**Unit – 7 Object Oriented System Development**

**Disadvantages of OOPs**

1. Compiler and runtime overhead. Object oriented program required greater processing overhead-demands more resources.
2. An object’s natural environment is in RAM as a dynamic entity but traditional data storage in files or databases.
3. Requires the mastery in software engineering and programming methodology.
4. Benefits only in long run while managing large software projects.
5. Re-orientation of software developer to object-oriented thinking.

**Application of OOPs**

1. User interface design such as windows.
2. Used in real time system or real business system environment.
3. Simulation and modeling.
4. Object-Oriented databases.
5. AI and expert systems.
6. Neural networks and parallel programming.
7. Decision support and office automation system.
8. CAM/CAD systems.

**Differences between Structured and Object Oriented Programming Language**

|  |  |
| --- | --- |
| **Structured Programming Language** | **Object Oriented Programming Language** |
| 1. It is a procedure language emphasize on doing the things only i.e. it focuses on algorithm. | 1. It is an object oriented technique in which emphasis is on data rather than procedure or algorithm. |
| 1. Larger programs are divided into smaller programs known as functions. | 1. Programs are divided into what we known as object. |
| 1. Most of the function shares the global data. Function transforms the data from one to another. | 1. Object may communicate with each other through function. |
| 1. Data undervalued. | 1. Data more valued. |
| 1. It is very difficult to add the new function and data once the program is completed. | 1. New data and function can easily be added whenever the need arises. |
| 1. It follows top-down approach. | 1. It follows bottom-up approach |
| 1. Example C,COBAL,FORTAN etc. | 1. Example C++,JAVA,C# etc. |

## Changing the default behavior of new and delete

Here's an example of what it would look like to overload new and delete for a particular class.

class Myclass

{

public:

void\* operator new(size\_t);

void operator delete(void\*);

};

**History of programming languages**

**The 1950s: The First**

**Programming Language**

• **Pseudocodes:** interpreters for assembly language

• **Fortran:** the first higher level programming

language

• **COBOL:** he first business oriented language

• **Algol:** one of the most influential programming

languages ever designed

• **LISP:** the first language outside the von Neumann

model

• **APL:** A Programming Language

**Fortran (1954-57)**

• FORmula TRANslator

• Developed at IBM under the

guidance of John Backus

primarily for scientific,

computational programming

• Dramatically changed forever the

way computers used

• Has continued to evolve, adding new features & concepts.

– FORTRAN II, FORTRAN IV, FORTRAN66, FORTRAN77, FORTRAN90

• Always among the most efficient compilers, producing fast

Cod

**COBOL**

• COmmon Business Oriented Language

• Principal mentor: (Rear Admiral Dr.)

Grace Murray Hopper (1906-1992)

• *Based on FLOW-MATIC* which had such

features as:

• Names up to 12 characters, with

embedded hyphens

• English names for arithmetic operators

• Data and code were completely separate

• Verbs were first word in every statement

• CODASYL committee (Conference on Data Systems

Languages) developed a programming language by the

name of COBOL

**basic**

Beginner's All purpose Symbolic Instruction Code

• Designed by Kemeny & Kurtz at Dartmouth for the GE

225 with the goals:

• Easy to learn and use for non-science students and as

a path to Fortran and Algol

• Must be “pleasant and friendly”

• Fast turnaround for homework

• Free and private access

• User time is more important than computer time

• Well suited for implementation on first PCs (e.g., Gates

and Allen’s 4K Basic interpreter for the MITS Altair

personal computer (circa 1975)

• Current popular dialects: Visual BASIC

**The 1970s: Simplicity,**

**Abstraction, Study**

• Algol-W - Nicklaus Wirth and C.A.R.Hoare

– reaction against 1960s

– simplicity

• Pascal

– small, simple, efficient structures

– for teaching program

• C - 1972 - Dennis Ritchie

– aims for simplicity by reducing restrictions of the type system

– allows access to underlying system

**Pascal**

• Designed by Wirth, who quit the ALGOL 68

committee (didn't like the direction of that

work)

• Designed for teaching structured programming

• Small, simple

• Introduces some modest improvements, such as

the case statement

• Was widely used for teaching programming ~

1980-1995.

**The 1980s: Consolidation**

**and New Paradigms**

• Ada

– US Department of Defence

– European team lead by Jean Ichbiah

• Functional programming

– Scheme, ML, Haskell

• Logic programming

– Prolog

• Object-oriented programming

– Smalltalk, C++, Eiffel

**Smalltalk (1972-80)**

• Developed at Xerox PARC by Alan Kay and

colleagues (esp. Adele Goldberg) inspired by

Simula 67

• First compilation in 1972 was written on a bet to

come up with "the most powerful language in the

world" in "a single page of code".

• In 1980, Smalltalk 80, a uniformly object-oriented

programming environment became available as the

first commercial release of the Smalltalk language

• Pioneered the graphical user interface everyone

now uses

• Saw some industrial use in late 80’s and early 90’s

**C++ (1985)**

• Developed at Bell Labs by Stroustrup

• Evolved from C and SIMULA 67

• Facilities for object-oriented programming, taken

partially from SIMULA 67, added to C

• Also has exception handling

• A large and complex language, in part because it

supports both procedural and OO programming

• Rapidly grew in popularity, along with OOP

• ANSI standard approved in November, 1997

**Java**

• Developed at Sun in the early 1990s

with original goal of a language for

embedded computers

• Principals: Bill Joy, James Gosling, Mike

Sheradin, Patrick Naughton

• Original name, Oak, changed for copyright reasons

• Based on C++ but significantly simplified

• Supports *only* OOP

• Has references, but not pointers

• Includes support for applets and a form of

Concurrency

**C# (C Sharp)**

• Microsoft and Sun were bitter rivals in the

90s

• C# is Microsoft’s answer to Java

• C# is very similar to Java with (maybe)

some minor improvements

• If you know Java, learning C# should be

easy

• However: both languages have extensive

libraries, and mastering them is a big part of

mastering the language.

