OBJECT ORIENTED PROGRAMMING CONCEPTS

Topic 2

Object Oriented Programming

Object-oriented programming (**OOP**) is a programming paradigm based on the concept of “objects", which may contain data, in the form of fields, often known as *attributes;* and code, in the form of procedures, often known as methods.

In OOP, computer programs are designed by making them out of objects that interact with one another.

STATE AND BEHAVIOR IN OOP

* State – State is Dynamic and is Represented by values of its attributes (data)

* Behavior – Behavior is static and is defined by the set methods in the class

Fundamentals of Object-Oriented Concepts/ features of c++

* Objects
* Classes
* Inheritance
* Encapsulation
* Message passing
* Information hiding
* Abstraction
* Polymorphism.

AN OBJECT- DEFINITION

a) A run-time entity that consumes the computer memory
b) An instance of a class.
c) An entity that has state, behavior and identity.

Types of objects

1. -Local objects – are used within a certain scope and would get destroyed if they go out of scope
2. -Global objects – used in the entire program and only destroyed when the program ends

Cohesion of objects – this is the degree to which the task assigned to an object seem to form a meaningful unit. Programmers should maximize cohesion

Coupling of objects – The degree to which the ability to fulfil a certain responsibility depends upon the actions of another object. Programmers should minimize coupling

A CLASS

* a) A template that is used to construct objects.
  b) A collection of objects that have similar state and behavior.
  c) A collection of objects of similar type.
*
* All members of a Class are private by default and classes are an independent unit that can implement inheritance and polymorphism
* Structs on the other hand have all members public by default ,a struct can be a member of a class or struct to another struct. But a Structs cannot implement inheritance or polymorphism

INHERITANCE

This is the ability to define new classes from already existing classes. It is a powerful concept in programming because it facilitates development of programs that are of high quality.

A class that is defined, created or derived from another class is referred to as a sub-class or a derived class or child class.

A class that is used to create or derive another class is referred to as a super-class, base or a parent-class.

BENEFITS OF INHERITANCE

* There is code sharing
* Software re use
* Improves reliability
* There is consistency of the interface
* Rapid prototyping is possible

TYPES OF INHERITANCE:

a)Single inheritance: The sub-class has one parent class
b) Multiple inheritance: The sub-class has more than one parent class.
c) Mult-level inheritance: The sub-class has one parent class and that parent class is a sub-class of another parent class
d) Hierarchical inheritance – This is where multiple sub classes inherit from one base class

e) Hybrid inheritance: It is a combination of single-inheritance, multiple inheritance and mult-level inheritance.- assignment to draw

A class that is derived from another class inherits properties of that base class.

ENCAPSULATION

This is enclosing state and behavior of an object or putting together state and behavior of objects in a single unit.This is implemented through the use of classes.

INFORMATION HIDING

This is prevention of state and behavior of an object from direct access by other objects which is done through the use of access specifiers that includes: private, public and protected.

ABSTRACTION

The act of showing only the general properties of an object leaving out the detailed properties. It is useful when designing complex programs.
A class is an example of an abstract data type(ADT)

MESSAGE PASSING

* This is the way in which objects interact with one another in an OOP program. They do this by sending messages to one another. E.g. if“ person” and “window” are objects in program, the “person” object may send a message to the “window” requesting it to close.
* A message is therefore a request to an object to invoke one of its methods. It contains the name of the method and the respective arguments(functions)

Cont..

A message is also be defined as a request for something to be done.
Message passing has the following information/properties/features.
a) The name of the object that the message is being sent to(reciever)
b) The name of the message itself
c) Information that the receiver of the message requires in order to respond to the message.

Ways in which references can be used in OOP

* When there is need to pass the address of an object to a function
* When there is need to return an address from a function

POLYMORPHISM

Polymorphism is the ability of two or more objects to respond differently when they are sent the same message.

steps that are followed when executing an oop program written in a high-level language

1. A text-editor is used to create a program in a high-level language such as Java or C++. At this stage program is referred to as source code which should be saved.
2. The source-code is compiled using a compiler. The compiler checks the source code for syntax errors, and if no error is found, it translate the program into equivalent machine code.
3. The Linker combines the machine code produced by the compiler with machine code from the libraries of that specific program to produce an executable file.
4. The executable file is loaded into the main memory for execution. This is done by a program referred to as loader.

