Introduction to Program Development:

Introduction to program:

Environment:

A wide picture of the need that has to be satisfied for which the program is being developed. It contains all scope of the need. It contains, Target, Decision Making Factors, Exceptions, Process, Dependencies

Pseudocode:

Set of instruction written in simple English

A clear flow of program without ambiguity

Problem Definition:

The thing must happen

The need to create a program to make things

Example: we need Canteen

Problem Description:

What are we going to create based on the need

Example:

The canteen needs to big enough to accommodate 100 people.

It should serve different varieties of food.

Problem Statement:

Defining Understanding a Problem

Identifing Input/Output of the Problem

Analysing Alternative ways to solve the Problem

Flowchart:

Figurative representation of the pseudocode.

Introduction to Pseudocode

Pseudocodes are used for effective understanding of the project.

It is written in simple English so its easy to understand by all types of people.

It is the flow of the project.

Do's and Don't in Pseudocode:

Do's :

Write comments. It should be in the following format

Start of the comments

detail explanation of the whole process

End of the comment

Don't use:

Keywords like JAVA, COMMIT, GoTo, Main etc

Short form like EN for Employee Name

Should not contain ambugity

Program Name:

It should be detailed and should contain the flow.

Contains the whole Process of the Project Like ( Welcome page...to end page), briefly written.

Don’t:

Should not use short forms for the Project

Sentence:

example : a+b

Statement:

Perform until the set conditions

Terminator:

Break the Statement

Delminators:

Using Colon, Semicolon( , :, :)

**GOOD PROGRAMMING PRACTICES**

**DOCUMENTATION:**

documentation is written text or illustration that accompanies computer software or is embedded in the source code. The documentation either explains how the software operates or how to use it, and may mean different things to people in different roles.

**COMMENTS:**

They are usually added with the purpose of making the source code easy to understand. Hence, add comments to your code in simple English language that describes the function of the code and the reason for your decision to do it in a particular way as well.

**INDENTATION:**

Leading white space (spaces and tabs) at the beginning of each statement, which is used to determine the group of statements, is known as “indentation‟.

**CHARACTERISTICS OF A GOOD PROGRAM**

A good programming language must be simple and easy to learn and use. It should provide a programmer with a clear, simple and unified set of concepts that can be grasped easily.

* A programming language must be simple, easy to learn and use, have good readability, and be human recognizable.
* Abstraction is a must-have Characteristics for a programming language in which the ability to define the complex structure and then its degree of usability comes.
* A portable programming language is always preferred.
* Programming language’s efficiency must be high so that it can be easily converted into a machine code and executed consumes little space in memory.
* A programming language should be well structured and documented so that it is suitable for application development.
* Necessary tools for the development, debugging, testing, maintenance of a program must be provided by a programming language.

#### **CHARACTERISTICS OF A GOOD PROGRAM**

**Readable**

* Readability in software programming can be defined by the ease with which the software is read and understood. Readability of software can be somewhat objective.
* Programmers who are “journeymen” and move from one project to another throughout their career tend to have an easier time reading a variety of software code.
* However, making software more readable helps in reviewing and maintaining it over the course of its life.
* Simplicity in logic, conditional statements, and the structure of the code all help with readability.

The following is an example of a proper “C” code segment, that isn’t entirely readable:

// Check for stuff to proceed

if((!((Engine\_Speed!=0)||(Vehicle Speed!=0))) || SecureTest!=FALSE ){

// ABC…

}

With a little better readability, the same conditional can be written as:

// Check for secure testing to be running, or if vehicle is stopped

// along with the engine not running. Then we can execute <ABC>

if (( Secure\_Test == TRUE ) || \

(( Vehicle\_Speed == 0 ) && ( Engine\_Speed == 0 )))

{

// ABC…

}

#### **Maintainability**

* Maintaining the code after it is written is a task that can become extremely difficult. Often, the code just doesn’t make sense to others who look at it.
* This can lead to incorrect interpretation, so even though a new feature goes into the code the existing code around it breaks.
* Another thing that can happen is if someone besides the author comes into the code to make a change; if he doesn’t understand the existing structure then another “if” condition could get placed at the bottom of the code just to avoid making any changes to the top part of the code.

with path to take.

