1st Class

2018-2019

Software Development Fundamentals

أساسيات تطوير البرمجيات

أستاذ المادة : د. يسرى حسين

Lecture One

Software development process

1- What is software?

Any computer system is made up of hardware and software.

The term hardware is fairly easy to understand, because you can see it. It is all the pieces of equipment that make up the system – the processor, monitor, keyboard, mouse, printer, scanner and so on.

Software : it is all the programs, instructions and data that allow the hardware to do something useful and interesting.

- Software Development Life Cycle ***(SDLC)***

The Software Development Life Cycle is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Is the big picture of creating an information system that handles a major task (referred to as an application). The **applications** usually consist of many programs. An example would be the Department of Defense supply system, the customer system used at your local bank, the repair parts inventory system used by car dealerships. There are thousands of applications that use an information system created just to help solve a business problem. Computer professionals that are in charge of creating applications often have the job title of **System Analyst**. The major steps in creating an application include the following and start at **planning step** shown in figure (1).

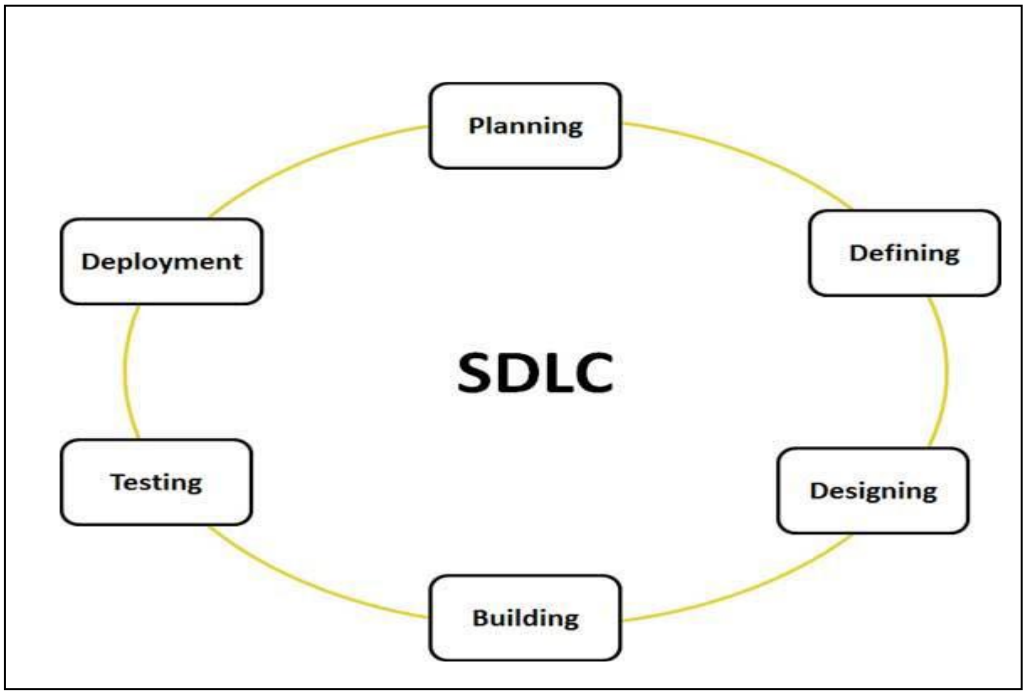


Figure (1) Software Development Life Cycle

Stage 1: ***Planning*** **and Requirement Analysis:** Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements: Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through 'SRS' – **Software Requirement Specification** document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the product architecture: SRS(**Software Requirement Specification**)is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a **DDS - Design Document Specification**. This DDS is reviewed by all the important stakeholders and based on various parameters as risk.

Assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product: In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers … etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product: This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance: Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations' business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

- During the **Maintenance phase**, it goes through a mini planning, analysis, design and implementation. The programs that need modification are identified and programmers change or repair those programs.

1- System analyst: Computer professional in charge of creating applications.

2- **Applications:** An information system or collection of programs that handles a major task.

3- **Life cycle:** Systems Development Life Cycle: Planning – Analysis - Design - Implementation – Maintenance.

4- **Implementation**: The phase of a Systems Development Life Cycle where the programmers would be assigned to write specific programs.

3- Modularization

1- Concept of Modularization: One of the most important concepts of programming is the ability to group some lines of code into a unit that can be included in our program. The original wording for this was a sub-program. Other names include: macro, sub-routine, procedure, module and function. Generally functions fall into **two categories:**

a- Program Control: Functions used to simply sub divide and control the program. These functions are unique to the program being written. Other programs may use similar functions maybe even functions with the same name, but the content of the functions are almost always very different.

b- Specific Task: Functions designed to be used with several programs. These functions perform a specific task and thus are use able in many different programs because the other programs also need to do the specific task. Specific task functions are sometimes referred to as building blocks.

The main program must establish the existence of functions used in that program. Depending on the programming language, there is a formal way to:

1. Define a function (it's definition or the code it will execute).
2. Call a function.
3. Declare a function (a prototype is a declaration to a complier).

Lectured Two

The Context of Software Development

A computer program, from one perspective, is a sequence of instructions that dictate the flow of electrical impulses within a computer system. These impulses affect the computer’s memory and interact with the display screen, keyboard, mouse, and perhaps even other computers across a network in such a way as to produce the “magic” that permits humans to perform useful tasks, solve high-level problems, and play games. One program allows a computer to assume the role of a financial calculator, while another transforms the machine into a worthy chess opponent. **<u>Note the two extremes here:</u>**

1- **At the lower,** more concrete level electrical impulses alter the internal state of the computer.

2- **At the higher,** more abstract level computer users accomplish real-world work or derive actual pleasure.

So well is the higher-level illusion achieved that most computer users are oblivious to the lower-level activity (the machinery under the hood, so to speak). Powerful software construction tools hide the lower-level details from programmers, allowing them to solve problems in higher-level terms.

The concepts of computer programming are **logical and mathematical** in nature. Computer programs can be developed without the use of a computer. Programmers can discuss the viability of a program and reason about its correctness and efficiency by examining abstract symbols that correspond to the features of real-world programming languages but appear in no real-world programming language.

1- Software:

A computer program is an example of computer software. Software makes a computer a truly universal machine transforming it into the proper tool for the task at hand. One can refer to a program as a piece of software as if it were a tangible object, but software is actually quite intangible. It is stored on a medium. A hard drive, a CD, a DVD, and a USB pen drive are all examples of media upon which software can reside.

The CD is not the software; the software is a pattern on the CD. In order to be used, software must be stored in the computer's memory. Typically computer programs are loaded into memory from a medium like the computer's hard disk.

2- Development Tools:

If very few humans can to speak the machine language of the computers' processors and software is expressed in this language, how has so much software been developed over the years?

Software can be represented by printed words and symbols that are easier for humans to manage than binary sequences. Tools exist that automatically convert a higher-level description of what is to be done into the required lower-level code. Higher-level programming languages like C++ allow programmers to express solutions to programming problems in terms that are much closer to a natural language like English.

