By: Mukunda Paudel
paudelmuku@gmail.com

PROGRAMMING TECHNOLOGY (PT)

Disclaimer

*This document is part of teaching materials for **Programming Technology** under the Pokhara university's syllabus for BE Computer II/II. This document do not cover all aspect of learning Programming Technology, nor are these be taken as primary source of information. As the core textbooks and reference books for learning the subject has already been specified and provided to the students, students are encouraged to learn from the original sources because this document cannot be used as a substitute for prescribed textbooks..*

*Various text books as well as freely available material from internet were consulted for preparing this document. Contents in This document are **copyrighted** to the instructor and authors of original texts where applicable.*

By: Mukunda Paudel
paudelmuku@gmail.com

Chapter-1 Introduction

What does *Procedural Language* mean?

- ❖ A procedural language is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program.
- ❖ It contains a systematic order of statements, functions and commands to complete a computational task or program.
- ❖ Procedural language is also known as **imperative language**.
- ❖ The procedural language segregates a program within variables, functions, statements and conditional operators.
- ❖ A procedural language, as the name implies, relies on predefined and well-organized procedures, functions or sub-routines in a program's architecture by specifying all the steps that the computer must take to reach a desired state or output.
- ✓ For example: BASIC, C, FORTRAN, Pascal etc.
- ✓ Procedures or functions are implemented on the data and variables to perform a task.

Why C is Called Procedure Oriented??

- ❖ Because **C** programs follow a **procedure** of steps written in it, called functions.
- ❖ It follows a top-down approach i.e. much importance is given to flow of program rather than on data on which functions operate.

The characteristics of POP are:

- ✓ Focus on process or doing things rather than data.
- ✓ The whole program is implemented based on actions such as reading, calculating and printing. Therefore, a bunch of functions are written to solve a program from top to down.
- ✓ Most of the functions share global data.
- ✓ Data moves openly from function to function.
- ✓ In the cases of large program, bringing change is difficult and time consuming.
- ✓ Appropriate and effective techniques are unavailable to secure data of a function from others.

Advantages of POP

- ✓ **Simple**. No need to think about code reusing for later use, just think about how to finish this task.
- ✓ **Ease of implantation** of compilers and interpreters.
- ✓ **Easy to** keep **track** of work flow.
- ✓ **Needs less memory**. There is no data abstraction leading to a large code size that consumes lots of memory.

Disadvantages of POP are:

1. **Bad code-reuse.**

   Not like class and objects that resemble the real world model, functions in PP are task-oriented and cannot be reused when meeting a different one. Except for some commonly used library functions, we need to write many repetitive code for similar problems.

2. **Hard to expand the application.**

   The entire process in PP is coded directly into the application, very-task related. For example, an application in POP for registering the user will not be used for logging on and off. Difficult to create new data types.

3. **Hard to maintain.**

   It will be a timing consuming solution to make changes. If one function accidentally changes value of the global data, all the functions who access to this global data must change their values.

4. **No information hiding.**

   Data is exposed to the whole program, so no security for data.

5. **Hard to design.**

   For example, in designing graphical user interface, we'd better mainly concentrate on how to design a menu, a button etc. since people will see them at first sight, not actions behind.

Object Oriented Programming Language

- Object-oriented programming (OOP) is a programming language model in which programs are organized around data, or objects, rather than functions and logic.
- An object can be defined as a data field that has unique attributes and behavior.
- **Simula** is credited as the first object-oriented programming language,
- Examples of popular object-oriented programming languages are Java and C++.
- Some other well-known object-oriented programming languages include Objective C, Perl, Python, JavaScript, Simula, Modula, Ada, Smalltalk, and the Common Lisp Object Standard

OBJECT

- Object is a collection of number of entities.
- Objects take up space in the memory.
- Objects are instances of classes.
- When a program is executed, the objects interact by sending messages to one another.
- Each object contain data and code to manipulate the data.
- Objects can interact without having known details of each other's data or code.

CLASS

- Class is a collection of objects of similar type.
- Objects are variables of the type class.
- Once a class has been defined, we can create any number of objects belonging to that class.
- E.g. grapes, bananas and orange are the member of class fruit.

NOTE: Classes are user define data types.

