Instructor Notes:

Add instructor notes here.

Instructor Notes:

None

Lesson Objectives

To Understand the following concepts

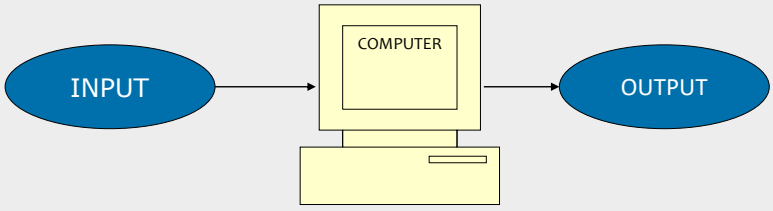
- Introduction to programs
- Types of projects
- SDLC process of waterfall model
- Introduction to Pseudocode
- Usage of variables and operators
- Introduction to control constructs

Instructor Notes:

1.1 Introduction to Programs

What is a Program

A program is a set of instructions for a computer to perform a specific task.
Programs can be written in one or more programming language
Computers accept input, process it and generate output.



What is a Program?

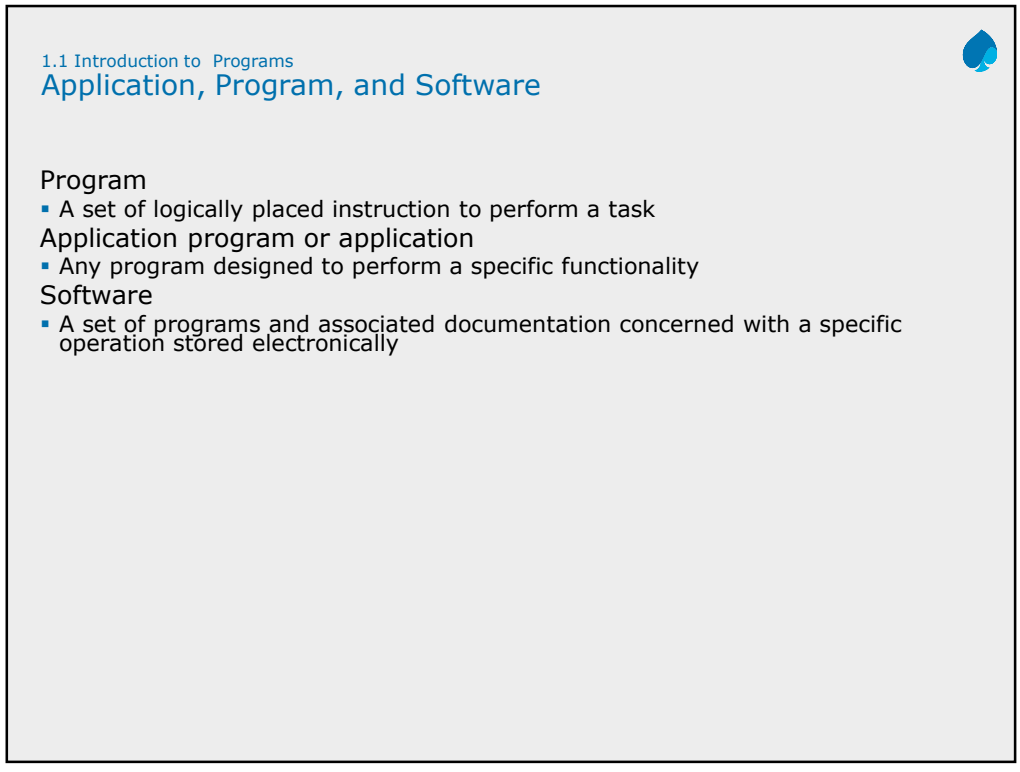
Set of instructions for a computer to perform a specific task.
Programs can be written in one or more programming language.

Example: Program to receive employee details as an input, then calculate and display the gross and net salary of an employee.

What is Programming Language?

- Used to feed instructions to the computer
- Can be categorized as Machine language, Assembly language, Compiled Languages, Interpreted Languages, Object Oriented Languages … etc
- Languages which are more simpler, easier are referred as High-Level Languages
- Low level languages provides little or no abstraction to the internal working of microprocessor

Instructor Notes:



Program:

Set of ordered instructions that enable a computer to carry out a specific task. A collection of instructions that tell the computer what to do.

Ex: Program to find a prime number, Program to print employee pay slip etc..

Application :

Any program designed to perform a specific function

Ex : Notepad, M S Paint etc

Software:

A set of programs and associated documentation concerned with a specific operation stored electronically

Ex : Microsoft Office, Oracle etc

Instructor Notes:

Discuss the comparison of Industry versus college programs

1.1 Industry versus College programs

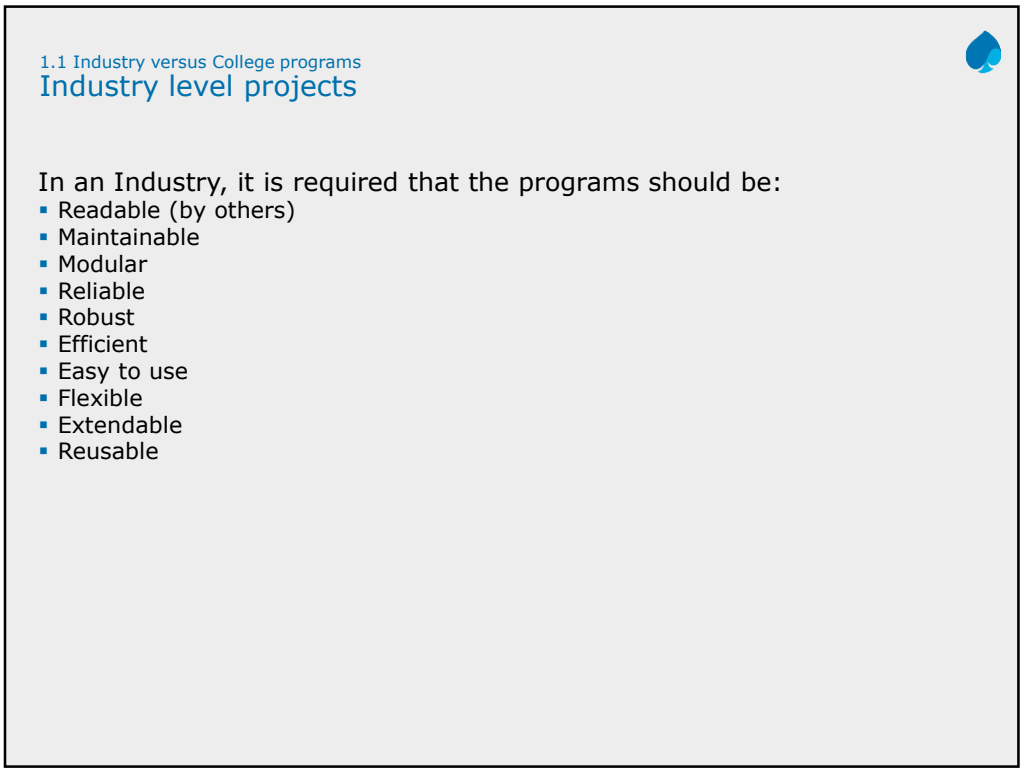
Industry level projects

Consider the scenario of an Industry:

- Programs have a long life (5 - 10 years!)
- Large applications: 10 - 500 person teams
- Entities: Users, Customer, Developers - Analyst, Designer, Programmer, Tester
- Varied application domains
- Mission critical applications
- Commercial gains and penalties
- Distributed architecture

- Consider the scenario of a College:
  - Throw away programs
    - These programs are not used by any one later
  - Same Users / Developers
  - Small assignments: 1-2 person teams
  - No commercial angle
  - Familiar application domain
  - Low criticality
  - Traditional single-machine architecture

Instructor Notes:



Industry level programs should be

- Readable : Easy to read and understand the code at any time since lifetime of projects will be longer in terms of years.
- Maintainable : If the program is easy to understand and If it is easy to modify then the program is called as maintainable
- Modular : A small unit of code for a single purpose
- Reliable : Reliability describes about the ability of a system will work perfectly as stated without failure or error
- Robust : The ability of a system to continue operating despite abnormalities in input, calculations, etc.
- Efficient : A task which gets done in the specified time with desired quality
- Easy to use: Program which is easy to use by the end users
- Flexible
- Extendable : If the additional features of a program is possible to be included without any side effects, then the program is called as extendable.
- Reusable : If the task written in program is called multiple times, then the program is reusable.