**What is a container class? What are the types of container classes?**

A class is said to be a container class which is utilized for the purpose of holding objects in memory or persistent media. A generic class plays a role of generic holder. A container class is a good blend of predefined behavior and an interface that is well known. The purpose of container class is to hide the topology for the purpose of objects list maintenance in memory. A container class is known as heterogeneous container, when it contains a set of different objects. A container class is known as homogeneous container when it contains a set of similar objects.

Containers are used to manage collections of objects of a certain kind. There are several different types of containers like deque, list, vector, map etc.

**What is container class? Explain containers of pointer.**

- Container class is one of the classes that were put into class libraries. To handle objects that contain other objects, container classes are used. A GUI class library contains a group of container classes.  
- Containers of pointers provide containers to hold the objects that are heap-allocated in manner that is exception-safe and with minimum overhead. The central idea is to make OOP easier in C++. This is done by establishing a standard a set of classes, methods to deal with OO specific problems.

**What are the C++ standardized container classes?**

The following are the standardized container classes :  
**1. std::map :**   
Used for handle sparse array or a sparse matrix.  
**2. std::vector :**  
Like an array, this standard container class offers additional features such as bunds checking through the at () member function, inserting or removing elements, automatic memory management and throwing exceptions.   
**std::string :**   
A better supplement for arrays of chars.

### Container class templates

**Sequence containers**:

[array](http://www.cplusplus.com/reference/array/array/)

Array class (class template )

[vector](http://www.cplusplus.com/reference/vector/vector/)

Vector (class template )

[deque](http://www.cplusplus.com/reference/deque/deque/)

Double ended queue (class template )

[forward\_list](http://www.cplusplus.com/reference/forward_list/forward_list/)

Forward list (class template )

[list](http://www.cplusplus.com/reference/list/list/)

List (class template )

**Container adaptors**:

[stack](http://www.cplusplus.com/reference/stack/stack/)

LIFO stack (class template )

[queue](http://www.cplusplus.com/reference/queue/queue/)

FIFO queue (class template )

[priority\_queue](http://www.cplusplus.com/reference/queue/priority_queue/)

Priority queue (class template )

**Procedure Oriented Paradigm :-**

The focus of **procedural** programming is to break down a programming task into a collection of variables, data structures, and subroutines, whereas in object-**oriented**programming it is to break down a programming task into objects that expose behavior (methods) and data (members or attributes) using interfaces.

**Procedural programming** is a programming paradigm, derived from structured programming, based upon the concept of the*procedure call*. Procedures, also known as routines, subroutines, or functions (not to be confused with mathematical functions, but similar to those used in functional programming), simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself. Procedural programming languages include [C](https://en.wikipedia.org/wiki/C_(programming_language)), Go, Fortran, Pascal, Ada, and BASIC.

Modularity is generally desirable, especially in large, complicated programs. Inputs are usually specified syntactically in the form  and the outputs delivered as [*return values*](https://en.wikipedia.org/w/index.php?title=Return_values&action=edit&redlink=1).

Scoping is another technique that helps keep procedures modular. It prevents the procedure from accessing the variables of other procedures (and vice versa), including previous instances of itself, without explicit authorization.

Less modular procedures, often used in small or quickly written programs, tend to interact with a large number of variables in the execution environment, which other procedures might also modify.

Because of the ability to specify a simple interface, to be self-contained, and to be reused, procedures are a convenient vehicle for making pieces of code written by different people or different groups, including through programming libraries.

**Object Oriented Analysis & Design**

Object–Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system’s object model, which comprises of interacting objects.

The main difference between object-oriented analysis and other forms of analysis is that in object-oriented approach, requirements are organized around objects, which integrate both data and functions. They are modelled after realworld objects that the system interacts with. In traditional analysis

**methodologies, the two aspects - functions and data - are considered separately.**

Grady Booch has defined OOA as, “Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain”.

The primary tasks in object-oriented analysis (OOA) are:

 Identifying objects

 Organizing the objects by creating object model diagram

 Defining the internals of the objects, or object attributes

 Defining the behavior of the objects, i.e., object actions

 Describing how the objects interact

**The common models used in OOA are use cases and object models.**

Object-Oriented Design

Object–Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis. In OOD, concepts in the analysis model, which are technology−independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.

The implementation details generally include:

 Restructuring the class data (if necessary),

 Implementation of methods, i.e., internal data structures and algorithms,

 Implementation of control, and

 Implementation of associations.

Grady Booch has defined object-oriented design as “a method of design

encompassing the process of object-oriented decomposition and a notation for

depicting both logical and physical as well as static and dynamic models of the

**Write a program to input two numbers and find out larger one using template function.**

function template

#include <iostream>

using namespace std;

template <class T>

T GetMax (T a, T b) {

T result;

result = (a>b)? a : b;

return (result);

}

int main ()

{

int i=5, j=6, k;

long l=10, m=5, n;

k=GetMax<int>(i,j);

n=GetMax<long>(l,m);

cout << k << endl;

cout << n << endl;

return 0;

}