Object-Based Programming in C++

Data-centric design philosophy. Programmer-defined types (a.k.a. user-defined types) created with `enum`, `struct`, or `class`.

- `enum`— for finite sets of values, e.g.,
  ```
  enum Color {RED, GREEN, BLUE};
  ```

- `struct`— Simple C-style objects, e.g.,
  ```
  struct Location{ int i, int j };  
  ```

- `class`— C++-style objects. Data members declared `private`. Clients must use the public member-function interface.

All three kinds of user-defined types are first class, but not all operators are automatically enabled.
First-Class Objects

- Can be defined as named variables.
- Can be received by a function as input parameters.
- Can be returned by a function as output parameter.
- Can be a member of another object.

Typically, first-class objects' values can be copied using the assignment operator (operator=). For structs and classes, only the assignment operator (operator=) is automatically made available.

Since enumerated types are stored as integers, all the integer operators ( + - = == < > etc.) are automatically available for them.
Overloaded Operators

• The name of the + operator is operator+, etc.
• All operators except operator= be explicitly defined.
• Can be defined as nonmember or member functions:
  Nonmember: a + b is the same as operator+(a,b)
  Member: a + b is the same as a.operator+(b)
• The calling object of a member-function binary operator is the **left-hand operand**.
• Arity, precedence, and associativity are preserved.
• Familiar binary operators are **not** automatically commutative: a+b is not necessarily b+a.
enum

    enum Day {SUN, MON, TUE, WED, THU, FRI, SAT};

— Stored internally as consecutive integers.

— No members or member functions, but can be parameters to
  or from user-defined functions and operators.

— Implicit conversion from the enum type to type int.

— Explicit conversion from type int to the enum type.

bool isWeekend( Day d ) { return (d == SAT || d == SUN ); }
bool isWeekday( Day d ) { return (d >= MON && d <= FRI ); }
Day operator+(unsigned i, Day d) { return Day( (i+unsigned(d))%7 ); }
Day& operator++( Day& d ) { return d = ( d==SAT ? SUN : Day(d+1) ); }
Sample Driver using enum Day type

Day firstDay = getDay();
Day nextDay = firstDay;
cout << " You entered the following day: "
   << dayToString( nextDay++ ) << ".\n";
cout << " The next day is " << dayToString(nextDay)
   << ", and the day after that is "
   << dayToString(++nextDay) << ".\n" << endl;

cout << " Enter a nonnegative integer number of days: ";
int i;
cin >> i;
if ( i < 0 ) fail("Please follow my instructions.\n");
unsigned ii = unsigned(i);

cout << '"' << ii << " days after " << dayToString(firstDay)
   << " is " << dayToString( ii + firstDay ) << ".\n";
Sample Trace

whale.2> opWeek
   Enter one of the following days:  sun mon tue wed thu fri sat :  sat
   You entered the following day: SAT.
   The next day is SUN, and the day after that is MON.

   Enter a nonnegative integer number of days: 10
   10 days after SAT is TUE.
   10 days after MON is THU.
struct

- Default member access is public via the dot operator (\(\cdot\)).

- Except for default member access, struct is grammatically identical to class.

- By convention, structs are used only for very simple types with only public data members.

```c
struct Location{ int i, j; }
Location findMarker(char A[][NCOLS], int nrows, int ncols, char mark){
    Location answer;  answer.i = answer.j = -1;
    for (int i = 0; i < nrows; ++i )
        for (int j = 0; j < ncols; ++j )
            if ( A[i][j] == mark ){ answer.i = i;  answer.j = j; }
    return answer;
}
```
class

• Default member access is private; i.e., data members can only be accessed by objects of the same class or by friends of the class.

• A nonstatic member belongs to an individual object or instance of the class. Different objects of the same class have different and separately stored nonstatic members. A public nonstatic member is accessed with the “dot” operator (.).

• A static member is shared by all objects of the class and is not replicated once for each instance. A public static member is accessed with the “scope resolution” operator (::).

• A class definition may contain other (nested) type definitions: class, struct, or enum.
A Constructor ...

- is a member function used to initialize an object of the class.
- has the same name as the class itself.
- has no return type, not even `void`.
- has special calling syntax rules.
- should use a `member initialization list` in its definition.
- can be overloaded.
Essential Member Functions for type X can be automatically generated and automatically called.

- Default Constructor: \texttt{x::x()} is called automatically when an object is initialized without initial values, e.g., \texttt{Date d;}.

- Copy Constructor: \texttt{x::x(const \&x)} is called automatically when an object is passed or returned by value.

- Destructor: \texttt{x::~x()} is called automatically when an object dies (goes out of scope or is deleted).

- Assignment operator: \texttt{const \&x \& x::operator=(const \&x)}

For objects with exogenous data, these operators must be explicitly defined or disabled to prevent disastrous unintended consequences.