Instructor Notes:

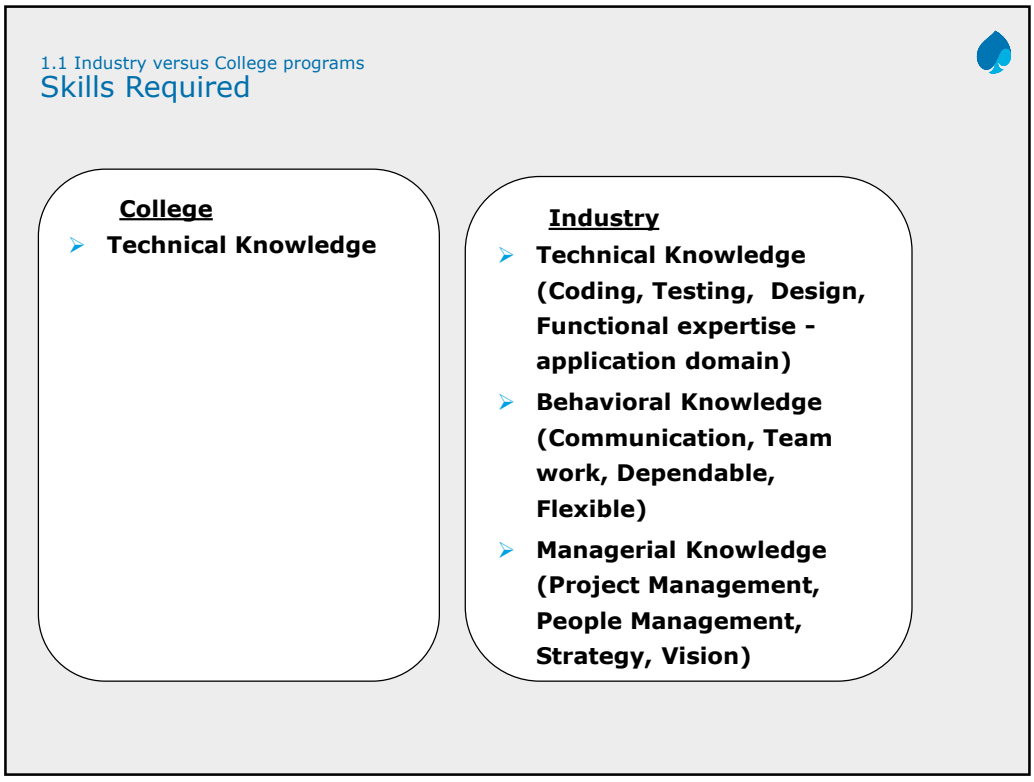
1.1 Industry versus College programs

Industry level projects

In an Industry, "coding standards" have to be maintained.
Coding Standards help to read others code, and maintain consistency. They entail standards regarding:

- naming conventions
- indentation
- commenting standards
- use of global variables
- modularity

Instructor Notes:



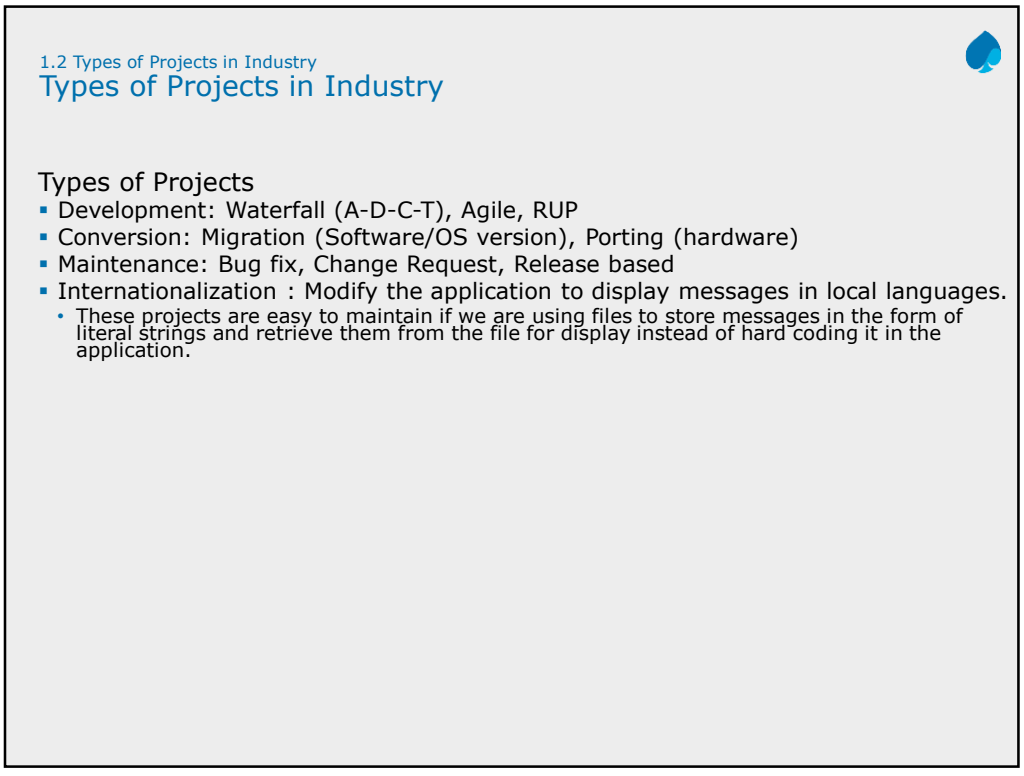
A person require much more technical knowledge for writing industry level applications as compared to writing programs in the college.

In college having good technical knowledge for a particular technology is enough to write program but in industry along with technical knowledge you require some other knowledge like

- What are good programming practices?
- How to write test cases?
- You also should have domain knowledge.
- As we need to work in a team in industry, you also need to improve your communication knowledge. You should be dependable and flexible.
- To reach at managerial level, you need to acquire people management skill, Project management skill over a period of time along with the technical skills

.

Instructor Notes:

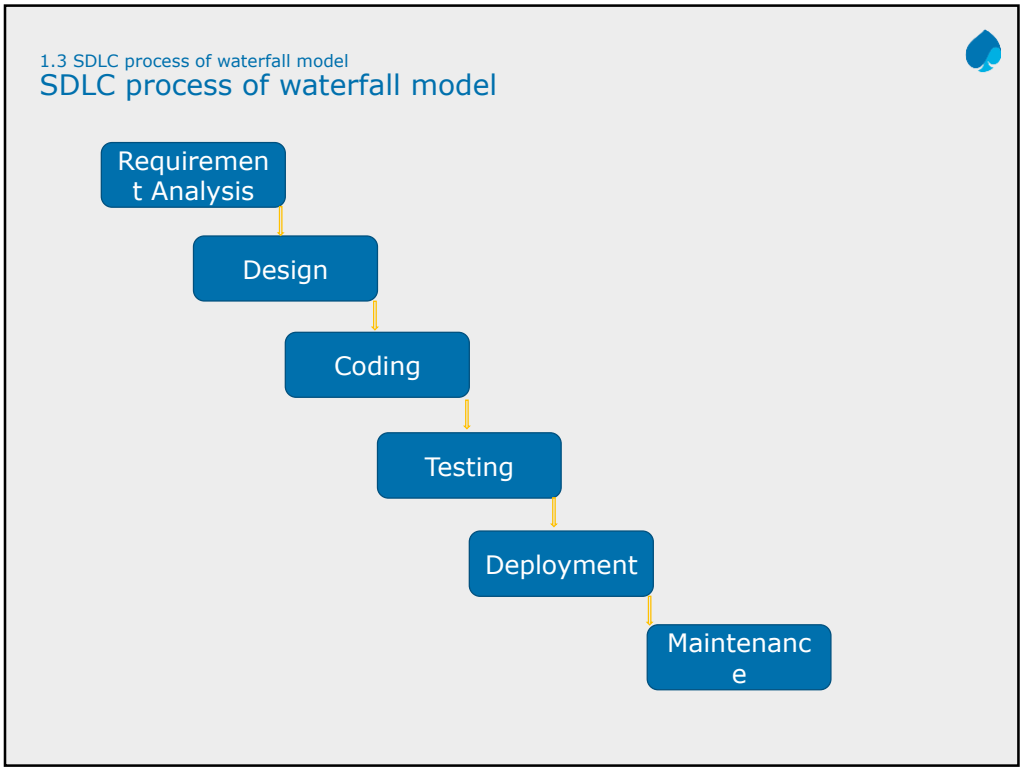


In an industry there are various types of projects like

- **Development Projects:** In these projects we may use different types of life cycles like Waterfall model, Agile, RUP (Rational Unified Process) you need to be familiar with these terms

- **Conversion Projects**
  - Migration - It refers to projects like upgrading a software to different version of the OS or DBMS systems
  - Porting – If there is major change in the system like, Hardware Platform has changed.
- **Maintenance projects**
  - Bug fix – If there is an unwanted behavior exists in an existing system due to bug(Problem), then fix the bug by doing changes.
  - Change request – If there are any changes in the functionality are requested from customer such projects are called as changed request type of project.
  - Release base – Products for which periodically the new release of the product is made
- **Internationalization :** Normally most of the applications display messages to user in English but in some cases we may need to change the application to display messages in local language based on the location where we are using it. In such type of projects we need to change the code accordingly. If we have hard coded the messages in the applications then maintenance will be the tedious activity. Hence the good solution for it is store the messages in a file in the form of literal strings and use these files for displaying messages.

