```c++
  int* d = new int;
  ```

- How do you free the memory for an integer on the heap?
  ```c++
  delete d;
  ```
Midterm 2: Dynamic memory allocation

• How do you allocate a 1-D array from the heap (e.g., short)?

• How do you free memory for a 1-D array on the heap?
Midterm 2: Dynamic memory allocation

• How do you allocate a 1-D array from the heap (e.g., short)?
  ```
  short* array = new short[17];
  ```

• How do you free memory for a 1-D array on the heap?
  ```
  delete [] array;
  ```
Midterm 2: C-style strings

• What kind of array is a C-style string?
• What library do you #include to access C-style string functions?

• What special item must a C-style string have? Why?

• cin >> c_string; reads user input and stops when?

• cin.getline(c_string, 10); reads how many characters from the user into c_string?
Midterm 2: C-style strings

• What kind of array is a C-style string?  
  \texttt{char[]}  

• What library do you \#include to access C-style string functions?  
  \texttt{\#include <cstring>}

• What special item must a C-style string have?  Why?  
  Null terminator ('\0' character), so functions know when string ends

• \texttt{cin >> c\_string;}  reads user input and stops when?  
  Stops at first whitespace (space, tab, newline, etc.)

• \texttt{cin.getline(c\_string, 10);}  reads how many characters 
  from the user into \texttt{c\_string}?
  9 characters and adds the null terminator '\0' to make 10
Midterm 2: 2-dimensional arrays

• How do you declare a static 2-D array (e.g., 4x5 double)?

• This memory is laid out in row-major or column-major order?

• How do you allocate memory for a dynamic 2-D array?

• How do you free memory for a dynamic 2-D array?
Midterm 2: 2-dimensional arrays

• How do you declare a static 2-D array (e.g., 4x5 double)?

```cpp
double array[4][5];
```

• This memory is laid out in Row-major or column-major order?

• How do you allocate memory for a dynamic 2-D array?

```cpp
double** array = new double*[4];
for (int i=0; i<4; i++)
    array[i] = new double[5];
```

• How do you free memory for a dynamic 2-D array?

```cpp
for (int i=0; i<4; i++)
    delete [] array[i];
delete [] array;
array = NULL;
```
Midterm 2: 2-dimensional arrays

• Given a 2-D (5x3) static array of ints, what type should be in the function definition to accept it?

• Given a 2-D (5x3) dynamic array of ints, what type should be in the function definition to accept it?
Midterm 2: 2-dimensional arrays

• Given a 2-D (5x3) **static** array of ints, what type should be in the function definition to accept it?

  ```c
  void my_fun(int arr[][3]);
  void my_fun(int arr[5][3]);
  ```

• Given a 2-D (5x3) **dynamic** array of ints, what type should be in the function definition to accept it?

  ```c
  void my_fun(int** arr);
  void my_fun(int* arr[]);
  ```
Week 8 continues

- Prepare for Midterm 2 (Friday, Feb. 28) – review practice questions and answers, ask questions on Piazza, come to office hours
- Attend Midterm 2 Review (Thurs., 6-7 p.m. in LINC 228)
- Attend lab 8 (laptop required)
- Continue working on Assignment 5 design (due Sunday, Mar. 1)

See you Friday! Bring your OSU student ID and #2 pencil(s)!