Some examples of the more popular of the hundreds of higher-level programming languages that have been devised over the past 60 years include FORTRAN, COBOL, C, Java, and C#. Most programmers today, especially those concerned with high-level applications, usually do not worry about the details of underlying hardware platform and its machine language.

- Consider the following program fragment written in the C++ programming language:

subtotal = 25;

tax = 3;

total = subtotal + tax;

These three lines do not make up a complete C++ program; they are merely a piece of a program. The statements in this program fragment look similar to expressions in algebra. Three words, **subtotal**, **tax**, and **total**, called variables, are used to hold information.

The higher-level language code is **called source code.** The compiled machine language code is **called the target code.** The **compiler translates** the source code into the target machine language.

- **Programmers have a variety of tools available to enhance the software development process.**
- **The complete set of build tools for mostly languages includes a:**

1- Editors: An editor allows the user to enter the program source code and save it to files. Most programming editors increase programmer productivity by using colors to highlight language features.

The syntax of a language refers to the way pieces of the language are arranged to make well-formed sentences. To illustrate,

the sentence **the tall boy runs quickly to the door.**

Uses proper English syntax. By comparison, the sentence **Boy the tall runs door to quickly the .** Is not correct syntactically. It uses the same words as the original sentence, but their arrangement does not follow the rules of English.

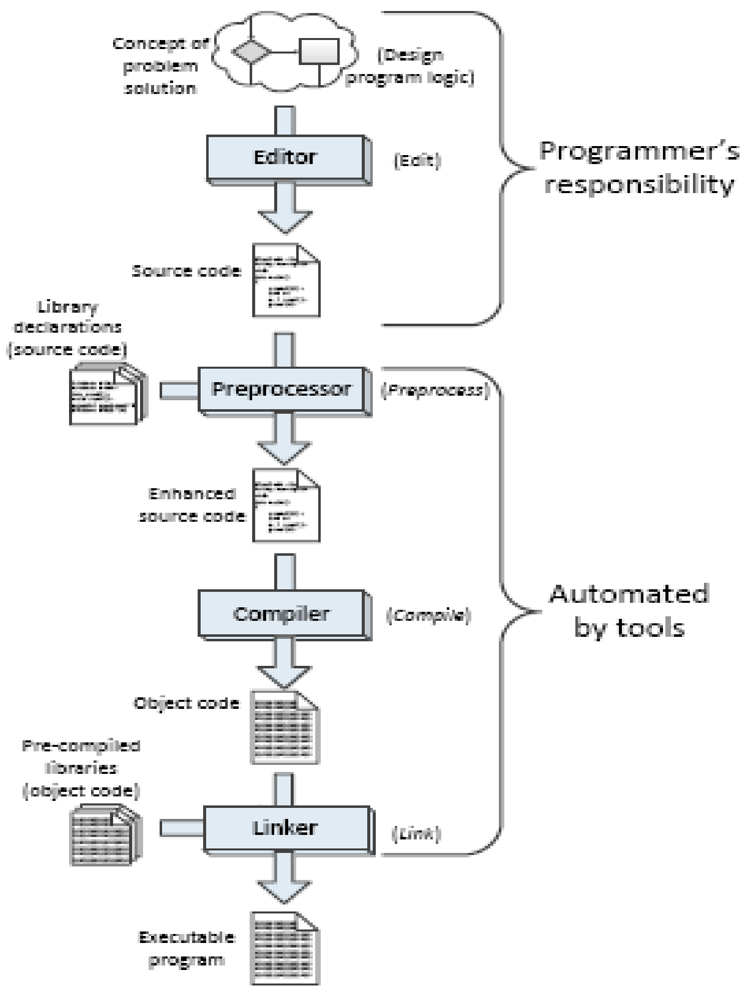


Figure (1) Source code to target code sequence

2- Compilers: A compiler translates the source code to target code. The target code may be the machine language for a particular platform or embedded device. The target code could be another source language; for example, the earliest C++ compiler translated C++ into C, another higher-level language. The resulting C code was then processed by a C compiler to produce an executable program. C++ compilers today translate C++ directly into machine language.

3- Preprocessor: preprocessor adds to or modifies the contents of the source file before the compiler begins processing the code.

We use the services of the preprocessor mainly to **#include** information about library routines our programs use.

4- Linker: linker combines the compiler-generated machine code with precompiled library code or compiled code from other sources to make a complete executable program. Most compiled – C++ code is incapable of running by itself and needs some additional machine code to make a complete executable program.

The missing machine code has been precompiled and stored in a repository of code called a library. A program called a linker combines the programmer's compiled code and the library code to make a complete program.

- **the preprocessor, compiler, and linker working as three separate programs (although they do); the tools we use make it appear as only one process is taking place; translating our source code to an executable program.**

5- Debuggers: A debugger allows a programmer to more easily trace a program's execution in order to locate and correct errors in the program's implementation. With a debugger, a developer can simultaneously run a program and see which line in the source code is responsible for the program's current actions. The programmer can watch the values of variables and other program elements to see if their values change as expected. Debuggers are valuable for locating errors (also called bugs) and repairing programs that contain errors.

6- Profilers: A profiler collects statistics about a program's execution allowing developers to tune appropriate parts of the program to improve its overall performance. A profiler indicates how many times a portion of a program is executed during a particular run, and how long that portion takes to execute. Profilers also can be used for testing purposes to ensure all the code in a program is actually being used somewhere during testing. This is known as coverage. It is common for software to fail after its release because users exercise some part of the program that was not executed anytime during testing. The main purpose of profiling is to find the parts of a program that can be improved to make the program run faster.

Lectured Three

Program Planning and Design

Program Design consists of the steps a programmer should do before they start coding the program in a specific language. These steps when properly documented will make the completed program easier for other programmers to maintain in the future.

- **There are three broad areas of activity:**

  1- Understanding the Program.

  2- Using Design Tools to Create a Model.

  3- Develop Test Data.

1- Understanding the Program:

If you are working on a project as a one of many programmers, the system analyst may have created a variety of documentation items that will help you understand what the program is to do. These could include screen layouts, narrative descriptions, documentation showing the processing steps, etc.

- Inputs
- Processing
- Outputs

This **IPO** approach works very well for beginning programmers. Sometimes, it might help to visualize the programming running on the computer. You can imagine what the monitor will look like, what the user must enter on the keyboard and what processing or manipulations will be done.

2- Using Design Tools to Create a Model:

At first you will not need a hierarchy chart because your first programs will not be complex. But as they grow and become more complex, you will divide your program into several modules (or functions).

The first modeling tool you will usually learn is **pseudocode**. You will document the logic or algorithm of each function in your program. At first, you will have only one function, and thus your **pseudocode** will follow closely the **IPO** approach above.

- There are **several methods** or **tools for planning** the logic of a program. They include: **flowcharting, hierarchy or structure charts**, **pseudocode,** etc. Programmers are expected to be able to understand and do **fowcharting** and **pseudocode.**

3- Develop Test Data:

Test data consists of the user providing some input values and predicting the outputs. This can be quite easy for a simple program and the test data can be used to check the model to see if it produces the correct results.