Instructor Notes:

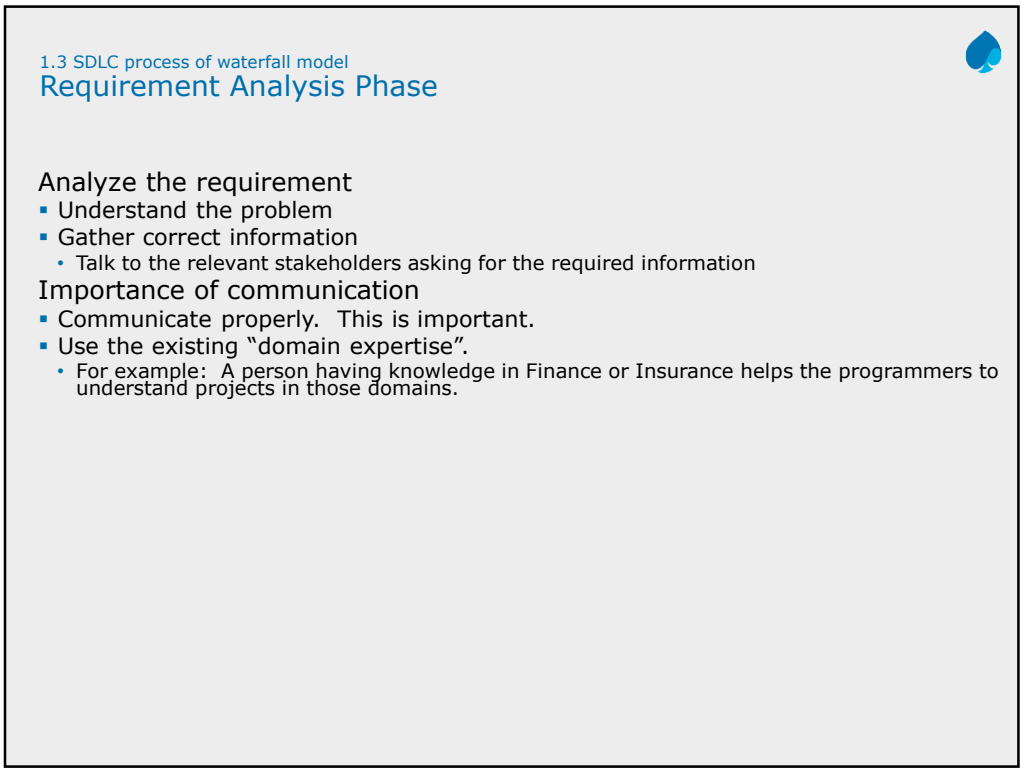


Waterfall Model: The waterfall model is a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, coding, testing (validation), deployment, and maintenance.

SDLC process of Waterfall Model:

- **Requirement Analysis :** Identify all the requirements. After requirement gathering, analyze the requirements for identifying their validity and the possibility of incorporating the requirements in the system to be developed.
- **Design:** It is a process of creating a detailed specification for a software module . It involves algorithmic design and other implementation specific approaches for a s/w component such as modularity , control hierarchy,, data structures etc. Designers/Technical leads ,senior developers , architects are involved in this phase
- **Coding :** Main objective of this phase is to translate the software design into code , each component identified in design is implemented as a program module following coding guidelines
- **Testing :** Process of checking what's been developed against the requirement.
- **Deployment :** Process of bringing the system into production environment
- **Maintenance :** The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness.

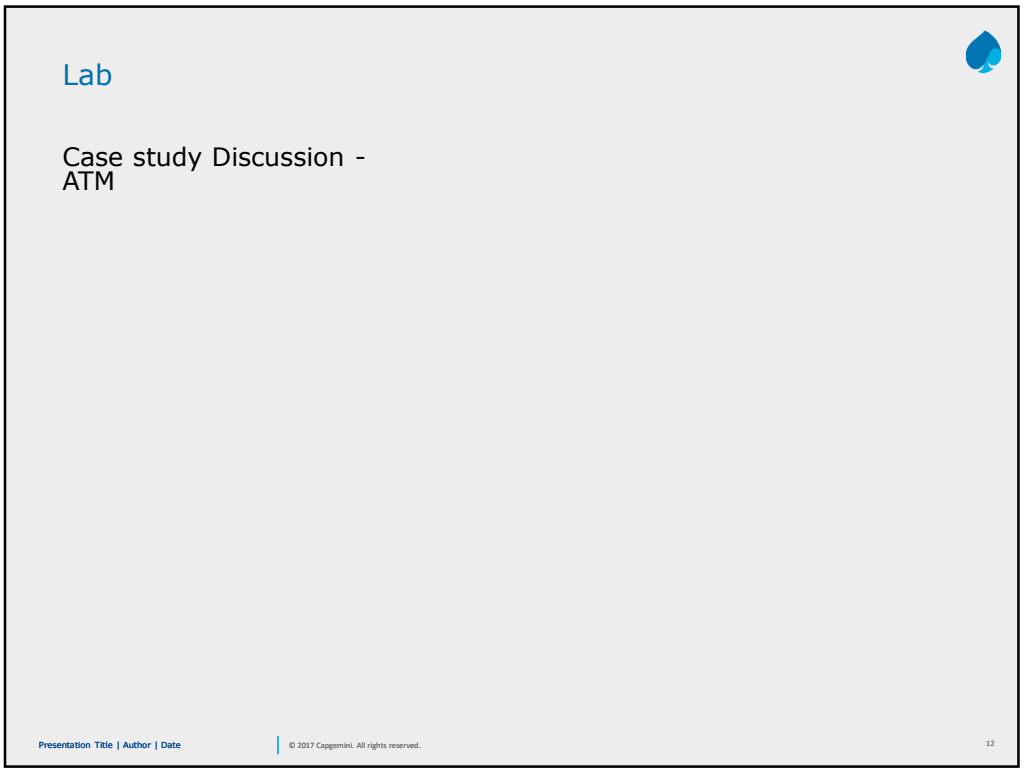
Instructor Notes:



Analyze the requirement:

- **Understand the Problem:**
    - Interpret the problem in your own words
    - Determine the outputs required
    - Identify the inputs required to obtain the desired output
    - List out the clarifications required
    - List the assumptions made
    - List the constraints / limitations
- **Gather Correct Information:** Gather the required information from the concern stakeholders by communicating with them.
- **Importance of proper communication:** User talks in language of business, and Programmer talks in the language of technology. Since there is a big gap between these languages, understanding the requirements properly is very important. This is possible through proper communication. The communication exercise is as follows:
    - Consider income tax calculation logic is written by you. If you want to check whether the Income Tax calculation written in the program, is correct as per the current Tax laws? Where can you get this information?
    - Draft an email to the relevant stakeholders, requesting for the required information.

Instructor Notes:



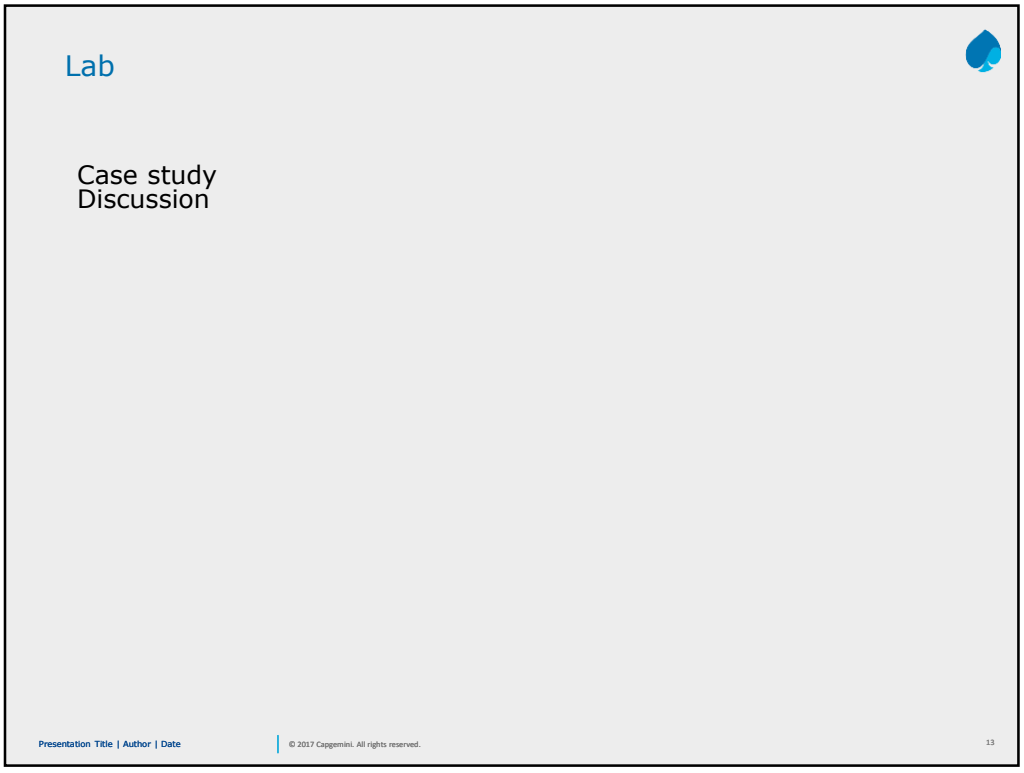
Case Study: The ATM application aims at performing ATM transactions and balance enquiry of an existing account holder in a user friendly way

Following is a list of functionalities of the system. Wherever, the description of functionality is not adequate; you can make appropriate assumptions and proceed.

- The user is requested for the card number and his personal pin number for authentication purpose.
- After authenticating the user, the application requests the user to choose any one of the following options:
    - BALANCE ENQUIRY
    - CASH WITHDRAWL
    - MINI STATEMENT
    - QUIT.
- When the user chooses one of the above options, say ‘1’, the balance of the user is retrieved and displayed. The application further requests the user whether he/she wants the report to be generated and responds accordingly.
- When the user chooses ‘2’, transaction is performed based on the request of the user with the help of the transaction file. Thus after the transaction is complete the user’s account is updated.

Instructor Notes:

Ask participants to form a team of 4 to 5 members and ask them to discuss among themselves to understand the different constructs to be used



Case Study – ATM (Contd..)