Consider the following code segment:

if ( GetEngineSpeed() > 700 )

{

// Execute All Speed Governor code

}

**Modular**

The module simply means the software components that are been created by dividing the software. The software is divided into various components that work together to form a single functioning item but sometimes they can perform as a complete function if not connected with each other.

**Cohesion**

Cohesion is a measure of the degree to which the elements of the module are functionally related. It is the degree to which all elements directed towards performing a single task are contained in the component. Basically, cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.

**Robust Programming**

Robust programming is a style of programming that focuses on handling unexpected termination and unexpected actions. It requires code to handle these terminations and actions gracefully by displaying accurate and unambiguous error messages. These error messages allow the user to more easily debug the program.

**Difference between correctness and Robustness:**

**Correctness:**

* Ability to perform according to specification.
* Software design must satisfy the requirements for the application
* Correctness of design usually means sufficient design.

**Robustness:**

* Robust program is a program that does not fail due to unexpected or unlikely input.

SDLC

SDLC produce high quality system that meets or exceeds customer expectations works effectively and efficiently in the current and planned information technology infrastructure and is inexpensive to maintain and cost effective to enhance.

PHASES:

Requirement

Analysis

Design

Development

Testing

Deployment and maintenance

REQUIREMENT:

Client needs are written as Business Requirement Specification (BRS).

Business Analyst gives that to development team.

ANALYSIS:

Product requirement are defined, documented an approved by the customer and Software Requirement Specification (SRS) is made.

Key people involved in SRS are Project manager, Business Analyst and senior members.

DESIGN:

High Level Design

Low Level Design

HIGH LEVEL DESIGN:

Done by architects and senior developers.

Gives architecture of the software to be developed.

Low Level Design:

Done by senior developers.

describes each and every feature in the product how it should work and how every component should work.

OUTCOME:

High level document and low-level documents are produced.

DEVELOPMENT:

Start building the software and start writing the code.

Outcomes are Source Code Documents and developed product.

TESTING:

When the software is ready it will be tested so it is error free and then goes to implementation.

Outcomes are Quality products and testing artifacts.

DEPLOYMENTS AND MAINTAINACE:

It will be done by deployment/ implementation engineers.

When the problem occurs, it needs to be solved from time to time

Maintenances should be done as per Service Level Agreements.

SDLC MODELS:

Waterfall

Spiral

V-Model

Agile

Prototype

Rapid Application Development (RAD)

Rational Unified Model

Hybrid Model etc

DESIGN STRATGIES

DIFFERENT DESIGN STRATEGIES ARE AS FOLLOW

1. Structure design

2. Function oriented design

3.Object oriented design

4. Design process

SOFTWARE DESIGN APPROACHES

a) Top-down design approach

b) Bottom-up design process

STRUCTURE DESIGN:

Based on divide & conquer strategy where problem is divided into several sub programs.

Each sup programs are individually solved until the whole problem is solved; the small pieces of problems are solved into solution modules.

Structured design emphasis that these modules be well organized in order to achieve precise solution.

These modules are arranged in hierarchy. They communicate with each other; a good structured design always follows some rules for communication among multiple modules

FUNCTION ORIENTED DESIGN:

System is comprised of many smaller sub-systems known as function. These function are capable of performing significant task in the system. The system is considered as top view of all function.

This design mechanism divides the whole system into small function. Which provides name of abstraction by concealing the information and their operation. These functional modules can share information among themselves by means of information passing & using information available globally.

Another characteristic of function is that when a program calls a functions, the function changes the state of the program, which sometimes is not acceptable by other modules. Function oriented design works well where the system state does not matter & program/function works on input rather than on a state.

DESIGN PROCESS:

The whole system is seen as how data flows in the system of means of data flow diagram.

Data flow design depicts how function changes data & state of entire system.

The entire system is logically broken down into smaller units know as functions on the basic their operation in the system.

Each function is then described at large.