Principles of OOP

Object-oriented programming is based on the following principles:

- **Encapsulation**

- ✓ Process of binding of member data and function into single unit.
- ✓ The implementation and state of each object are privately held inside a defined boundary, or class.
- ✓ Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions, or methods.
- ✓ Encapsulation also leads to *data abstraction or hiding*.
- ✓ This characteristic of data hiding provides greater program security and avoids unintended data corruption

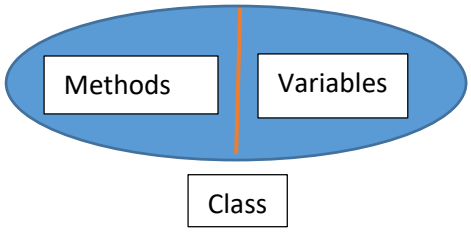

Figure: Encapsulation in OOP

- **Abstraction**

✓ Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

- **Polymorphism**
  - ✓ Ability of a message to be displayed in more than one form.
  - ✓ Objects are allowed to take on more than one form depending on the context.
  - ✓ The program will determine which meaning or usage is necessary for each execution of that object, cutting down on the need to duplicate code.
  - ✓ Polymorphism is extensively used in implementing inheritance.

Advantages of object-oriented programming

1. Improved software-development productivity:

2. Improved software maintainability:

3. Faster development:

4. Lower cost of development:

5. Higher-quality software:

Disadvantages of object-oriented programming

1. Steep learning curve:

2. Larger program size:

3. Slower programs:

4. Not suitable for all types of problems

Event Oriented Programming:

- It is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads.
- Event-driven programming is the dominant paradigm used in graphical user interfaces and other applications (e.g. JavaScript web applications) that are centered on performing certain actions in response to user input.
- In EOP we have a server, whether it be on a communications port or a user interface, waiting for an input command. It will then process that command and display/produce desired results.

In EOP control flow of the program is determined by the occurrence of **events**.

  - ✓ These events are monitored by code known as an event **listener**.
  - ✓ If it detects that an assigned event has occurred, it runs an event **handler**
  - ✓ Handler is a callback function or method that is triggered when the event occurs
- Event-driven programming is currently the default paradigm in software engineering.

- As the name suggests, it uses events as the basis for developing the software. These events can be something the users are doing - clicking on a specific button, picking an option from drop-down, typing text into a field, giving voice commands, or uploading a video or system-generated events such as a program loading.
- **Operating Systems** can be referred to as an event driven program.
- An event driven programming is when the application runs constantly and depends on **events** happening in order to execute code and functions.

Generally, there is a main loop in an event-driven application that used to **listen for events** and triggers a callback function when there are events is detected.

Key features / characteristics of event-driven programming (*imp*)

- **Service Oriented**

Service oriented is a key features in event-driven programming that used to write programs that are made for services and it takes does not slow down the computer as service oriented only consume little of the computer processing power and usually services run in the background of OS.

- **Time Driven**

In event driven programming, time driven is a paradigm, it's a code that runs on a time trigger, time driven can be a specific code that runs on a specific time, which could be once an hour, once a week or once a month, this means it's a pre-set to do task.

For example, windows update is the example of time driven, which user can set when to update or when to check and download the update.

- **Event Handlers**

Event handlers is a type of function or method that run a specific action when a specific event is triggered.

For example, it could be a button that when user click it, it will display a message, and it will close the message when user click the button again, this is an event handler.

- **Trigger Functions**

Trigger functions in event-driven programming are a functions that decide what code to run when there are a specific event occurs, which are used to select which event handler to use for the event when there is specific event occurred.

- **Events**

Events include mouse, keyboard and user interface, which events need to be triggered in the program in order to happen, that mean user have to interacts with an object in the program, for example, click a button by a mouse, use keyboard to select a button and etc.

- **Simplicity of Programming and Ease of Development**

Simple and easier to program compared to other type of programming as it's very visual.

For example you can place a button by just select it and place it onto a form and write a code for it.

Event-driven programming also easy for user to insert a pre-written code scripts into an existing application because it allows user to pause the code while it's running. Which make developing using event-driven programming is easy.

Difference between OOP and EOP imp

OOP	**EOP**
1. Object Oriented Programming is defined by the pairing together of data and actions into a model of a real world object.	**1.** Event driven programming is a style of programming in which we have a server, whether it be on a communications port or a user interface, waiting for an input command. It will then process that command and display/produce desired results.
2. A program in an object oriented language is not necessarily event driven,	**2.** Most event driven languages are object oriented.