- When user wants to generate a record for his/her last 5 DAYS transaction, mini statement is opted where details retrieved from Transaction History File.

Updating balance is done when recharge is successful in Master and Transaction File. Thus, the application requests the user for further processing and responds based on the input from the user.

Case Study 1: In a firm there are 10 salespeople and incentive is paid on the portion of sales that exceeds two thirds of the average sales of the group. List the salesperson receiving incentives along with their incentive amount. Incentive amount is 20% the amount of sales that exceeds the two thirds of the average.

Case Study 2: All candidates have to take three tests. Selection for the interview round is done based on the scores of all the three tests. The individual score in each test has to be greater that 75 and the average score across the three tests should be a minimum of 80. An interview call letter is to be sent to candidates who have been selected and a reject letter is to be sent to the others. The interview stage involves two rounds:

- *Round 1:* qualifying criterion: rating of greater than five on a scale of 1 to 10
- *Round 2:* qualifying criterion: rating of greater than seven on a scale of 1 to 10Candidates go through both rounds of interview.

Selected candidates are sent offer letters and the rest are sent reject letters.

Represent the logic for finding the number of candidates who have been sent interview call letters, who have been sent reject letters and who have been sent offer letters.

Instructor Notes:

Explain about flowchart and Algorithm. Pseudocode need to be discussed later.

1.3 SDLC process of waterfall model

Design Phase

Design includes translation of the requirements into a logical structure that can be implemented in a programming language.

The programmer designs the application flow/program using design tools such as

Flowchart

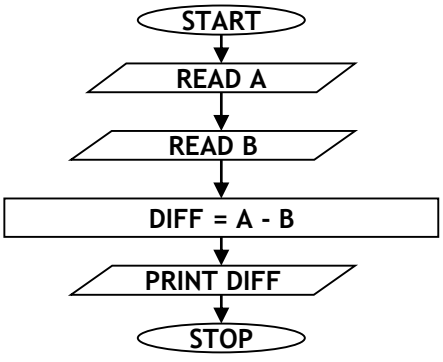
- A flow chart, or flow diagram, is a graphical representation of a process or system that details the sequencing of steps required to create output.

Algorithms

- An algorithm is a set of instructions for solving a problem. It is a basic technique of how to do a specific task.

Pseudo code

Flowchart: A flow chart, or flow diagram, is a graphical representation of a process or system that details the sequencing of steps required to create output. A typical flow chart uses a set of basic symbols to represent various functions, and shows the sequence and *interconnections* of functions with lines and arrows. Ex: Flow chart to calculate difference b/w two numbers:



Algorithm: An algorithm is a set of instructions for solving a problem. It is a basic technique of how to do a specific task. It takes input, processes it according to a set of instructions, and generates output. An algorithm must provide correct output for every possible input condition. An algorithm must have a definite end point so that when the input has been processed and the desired output achieved, the process stops.

Example: Algorithm to calculate the difference between two numbers

```
Accept two numbers as num1 and num2
Find the difference between num1 and num2
PRINT DIFFERENCE
```

Instructor Notes:

•Explain What do you mean by test case
•Give one example
•Inform participants that testing will be seen later in detail

1.3 SDLC process of waterfall model

Design Phase

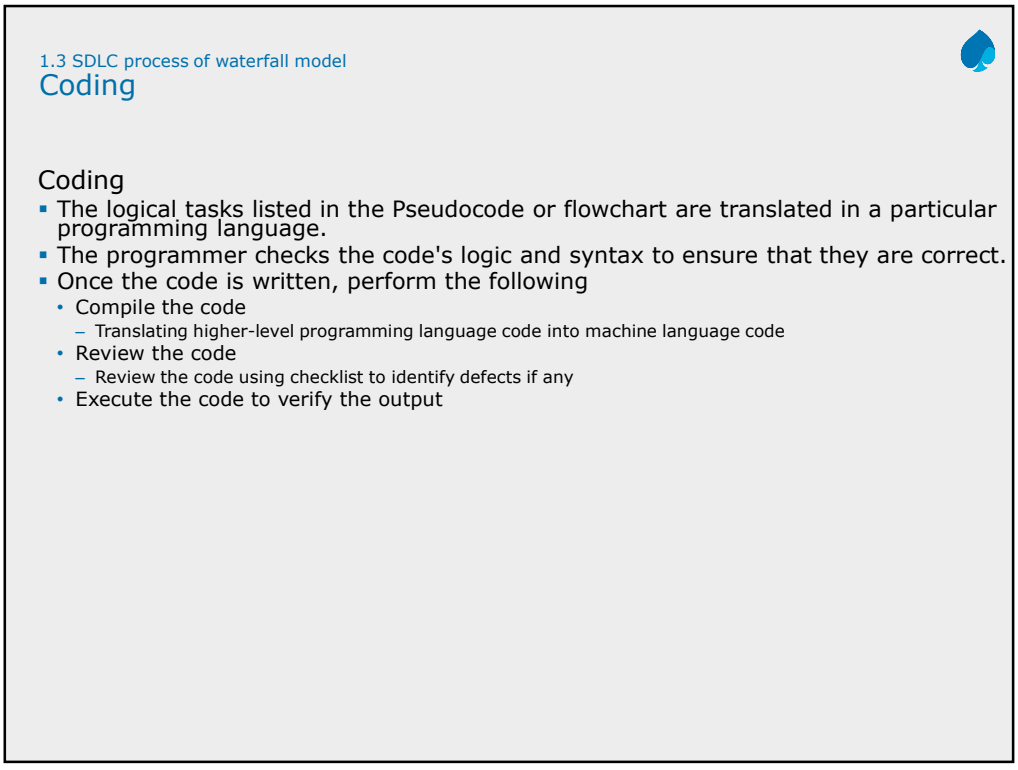
Designing the test cases

- Documenting the test cases is very important, as well.
- Identify and document the Test cases that will adequately test the program.
- Different formats are available to document Test cases.
- Normally an excel sheet is used for documenting Test cases.
- One of the formats for documenting Test cases is the "3 column format". The column headings are:
  - Test Case No
  - Test Case Description
  - Expected Result

Test case
"A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement."
See the sample test cases given below for checking valid date.

Test Case ID	Test Condition / Scenario	Test Steps	Input/Test Data	Expected Result
TC_1	To check the valid Date	Enter date	10/04/2005	Date should be accepted.
TC_2	To Check the Date when days are entered as Alphabets	Enter day as alphabets. Enter valid month & year	Ten/04/2005	"Enter Valid date"
TC_3	To Check the Date when days are entered as Special Characters	Enter day as special char. Enter valid month & year	!)/04/2005	"Enter valid date"

Instructor Notes:



Compiling the code:

- Computers can execute machine language code only.
- The process of translating higher-level programming language code into machine language code is called compiling the program.
  Special programs, called compilers and interpreters, perform this task.
- Code that has errors in it cannot be compiled. The compiler will flag the error, and the programmer must fix the error before trying to compile the code again

Execute the program and verify the output:

- Once a program is compiled, it can be executed.
- The programmer checks the program's output to ensure that it is correct
- Logic of the program can be checked by using sample inputs and comparing the output obtained with the expected output identified during creation of test cases.

Instructor Notes:

•Explain importance of self review and use of check list.

1.3 SDLC process of waterfall model

Review

To identify errors , either in the code or in other artifacts, self review is very important

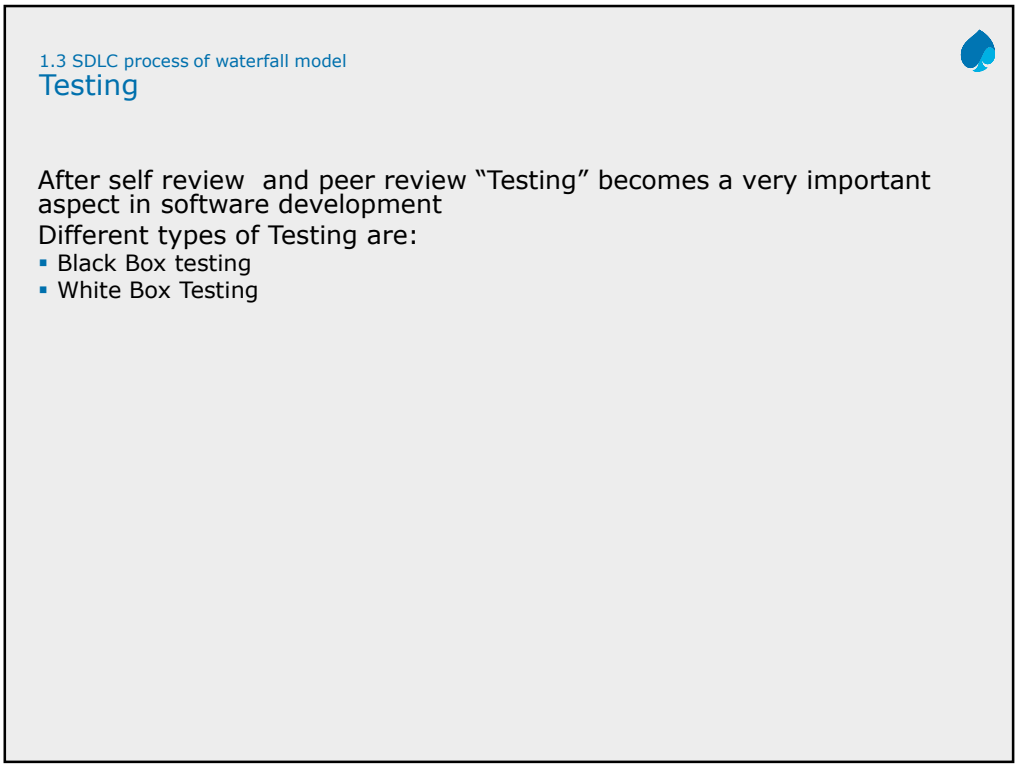
Self review

- Process of reading code line by line to check for:
  - Flaws or potential flaws
  - Consistency with the overall program design
  - The quality of comments
  - Adherence to coding standards
- Is done by the developer with the help of checklist
- If the evaluation is done by other people in the team ,it's called as peer review

Self review is a process done by the author himself with the aid of tools like checklist, review guidelines, etc..