Software design process can be perceived as series of well-defined steps. Through it varies according to design approach (function & object oriented, yet it may have the following steps:

A solution design is created from requirement or previous used system and/or system sequence diagram.

Object are identified & grouped into classes on behalf of similarity in attributes characteristics.

Class hierarchy & relation among them is defined.

Application framework is defined.

OBJECT ORIENTED DESIGN:

OOD works around the entities & their characteristic instead of functions involved in the software system. This design strategies focuses on entities & it's characteristic.

IMPORTANT CONCEPTS OF OBJECT-ORIENTED DESIGN:

Object - All entities involve in the solution design is known as objects.

Classes – A class is a generalized description of an object. An object is an instance of a class. Class defined all the attributes, which an object can have and methods, which defines the functionality of the object.

Encapsulation – In OOD, the attributes & method are bundled together is called encapsulation.

Inheritance – sub classes can import implement and re-use allowed variables and methods from their immediate super class. Easy to define specific class and to create generalized classes from specific ones.

Polymorphism – same method name with different arguments. Allows a single interface performing tasks for different types.

SOFTWARE DESIGN APPROACHES:

Here are two generic approaches for software designing:

TOP-DOWN DESIGN:

We know that system is composed of more than one-system and it contains a number of components.

BOTTOM-UP DESIGN:

Starts with most specific and basic components. Its proceeds with composing higher level of components by using basic or lower-level components until the desired software is not evolved.

What is STLC?

STLC defines a series of activities conducted to perform Software Testing is a process of evaluating the functionality of a software application to find any software bugs.

Importance of STLC

⦁ Points out the defects and error.

⦁ Ensures customer readability and satisfaction.

⦁ Delivers Quality products.

⦁ Allows the software to stay in business.

⦁ Prevents from unwanted software failure and hence its cost overhead.

Phases

1.Requirements and Analysis

2.Test Planning

3.Testcase Development

4.Environment setup

5.Test Execution

6.Test Cycle closure

SDLC Vs STLC

1.origin

Software Development Lifecycle Since it is deals with the development of the software itself.

STLC is begins with the software is to be tested

2.Relationship

SDLC can be consider as the parent of the software testing life cycle. This is because of part of software development. STLC can be consider as the child of SDLC

3.Intension

SDLC is to be able to develop the quality software smoothly. Without SDLC there are chances of losing track, leading towards to know undesired results. STLC is formulated towards to detect analyze and debug any errors a shortcoming of the software that is being developed.

4.Phases

1.Planning

2.Requirement Analysis

3.Designing

4.Coding

5.Testing

6.Deployment

7.Maintainance Requirement Analysis

Test Planning

Test case development

Environment setup

Test execution

Test cycle closure

5. Requirement Analysis

1.Performed by system engineers or Business analyst.

2. Feasibility study

3. Time Constraints

4.Requirement Elicitation

5.Scheduling 1. Performed by quality assurance team.

2. Understanding testing requirements

3.Interaction with stockholders such as clients, BAS, technical leads etc.,

6.Design Phase

Performed by developers, technical architects.

Decisions based on high and low designing.

Parameters such as risk, technologies to be used, the capability of the team etc., are decided upon Performed by test Architects.

Understanding of high levels and low-level testing points takes place.

Timelines are finalized.

Coding Phase

Design is working out in terms of software.

Performed by software developers

Various tools such as IDE's text editors, compilers, debuggers, interpreters etc., are used Performed by test team

Test cases and data is prepared.

Testing Phase

Deals with actual testing of the software

Can be done suing unit testing, integration testing, system testing, etc.,

Bugs are fixed by the software developers. System integration takes place

Errors and shortcomings are reported retested and fixed.

Also involved regression testing.

Capability Maturity Model

It is a benchmark for measuring the maturity of an organization’s software process

A methodology used to develop and refine an organization’s software development process

It was developed by the software development community the software engineering institute(SEI) and the Carnegie Mellon university ,under direction of the US department of defense

A Maturity model provides

A place to start the benefit of a community prior experiences

A common language and a shared vision

A framework for prioritizing actions

A way to define what improvement means for your organization .

