***Prof. Croker's Glossary of Object-Oriented Terms***

**actual parameter** - A parameter value that is used in a function call or in a message sent to an object.

**application software** - Software that is used to perform some task in furtherance of a user's, generally, non-computer-related, goals.

**array** - A collection of objects, referred to as *elements*, all of which are of the same class and are associated with the same identifier. Each element of an array is associated with unique integer-valued *subscript/index*. Array elements are assigned contiguous memory locations.

**assembler** - A language processor that translates each statement of a non-machine language program, the source, into one machine language statement. The source language is called an *assembly language*, or a *second generation language* (*2GL*).

**associativity** - Specifies the order in which a sequence of operators of the same precedence are evaluated. Can be *left-to-right* or *right-to-left*.

**base class** - see ***inheritance***

**class** - A collection of objects all of which have the same types of data members, and the same member functions. In *C++* a class serves as a template for its objects.

**compiler** - A language processor that translates each statement of a non-machine language program, the source, into one or more machine language statements. The source language is called a *high level language*, or a *third generation language* (*3GL*).

**compound statement** - A sequence of one or more statements that are syntactically equivalent to a single statement.

**constant** - A data object whose value cannot change during the execution of a program.

**constructor** - A member function of an object that is executed whenever that object comes into existence. Often used to initialize an object's data members.

**control construct** - A statement or unit within a program that is used to alter the default order in which program statements are executed.

**data-hiding** - see ***encapsulation***

**data type** - A set of values. Optionally may include the set of operators used for manipulating those values.

**declaration statement** - A statement that is used to specify the attributes (i.e., properties) of an identifier.

**definition statement** - A declaration statement that is implemented (i.e., causes the allocation of memory).

**derived class** - see ***inheritance***

**degree of an operator** - The number of values associated with the evaluation (i.e., the use) of that operator. Usually *unary* (1), *binary* (2), or *ternary*(3).

**destructor** - A member function of an object that is executed whenever that object goes out of existence. Usually used to deallocate any dynamically allocated memory associated with that object.

**dynamic allocation** - A process of obtaining access to additional memory during program execution.

**encapsulation** - The ability to define objects (and also functions) in such a way that they can be referenced by some statements within a program but not by others. Also referred to as ***data hiding***.

**enumerated type** - A data type in which each value corresponds to an integer value.

**expression** - A well-structured (i.e., syntactically correct) sequence of objects and operators that evaluates to a single value. Must include at least one object.

**external object** - An object that is defined outside of all functions.

**formal parameter** - A parameter that is specified as part of a function definition.

**friend** - A specification in a designated class of a non-member function of that class (a **friend function**) or of a second class (a **friend class**) that gives the friend function, or the member functions of the friend class access to all of the members of objects of the designated class.

**function** - A named collection of statements, and possibly data, that is used to perform some task for a program.

**function overloading** - see ***polymorphism***

**identifier** - A sequence of characters used to name a component of a program.

**inheritance** - The ability to define a new class in terms of one or more existing classes. Objects of the new class, referred to as the *derived* class, includes the same type of data members and the same function members as objects of each of the existing classes, referred to as *base* classes, as well as members specific to objects of the derived class.

**internal object** - An object that is defined inside of a function.

**language processor** - Any program whose sets of inputs and outputs each includes a program.

**lifetime of an object** - see ***storage class***

**linker** - A language processor whose inputs includes one or more compiled or assembled programs, and whose output is an executable program.

**literal** - A data item/object whose value is as written.

**machine language** - A programming language that can be acted on (i.e., executed) directly by the cpu. Also called a *first generation language* (*1GL*).

**member** - A data or function component of an object.

**message** - A call (i.e., the invoking) of a member function of an object. The name of the message is the same as the associated member function.

**method** - The specification of a member function, and how (i.e., its definition) it carries out its task.

**object** - A collection of data (i.e., other objects) and functions used for interfacing with and otherwise manipulating that data. In *C++* these data and functions are referred to as *member data* and *member functions*, respectively.

**object program** - An output program of a language processor. This term is most commonly used when referring to the output of a compiler or an assembler.

**operating system** - Software that is used to allocate and manage the resources, both hardware and software, of a computer system.

**operator overloading** - see ***polymorphism***

**operator precedence** - A property of an operator that is used to specify its order of evaluation relative to other operators that are included within the same set of parentheses in an expression. Operators with higher precedence are evaluated before those with lower precedence.

**overloading** - see ***polymorphism***

**parameter** - A mechanism that allows a function to exchange data with other parts of a program. In *C++* parameters are used for function-to-function communication of data. Zero or more parameters may be associated with a function.

**pointer** - An object whose value denotes the address in memory of some other object.

**polymorphism** - The ability to use the same identifier to name two or more functions within the same scope of a program; gives the appearance that a single function exhibits different behavior depending on the nature (i.e., classes/types and number) of its parameters.

**preprocessor** - A language processor that is used to translate and perform other types of manipulations of a program before it is compiled or assembled.

**program** - The most inclusive algorithmic unit that when compiled and/or assembled, and linked results in a single executable (program) file. Standard unit of execution from the perspective of the operating system.

**prototype** - A function declaration. In *C++* a statement that specifies the name of the function, the type of value, if any, returned by the function, and the number and types of parameters used when invoking the function.

**scope** - A property of an identifier, and thus the associated component of the program denoted by that identifier, indicating which statements within the program can reference that identifier.

**semantics** - A feature of a language that denotes the meaning associated with various syntactic constructs of the language.

**signature** - The name of a function together with the sequence of its parameter types. Used in function calls to overloaded functions to determine which variant to invoke.

**source program** - An input program to a language processor. This term is usually used when referring to input of a compiler or an assembler.

**state** - A property of any data containing component of a program (e.g., an object or function), or the program itself, that refers collectively to the value of its data at a given instance.

**statement** - In *C++*, any well-formed expression that is terminated by a semicolon ("*;*").

**static allocation** - The process of relating objects with memory at compile-time.

**storage class** - Property of an object that determines when the object comes into existence and goes out of existence (i.e., its lifetime), and any default initial value of that object. An object comes into existence when it assigned to a specific memory location, and goes out of existence when that location is disassigned.

**stub** - A skeleton function (i.e., one that performs little or no processing) created to stand in for one function (the target) to allow the testing of a second function that is designed to call the target function. Through the use of a stub the second function can be tested even when the target function is not available or has not been tested.

**syntax** - A feature of a language that denotes the grammatical structure of the language. A set of rules governing the correct groupings of the languages elements.

**systems software** - Software whose purpose is to facilitate the use of a computer system.

**template** - A general set of code written using class/data type parameters (or surrogates) that can later be substituted with specific classes/data types resulting in an instantiated set of code that can be used in a program.

**virtual function** - A designation for a function, say ***fcn()***, of a base class having the property that whenever an object of a derived class contains a redefinition (i.e., a function with the same prototype, and thus the same name) of this virtual function, any reference to ***fcn()*** of a derived class object through a base class pointer will cause the derived class variant to execute instead of the base class variant.

**visibility** - The designation of the scope of the members of a class. Can be *private* (the default), *protected* or *public*. The default for classes is *private*.