Pseudocode:

Pseudocode is one method of designing or planning a program. Pseudo means false, thus pseudocode means false code. A better translation would be the word fake or imitation. Pseudocode is fake (not the real thing). It looks like (imitates) real code but it is NOT real code. It uses English statements to describe what a program is to accomplish. It is fake because no complier exists that will translate the pseudocode to any machine language. Pseudocode is used for documenting the program or module design (also known as the algorithm).

Example Pseudocode:

```
Input
  display a message asking the user to enter the first age
  get the first age from the keyboard
  display a message asking the user to enter the second age
```

```
    get the second age from the keyboard
Processing
    calculate the answer by adding the two ages together and dividing by two
Output
    display the answer on the screen
    pause so the user can see the answer
```

Definition pseudo: Means false and includes the concepts of fake or imitation.

Test Data:

Test data consists of the user providing some input values and predicting the outputs. This can be quite easy for a simple program and the test data can be used twice.

1. To check the model to see if it produces the correct results (model checking).
2. To check the coded program to see if it produces the correct results (code checking).

Test data is developed by using the algorithm of the program. This algorithm is usually documented during the program design with either flowcharting or pseudocode.

Example: Pseudocode using an IPO Outline for Painting a Rectangular Building

```
Input
display a message asking user for the length of the building
get the length from the keyboard
display a message asking user for the width of the building
get the width from the keyboard
display a message asking user for the height of the building
```

get the height from the keyboard

display a message asking user for the price per gallon of paint

get the price per gallon of paint from the keyboard

display a message asking user for the sq ft coverage of a gallon of paint

get the sq ft coverage of a gallon of paint from the keyboard

<u>Processing</u>

calculate the total area of the building by:

multiplying the length by height by 2

then multiply the width by height by 2

then add the two results together

calculate the number of gallons of paint needed by:

dividing the total area by the coverage per gallon

then round up to the next whole gallon

calculate the total cost of the paint by:

multiplying the total gallons needed by the price of one gallon of paint

<u>Output</u>

display the number of gallons needed on the monitor

display the total cost of the paint on the monitor

pause so the user can see the answer

Lectured Four

Top down design

Top-down design is basically a decomposition process which focuses on the flow of control or on the control structure of the program; at later stages it concerns itself with code production. The first step is to study the overall aspects of the task at hand and break it into a number(perhaps 3 to 1 of independent constituent functions or modules. this is the first step in the decomposition. the second step is to break each one of these modules further into independent sub modules. the process is repeat 5ed until one obtains modules which are small enough to grasp mentally and to code at one sitting in a straightforward uncomplicated manner. clearly, one module of the structure may extend to a lower4 level than the next.

One important feature of top down design is that at each level the details of the design at lower levels are hidden. One the necessary data and control which must be passed back and forth over the interfaces are defined .furthermore, if a data structure is contained wholly within a lower level module, it need not be specified until that level is reached in design process.

However, if data must be shared by several modules at some level, then the data structure must be chosen before processing to a lower level. The design will include both the data structure and the means of data access for each involved module.

One strong point of the top down method is that postpones details of decisions until the last stages of the design. This allows one to accommodate easily small design changes or improvements of technology partway through the design. There is also a concomitant danger that the specifications will be incompatible or unrealizable and that this will not be discovered until late in the design process.

Example:-

A simple example which illustrates top-down is given in figure 1.the program is to solve for and classify the roots of the cubic equation

$$Ax^3 + Bx^2 + Cx + D = 0$$

The top down solution does not commit the design to a specific approach at the first decomposition level. This will come when we go to the second level of detail.

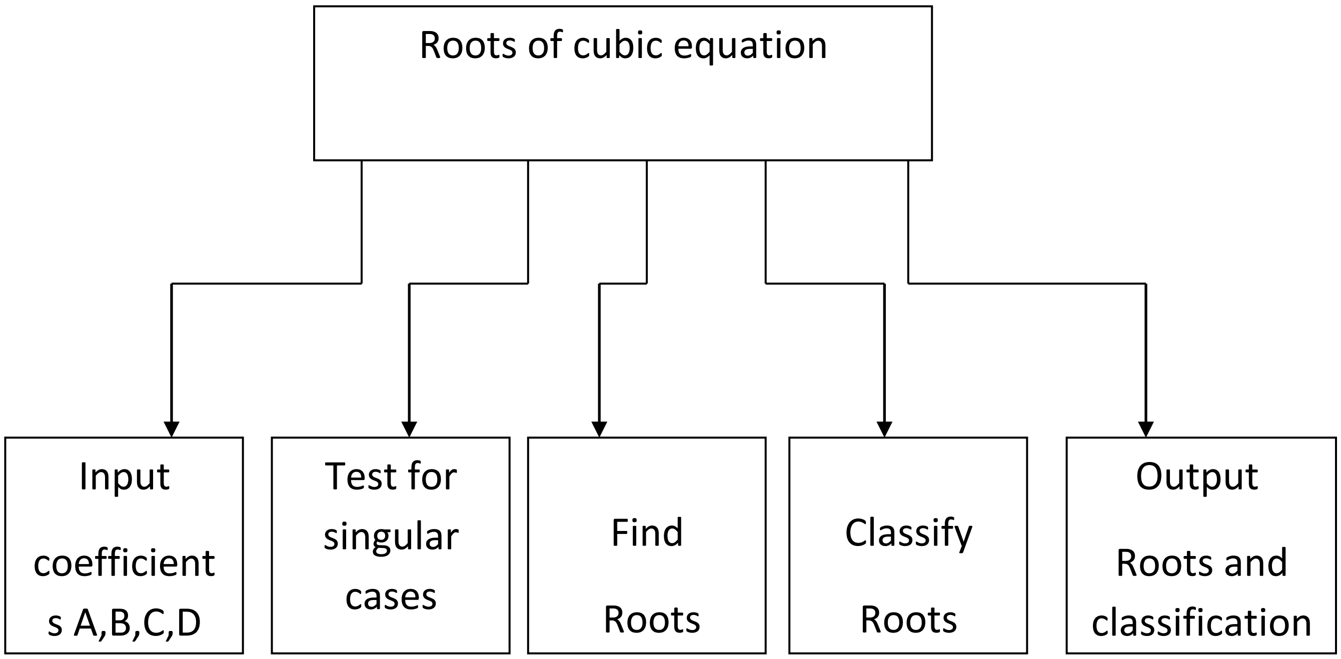


Figure 1: first decomposition level for a top-down design

Computer program developments

You will understand the steps that are used by programmers to develop computer programs and to ensure that they function properly.

You will see how the programmer uses the flowcharts, input design, output design, and file design results of the design phase activities to develop a computer program.

Algorithm a set of rules or instructions used to accomplish a task.

Coding is the process of writing instructions in a programming language, such as C++.

Debugging is the process of testing a computer program for errors and correcting any errors found.

Steps in Computer Program Development

The steps in the development of each of the computer programs that make up the computer program component of a system are

(1) Define the function of the program

(2) Plan the logic of the program

(3) Code the program

(4) Test and debug the program

(5) Complete the documentation.

Although the programmer is responsible for writing the computer program, the systems analyst must communicate the computer program requirements to the programmer. The function of each program was defined for the programmer when functions were allocated during system design.