3. Object oriented programming focuses on performing actions and manipulation of data that is encapsulated in objects within a sequential series of steps	**3.** While event driven is more dynamic and relies on event triggering and event handling to determine the sequencing of the program.
4. OOP is organized around objects rather than actions (which are used in event driven programming) and data rather than logic.	**5.** In EOP the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs or threads
6. Object oriented programming is based on execution of programs by invoking methods of object	**7.** Event driven programming is based on execution by listening events.
8. Object oriented is looking at the actors and creating objects to represent those actors. For example, an emulator for a movie line queue. You've got people, the line, and a ticket booth. So, you build objects around those actors.	**9.** Event driven, is used when you have spontaneous moments that need to be taken care of. You can add event handlers to both procedural and object oriented paradigms. So, as a standalone, it's not a complete paradigm
10. Example C++, Java, Python, Simula, Modula etc.	**11.** Virtually all object-oriented and visual languages support event-driven programming. Visual Basic, Visual C++ and Java are examples of such languages.
12. In OOP, classes can be created by own	13. In event-driven programming user must use the defined classes.
14. OOP is complex to program In comparison with EOP.	15. Event-driven programming is simple and easier to program compared to other type of programming as it's very visual

Aspect Oriented Programming (AOP)

- **Aspect** Oriented Programming (**AOP**) compliments OOPs in the sense that it also provides modularity.
- **AOP** breaks the program logic into distinct parts (called concerns).
- It is used to increase modularity by cross-cutting concerns.

- AOP is some kind of "meta-programming".
- Everything that AOP does could also be done without it by just adding more code. AOP just saves you writing this code.
- Aspect oriented programming provides a nice way to implement cross-cutting concerns like logging, security.
- These cross-cutting concerns are pieces of logic that have to be applied at many places but actually don't have anything to do with the business logic.
- You shouldn't see AOP as a replacement of OOP, more as a nice add-on that makes your code cleaner, loosely-coupled and focused on the business logic**.**

By applying AOP you will get 2 major benefits

1. The logic for each concern is now in one place, as opposed to being scattered all over the code base.
2. Classes are cleaner since they only contain code for their primary concern (or core functionality) and secondary concerns have been moved to aspects.

AOP does not *replace* OOP but adds certain decomposition features that address the so-called crosscutting concerns.

Features of AOP

- Main goal of Aspect Oriented Programming (AOP) is to **keep all the code relating to a single feature in one module of code.**
- AOP introduces some new terms. A **"join point"** is a place in the control flow where code is instrumented, like a function call or return. A "**point cut"** is a selection of join points according to some query, which are tied to a common instrumenting code. (A single instrumenting function can handle a large point cut of different join points.)
- AOP allows for a **"separation of concerns"** because new features are introduced as aspects tied to the execution of existing code, but without touching any of it, thereby keeping code bases completely separate.

By: Mukunda Paudel
paudelmuku@gmail.com

Subject Oriented Programming Language

- **Subject-oriented programming** is an **object-oriented** approach in which different subsystems known as subjects are divided to create new subjects based on the composition expression.
- **Subject-oriented programming** is largely **oriented** toward dividing an **object-oriented** system into subjects.
- It provides a compositional view of the application development.
- The main objective of subject-oriented programming is to help in evolving suites and in facilitating the development of cooperating applications.
- The two ways in which applications cooperate are: **by sharing objects** and by **jointly helping in the operation executions**.
- The subjects in subject-oriented programming can be used to form larger subjects along with combining their functionalities. This helps in reusing the functionalities available for subjects in larger subjects.
- Moreover, based on composition rules, the subjects are composed in a system and this divide approach helps in extending and maintaining large object-oriented environments. The application source code is not necessary, and it helps in extending existing applications with new and unplanned functionalities.
- The subject-oriented approach helps in bringing a model and in focusing on issues related to composition within an application. It brings in composition rules and compositors, as well as helps in better creation of objects, considering deletion and finalization protocols.
- It also helps in class and interface matching and also in taking care of implementation issues.

