Memory Layout for C++

Three types of memory
- Static - lifetime is entire program execution
  - Code
  - Static variables
  - Top-level variables
  - Constants (strings, numbers etc.)
- Stack - lifetime is until function/block is exited
  - Local variables of functions
  - Blocks
    (Some allocation functions allocate here)
- Dynamic or Heap - lifetime is until deallocated
  - Allocated by new (and malloc)
  - Released by delete (and free)

Representation Of Some C++ Types

Basic types (int, float, double, char, bool, ...)
- A sequence of bytes representing the value

Pointers (TYPE *)
- Bytes representing the address of the pointed-to TYPE
- All pointers are the same size
- A zero pointer is legal and points to nothing

References (TYPE &)
- These are synonyms for another name and may not exist in memory
- Can be represented the same way as a pointer with automatic dereferencing
- enum
  - An integer representing the alternative

Arrays (TYPE x[ ])
- A sequence of TYPE
- There may be padding between the elements of the array to align elements on word boundaries

Functions and Methods
- Actual code
- Programmer never uses code as data
- Most use of functions and methods is through a pointer

Compound Types

struct and class
- A sequence of the fields of the class or struct
- The fields may be rearranged and aligned on word boundaries
- Static fields do not appear in the object
- Methods do not appear in the object
  - If there are any virtual functions then there is a "virtual function table" pointer that points to an array of function pointers
  - This is how polymorphism works

union
- Same as a struct or class except every member is assigned to the same address in the object - all members are on top of one another

"Null Terminated Byte Strings"
- Same as an array of char
- Last character is a null (zero) character