Checklist is a tool with which review can be performed. Checklist is available as an excel document used to verify that all the requirements mentioned in requirement specification is covered and best practices, coding standards are followed.

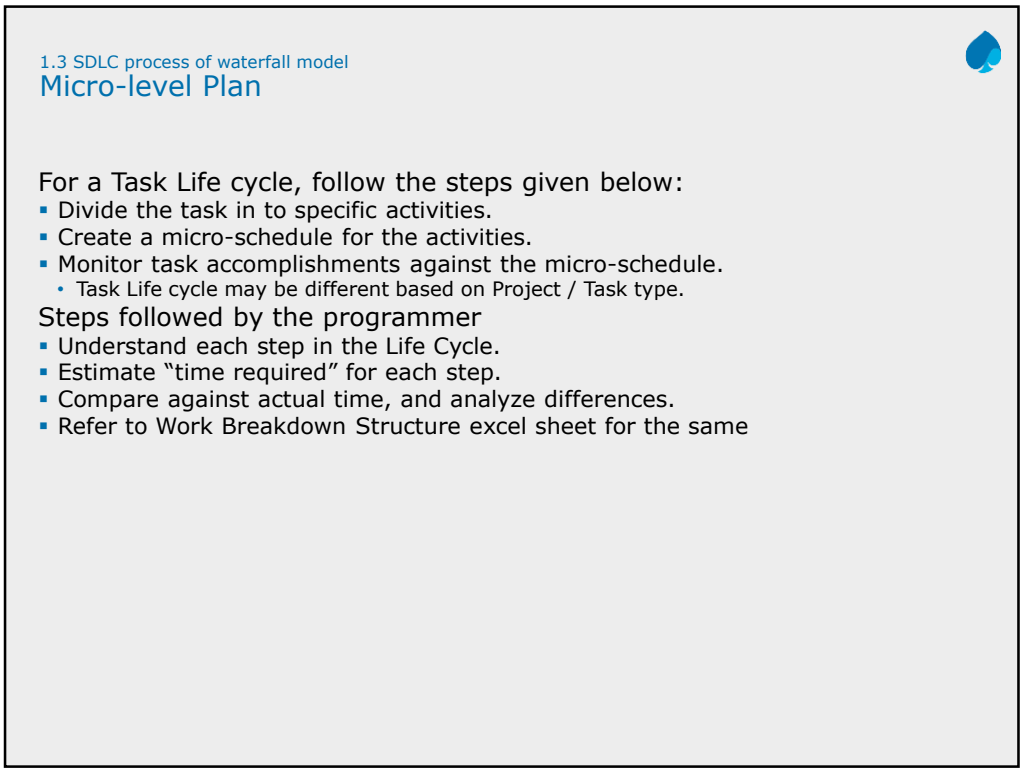
Instructor Notes:



Different types of testing:

- **Black Box testing** is a software testing method focused on testing whether the input is properly accepted, and output is correctly produced in an application. For an Example, if you want to validate the given input, then use black box testing.
- **White Box testing** is also a software testing method focused on testing the internal structure of the code. For an example, any unreachable path is identifiable using white box testing.

Instructor Notes:



If any of the task is assigned to a programmer then to complete it in time follow the task life cycle.

1. The task life cycle means, divide the task in to smaller activities which should be performed in the sequence to complete the task.
2. Decide the time lines for every activity and prepare a micro schedule for the activity
3. Monitor the accomplishment for every activity against the schedule prepared in the previous step. So that we can take care of timely completion of the task. It will also help to improve our ability to estimate the timelines to complete the task.

Task life cycle will be different for different project or different task type

And hence programmer should first understand each step in the life cycle. Estimate the time required for each step and monitor the task completion against the plan.

Instructor Notes:

1.3 SDLC process of waterfall model

Deliverables

Note that following are the primary deliverables for development tasks:

- Test cases (black box and white box)
- Documented source code
- Code review and Testing results
- Timesheet data (Effort), and Defect data (logging and closure)
- Check-in in the Source Code Control system
- Code Integration Test results (Pass)
- Task closure in PMS

We need to submit above mentioned deliverables.
In defect data we need to log the defect and also mention the closure of it.
Submit the code to configuration management (Check-in in the Source Code Control system). Once the integration test is pass then close the task in PMS (Project Management system). i.e we are ready to work on another task.

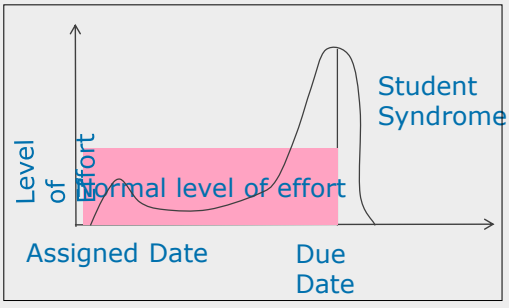
Instructor Notes:

1.3 SDLC process of waterfall model

Student Syndrome

Question: If you have 16 days to execute a 10 day project, when do you start?

- Immediately! or
- After 6 days, or
- After 10 days (since you know you are faster than an average developer, and can probably do it in 6 days!)



Student Syndrome:

- In the above graph. If you observe people are working with less efforts than expected in the beginning and near the dead line people are slogging to complete the work in time.
- This is similar to how a student prepare for the exams for day and night just few days before the exam.
- Avoid student syndromes. While working in project.
- Parkinson's Law:
  - "Work expands to fill (and often exceed) the available time."
- Murphy's Law:
  - "If anything can go wrong, it will!"

Instructor Notes:

1.4 Introduction to Pseudocode

What is a Pseudocode?

A pseudocode is an algorithm expressed in a natural language rather than in a programming language.
It is written in the design phase.
It is an outline of a computer program, written in a format that can easily be converted into programming statements.

What is Pseudocode?

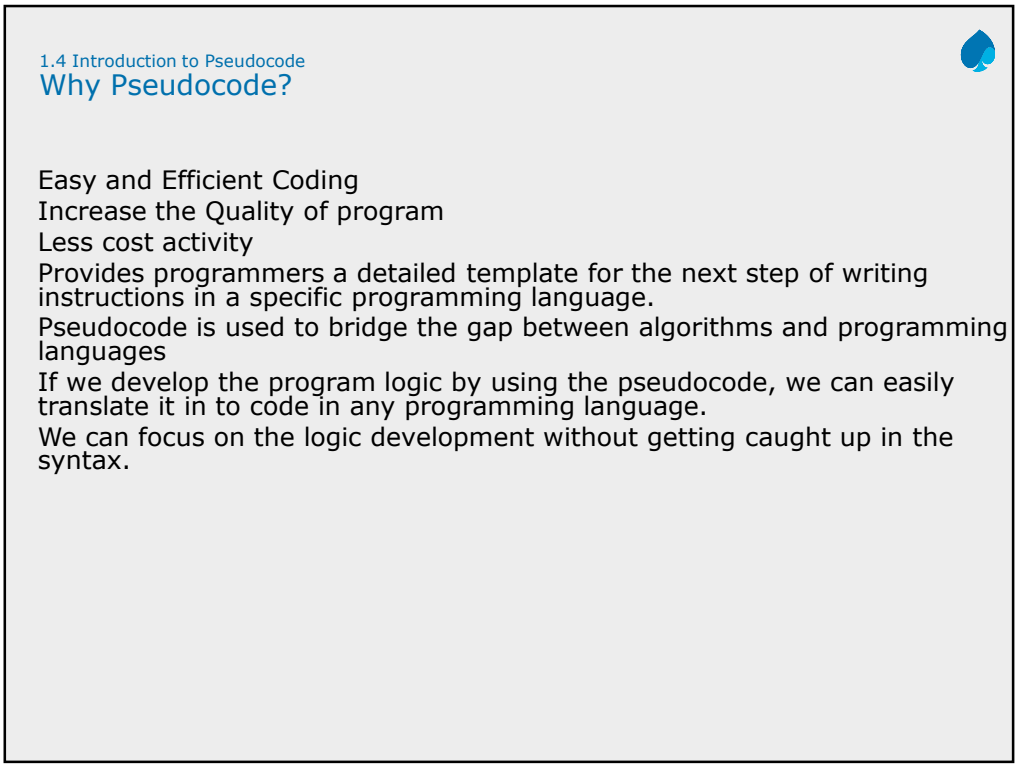
- Pseudocode is a natural language description of what a computer program must do rather than in a programming language.
- It allows the developer to focus on the logic of the code without being distracted by details of programming language syntax.
- At the same time, the pseudocode needs to be complete. It describe the entire logic of the task so that implementation becomes easier for translating the task line by line into source code.
- The pseudocode describes the logic of the program and acts as a blueprint for the source code to be written by the programmer.