By: Mukunda Paudel
paudelmuku@gmail.com

Basic Difference between different programming patterns:

Event Driven Programming Paradigm	Procedural Programming Paradigm	Object Oriented Programming Paradigm	Declarative Programming Paradigm
Provides graphical user interface to create the programs.	Provides character user interface to write the commands	Provides command writing in modules	Provides talking user actions and words as inputs.
Actions are defined on events These events could be occurred by mouse clicking and moving or keyboard strokes	Commands are written in linear fashion and executed also in linear fashion	Objects and functions are prepared for interaction to perform specific tasks	Facts and rules are defined and analyzed to solve a problem.
Focuses on selecting user Interface.	Focuses on sequential execution of steps	Focuses on objects or data and facilitate to secure it from unauthorized access	Focuses on using and analyzing facts and rules for describing the problem
Most common languages which follow this paradigm are Visual Basic and C	Most common languages which follow this paradigm are Basic, Fortran and COBOL	Most common languages which follow this paradigm are Smalltalk, C++ and JAVA	Most common language which follows this paradigm is Prolog

1.2 An integrated development environment (IDE)

- IDE is a **software** application that provides comprehensive facilities to computer programmers for **software development**.
- An **IDE** normally consists of at least a source code editor, build automation **tools**, and a debugger.
- The **integrated development environment** is designed to maximize **programmer productivity** by providing tight-weave components with the same user interface. The **IDE** presents a single program wherein all developments are carried out.
- An **IDE** brings many of those **development**-related tools together as a single framework, **application** or service.
- The integrated toolset is designed to simplify **software development** and can identify and minimize coding mistakes and typos.
- Some **IDEs** are open source, while others are commercial offerings.
- Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms.
- Without an IDE, a developer must select, deploy, integrate and manage all of these tools separately. An IDE brings many of those development-related tools together as a single framework, application or service.
- The integrated toolset is designed to simplify software development and can identify and minimize coding mistakes and typos.
- **Examples** of **IDE's** include NetBeans, Eclipse, IntelliJ, and Visual Studio.

Common features of IDE

1. An IDE typically contains a code editor, a compiler or interpreter, and a debugger, accessed through a single graphical user interface (GUI).

2. The user writes and edits source code in the code editor. The compiler translates the source code into a readable language that is executable for a computer. And the debugger tests the software to solve any issues or bugs.

3. An IDE can also contain features such as programmable editors, object and data modeling, unit testing, a source code library and build automation tools.

4. An IDE's toolbar looks much like a word processor's toolbar. The toolbar facilitates color-based organization, source-code formatting, error diagnostics and reporting, and intelligent code completion.

5. IDEs are typically designed to integrate with third-party version control libraries, such as GitHub and Apache's Subversion.

6. An IDE can support model-driven development (MDD). A developer working with an IDE starts with a model, which the IDE translates into suitable code. The IDE then debugs and tests the model-driven code, with a high level of automation. Once the build is successful and properly tested, it can be deployed for further testing through the IDE or other tools outside of the IDE.

Benefits of using IDEs

1. An IDE can improve the productivity of software developers thanks to fast setup and standardization across tools.
2. Without an IDE, developers spend time deciding what tools to use for various tasks, configuring the tools and learning how to use them. Many or even all of the necessary dev-test tools are included in one integrated development environment.
3. IDEs are also designed with all their tools under one user interface. An IDE can standardize the development process by organizing the necessary features for software development in the UI.

Component of Visual Programming:

- In computing, a **visual programming** language (VPL) is any **programming** language that lets users create programs by manipulating **program** elements graphically rather than by specifying them textually.
- A visual programming language (VPL) is a programming language that uses graphical elements and figures to develop a program.
- A VPL employs techniques to design a software program in two or more dimensions, and includes graphical elements, text, symbols and icons within its programming context.

- A visual programming language is also known as an executable graphics language.
- A **visual language can be one of a few types,** such as

  - ✓ Icon-based languages,
  - ✓ diagramming languages and
  - ✓ Form-based language.

- Visual languages should not be confused with GUI-based programming languages, as they only provide graphical program authoring services. However, their code/context is completely textual.
- Kodu (made specifically for creating games. It is designed to be accessible for children and enjoyable for anyone.), Blockly and executable UML are popular examples of visual programming languages.
- E-draw software is an example of visual programming and AI BIRD is another example.
- **Components** that support **visual** composition implement a set of interfaces defined by the **visual programming** environment that supports their use.
- **Visual components** are supported by a number of IDEs. Typical examples are **Visual** Basic (VB), Delphi, JBuilder, Latte, and **Visual** Café.
- The programming environment runs on the Xbox, allowing rapid design iteration using only a game controller for input.