Difference between a class and an object

Class

* A class is a user defined data type in C++ and can be created to solve a particular type of problem
* All the attributes of a class are fixed and don't change before, during and after the execution of the program
* The class to which an object belongs is usually static/still

Object

* Every object belongs to a class and every class contains one or more related objects
* Objects are created and eventually destroyed and during their lifetime the attributes of objects may change
* Objects have limited lifespan to the time they are destroyed

Note

* Virtual base class – used when there is need to implement multiple inheritance eg a child class inherits properties of a parent ,grand parent through two parents

Tokens -They are smallest individual units in a program ,they are divided into the following

* 1. Keywords
* 2. Identifiers
* 3. Constants
* 4. String
* 5. Operators
* 6. Literals
* 7. Escape characters

1. Keywords

* Key words are also identifiers but cannot be user defined since they are used by the language as a reserved word for use as key words e.g. auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, else, explicit, , float, for, friend, goto, if, inline, int, long, namespace, new, operator, private, protected, public, return, sizeof, static, static_cast, struct, switch, virtual, void, while e.t.c

Terms in OOP

Access specifiers- A Keyword that defines the extent to which a program element can be reached within a program. In C++ access specifiers are private, protected and public

Pass by value – Involves passing a copy of a memory location to a sub program

Pass by reference – Involves passing an address of an object to a function and when there is need to have a return value

Arguments – Instructions that a particular program is meant to do. Another word for arguments is instructions/program statements.

2. Identifiers

Identifiers refer to t he names of variables, functions, array, keywords, classes etc created by the user(programmer) . They can be defined as the name of the variable and some other program elements.

Rules for naming a identifiers

1. only alphabetic characters, digits and underscore are permitted
2. the name cannot start with a digit
3. upper case and lower case are distinct
4. a declared keyword cannot be used as a variable name
5. Variable names should not be underlined
6. Should not exceed 128 characters

3. Constants

* This are the values which are set and will remain unchanged through out the program Types of constants

1. String constant
2. Numeric constants
3. Character constant

String constant: is a sequence of alphanumeric characters enclosed in double quotation marks whose maximum length is 255 characters Examples “result” “ksh 3000” “my program”

Ways are there to initialize an int with a Constant?

1. The first format uses traditional C notation e.g.
   int result = 10;
2. The second format uses the constructor notation e.g.
   int result (10);

Cont..

Numeric constants: there are four types of numeric constants: integer constant, floating point constant, hex constant, octal constant .The variable can be declared as integer in the following ways .. Eg

int x,y;

short int x,y;

long int x,y ;

4. Literals

* Literals are used to express particular values within the source code of a program. We have already used these previously to give concrete values to variables or to express messages we wanted our programs to print out, for example, when we wrote: a = 5; the 5 in this piece of code was a literal constant value.
  Assignment (=) The assignment operator assigns a value to a variable eg

a = 5; //This statement assigns the integer value 5 to the variable a

Comparison between structured and OOP

* keywords and identifiers
* Comments - comments improve readability/understandability of a program. They
  enables other programmers to understand the program better

* literals
* constants
* punctuators

Escape characters/Escape codes

Character and string literals have certain peculiarities, like the escape codes. These are special characters that are difficult or impossible to express otherwise in the source code of a program, like newline (\n) or tab (\t). All of them are preceded by a backslash (\). Here you have a list of some of such escape codes

\n	Newline
\r	carriage return
\t	Tab
\v	vertical tab
\b	Backspace
\f	form feed (page feed)
\a	alert (beep)
\'	single quote (')
\"	double quote (")
\?	question mark (?)
\\	backslash (\)

operators

Arithmetic operators (+, -, *, /, %)

The five arithmetical operations supported by the C++ language are:

+	addition
-	subtraction
*	multiplication
/	division
%	modulo

Operations of addition, subtraction, multiplication and division literally correspond with their respective mathematical operators. The only one that you might not be so used to see may be *modulo*; whose operator is the percentage sign (%). Modulo is the operation that gives the remainder of a division of two values.