Detailed data **F**low **D**iagrams are prepared for each program from the decomposed DFDs created during the design phase. These DFDs define the function of each program.

In program planning, the logic to be used to solve the problem is developed. Algorithms, computer program logic flowcharts, and structure charts are useful tools for program planning. Algorithms are sets of rules or instructions used to accomplish tasks. They may be stated as formulas, decision tables, or narratives. The program logic flowchart, pseudo code, structure chart, and algorithms that result from program planning are retained and become part of the project documentation.

The next step, writing, or coding, a program, is the actual writing of computer instructions. These instructions will be translated to machine code and followed by the computer; they should follow the steps of the program logic plan.

Several programming languages, particularly c++ and visual studio, are commonly used to solve business problems. In addition to these traditional languages, organizations using data base management systems may choose to generate programs using the query language of the data base management systems. These query languages are a part of a package of programming tools known as fourth-generation languages. Each language has its advantages and disadvantages.

Testing and debugging a program involve:

(a) Translating the coded program into machine language, a process called compilation

(b) Testing the translated program with sample data and checking the result. If the results of testing are not correct, the program is said to have "bugs." Debugging is the process of correcting computer programs to obtain correct results.

Testing must be planned and structured to reduce the chance that errors will be overlooked.

The last step is to complete the documentation for the program. The documentation must include a statement of the purpose of the program (from step 1), a description of the solution logic (step 2), a listing of the program instructions (step 3), and sample outputs from the completed programs (step 4). Information provided to the programmer by the analyst, such as descriptions of program inputs, outputs, and files, should be included. Instructions to operators explaining how the program is to be used must be written before the program documentation is complete.

Recursion

Recursion, which is facilitated by automatic storage, is an important feature that allows the user to handle a broad range of problems. Many of the routines we must write are Recursive.

Many algorithms are much more convenient to describe and program in a recursive manner. For example, the standard mathematical definition of factorial is as follows:

$$n! = \begin{cases} 1 & \text{if } n=0 \\ n*(n-1)! & \text{if } n\neq 0 \end{cases}$$

A straightforward implementation of this function is shown in figure 2:

```
N_Factorial: procedure (n)        recursive;
If N=0 then return (1)
Else
Return (N*N_Factorial (N-1));
End;
```

Figure 2: Example of recursive procedure

Demonstrate the key issues in recursive procedures. The algorithm simply notes

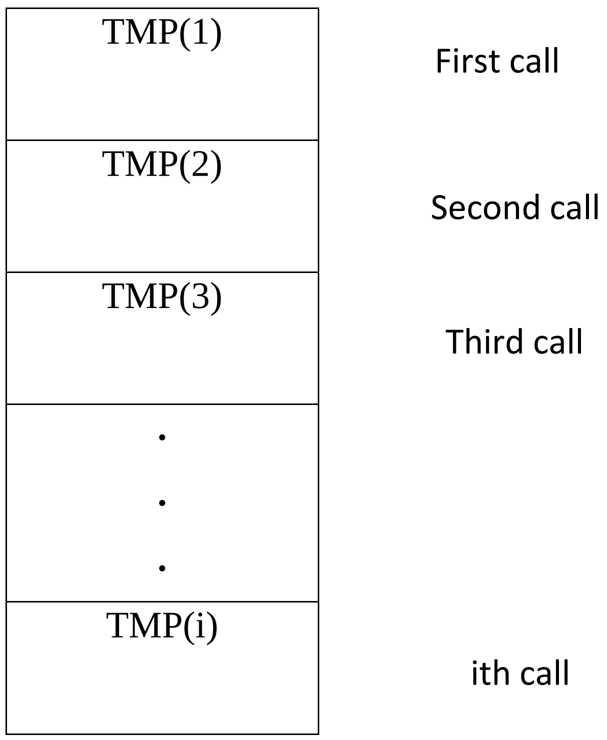
N!=N*(N-l)!Thus,

3! =3*2! =3*2*l!=3*2*1*0!=3*2*I*1.

Thus, we may subtract one from N and call factorial again. If N = 0, then return.

The procedure of Figure 3 computes 3 factorial and leaves the results in F. Storage is assigned TMP at the end of the program. Figure 3 depicts the values of these variables during the calls.

Each call sets I = 1+1, thereby asserting a new TMP with each call. A return sets I I-I, thus re-establishing the TMP associated with the previous call.



This scheme of storing variables is loosely called a stack, where setting I= I+1 (creating a new, TMP) corresponds to push down, and returning (I = I-1) corresponds to pop.

```
fact:   procedure;
              declare n fixed bin static intial(3);
              declare (f,tmp) fixed bin static
               if  n=0      then   f= 1;
           else do;
             tmp= n;
              n =n-1;
        call fact;
          f=f*tmp    end;
```

return; end;

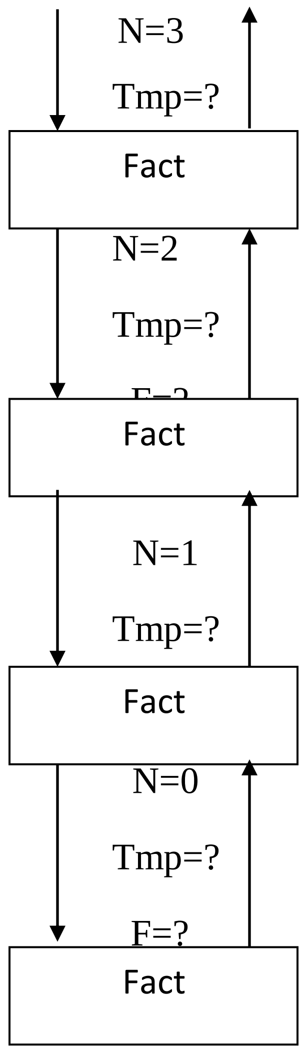


Figure 3: computation of N!

Lecture Five

Structured Programming versus Object-Oriented Programming

The structured programming approach to program design was **based on the following method:**

- To solve a large problem, break the problem into several pieces and work on each piece separately;
- to solve each piece, treat it as a new problem that can itself be broken down into smaller problems;
- repeat the process with each new piece until each can be solved directly, without further decomposition.

This approach is also called **top-down program design**.

The following is a simple example of the structured programming approach to problem solving.

Write a program for a computer to execute to display the average of two numbers entered through a keyboard connected to the computer. The average is to be displayed on a VDU that is also connected to this computer.

<u>The top-down solution is arrived at as follows:</u>

Top level: 0. Display average of two numbers entered through keyboard

Next level: 0.1. Get two numbers through keyboard
0.2. Calculate average of these two numbers

0.3. Display average on VDU

The three steps in *next level* can now be coded in a programming language such as c++.