Components of Visual Programming / Graphical Visual Interface

Following are the popular components of Visual programming

1. **Window:**

   - ✓ A window sometimes also called a form is the most important of all the visual interface components.
   - ✓ A window plays the role of the canvas in a painting.
   - ✓ Without the canvas there is no painting. Similarly, without the window there is no User interface.
   - ✓ The components that make up the user interface have to be placed in the window and cannot exist independent of the window.

2. **Buttons**

   - ✓ A button is used to initiate an action.

- ✓ The text on the button is inductive of the action that it will initiate. Clicking on a button initiate the action associated with the button.

3. **Text boxes**:

- ✓ Text boxes are used to accept information from the user.
- ✓ The user Interface will display one text for each piece of information

4. **List Boxes or Pop-up Lists**:

- ✓ List boxes are used to present the user with the possible options.
- ✓ The user can select one or more of the listed options.

5. **Label**:

- ✓ The label is used to place text in a window.
- ✓ Typically label is used to identify controls that do not have caption properties of their own.

6. **Radio button or Option Button**:

- ✓ The radio buttons, also referred as option buttons
- ✓ Used when the user can select only one of multiple options.

Advantages of Visual Programming

Visual programming enables visual development of graphical user interface. Such user Interfaces are easy to use and easy to learn.

1. **Ready to Use Component**

- ✓ One of the principal advantages of visual programming environment is that the Programmer need not write code to display the required component.
- ✓ The Visual programming environment displays a list of available components. The programmer picks up the required components from this list.
- ✓ The components can be moved, resized and even deleted, if so required.
- ✓ There is no restriction in the number of controls that can be placed this way.

2. **Built-in Code**

- ✓ The interface components provided by the visual programming environment have some code built into them.
- ✓ **For example**, a button 'knows' when it has been clicked upon. In the case of Conventional programming tools, the programmer has to write code to determine the component that has been clicked and then execute the appropriate code.

Disadvantages of Visual Programming

While Visual programming makes it very simple to create complex user interfaces, it suffers from some disadvantages.

- ✓ As the name implies, the entire process of developing an application using a visual development environment is visual. Thus, the development environment in itself is highly graphical in nature and therefore **requires more memory.**
- ✓ Visual development environments **require computers of higher configuration** in comparison to the conventional programming tools.
    - ➢ Larger capacity hard disk
    - ➢ More RAM
    - ➢ Faster Processor
- ✓ Primarily, Visual development environment can be used only with GUI operating system such as windows.

Virtual Machines and Run Time Environment:

- ❖ A virtual machine (VM) is an **operating system** (OS) or application environment that is installed on software, which imitates dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware.
- ❖ A virtual machine (VM) not only exhibits the behavior of a separate computer, but is also capable of performing tasks such as running applications and programs like a separate computer.

- A virtual machine, usually known as a guest is created within another computing environment referred as a "host." Multiple virtual machines can exist within a single host at one time.
- A virtual machine is also known as a guest.
- Virtual machines are becoming more common with the evolution of virtualization technology.
- Virtual machines are often created to perform certain tasks that are different than tasks performed in a host environment.
- Virtual machines are implemented by software emulation methods or hardware virtualization techniques.
- VMs have multiple uses, but in general they are deployed when the need for different operating systems and processing power are needed for different applications running simultaneously.

For example, if an enterprise wants to test multiple web servers and small databases at the same time. Similarly, if an enterprise wants to use the same server to run graphics-intensive gaming software and customer service database.

Depending on their use and level of correspondence to any physical computer, ***virtual machines can be divided into two categories***:

1. **System Virtual Machines**: A system platform that supports the sharing of the host computer's physical resources between multiple virtual machines, each running with its own copy of the operating system. The virtualization technique is provided by a software layer known as a hypervisor, which can run either on bare hardware or on top of an operating system.
2. **Process Virtual Machine:** Designed to provide a platform-independent programming environment that masks the information of the underlying hardware or operating system and allows program execution to take place in the same way on any given platform.

