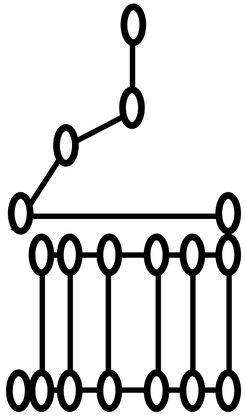
Lecture: Introduction to Classes and Objects

ENGR 2730:Computers in Engineering



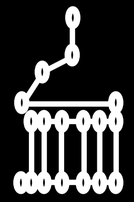
Recall “basic” built-in data types

Examples:

```
char
short
int
long
float
double
long double
```

What if we want to store a collection of data items?

What if we want to define our own types?



Simple Class and Structure Definitions

- Classes and structures group variables together to form a new variable type.
- The difference between classes and structures is that methods can be added to classes but not to structures.
- Example: a poker card has a value, suit and face.

Class

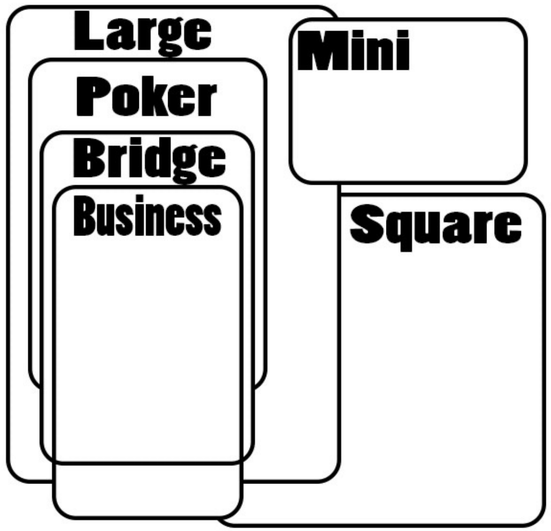
```
/* Class Card is a compound data type
   to model a playing card */
class Card {
public:
    int value;        /* value of the card */
    char suit;        /* card suit */
    char face;        /* 2-9, ,J,Q,K,A
};
```

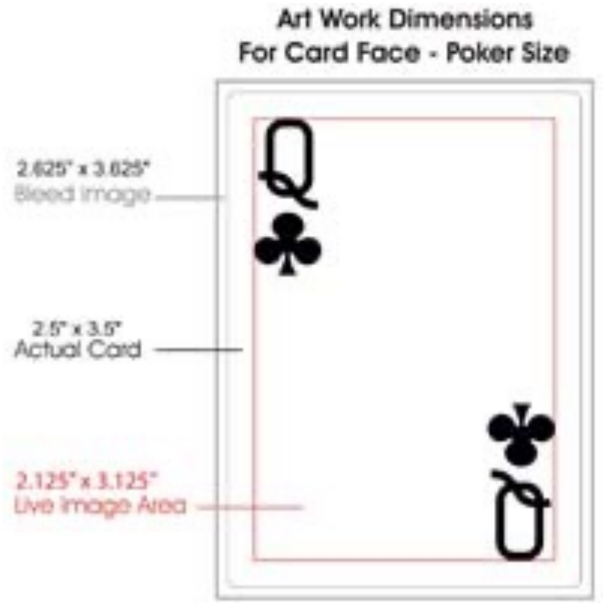
Structure

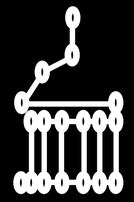
```
/* struct card is a compound data type
   to model a playing card */
struct Card {
    int value;        /* value of the card */
    char suit;        /* card suit */
    char face;        /* 2-9,T,J,Q,K,A
} ;
```

Other attributes of a poker card

- What are some other attributes of a poker card besides value, suit and face?
- Color (Red, Black)
- Suit symbol ♠ ♦ ♣ ♥
- Picture
- Card Width
- Card Height
- Picture Width
- Picture Height