Top-down program design is a useful and often-used approach to problem solving. However, it has limitations:

- It focuses almost entirely on producing the **instructions** necessary to solve a problem. The design of the **data structures** is an activity that is just as important but is largely outside of the scope of top-down design.
- It is difficult to reuse work done for other projects. By starting with a particular problem and subdividing it into convenient pieces, top-down program design tends to produce a design that is unique to that problem. Adapting a piece of programming from another project usually involves a lot of effort and time.
- Some problems by their very nature do not fit the model that top-down program design is based upon. Their solution cannot be expressed easily in a particular sequence of instructions. When the order in which instructions are to be executed cannot be determined in advance, easily, a different approach is required.

Top-down design was therefore combined with **bottom-up design**.

<u>**In bottom-up design**</u>, the approach is to start “at the bottom”, with problems that have already been solved and for which a reusable software component might exist. From this position, the software engineer works upwards towards a solution to the overall problem.

It is important in this approach that the reusable components are as “modular” as possible.

A module is a component of a larger system that interacts with the rest of the system in a simple and well-defined manner.

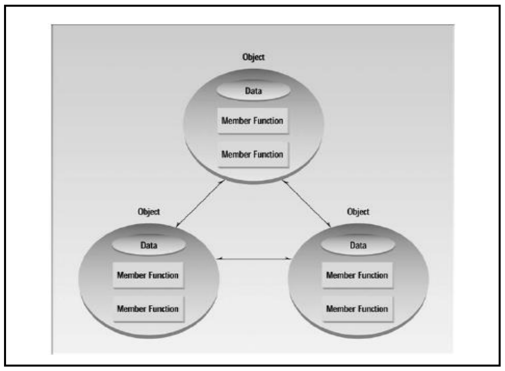
The idea is that a module can be “plugged into” a system. The details of what goes on inside a module are not important to the system as a whole, only that the module fulfils its function correctly. For example, a module might contain procedures to print a list of students, to add a new student, edit a student’s details and to return a list of specified students. How the module stores the master records of student details is hidden from applications/systems that use this module. Similarly, the detail of how the various procedures are coded is also hidden. This is called **information hiding**. Applications only require knowledge of what procedures are available from the module and the data that can be accessed. This information is published. It is often called the **module’s interface or interfaces**.
A common format for a software module is a module containing some data, along with some subroutines (subprograms/procedures/functions) for manipulating that data. The data itself is often hidden from view inside the module forcing a program using the module to manipulate the data indirectly, by calling the subroutines provided by the module for this purpose. The advantages of this approach are as follows:

- the data is protected, since it can be manipulated only in known, well-defined ways;

- it is easier to write programs to use a module because the details of how the data is represented and stored need not be known;
- the storage structure of the data and the code for the subroutines in a module may be altered without affecting programs that make use of the module as long as the published interfaces and the module's functionality remain the same.

The Object-Oriented Approach

The fundamental idea behind object-oriented languages is to combine into a single unit both data and the functions that operate on that data. Such a unit is called an object. An object's functions, called member functions in C++, typically provide the only way to access its data. If you want to read a data item in an object, you call a member function in the object. It will access the data and return the value to you. You can't access the data directly. The data is hidden, so it is safe from accidental alteration. Data and its functions are said to be **encapsulated** into a single entity. **Data encapsulation and data hiding are key terms in the description of object-oriented languages.** If you want to modify the data in an object, you know exactly what functions interact with it: the member functions in the object. No other functions can access the data. This simplifies writing, debugging, and maintaining the program. A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions.



The organization of an OOP C++ program

OOP: An Approach to Organization

Object-oriented programming is not primarily concerned with the details of program operation. Instead, it deals with the overall organization of the program. Most individual program statements in C++ are similar to statements in procedural languages, and many are identical to statements in C. Indeed, an entire member function in a C++ program may be very similar to a procedural function in C. It is only when you look at the larger context that you can determine whether a statement or a function is part of a procedural C program or an object-oriented C++ program.

Characteristics of Object-Oriented Languages

- Objects
- Classes
- Inheritance
- Reusability
- Creating New Data Types
- Polymorphism and Overloading

Objects

When you approach a programming problem in an object-oriented language, you no longer ask how the problem will be divided into functions, but how it will be divided into objects. Thinking in terms of objects, rather than functions, has a surprisingly helpful effect on how easily programs can be designed. This results from the close match between objects in the programming sense and objects in the real world.

Classes

In OOP we say that objects are members of classes. What does this mean? Almost all computer languages have built-in data types. For instance, a data type int, meaning integer, is predefined in C++ . You can declare as many variables of type int as you need in your program:

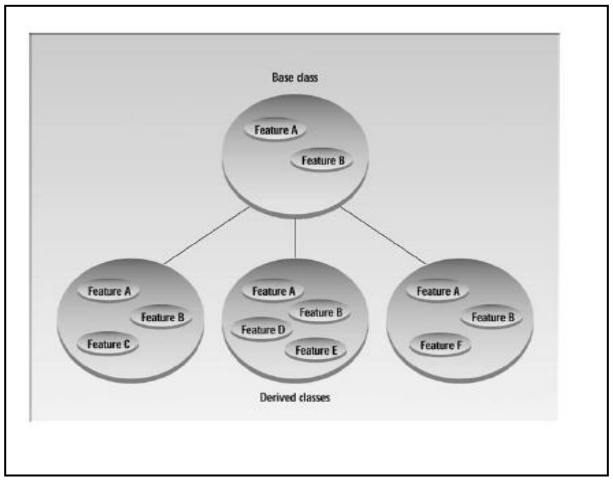
```
int day;
int count;
int divisor;
int answer;
```

In a similar way, you can define many objects of the same class. It specifies what data and what functions will be included in objects of that class. Defining the class doesn't create any objects, just as the mere existence of data type int doesn't create any variables. A class is thus a description of a number of an object is often called an "**instance**" of a class.

Inheritance

The idea of classes leads to the idea of inheritance. In our daily lives, we use the concept of classes divided into subclasses. We know that the animal class is divided into mammals, amphibians, insects, birds, and so on. The vehicle class is divided into cars, trucks, buses, motorcycles, and so on.

In C++ the original class is called the base class; other classes can be defined that share its characteristics, but add their own as well. These are called derived classes. Don’t confuse the relation of objects to classes, on the one hand, with the relation of a base class to derived classes, on the other. Objects, which exist in the computer’s memory, each embody the exact characteristics of their class, which serves as a template. Derived classes inherit some characteristics from their base class, but add new ones of their own.



Reusability

Once a class has been written, created, and debugged, it can be distributed to other programmers for use in their own programs. This is called reusability. It is similar to the way a library of functions in a procedural language can be incorporated into different programs.

Creating New Data Types

One of the benefits of objects is that they give the programmer a convenient way to construct new data types. Suppose you work with two-dimensional positions (such as x and y coordinates, or latitude and longitude) in your program. You would like to express operations on these positional values with normal arithmetic operations, such as:

position1 = position2 + origin

where the variables position1, position2, and origin each represent a pair of independent numerical quantities. By creating a class that incorporates these two values, and declaring position1, position2, and origin to be objects of this class, we can, in effect, create a new data type. Many features of C++ are intended to facilitate the creation of new data types in this manner.