Levels of Capability Maturity Model

Initial

Repeatable

Defined

Managed

Optimizing

Level 1

Initial processing is unpredictable poorly controlled and reactive

Level 2

Repeatable processing are characterized for specific projectors and organization is often reactive

Level 3

Defined projects tasks their process from the organization development methodology

Level 4

Managed process are measured and controlled

Level 5

Optimizing focus on process improvements

Manifesto for Agile software development

We are uncovering better ways of developing software by doing it and helping others do it . Through this work we have come to value .

Individuals and interactions over processes and tools working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan .

That is ,while there is value in the items on that right ,we value the items on the left more .

Twelve principles of Agile Software

View Signatories

About the Authors

About the Manifesto

Afrikaans

Albanian

Amharic

Azarbaycanca

Principles behind the agile Manifesto

We fellow these principles our highest priority to satisfy the customer though early and continuous delivery of valuable software .

Welcome changing requirements ,even late in development Agile processes harness change for the customer’s competitive advantage .

Deliver working software frequently ,from a couple of weeks to a couple of months ,with a preference to the shorter timescale .

Business people and developers must work together daily throughout the project .

Build project around motivated individuals .given them the environment and support they need ,and trust to get the job done .

The most efficient and effective method of conveying information to and within a development team is face to face conversation .

Working software is primary measure of progress

Agile processes promote sustainable development. The sponsors ,developers ,and users should be able to maintain a constant pace indefinitely

Continuous attention to technical excellence and good design enhances agility.

Simplicity the art of maximizing the amount of work not done is essential .

The best architectures ,requirements ,and designs emerge from self- organizing teams .

At regular intervals ,the team reflects on how to become more effective , hen tunes and adjusts its behavior accordingly

SCRUM

Milestones

Sprint

Product backlog

Burn down chart

User story

Wish list

Product roller

Scrum master

Team

Release planning

Release backlog

Story point-smallest component

Estimate in hours

Sprint 1 ,sprint 2,….Sprint n

Sprint (2 days to 30 days )

Sprint length α Sprint recycle

Burndown chart(description of sprints progress)

Graph (work remaining vs number of days)

Always shows percentage of total project done

Slop of the graph is called project velocity or burndown velocity or rate of productivity which is equal to number of hour per day

Burn down chart work remaining divided by rate equal no of days to completion

1. Inventory Control System

Environment:

People need to buy Products they need. Every products they might need different types of Products such as Groceries, daily needs and so on.They might buy many products if wide range of products available.

Program Name:

To do one Inventory Control Program to Figure out the Commulative amount for items with tax slabs.

Problem Definitions:

Need to build an inventory control system that contains different category of items grouped in a storage.

Problem Description:

When people buy an item or many items from the inventory, the total value produced should be the amount and the tax of the particular products added together.

Pseudo Code:

⦁ Start

⦁ To givein the item details for different variable with default values and display it

⦁ To typein the item details with Tax rate for different variables with default values and display it

⦁ To calculate the Tax values with cummulative amount and display the item details

⦁ Input Item records

⦁ Set the item name

⦁ Set the item id

⦁ Set the item description

⦁ Set the item cost

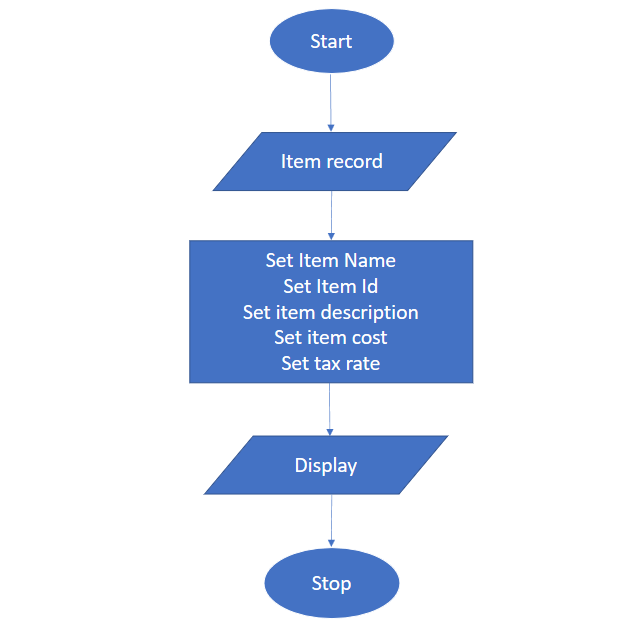
⦁ Set the tax rate

⦁ Store the above details in different variables

⦁ End

Flow Chart:

Inventory Control System:



Taxable Inventory System:

⦁ start

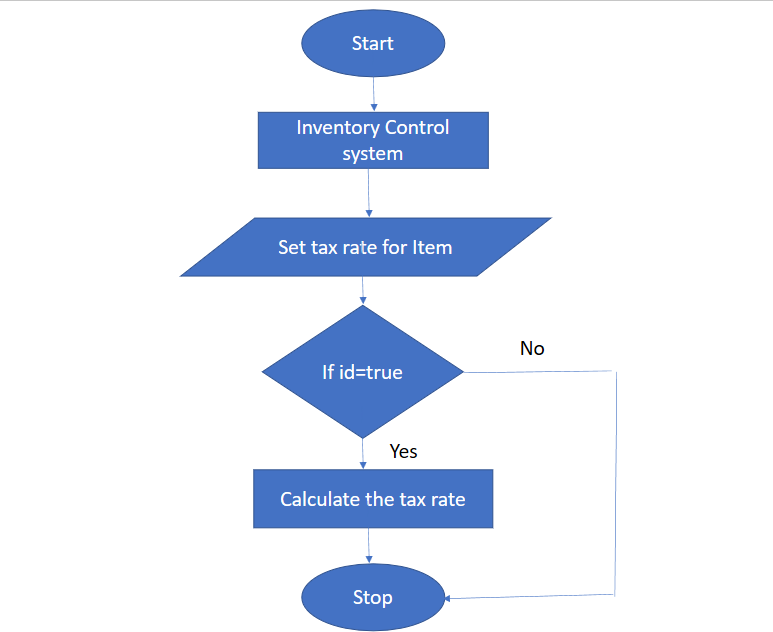
⦁ Input inventory control sytem

⦁ Set tax rate

⦁ check item id

⦁ calculate the tax rate

⦁ End



Billing Inventory System:

⦁ Start

⦁ Input inventory item system

⦁ input item ID

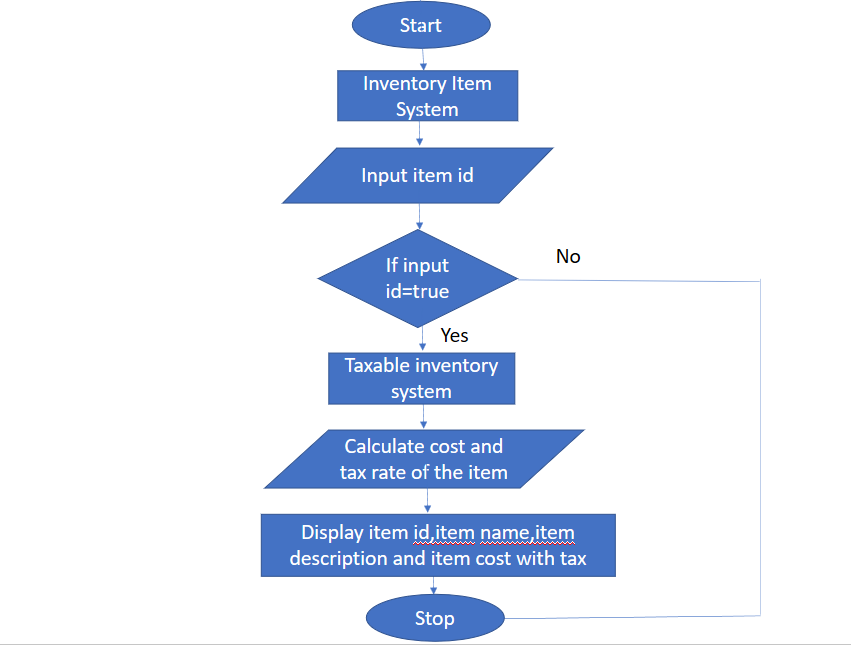
⦁ check the item ID

⦁ Get taxable input system

⦁ calculate the cost rate and tax value

⦁ Display item ID, item name, item description and item cost

⦁ END



Train ticket booking via online:

Project Name – Train ticket booking for passenger through online

Task- Visiting Native Place

Target- Online Ticket booking

Decision Making Process- First /second class

Exception-Second Class

Actual Process- Travelling

Dependency-No.Of.Passengers

Environment

To visit native ,So I need to travel from Chennai to Villupuram via train, In such case I usually prefer first class as my comfort. In case of rejection I am okay with second class, it depends upon the number of passengers .

Pseudo Code

-Initiate the program ticket booking for train

-Initiate input from the counter

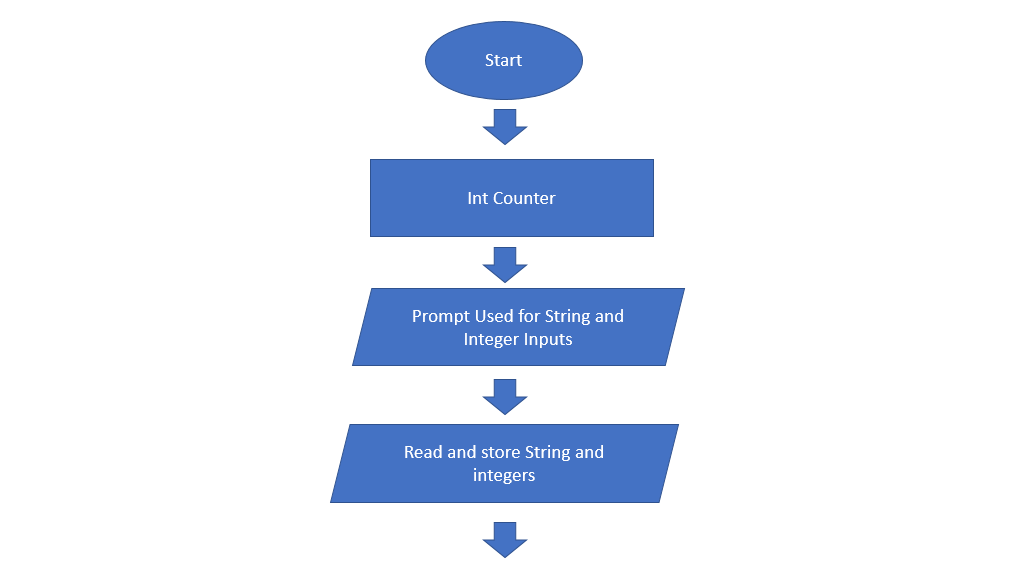
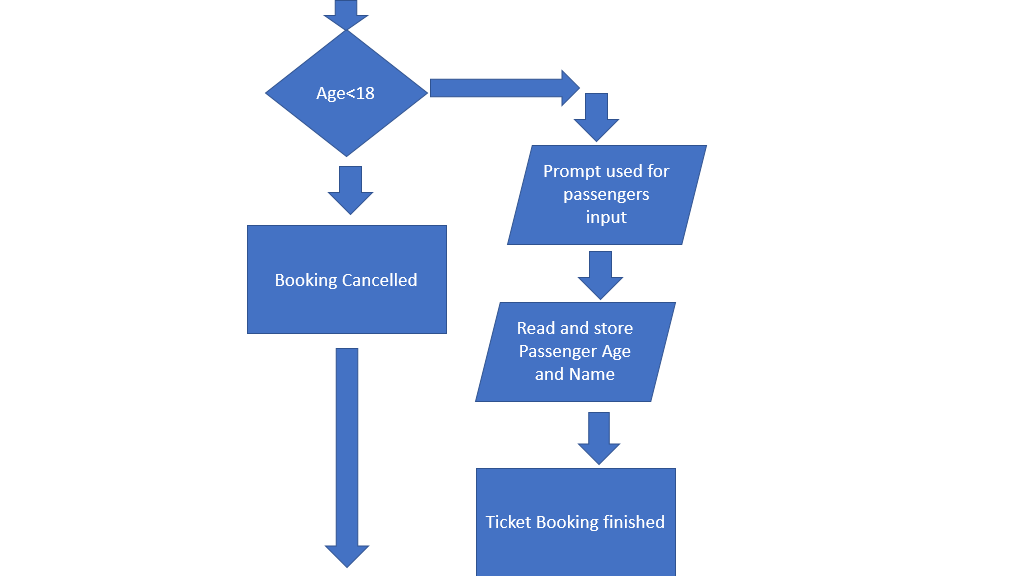
-Set name and age input from the user