Declaring variables for custom class data types

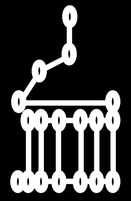
```
/* Class Card is a compound data type to model a playing card */
 class Card {
 public:
    int value;        /* value of the card */
    char suit;        /* card suit */
    char face;        /* 2-9,T,J,Q,K,A */
};
```

```
int main()
{
    Card AceOfHearts;
    Card queenOfClubs;
    Card TenOfHearts = {10, ‘H’, ’T'}; /* A + initializer list */
    Card ThreeOfSpades = {3, ’S’, ‘3’};   /* B + initializer list */
}
```

Declaring variables for struct data types

```
/* struct card is a compound data type to model a playing card */
struct Card {
    int value;        /* value of the card */
    char suit;        /* card suit */
    char face;        /* 2-9,T,J,Q,K,A
} ;
```

```
int main()
{
    Card AceOfHearts;
    Card queenOfClubs;
    Card TenOfHearts = {10, ‘H’, ’T'}; /* A + initializer list */
    Card ThreeOfSpades = {3, ’S’, ‘3’};   /* B + initializer list */
}
```



Classes are to objects as blueprints are to houses

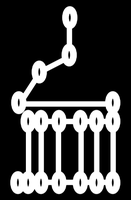
In C++, the unit of programming is the "class" from which objects are eventually instantiated (i.e., "created")

A class is a "plan" for building an object of the class.

An analogy

Class (the "blueprint")

Object (the actual "house")



C++ has a number of classes that encapsulate data (attributes) **and** functions (behaviors)

- Classes encapsulate:
  - attributes - data members
  - methods - member functions
- Classes hide the implementation details
- You must build an object from a class to interact with the class
  - You instantiate an object
- Objects are variables in your program
- Objects have its own copy of the class attributes (data members)
  - Member functions are called using the dot operator

Example Object: Complex number

A complex number has a real and imaginary part
Examples: 3+j8, 4, 1-j, 1+j

What are some things we can do with complex numbers?
We can add, subtract, multiply and divide them.
Examples:
(3 + j8) + (1 – j) = 4 + j7
(1 + j5) * (1 – j5) = 6 + j0

But how can we add, subtract, multiply and divide complex numbers
in a computer program?

Answer: We can create a new variable type called Complex.
Then we can then make Complex variables (objects) and do arithmetic with these objects.

Simple Complex Number Example

```
#include <iostream>

using namespace std;

class Complex{
public:
    double real;
    double imag;
};

Complex addComplex(Complex x, Complex y);
```

```
int main() {

    Complex p(1, 5);
    Complex q(2, -3);

    Complex s = addComplex(p, q);

    cout << "real part of s = " << s.real << endl;
    cout << "imag part of s = " << s.imag << endl;

    return 0;
}

Complex addComplex(Complex x, Complex y){
    Complex result;

    result.real = x.real + y.real;
    result.imag = x.imag + y.imag;

    return result;
}
```

Complex Number Example #2

A complex number object can be much more than a grouping of two numbers that represent the real and imaginary parts.

One improvement that we can make is to program an object so that it can tell us information about itself.

For example, we can modify the complex number class so that complex number objects can report their magnitude and phase.

Suppose we have a complex number objects x = 2 + j2 and y = 1 + j5.

We should be able to ask x and y questions about themselves.

Question: x what is your magnitude? Answer: 2.8284
Question: y what is your magnitude? Answer: 5.0990
Question: x what is your angle? Answer: 0.7854
Question: y what is your angle? Answer: 1.3734

Complex Number Example #2

```
#include <iostream>
#include <cmath>

using namespace std;

class Complex{
public:
    double real;
    double imag;

    double getPhase() {
        return atan2(imag, real);
    }

    double getMagnitude() {
        return sqrt(real*real + imag*imag);
    }
};
```

```
int main() {

    Complex x(2, 2);
    Complex y(1, 5);

    cout << "magnitude of x = " << x.getMagnitude() <<
endl;
    cout << "magnitude of y = " << y.getMagnitude() <<
endl;
    cout << "phase of x = " << x.getPhase() << endl;
    cout << "phase of y = " << y.getPhase() << endl;

    return 0;
}
```

Output:

magnitude of x = 2.82843
magnitude of y = 5.09902
phase of x = 0.785398
phase of y = 1.3734