Object Orientation

Why We Do It
Typical “Early” Code

• Often, when programming is taught, the majority of the focus is on learning to use basic data types, programming logic, and functions.

• Much of the program is often thrown into one main method, which might call one or two other functions.
Typical “Early” Code

• It’s readily apparent that any program has (at least) two fundamental component categories that the user must define and manage.
  – Data – the information received, output, and maintained by the program
  – Functions/Methods – the programming logic that manipulates data as needed.
Typical “Early” Data

- For (nearly) any program to serve a useful purpose, it will need to meaningfully store and use some type of data.
  - What are some of the basic data types that you’ve used to this point in programming?
Typical “Early” Data

- int; long; char; float; double; bool; string; and array [];
  - Combination of these types can represent more complex types
  - Two basic types: integers and float
  - Integers are whole numbers
  - Integers can be signed or unsigned
## Typical Ranges

- Typical ranges on a 32-bit machine:

<table>
<thead>
<tr>
<th>Type</th>
<th>Smallest Value</th>
<th>Largest Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>short int</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>unsigned short int</td>
<td>0</td>
<td>65,535</td>
</tr>
<tr>
<td>int</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>unsigned int</td>
<td>0</td>
<td>4,294,967,295</td>
</tr>
<tr>
<td>long int</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>unsigned long int</td>
<td>0</td>
<td>4,294,967,295</td>
</tr>
</tbody>
</table>
C++ vs. C languages

```c++
#include <iostream>

using namespace std;

class cl { int i; // private by default
    public: int get_i(); int put_i(int j); }

int cl::get_i() {return i;}  
int cl::put_i(int j) {i = j;}

int main()
{
    cl s;
    s.put_i(10);
    cout << s.get_i() << endl;
    return 0;
}
```

```c
#include <iostream>

int main()
{
    int l, j=10;

    i=j;
    printf ("%d \n");
    return 0;
}
```
A Rough Exercise

• Suppose we wanted to write a program for playing a card game of some sort.
  – Like with Hearts or Spades, the full deck is dealt to four players.
• Disregarding the rules of the game... how would we manage the cards?
A Rough Exercise

• Cards are important information/data to keep track of for a card game.
• What manipulates cards, and how would this have to be coded?
  – Shuffling
  – Dealing
  – Each player has a separate hand...
A Rough Exercise

- The following program illustrates both two-dimensional arrays and constant arrays.
- The program deals a random hand from a standard deck of playing cards.
- Each card in a standard deck has a suit (clubs, diamonds, hearts, or spades) and a rank (two, three, four, five, six, seven, eight, nine, ten, jack, queen, king, or ace).
A Rough Exercise

- The user will specify how many cards should be in the hand:
  
  Enter number of cards in hand: 5
  
  Your hand: 7c 2s 5d as 2h

- Problems to be solved:
  
  * How do we pick cards randomly from the deck?
  
  * How do we avoid picking the same card twice?
```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define NUM_SUITS 4
#define NUM_RANKS 13

int main(void)
{
    bool in_hand[NUM_SUITS][NUM_RANKS] = {false};
    int num_cards, rank, suit;
    const char rank_code[] = {'2','3','4','5','6','7','8','9','t','j','q','k','a'};
    const char suit_code[] = {'c','d','h','s'};
    ```
srand((unsigned) time(NULL));

printf("Enter number of cards in hand: ");
scanf("%d", &num_cards);

printf("Your hand:");
while (num_cards > 0) {
    suit = rand() % NUM_SUITS;   /* picks a random suit */
    rank = rand() % NUM_RANKS;   /* picks a random rank */
    if (!in_hand[suit][rank]) { /* picks a random rank */
        in_hand[suit][rank] = true;
        num_cards--;
        printf("%c%c", rank_code[rank], suit_code[suit]);
    }
}
printf("\n");

return 0;
A Rough Exercise

• Now, consider the complexity of what we’ve put forth.
• There were many servers for competitive card-game playing
  – Imagine having to code like this for thousands of simultaneous games
  – How would that work?