Polymorphism and Overloading

Note that the = (equal) and + (plus) operators, used in the position arithmetic shown above, don't act the same way they do in operations on built-in types such as int. The objects position1 and so on are not predefined in C++, but are programmer-defined objects of class Position. How do the = and + operators know how to operate on objects? The answer is that we can define new behaviors for these operators. These operations will be member functions of the Position class. Using operators or functions in different ways, depending on what they are operating on, is called polymorphism (one thing with several distinct forms). When an existing operator, such as + or =, is given the capability to operate on a new data type, it is said to be overloaded. Overloading is a kind of polymorphism; it is also an important feature of OOP.

A Simple Class

```
#include <iostream>
using namespace std;
class smallobj    //define a class
{
private:
int somedata; //class data
public:
void setdata(int d)     //member function to set data
{ somedata = d; }
void showdata()         //member function to display data
{ cout << “Data is “ << somedata << endl; }
};
int main()
{
smallobj s1, s2;           //define two objects of class smallobj
s1.setdata(1066);     //call member function to set data
s2.setdata(1776);
s1.showdata();         //call member function to display data
s2.showdata();
return 0;
}
```

Classes and Objects

An object has the same relationship to a class that a variable has to a data type. An object is said to be an instance of a class. In SMALLOBJ, the class—whose name is smallobj—is defined in the first part of the program. Later, in main(), we define two objects—s1 and s2—that are instances of that class.

Each of the two objects is given a value, and each displays its value. Here's the output of the program:

```
— Data is 1066          object s1 displayed this

— Data is 1776          object s2 displayed this
```

Defining the Class

Here's the definition (sometimes called a specifier) for the class smallobj, copied from the SMALLOBJ listing:

```
class smallobj //define a class
{
private:
int somedata; //class data
public:
void setdata(int d) //member function to set data
{ somedata = d; }
void showdata() //member function to display data
{ cout << "\nData is " << somedata; }
};
```

The definition starts with the keyword class, followed by the class name—smallobj in this example. Like a structure, the body of the class is delimited by braces and terminated by a semicolon.

Private and Public

The body of the class contains two unfamiliar keywords: private and public. What is their purpose?

A key feature of object-oriented programming is data hiding. This term does not refer to the activities of particularly paranoid programmers; rather it means that data is concealed within a class so that it cannot be accessed mistakenly by functions outside the class. The primary mechanism for hiding data is to put it in a class and make it private. Private data or functions can only be accessed from within the class. Public data or functions, on the other hand, are accessible from outside the class.

Class Data

The smallobj class contains one data item: somedata, which is of type int. The data items within a class are called data members (or sometimes member data). There can be any number of data members in a class, just as there can be any number of data items in a structure. The data member somedata follows the keyword private, so it can be accessed from within the class, but not from outside.

Member Functions

Member functions are functions that are included within a class. There are two member functions in smallobj: setdata() and showdata().

```
void setdata(int d)
```

```
{
somedata = d;
}
        —    And
void showdata()
{
cout << "\nData is " << somedata;
}
```

Because setdata() and showdata() follow the keyword public, they can be accessed from outside the class.

Defining Objects

The first statement in main()

```
—
— smallobj s1, s2;
```

defines two objects, s1 and s2, of class smallobj. Remember that the definition of the class smallobj does not create any objects. It only describes how they will look when they are created, just as a structure definition describes how a structure will look but doesn't create any structure variables. It is objects that participate in program operations. Defining an object is similar to defining a variable of any data type: Space is set aside for it in memory. Defining objects in this way means creating them. This is also called instantiating them. The term instantiating arises because an instance of the class is created. An object is an instance (that is, a specific example) of a class. Objects are sometimes called instance variables.

Calling Member Functions

The next two statements in main() call the member function setdata():

```
s1.setdata(1066);
s2.setdata(1776);
```

These statements don't look like normal function calls. Why are the object names s1 and s2 connected to the function names with a period? This strange syntax is used to call a member function that is associated with a specific object. Because setdata() is a member function of the smallobj class, it must always be called in connection with an object of this class. It doesn't make sense to say setdata(1066); Some object-oriented languages refer to calls to member functions as messages.

Example2

```
#include "stdafx.h"
#include <iostream.h>
class Point {
int xVal, yVal;
public:
void SetPt (int, int);
void OffsetPt (int, int);
};
void Point::SetPt (int x, int y)
{
xVal = x;
yVal = y;
cout<< "x="<<x <<"  "<< "y="<<y<<endl;
}
void Point::OffsetPt (int x, int y)
{
xVal += x;
yVal += y;
cout<< "x="<<xVal <<"  "<< "y="<<yVal<<endl;;
}
int main(int argc, char* argv[])
{
Point pt; // pt is an object of class Point
pt.SetPt(10,20); // pt is set to (10,20)
pt.OffsetPt(2,2); // pt becomes (12,22)
        return 0;
}
```

Lecture Six

Web Applications, Web Site and Desktop Application

6.1- What is a Web Application?

Many terms are bantered about in the online world; website, web application, web 2.0, etc. But what is truly the difference between a website and a web application (web app)? You can think of the difference as being one of interactivity and data manipulation.

6.2- What's the major difference?

A standard **website** is generally content-centric. That means it focuses on providing the web user with information, generally in a static layout with static links to other pages filled with static content. In a nutshell, there's not much to do aside from read each page. A **web application** means there is more to be done. It means some task can be accomplished, a goal can be attained or some expectation can be met. That is all rather esoteric so let's delve further into it with some sort of an example.

This article, which you are undoubtedly reading on a website, is a classic example of a content-centric page. It contains words and perhaps images, but it only allows you to read or view the content, there is no manipulation or interaction except perhaps some form of commenting, rating system or ability to share it with others through various social networking sites. There is very little interaction.

6.3- What Makes Something A Web Application

A web application on the other hand might be something like Gmail where you have a specific instance of the application that you alone see. Your email and your interaction with the site is completely separate from that of others. You see the page differently than others do and are able to affect changes to it for yourself and no others. It is interactive in that you can send and receive information in the form of emails, attachments etc.

6.4- The main differences of a web application are basically that:

Each user has a session-based relationship. That means the application is somehow aware of who you are and loads a specific set of variables for your interface.

Each user can change the interface for their own session. This generally manifests itself in things like themes, colors, organization of elements, etc.

Users can permanently create, store and change data. This can be as simple as an email message or as complex as a multi-page spreadsheet in a web application like Google Docs or even an image in Pixlr or a video at Animoto

6.5- Content and Interaction

What if a site has both content and interactivity? Take Amazon.com for example. There are numerous pages where you read product descriptions, see photos, read user reviews and comments. However, you can login and Amazon knows who you are (figuratively) and changes your environment based on your identity. You can also create and edit wish lists (data). That means it fulfills all of the requirements for it to be a web application and therefore should be classified as one.

Web applications need not be extremely interactive or offer a multitude of things to do and ways to interact to be considered an application. They need only meet specific requirements that have been outlined above. Therefore an application may be as simple as an email program or as complex as a complete Enterprise Resource

Planning package which allows you to interact with all facets and departments in a multinational corporation.