Advantages of a virtual machine

- ✓ Allows multiple operating system environments on a single physical computer without any intervention
- ✓ Virtual machines are widely available and are easy to manage and maintain.
- ✓ Offers application provisioning and disaster recovery options

Drawbacks of virtual machines

- They are not as efficient as a physical computer because the hardware resources are distributed in an indirect way.
- Multiple VMs running on a single physical machine can deliver unstable performance

Virtual machines are an isolated environment from the physical operating system, so you can run potentially dangerous stuff, such as malware, without fear of compromising your main OS. They're a **safe** environment, but there are exploits against virtualization software, allowing malware to spread to the physical system.

Run Time Environment:

- A configuration of hardware and software. It includes the CPU type, operating system and any **runtime** engines or system software required by a particular category of applications.
- A runtime environment is the execution environment provided to an application or software by the operating system. In a runtime environment, the application can send instructions or commands to the processor and access other system resources such as RAM, which otherwise is not possible as most programming languages used are high level languages.
- The runtime environment provides a state for the target machine to have access to resources such as software libraries, system variables and environment variables, and provide all necessary services and support to the processes involved in the execution of the application or program.
- In certain software or applications such as Adobe Flash Player or Microsoft PowerPoint Viewer the runtime environment is available to end users as well.
- Software developers need a runtime environment to test their software's functioning. As a result, all software development applications include a runtime environment component which allows the testing of the application during execution.
- Tracking bugs or debugging for any errors are done in most applications with the help of runtime environments.
- Runtime execution continues even if the application or program crashes.

- Most runtime environments are capable of reporting of why an application or program crashed.
- One of the more popular runtime environments is Java, which helps Java applets and applications to be executed in any machine which has a Java runtime environment installed.

The **Java Runtime Environment** (**JRE**) **is** a set of software tools for development of **Java** applications. It combines the **Java** Virtual Machine (JVM), platform core classes and supporting libraries. **JRE is** part of the **Java** Development Kit (JDK), but can be downloaded separately

Document View Architecture:

- The **Document/View architecture** is the foundation used to create applications based on the Microsoft Foundation Classes library.
- It allows to make distinct the different parts that compose a computer program including what the user sees as part of application and the document a user would work on. This is done through a combination of separate classes that work as an ensemble.

The parts that compose the Document/View architecture are a *frame*, one or more documents, and the view. Put together, these entities make up a usable application.

Document

- A **document** is similar to a bucket.
- For a computer application, a document holds the user's data.
- To create the document part of this architecture, you must derive an object from the **CDocument** class.

View

- A **view** is the platform the user is working on to do his or her job.
- To let the user do anything on an application, you must provide a view, which is an object based on the **CView** class.
- You can either directly use one of the classes derived from **CView** or you can derive your own custom class from **CView** or one of its child classes.

Frame

- **Frame** is a combination of the building blocks, the structure, and the borders of an item.
- Frame gives "physical" presence to a window.
- It also defines the location of an object with regards to the Windows desktop.

- The document/view architecture simplifies the development process.
- Code to perform routine tasks such as prompting the user to save unsaved data before a document is closed is provided for us by the framework.
- So is code to transform our application's documents into OLE containers, simplify printing, use splitter windows to divide a window into two ormore panes, and more.