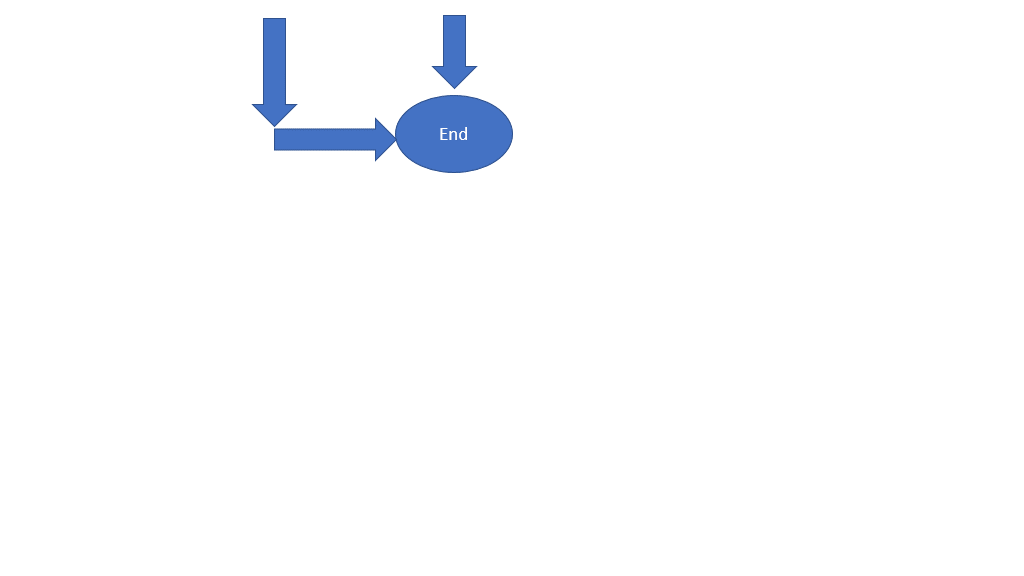
-Set the number of tickets needed and input the needed passengers information

-Set the age limit higher than 18 to access the ticket booking or Cancel the ticket booking process

-Output the passengers details and complete the ticket booking process

Flow Chart



2.Matrimonial Site

**MATRIMONIAL WEBSITE**

ENVIRONMENT: -

1)In this fast era all are familiar with matrimonial website concept.

2)As it is very hard to physically find life partner as per our choice or like even it is impossible now.

3)This matrimonial site helps for both groom and bride to find their life partner as per their criteria and choice without broker contact.

PROGRAM NAME: -

These procedures to be followed to preserve the matrimonial website:

L1)First to access the matrimonial website the groom or bride must enroll their respective details such as name,age,city,region,contactdetails,personaldetails,role,salarydetails.

L2)Make a suitable username and password to create an account.

L3)then compile the respective details in the profile board to search for the desired bride and groom.

PSEDO CODE: -

L1.1)Give the name,

Set the age,

Set the gender,

Set region,

Set the caste,

Give the contacts details,

Set salary for role,

Give the location details.

L2.1)Give a accceptable username and set a strong password,

Now login.

L3.1)upload photo,

Describe interests,

Specify the height,

Specify the colour,

State educational qualifications,

Mention the horoscope details,

Find the bride and groom with required specification.

**FLOW CHART**