PseudoCode for Finding Whether a number is odd or even:

```
BEGIN
        PROMPT "Enter the number" AND STORE IN num
        IF (num MOD 2 == 0) THEN
                DISPLAY "Even Number"
        ELSE
                DISPLAY "Odd Number"
        ENDIF
END
```

PROMPT is a Pseudocode keyword used to take inputs from the keyboard and store in a variable

Instructor Notes:



Why Pseudocode ?

- Easy and Efficient Coding – Easy to solve the task by focusing only on logic of the code with pseudo code rather than any other programming language.
- Increase the Quality of program – Easy way for Analyst to ensure the code matches with design specifications. Once it is matched, then they can easily convert the pseudocode into project specific Language. Thus it helps to ensure requirements are met and that program code meets good software development practice.
- Less Cost activity. Since Catching Logical errors is less tedious than catching them in development process.

Instructor Notes:

1.4 Introduction to Pseudocode

How to write Pseudocode?

All statements are written as sentence.
No variable declarations.
Use unique variable names but there is no need to declare them before they are used.
There is no universal "standard" Code for writing Pseudo Code.

Rules to be followed While writing pseudo Code:

- All statements are written as sentence. Use Words and Phrases which are in line with basic computer operations.
- No variable declarations. Use unique variable names but there is no need to declare them before they are used.
- There is no universal "standard" Code for writing Pseudo Code. But for understandable by others in the project follow the common coding standards specific to the project.
- Some of the Common Coding Notations are

Pseudocode Keyword	**Function / Operation**
DISPLAY/PRINT	Output to screen
PROMPT/ACCEPT	Display a prompt and store into a variable
EQUALS or =	Assignment operation
READ	Read from data source (File)
WRITE	Write to data source (File)
INITIALIZE/SET	Give data an initial value

Instructor Notes:

1.4 Introduction to pseudocode

Best practices of writing pseudocode

There is no absolute standard for pseudocode, these best practices can be followed:

- Use simple English
- Write each instruction on a separate line
- Declare variables in the format of "DECLARE variablename as basictype", if required
- Use "initialize" keyword to initialize value to a variable.
- Capitalize keywords
- Follow indentation strictly
- Group instructions into modules.
- Always use terminators for loops and iteration like ENDLOOP, ENDIF
- Provide only one entry and one exit point in a Pseudocode using BEGIN and END keyword.
- Every program and module should have a header preceding it.
- Module and variable names should be meaningful.
- Follow all the programming best practices like readable, maintainable, etc.

Pseudocode Example with best practices:

```
BEGIN
        DECLARE num AS INTEGER
         INITIALIZE num TO 0
         PROMPT "Enter the number" AND STORE IN num
         IF(num > 0) THEN
                  DISPLAY "Positive Number"
         ELSE
                  DISPLAY "NegativeNumber"
         ENDIF
END
```

Instructor Notes:

1.4 Introduction to pseudocode

Example of pseudocode

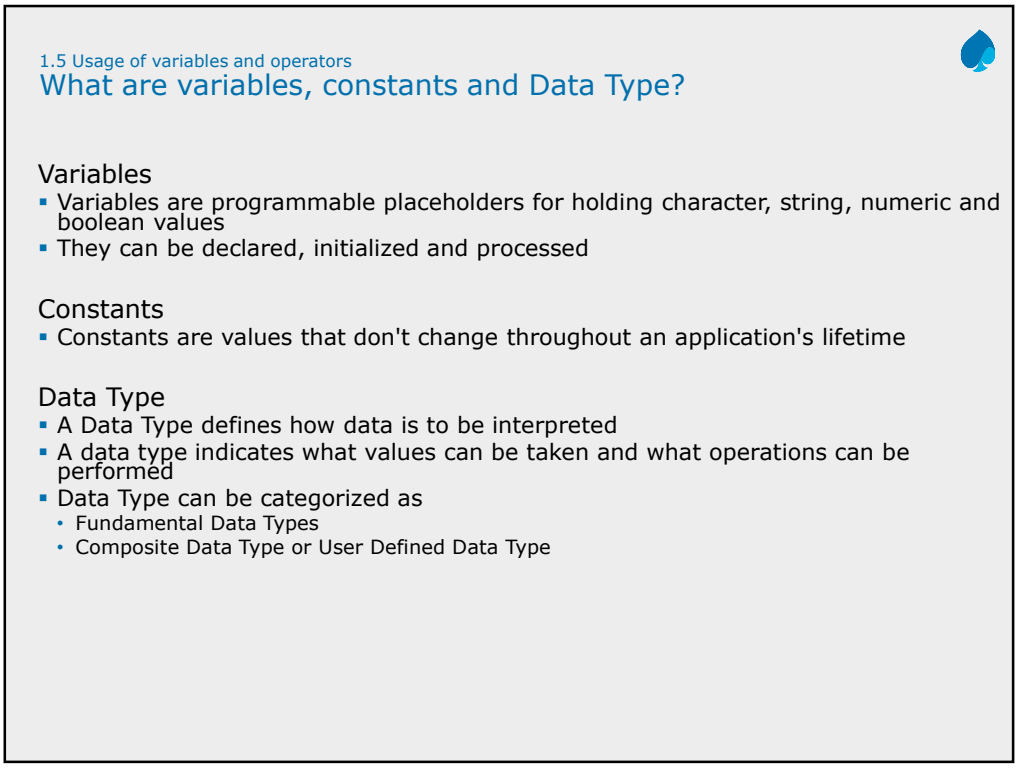
```
BEGIN
        DECLARE CONSTANT interest_rate =0.5
        INITIALIZE Amount = 0
        INITIALIZE Interest = 0
        INITIALIZE Ctr=0
        WHILE Ctr  <10
        DO
                PRINT "Enter amount to find interest"
                ACCEPT Amount
                CALCULATE Interest = Amount *
    interest_rate
                DISPLAY Interest
                Ctr=Ctr+1
        END WHILE
END
```

The above pseudocode is used to calculate interest for a given amount based on fixed interest rate. This process will be repeatedly executed for 10 times.

Fixed Interest Rate is 0.5.

Formulae used to calculate interest is Amount * Fixed Interest Rate.

Instructor Notes:



Variable:

Variable's value can be changed at any point in a program.
Each new value must be of the initial type.
Variable describes data that may change while the program is running

Constants:
Constant describes data that is set before the program is used and will remain the same while the program is running
For example, the mathematical symbols pi and e have well-defined values, which are invariant

Instructor Notes:

1.5 Usage of variables and Operators

Fundamental Data Types

Fundamental data types are the data type provided as basic building blocks They are also known as primitives or basic data type, following

Data Type	Description
Character	A character type (typically called "char") may contain a single letter, digit, punctuation mark, or control character. Some languages have two or more character types, for example a single-byte type for ASCII characters and a multi-byte type for Unicode characters
Integer	An *integer* data type can hold a whole number
Real	A *real* type stores rational number having fractional part
Boolean	A *boolean* type, typically denoted "bool" or "boolean", is a single-bit type that can be either "true" or "false".

Character
The char data type represents single characters, such as 'm'.
A variable made up of a number of characters is called a string.
Integers
Integers are positive and negative whole numbers
There are different sizes of integers available, including
Short integers and
Long integers
Real
Real data are numbers with a fractional part, for example 8.65 and -0.03
They also include numbers that have no fractional part but are expressed as a whole number with a decimal point, such as 4.0 or -1.0
Boolean
Boolean values are logical values and can be either true or false.

Pointer Data type is integer type
Named Constants are also integers

Instructor Notes:

•Record and arrays will be discussed in detail later

1.5 Usage of variables and Operators

Composite Data Types

A composite data type helps in grouping logically related data as one unit
The data types derived/created from the fundamental data types are called Composite or User Defined data types.
Example:

- Arrays
- String
- Records

```
RECORD Employee
        DECLARE Empno AS
INTEGER
        DECLARE Emp_name AS
STRING
        DECLARE Salary AS INTEGER
END RECORD
```

"Structured data" refers to data that's built up from other types. Use them to clarify relationships between related items
Example :

```
Name  =  InputName;
Address =  InputAdress;
Phone = InputPhone;
Title = InputTitle;
Department = InputDepartment
Bonus = InputBonus
```

```
Employee.Name  = InputName;
Employee.Address = InputAddress;
Employee.Phone = InputPhone;
Supervisor.Title = InputTitle;
Supervisor.Department  = InputDepartment;
Supervisor.Bonus = InputBonus;
```

Why to create user defined datatypes?

- To increase reliability
  One could specify the range of values that a variable of a User Defined data type can take.
- To make up for language weakness
  If a language does not support a type the user wants, it is possible to create it yourself.
  Example, a student in 'C

Instructor Notes:

1.5 Usage of variables and Operators

Statement ,Expression, and Operators

Statement: A statement is a unit of code which performs an operation.
Example: PRINT sum, name = "Rama" etc.
Operators and operands: Operators are special symbols that represent computations like addition and multiplication. The values the operator is applied to are called operands.
Expression: An expression is a combination of operators and operands that ascertains a value
Based on operation need to be performed, following operators can be used:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Ternary/Conditional Operators

Operators and operands, Expression
An operator is a symbol that represents a specific action that must be performed on data

Ex:In the expression *sum=num1+num2*
sum,num1 and num2 are the *operands* and + is the *operator*

Operators have rules of precedence and associativity that are used to determine how expressions are evaluated.
Expressions**:**
An Expression is a combination of constants and variables together with the operators.
Constants and variables by themselves are also considered as expressions. An expression that involves only constants is called a **Constant Expression**.
Note: Balanced parentheses can be used in combining constants and variables.
Operators have been classified into categories based on the operation that they perform. Refer the slide for the different categories.