6.6- Understanding Desktop Application

A desktop application means any software that can be installed on a single computer (laptop or a desktop) and used to perform specific tasks. Some desktop applications can also be used by multiple users in a networked environment. Web application development, however, soon started replacing desktop applications for reasons of portability and better functions from usability point of view. Web application development is usually made on client-server architecture and use a web-browser as the client interface. This is one of the reasons why web applications are so widely getting popular. Though web applications offer a slight advantage over desktop applications, there is a very narrow chance of desktop applications becoming outdated. The primary reason for this could be the security issues and legalities associated with web based applications.

Following is a basic comparison on desktop and web based applications based on certain parameters:

Maintenance - web based applications need to be installed only once where as desktop applications are to be installed separately on each computer. Also updating the applications is cumbersome with desktop applications as it needs to be done on every single computer which is not the case with web applications.

Ease of use - desktop applications are confined to a physical location and hence have usability constraint. Web applications development on the other hand makes it convenient for the users to access the application from any location using the Internet.

Security - web applications are exposed to more security risks than desktop applications. You can have a total control over the standalone applications and protect it from various vulnerabilities. This may not be the case with web applications as they are open to a large number of users in the Internet community thus widening the threat.

Connectivity - web application development relies significantly on Internet connectivity and speed. Absence of Internet or its poor connectivity can cause performance issues with web applications. Desktop applications are standalone in nature and hence do not face any hindrances resulting from Internet connectivity. Connectivity also significantly affects the speed at which desktop and web applications operate.

Cost factor - web application development and its maintenance involve higher costs and mostly recurring in nature. Desktop applications are purchased one time and there are not continually occurring charges. However, in certain cases,

Therefore the difference between Desktop Application and Web Applicatio, the desktop applications truly know who you are? The answer is generally no and that is one major defining factor between them and web applications. When you go to a shared computer at a web cafe and load Photoshop, it does not differentiate you from any other user who logs into that computer and will load the same interface for each user.

When you log into a web application it does have some set of information about you that delineates you from other users of that web application. That means that you can have an environment customized to your specific user identity. This can be done to some degree in many desktop applications but is not always available. In web applications it must be in order to be a full-featured and full-fledged web application.

6.7- Development Considerations

What this all means is that web application development has a differing set of factors to take into account and they must include the ability to recognize individual users (generally through a username and password) whereas a desktop application doesn't necessarily need that functionality in order to still be considered a desktop application. Additionally, a web application, as the name suggests, is available on the web and can be accessed from any computer generally without the installation of any local files. A desktop application must generally have some files installed on the computer itself in order to function.

6.8- Web Application Programming Languages

Another major point for a web application is that it is programmed in a language that is understandable by a web browser. Since they are applications on the web they must be accessed somehow. The standard interface is through a web browser. Web browsers understand a finite amount of languages which means that web applications must be programmed in one of them to be understood. The following is a list of dominant languages that web applications can be programmed in:

HTML, DHTML, XHTML

XML

Flash

Javascript

Java

PHP

ASP

ActiveX

AJAX (a combination of javascript and XML)

6.9- Web Application Structure

Web applications now come in several varieties including 2-tier and 3-tier which refer to the number of levels of the applications. The three-tiered approach is most common at present and represents presentation, application and storage. The presentation layer is the web browser while the application layer resides on the server and includes the files in the particular programming languages and/or some sort of server technology that helps translate information to the other layers (Ruby on Rails, PHP, etc). The storage layer is generally some sort of database which stores the information that is passed to the other layers. The application layer is basically the brains of the web application and allows the other two layers to interact in a more user-friendly way by supplying both with the required information.

With the advent of Web 2.0, web applications have become abundant. Web 2.0 itself means there is the ability to share information, collaboration across multiple computers, operation across multiple operating systems and a user interface that can be edited by the user. If you're doing more than just reading content on a site, if you're interacting with other users and/or editing the colors, layout and options of your web interface, you're most likely using a web application.

Lecture Seven

Understanding Data Base

1- Introduction

Today, people use computers to perform many tasks formerly done with other tools. Computers have replaced typewriters for creating and modifying documents.

They’ve surpassed electromechanical calculators as the best way to do math. They’ve also replaced millions of pieces of paper, file folders, and file cabinets as the principal storage medium for important information. Compared to those old tools, of course, computers do much more, much faster — and with greater accuracy. These increased benefits do come at a cost, however. Computer users no longer have direct physical access to their data. When computers occasionally fail, office workers may wonder whether computerization really improved anything at all. In the old days, a manila file folder only “crashed” if you dropped it — then you merely knelt down, picked up the papers, and put them back in the folder. Barring earthquakes or other major disasters, file cabinets never “went down,” and they never gave you an error message.

A hard drive crash is another matter entirely: You can’t “pick up” lost bits and bytes. Mechanical, electrical, and human failures can make your data go away and never to return.

If you are storing important data, you have four main concerns:

1- Storing data needs to be quick and easy, because you’re likely to do it often.

2- The storage medium must be reliable. You don’t want to come back later and find some (or all) of your data missing.

3- Data retrieval needs to be quick and easy, regardless of how many items you store.

4- You need an easy way to separate the exact information that you want today from the tons of data that you don't want right now.

What is data?

Data can be defined in many ways. Information science defines data an unprocessed information.

What is information?

Information is data that have been organized and communicated in a coherent and meaningful manner.

- Data is converted into information, and information is converted into knowledge.

- Knowledge is information evaluated and organized so that it can be used purposefully as shown in figure (1.1)

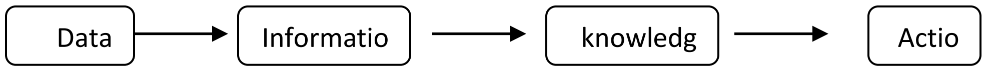


Figure (1)

2 - What is a DataBase ?

A **database** is an organized collection of data for one or more uses, typically in digital form. The data can be textual, like order or inventory data, or it can be pictures, programs or anything else that can be stored on a computer in binary form.

One way of classifying databases involves the type of their contents, for example: bibliographic, document-text, statistical.

The purpose of a database is to store and retrieve related information, so databases are designed to offer an organized mechanism for :

- Storing
- managing
- and retrieving information.

3- Files System:

The File is a block of arbitrary information, it is a place that application programs stores there data in it. These application programs either database application or non-database application. Each file has a format. The information stored in the file can be organized in a record, which is a collection of fields.

The file system is typically described as various files and a number of different application programs are written to read from and add to the appropriate files.

File System Disadvantage:

- Program dependence: Each file has a format, the non-database application must know exactly the format of the file to deal with it. Any other application cannot access the file unless knowing the format of the file.
- When file format updated, then the application program must be updated, it is complicated to update all programs when data format is update.
- Security problems existed. Any one can write a program to read the data in the file.

- Data redundancy , if there are application A deals with file A and application B deals with file B, if application A store an information in file A, and if application B need this information , application B can not access file A , so application B must record the same information in file B.

<u>Some basic Definitions:</u>

Field: one category of information (one data value), i.e., Name, Address, Semester Grade, Academic topic.

Record: Collection of fields i.e., one student's information, a recipe, a test question.

A File: A group or collection of similar records, like student File.

