Basics

- #include includes libraries
  - ↳ iostream, string, cctype, cstring, cmath
- namespace - collection of classes and functions
- Basic types: int, double, char, bool
- Integer division truncates after the decimal

Strings

- Access specific characters with [ ]
- .size() returns # of characters in a string
- .substr(startIndex, length) returns a string including startIndex of size length
- Necessary to use cin.ignore(...) if after cin an int/double, you need to use getline(...)
  - ↳ getline(...) consumes '\n'

Conditionals

- If statements run code within the curly braces following it
  - ↳ if no braces, it will only run the first line of code following it
- Value tested in switch statements must be convertible to an integral type
  - ↳ int, char, short, long, etc.
  - ↳ NOT strings
- Break statement needed to leave the switch, or else all following cases will also be executed
- And operator has a higher precedence than the Or operator
- Not (A and B) → (not A) or (not B)
- Not (A or B) → (not A) and (not B)
- Big difference between = and ==
- Any non-zero value is "true"

Loops

- In while loops, always ensure the condition will eventually be false
- In for loops:
  - ↳ the declaration is run 1st
  - ↳ the condition is evaluated before any code is run
  - ↳ the action is run after the code block

Arrays

- Can declare:
  - ↳ int arr[10];
  - ↳ string words[constant var];
  - ↳ int arr[] = {1, 2, 3};
    - ↳ non-constant variables cannot be used for the size
- Arrays are passed by reference by default
- Arrays do not pass their size to functions
- Arrays can be of longer length/larger size than the # of interesting elements they hold
  - ↳ # of interesting elements = position of the next available spot
- ALWAYS make sure the subscript (array[subscript]) is within bounds
  - ↳ not negative, not over the array's declared length
- Arrays cannot be printed directly, you must use a loop
- It is impossible to check if an index is too big for the array
- You cannot pass const arrays into functions that do not promise to leave it unchanged
  - ↳ compile error

Characters

- String ≠ character
- 0 → 9 are contiguous
  - ↳ 'number' - '0' = int value of number
- Letters/symbols are not contiguous
- Letters organized lexicographically
  - ↳ 'a' < 'z'
- No guarantee on order of uppercase/lowercase
- Shorter < longer for strings

C Strings

- Represented w/ arrays of chars
  - ↳ uninitialized values given a neutral value
  - ↳ always ends in a '\0'
    - ↳ automatic if space
- char s[100] → not empty
- getline(...) → cin.getline(arr, limit)
- s = t doesn't work
- #include <cstring>
  - ↳ strlen(cstring) → returns length
  - ↳ strcpy(s, t) → copies t to s
    - ↳ undefined if destination isn't large enough
  - ↳ strcat(s, "!!!") → appends s with "!!!"
    - ↳ 2nd argument must contain a null byte
  - ↳ strcmp(x, y) → negative if x < y, 0 if x == y, positive if x > y

2-D Arrays

- int x[rows][cols]
- In function declarations, you only need to write the # of columns
  - ↳ int f(int x[][numCols])
- 2-D arrays are an array of arrays
  - ↳ x[2] → 2nd row of 2D array

Misc.

- ALWAYS check division by 0 if using division with a variable
- cout.setf(ios::fixed);
  cout.precision(# of decimal places)
- Nested function declarations are not allowed
- static_cast<type>(x) turns x into type if possible
- When constructing C strings ALWAYS remember the null byte
- strlen doesn't include the null byte
- .size() - 1 - i to access the end of a string

Pointers

- & - used to determine the address of a variable to store in the ptr.
- * - accesses the value of the variable the ptr. points to
- A ptr. to a double cannot point to an int, etc.
- &a[i] + j = &a[i+j]
- &a[i] < &a[j] if i < j
- &a[i] - &a[j] = i - j
- Deleteing a variable that a ptr. points to leads to undefined behavior
- Local variables are in the stack
- Dynamic variables are in the heap

Structs

- Don't forget the semicolon
- Primitive data members are uninitialized by default
- Classes will call their default constructor (strings are "")
- Data members default to public

Classes

- Data members default to private
- Generally used for more complex structures
- Constructors are member functions that initialize objects
  - ↳ compiler has one by default
  - ↳ defined in/outside the class
  - ↳ do not call with .
- Destructors are called when objects pass out of scope
- Copy constructors construct an exact copy of the object
  - ↳ called when functions return a class, passing-by-value, etc.
  - ↳ the default copy constructor doesn't work with dynamic variables/ptrs