Instructor Notes:

1.5 Usage of variables and Operators

Arithmetic Operators

Arithmetic Operators:

- Are used for arithmetic calculation such as addition, multiplication, subtraction and division. (Binary Operators)

	*	- Multiplication
Priority-High	/	- Division
	%	- Modulus Division (only for integers)
Priority-Low	+	- Addition
	-	- Subtraction

Types of Operators: Arithmetic Operators:

There are five types of arithmetic operators that are used for arithmetic calculations such as, addition, subtraction, multiplication and division.

These five operators are binary operators that is, they require two operands.

Each of these operators work with values of type integers, Real and character.

Instructor Notes:

1.5 Usage of variables and Operators

Relational Operators

Relational operators are used to compare two operands to check whether they are equal, unequal or one is greater than or less than the other

Relational Operator	Meaning	Relational Expression
<	Less than	expr1 < expr2
<=	Less than or equal to	expr1 <= expr2
>	Greater than	expr1 > expr2
>=	Greater than or equal to	expr1 >= expr2
==	Equal to	expr1 == expr2
!=	Not equal to	expr1 != expr2

Types of Operators: Relational Operators:

Relational Operators are used to compare two operands to check whether they are equal, unequal or one is less than or greater than the other.

There are six relational operators for comparing the values of two expressions and the expression so formed is called a Relational Expression.

Table provided in the slide shows relational operators and how they can be used to compare expressions expr1 and expr2.

Instructor Notes:

1.5 Usage of variables and Operators

Logical Operators

Logical Operators are:

- Used to combine two or more expressions to form a single expression
- Evaluated left to right, and evaluation stops as soon as the truth or the falsehood of the result is known

Operator	Name	Meaning
&&	Logical AND	Conjunction
||	Logical OR	Disjunction
!	Logical NOT	Negation

Types of Operators: Logical Operators:
There are three logical operators for combining expressions into logical expressions.
Table in the slide shows available logical operators

The logical operators **&&** and **||** are binary operators, and ! is a unary operator.
The value of a logical expression is either '1' or '0', depending upon the logical values of the operands. The operands may be of any arithmetic type. The result is always an integer.

Logical AND Operator(&&):
The && operator combines two expressions into a logical expression and has the following operator formation:

expr1 && expr2

An expression of this form is evaluated by first evaluating the left operand. If its value is 0 (false), then right operand is not evaluated and the resulting value is 0 (false).

If the value of left operand is nonzero (true), the right operand gets evaluated. The resulting value is 1 (true) if the right operand has nonzero value (true) and 0 (false) otherwise.

Instructor Notes:

1.5 Usage of variables and Operators

Ternary/Conditional Operator

Ternary Operators:

- Provide an alternate way to write the if conditional construct
- Take three arguments (Ternary operator)

Syntax:

```
expression1 ? expression2 : expression3
```

If expression1 is true (i.e. Value is non-zero), then the value returned would be expression2 otherwise the value returned would be expression3

```
BEGIN
        DECLARE number1, number2 AS INTEGER
        PROMPT "Enter number" AND STORE IN number1
        number2 = (number1>5 ? 3 : 4)
        PRINT number2
END
```

This statement will store 3 in 'number2' if 'number1' is greater than 5, otherwise it will store 4 in 'number2'.

Types of Operators: Ternary Operators:

Simple condition operations can be carried out using the conditional operator (? :). The conditional operator is a ternary operator that is, it takes three arguments. It has following operator formation:

expression1 ? expression2 : expression3

Where ? and : are the two symbols that denote this operator. A conditional expression is evaluated by first evaluating expression1. If the resulting value is true, then expression2 is evaluated and the value of the expression2 becomes the result of the conditional expression. Otherwise, expression3 is evaluated and its value becomes the result.

This is used to assign one of the two values to a variable depending upon some condition.

For example:

big = num1 > num2 ? num1 : num2 ;

assigns value of num1 to big if num1 is greater than num2, else assigns the value of num2 to big.

Instructor Notes:

1.6 Introduction to control constructs

Introduction to Control Constructs

There are basically three control constructs used to write algorithms:

- Sequence: The instructions are executed in the sequence in which they appear, and the program does not skip or repeat any of the instructions
- Selection: Selection implies that a choice will be made, which depends on the value of a condition specified by the programmer
- Repetition: Repetition repeats a section of code while a certain condition holds true.

Instructor Notes:

1.6 Introduction to control constructs

Control Constructs - Sequence

A computer program executing in sequence performs each instruction once only. The instructions are executed in the sequence in which they appear, and the program does not skip or repeat any of the instructions

```
BEGIN
        READ num1
        READ num2
        CALCULATE Difference = num1-num2
        PRINT Difference
END
```

Pseudocode given in the slide is used to find the difference between two numbers. In the above pseudocode, each instruction will be executed sequentially.

Instructor Notes:

1.6 Introduction to control constructs

Control Constructs - Selection

Selection implies that a choice will be made, which depends on the value of a condition specified by the programmer

Two forms of selection are there:

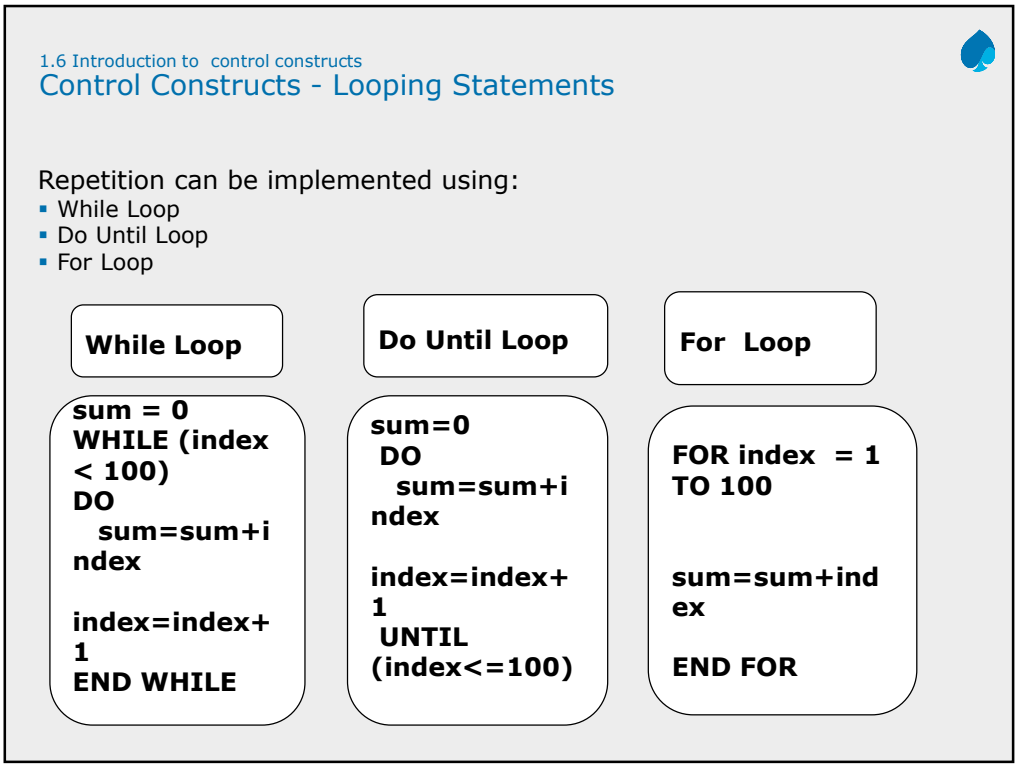
- **If…then**

```
IF num = 0 THEN
    PRINT " Number is
zero"
END IF
```

- **If…then…else**

```
IF num > 0 THEN
    PRINT " Number is
positive"
ELSE
    PRINT " NUMBER is
negative"
END IF
```

Instructor Notes:



Control Constructs – Looping Statements

Do Until Loop: Set of statements inside the block is executed and then the condition is checked. Executed till the condition is true. Block will be executed at least once irrespective of the condition

Example:

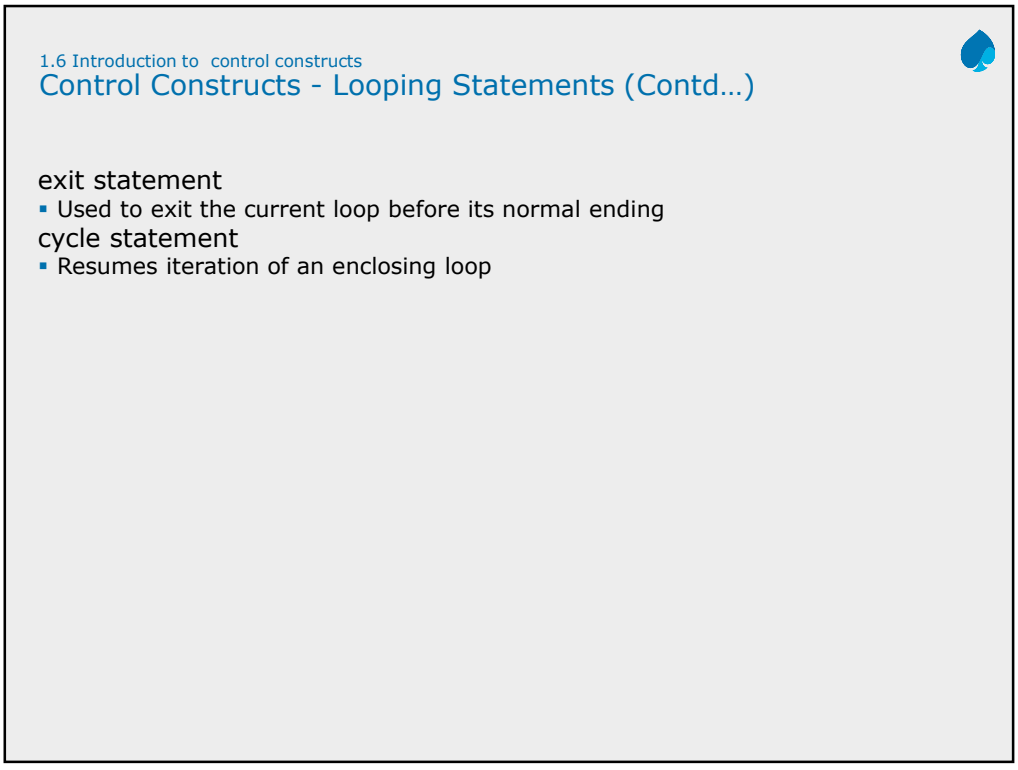
```
seats_allocated = 0
 DO
  GET booking
  PRINT ticket
  ADD 1 to seats_allocated
UNTIL (seats_allocated < 60)
```

For Loop:Consists of set of statements that are executed for a fixed number of iterations Must specify a starting value, ending value and incrementing value

Example:

```
FOR seats_allocated = 1 to 25
   Get booking
   PRINT ticket
END FOR
```

Instructor Notes:



Control Constructs – Looping Statements (contd..)

Example: Cycle and exit statement(check no is odd or even until user wants to stop)

```
BEGIN
   WHILE(TRUE)
      PRINT "Enter a NUMBER"
      ACCEPT num
      IF (REMAINDER of num/2 = 0) THEN
            PRINT "Number is EVEN"
      ELSE
            PRINT "Number is ODD“
      END IF
       PRINT "Enter u'r Choice , Continue with another number?[Y/N]"
       ACCEPT choice
        IF(choice='Y') THEN
                  cycle
        ELSE
                  exit
    END WHILE
END
```

In above example cycle statement will transfer the control to the beginning of the while loop. Exit statement will transfer the control out of while loop.

Instructor Notes:

Explain the scenario when particular conditional statement has to be used. Also Explain the example given in notes page.

1.6 Introduction to control constructs

Guidelines for Conditional Statements

When to use the if statement
- When only one condition is being checked

When to use if else statement
- When more conditions are being checked and the subsequent conditions are related to the first condition

When to use multiple if statements
- When more conditions are being checked and the subsequent conditions are not related to the first condition

In case of multiple or nested IF conditions, implement the most common conditions at the beginning.

Consider Pseudocode has to be written for accepting a number from user and need to identify whether the given number is falling under any of the below mentioned category
- zero
- negative
- Positive

Think which condition statement is more suitable for this?

Answer: As the given number has to be compared against minimum 2 conditions, IF-ELSEIF-ELSE is more suitable

Pseudocode:

```
BEGIN
        DECLARE num AS INTEGER
        PROMPT “Enter a number” AND STORE IN num
        IF ( num== 0 )THEN
                PRINT "The value you entered was zero.“
        ELSE IF ( num< 0 ) THEN
                PRINT  "The value is negative.“
        ELSE
                PRINT  "The value is positive.“
        END IF
END
```

I

Instructor Notes:

Explain the scenario when particular looping statement has to be used. Also Explain the example given in notes page.

1.6 Introduction to control constructs

Guidelines for Looping Statements

When to use a for loop
- When the iterative task is to be performed for <n> number of times

When to use a while loop
- When the question whether to continue the loop or not is asked at the beginning of the iterative task
- When to use a do while loop

When the question whether to continue the loop or not is asked at the end of the iterative task

Consider Pseudocode has to be written for finding sum of n numbers by accepting "n" value from the user. For an Example, if given "n" value is 5, then sum of value is 15(1+2+3+4+5).

Think which looping statement is more suitable for this scenario?

Answer: As the given number has to be used as upper limit for process to be executed repeatedly, use for loop to describe lower limit and upper limit.

Pseudocode:

```
BEGIN
        DECLARE num, count, sum AS INTEGER
        PROMPT "Enter the value of n" AND STORE IN num
        FOR COUNT = 1 TO num
                sum+=count;
        END FOR
        PRINT "Sum is " + sum
END
```

Instructor Notes:

Demo : Variables, Operators and Control Constructs

Refer the pseudo code available in the below listed files for understanding the usage of variables, operators and control constructs

- ArithmeticOperators
- TernaryOperators
- IF-ELSEIF-ELSE
- LogicalOperators
- DoUntil
- WHILE-CYCLE-EXIT
- ForLoop

Faculty will be sharing the below mentioned files. Refer the same to understand the usage of variables, operators and control constructs

- ArithmeticOperators.txt
- TernaryOperators.txt
- IF-ELSEIF-ELSE.txt
- LogicalOperators.txt
- DoUntil.txt
- WHILE-CYCLE-EXIT.txt
- ForLoop.txt

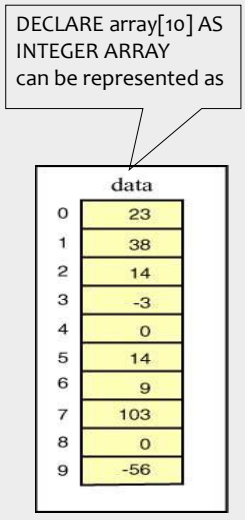
Instructor Notes:

1.7 Introduction to Arrays

Introduction to Arrays

Array

- It is an object that is used to store a list of values.
- It is made out of a contiguous block of memory that is divided into a number of "slots."
- Each slot holds a value, and all the values are of the same type, addressable by index or subscript, usually starting with 0
- are useful when a defined set of data has to be processed systematically
- make a program handle large amount of data without having to write unnecessary code



When to go far an Array?

- When there is a finite set of logically related information that needs similar processing
- For example calculating the grade of all students of a particular class
- All data's are similar and of one data type

Operations on an array:

The frequent operations which we perform on an array are

- Searching: searching the array for particular element
- Sorting : arranging data in particular order (ascending/descending)

Instructor Notes:

1.7 Introduction to Arrays

Example of Arrays

Find out the maximum number among 10 numbers

```
BEGIN
      DECLARE numbers[10] AS INTEGER ARRAY
      DECLARE max AS INTEGER
      INITIALIZE max TO 0
      FOR index=0 TO 9
          ACCEPT numbers[index]
      END FOR
      max=numbers[0]
      FOR index=0 TO 9
            IF numbers[index] > max THEN
                  max=numbers[index]
            END IF
      END FOR
      PRINT max
END
```

The above pseudocode logic is used to accept 10 numbers from user and find the largest number among 10 numbers. For storing the accepted 10 numbers of same type, array is used in this pseudocode.

Instructor Notes:

Lab

Basic program development with pseudocode Lab exercises - Lab 1

Write pseudocode for all the assignments mentioned in Lab 1 for practicing on the usage of variables, operators, control constructs and arrays.

Instructor Notes:

Summary

In this lesson, you have learnt about:

- Introduction to programs
- What is a program?
- What is an application?
- Industry Vs College Programs
- Types of projects
- SDLC process of waterfall model
- Introduction to Algorithm and Pseudocode
- Usage of variables, datatypes and constants
- Introduction to control constructs

Instructor Notes:

Question 1 : B
Question 2 : B

Review Question

Question 1: What are different types of testing techniques?

- A. Self testing
- B. Black box testing
- C. Red box testing
- D. None of the above

Question 2: A task which gets done in the specified time with desired quality defines ------

- A. Maintainable
- B. Efficient
- C. Robust
- D. Readable

Instructor Notes:

Question 3: A
Question 4: A,B

Review Question

Question 3: There is no absolute standard for pseudocode

- A. True
- B. False

Question 4: Identify guidelines for using variables

- A. Assigning values to variables just before values are used
- B. Prefer local variables over global ones
- C. Initialize variables used in looping constructs at the top of the code block
- D. Maximize the lifetime of local variables

Instructor Notes:

Question 5 : A,C

Review Question

Question 5: How does the following code breach best practice guidelines

```
myarray[10]= new Array{1,1,2,3,5,8,13,21,34,55};
index=0;
while(index<10)
index=index+1;
//dosomething
```

- A. The array should have a clearer name
- B. The index is never incremented
- C. The index Is zero based so the loop passes beyond the last element

Instructor Notes:

Question 6 :

1. C
2. A
3. B

Review Question

Question 6: Match the looping structure to their definitions

1. For executing	A. Condition controlled for choice once
2. IF executing times	B. Condition controlled for choice multiple
3. While	C. Count controlled

Question 7: You need to create a loop whose exit condition depends solely on the maximum number of employees being reached when you increment the number of employees by one. Which construct should you choose to ensure greatest clarity

- A. For
- B. DO UNTIL
- C. while