Objects: Instances of classes, responsible for themselves

Keywords
Public: Accessible
Private: Variables/methods that only the class members can access

Primitive Data Types
Bit - a single ‘0’ or ‘1’
Byte - a collection of 8 bits
char - 1 byte - can handle 256 ASCII characters
int - 4 bytes

Arrays
Linear organizations of collections of variables of the same type
`int my_array [100];` creates a space for 100 ints to live
`int* myarray = new int(100);` does the same thing, but creates the memory on the heap rather than the stack

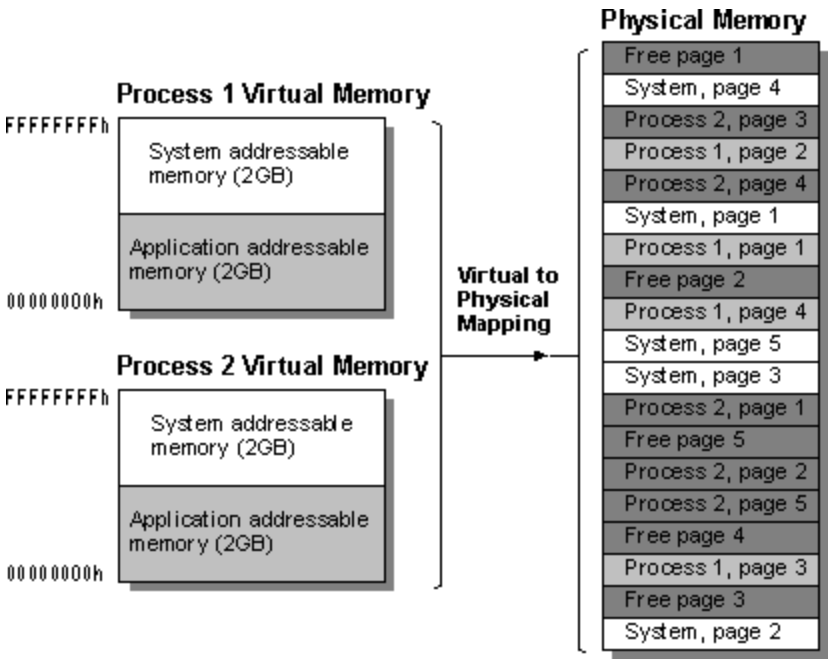
Number Conversion
11 in Binary => 3 in Decimal
11 in Hexadecimal => 17 in Decimal

Pointers - point to memory addresses
A string in C++ is a char*, a pointer to the first char in a sequence.
`char arr[5]` is the same as `char* arr = new char(5);`, but the first call lands in the stack and the second one lands in the heap.

Stack v Heap
Using the `new` keyword makes your object be created in the heap, must use `delete` to get rid of it explicitly.
Stack automatically allocates and de-allocates based on scope.
Heap is explicitly allocated (new) and de-allocated (delete). Used when you don’t know how much memory you need at run time.

Shallow Copy - Creates a pointer to the original object
Deep Copy - Copies the entire object

Real World Pointers
Virtual to physical memory -



Disk Organization - first possible fit with lots of pointers

Student example:
// Real world example with public and private - only allow an external library to interact

```
#include <stdio.h>
#include <stdlib.h>

#DEFINE MAX_NAME_LENGTH 5

public class student
{
  public:
        char name [MAX_NAME_LENGTH];
        int student_id;
        bool is_currently_enrolled();
        bool enroll_student();
  private:
        bool currently_enrolled;
        bool student_has_paid;
};

bool student::is_currently_enrolled()
{
  return currently_enrolled;
```

```
}

bool student::enroll_student()
{
  if ( student_has_paid )
  {
     currently_enrolled = true;
  }
  return currently_enrolled;
}
```