Logical operators (!, &&, ||)

The Operator ! is the C++ operator to perform the Boolean operation NOT, it has only one operand, located at its right, and the only thing that it does is to inverse the value of it, producing false if its operand is true and true if its operand is false. Basically, it returns the opposite Boolean value of evaluating its operand. For example:

```
!(5 == 5)   // evaluates to false because the expression at its right (5 == 5) is true.
!(6 <= 4)   // evaluates to true because (6 <= 4) would be false.
!true       // evaluates to false
!false      // evaluates to true.
```

The logical operators && and || are used when evaluating two expressions to obtain a single relational result. The operator && corresponds with Boolean logical operation AND. This operation results true if both its two operands are true, and false otherwise. The following panel shows the result of operator && evaluating the expression a && b:

Cont..

The operator || corresponds with Boolean logical operation OR. This operation results true if either one of its two operands is true, thus being false only when both operands are false themselves. Here are the possible results of a || b:

* Note : logical operators are simply 3
* ! For NOT
* && for AND
* || for OR

BASIC DATA TYPES

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	3.4e +/- 38 (7 digits)
double	Double precision floating point number.	8bytes	1.7e +/- 308 (15 digits)
long double	Long double precision floating point number.	8bytes	1.7e +/- 308 (15 digits)
wchar_t	Wide character.	2 *or* 4 bytes	1 wide character

Enumerated data type

This is a user defined type which provides a way for attaching names to numbers, thereby increasing comprehensibity of the code.

The enum keyword automatically enumerates a list of words by assigning them values 0.1.2, and so on

Example

Enum shape {circle, square, triangle};

Enum color {red, blue, yellow}

Enum position {off, on}

Data type.. Cont..

Properties of Abstract data types (ADTs)

* They export a type
* They export a set of operations through their interface
* Operations of the interface are the only access mechanism to this types data structures
* They are Axioms(proverbs or sayings) and precondition that define the application domain of the data type

Static and dynamic memory allocation

Static

* The memory is allocated is predicted and pre-known.
* This memory is allocated during compilation
* All the declared variables are declared normally, are allocated memory needed statically

Dynamic

* The amount of memory to be allocated is not known
* This memory is allocated during run time as and when required
* The memory size is dynamically allocated as per the operators and other variables that get involved

C++ features that support Object Oriented Programming(oop)

* It gives the easiest way to handle data hiding and encapsulation with the help of powerful keywords such as class, private, public, protected e.t.c
* Inheritance, with options of single, multiple, multilevel, hyrachical e.t.c where base classes and derived classes are applied.
* Polymorphism through virtual functions, virtual base classes and virtual destructors
* It provides overloading of operators and functions
* C++ focusses on functions and class templates in handling data types
* Provides friends, static methods, constructors and destructors for class objects

Difference between Declaration and Definition of a variable.

Declaration of a variable

The declaration of a variable is specifying the data type of a variable and the variable name. As a result of the declaration, we tell the compiler to reserve the space for a variable in the memory according to the data type specified. E.g.

int Result;

char c;

int a,b,c;

Definition of a variable

Definition is an implementation of the declared variable where we tie up appropriate value to the declared variable so that the linker will be able to link references to the appropriate entities e.g.

Result=21

C='M'

Friend function

C++ class does not allow its private and protected members to be accessed outside the class. But this rule can be violated by making use of the "**Friend**" function, which is an external function that is a friend of the class.

For friend function to access the private and protected methods of the class, we should have a prototype of the friend function with the keyword "friend" included inside the class.

Friend function can be accessed from outside the same class.

Non-member operator overloading functions are made to be friend function so that they can access data members of other classes.

Friend class

Friend classes are used when we need to override the rule for private and protected access specifiers so that two classes can work closely with each other. Therefore we can have a friend class to be a friend of another class. This way, friend classes can keep some properties private where others are left to be the way they were.

Revision Questions

1. Differentiate an object and a class - a class is a collection of objects that have similar properties while an object is an instance of a class

2. Define a compiler- A compiler is a program that checks for syntax errors and translate program source code into machine code
3. Compare and contrast an object-oriented program and a structured program
4. Explain how inheritance improve quality of a program
5. Explain the importance of polymorphism and encapsulation