Digital databases are managed using **<u>database management systems (DBMS)</u>** , which store database contents, allowing data creation and maintenance, and search and other access to the database.

4 What is DBMS ?

A **Database Management System** (**DBMS**) is a set of computer programs that controls the –

- Creation of the database
- The storing and organization of the data in the database
- Maintenance the database
- Searching ,data retrieval and the use of a database.

The DBMS accepts requests for data from an application program and instructs the operating system to transfer the appropriate data as shown in figure (1.2)

ADVANTAGES OF A DBMS

1- Database Development: It allows organizations to place control of database development in the hands of database administrators (DBAs) and other specialists.

2-Data independence: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.

3-Efficient data access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. It allows different user application programs to easily access the same database. Instead of having to write computer programs to extract information, user can ask simple questions in a query language.

4-Data integrity and security: If data is always accessed through the DBMS, the DBMS can enforce :

- integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded.

- Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.

5-Crash recovery: the DBMS protects users from the effects of system failures.

6- Data administration and Concurrent access: When several users share the data(more than one user access the database at the same time), DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time.

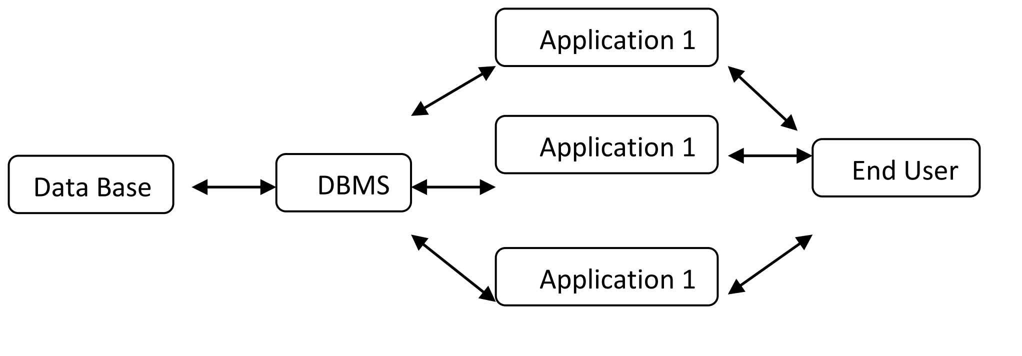


Figure (2) DBMS

5- Data Abstraction

The major purpose of a database system is to provide users with an abstract view of the data. That is , the system hides certain details of how the data are stored and maintained.

For the system to be usable, it must retrieve data efficiently. This had led to the design of complex data structure to represent the data in the data base. Since many database-user are not computer trained , data base developers hide the complexity from users through several levels of abstraction as shown in figure (1.3).

**** Physical level** : the lowest level of abstraction, describe ***how*** the data are actually stored. At the Physical level , complex low level data structure are described in detail. In this level storage locations used to specify where the data are and the data been described by words and bytes.

**** Logical level** : This level describe ***what*** data are stored in the database and what relationship exist among those data. The entire database is describe in terms of small number of relatively simple structure. The logical level is used by the database administrator, who must decide what information is to be kept in the database.

**** View level** : The highest level of abstraction describes only part of the entire database. Many users of the database system will not be concern with all the information. Instead the user need to access only part of the database , so a view level of abstraction is defined .

The system may provide many views for the same database.

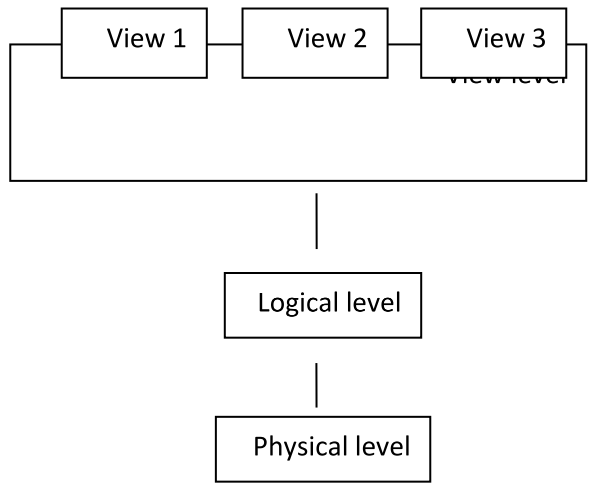


Figure (3) The three level of abstraction

6- The Relational model

The Relational Model is a clean and simple model that uses the concept of a relation using a table rather than a graph or shapes. The information is put into a grid like structure that consists of columns running up and down and rows that run from left to right, this is where information can be categorized and sorted.

The relational model used the basic concept of a relation or table. The columns or fields in the table identify the attributes such as name, age, and so. A tuple or row contains all the data of a single instance of the table such as a person named Doug. In the relational model, every tuple must have a unique identification or key based on the data as shown in figure 2.3 , a social security account number (SSAN) is the key that uniquely identifies each tuple in the relation. Often, keys are used to join data from two or more relations based on matching identification.

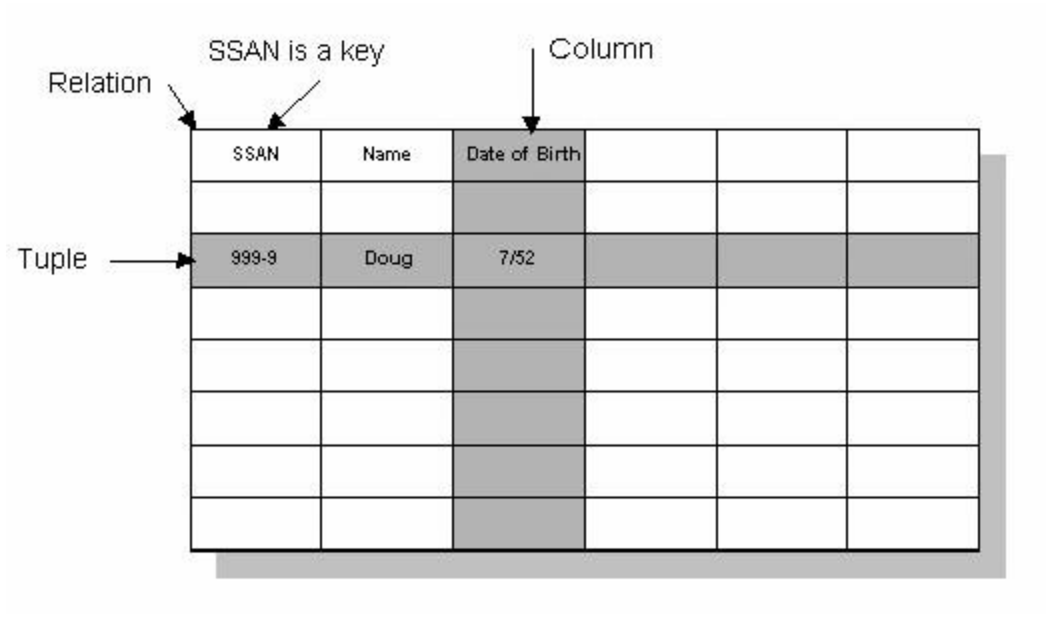


Figure (4) Relation (Table)