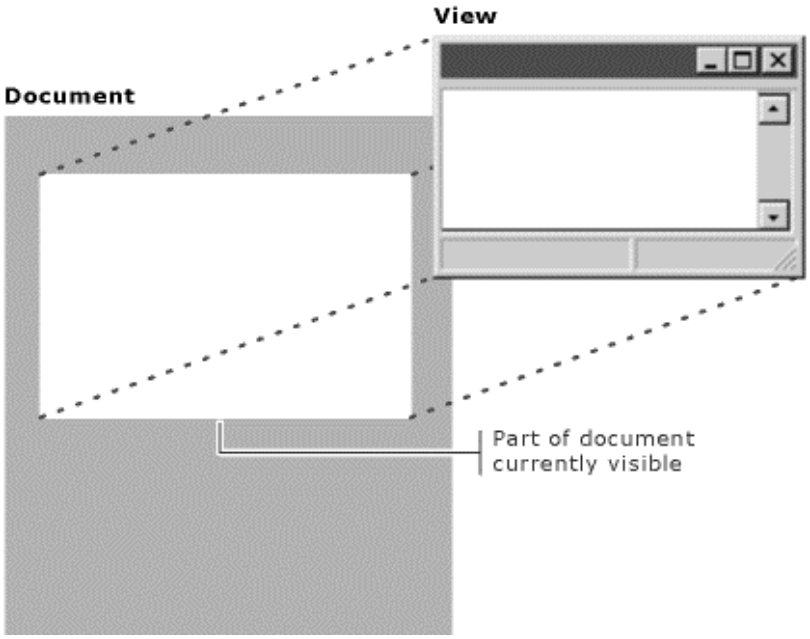

Figure: Relationship between a document and its view

MFC supports **two types of document/view applications**

1. **Single Document Interface (SDI)**
   - The expression **Single Document Interface** or SDI refers to a document that can present only one view to the user.
   - This means that the application cannot display more than one document at a time.

- If you want to view another type of document of the current application, you must create another instance of the application.
- Notepad and WordPad are examples of SDI applications.

2. **Multiple Document Interface (MDI)**

- An application is referred to as a **Multiple Document Interface**, or MDI, if the user can open more than one document in the application without closing it.
- To provide this functionality, the application provides a parent frame that acts as the main frame of the computer program. Inside this frame, the application allows creating views with individual frames, making each view distinct from the other.
- In Multiple Document Interface (MDI) applications, there is one main frame per application.

Four **Major Class** in Document View Architecture:

1. **CDocument (or COleDocument)**

- ✓ This class supports objects used to store or control your program's data and provides the basic functionality for programmer-defined document classes.
- ✓ A document represents the unit of data that the user typically opens with the Open command on the File menu and saves with the Save command on the File menu.

2. **CView (or one of its many derived classes**)

- ✓ This class provides the basic functionality for programmer-defined view classes.
- ✓ A view is attached to a document and acts as an intermediary between the document and the user: the view renders an image of the document on the screen and interprets user input as operations upon the document. The view also renders the image for both printing and print preview.

3. **CFrameWnd (or one of its variations)**

- ✓ This class supports objects that provides the frame around one or more views of a document.

4. CDocTemplate (or CSingleDocTemplate or CMultiDocTemplate)

- ✓ This class supports an object that coordinates one or more existing documents of a given type and manages creating the correct document, view, and frame window objects for that type.

Other Classes

1. CString.

- ➢ Microsoft Foundation Class (MFC) library provides a class to manipulate string called **CString**.

Features of CString.

- ✓ CString does not have a base class.
- ✓ A CString object consists of a variable-length sequence of characters.
- ✓ CString provides functions and operators using a syntax similar to that of Basic.
- ✓ Concatenation and comparison operators, together with simplified memory management, make CString objects easier to use than ordinary character arrays.

2. CArray

- ➢ **CArray** is a collection that is best used for data that is to be accessed in a random or non-sequential manner.
- ➢ CArray class supports arrays that are like C arrays, but can dynamically shrink and grow as necessary.

Features of CArray

- ✓ Array indexes always start at position 0.
- ✓ You can decide whether to fix the upper bound or enable the array to expand when you add elements past the current bound.

- ✓ Memory is allocated contiguously to the upper bound, even if some elements are null.

Advantages of Document View Architecture.

- ❖ The key advantage to using the MFC document/view architecture is that the architecture supports multiple views of the same document particularly well.

Suppose an application lets users view numerical data either in spreadsheet form or in chart form. A user might want to see simultaneously both the raw data, in spreadsheet form, and a chart that results from the data. We display these separate views in separate frame windows or in splitter panes within a single window. Now suppose the user can edit the data in the spreadsheet and see the changes instantly reflected in the chart.

- ❖ In MFC, the spreadsheet view and the chart view would be based on different classes derived from **CView**. Both views would be associated with a single document object. The document stores the data (or perhaps obtains it from a database). Both views access the document and display the data they retrieve from it.
- ❖ When a user updates one of the views that view object calls **CDocument::UpdateAllViews**. That function notifies all of the document's views, and each view updates itself using the latest data from the document. The single call to **UpdateAllViews** synchronizes the different views.

This scenario would be difficult to code without the separation of data from view, Particularly if the views stored the data themselves. With document/view, it's easy. The framework does most of the coordination work for us.

****End of Chapter 1****