Application Workflow

There are a number of steps to follow when planning a big project. A good programmer would give proper attention to each step and in doing so will save themselves time in the development and testing stage. By planning well in advance problems may be anticipated and dealt with before they manifest into bigger issues.

Steps to **Application Workflow**:

1. Understand the Problem
2. Plan the Program (Flowcharting/Pseudocode)
3. Write the Code
4. Translate the Program into Machine Language (Compile the Code) - *if applicable*
5. Test the Program. Debug!
6. Put the Program into Production

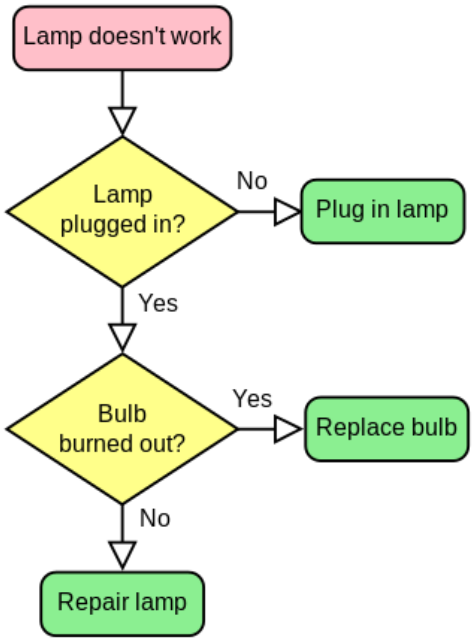
Flowchart

A flowchart is a type of diagram that represents a process or workflow (some people also say an algorithm) showing all the steps involved. These steps are represented by boxes and are positioned in order with arrows linking them. The arrows show the general flow of the process.

Ultimately a flowchart is an illustration of a solution to a problem. They are used in analyzing, designing, documenting or managing a process or program in various fields including web development.

Flowcharts in different fields tend to differ slightly with the type of symbols or boxes used to illustrate a particular event. In general however

- a **processing step** (sometimes referred to as an ***activity***) is denoted by a rectangular box,
- and a **decision** is denoted by a diamond.



Pseudocode

Pseudocode is a detailed and readable description of a computer process or algorithm. It is expressed in natural language as opposed to a computer language so as to be easily understood. Think of it as instructions written in plain enough English.

Pseudocode typically omits details essential for a machine to understand such as variable declarations or system specific code. There is no real standard syntax for pseudocode.

Generally speaking the comments you should be making in your code would resemble pseudcode in that they explain the process of what is happening.

```
procedure fizzbuzz
  For i := 1 to 100 do
    set print_number to true;
    If i is divisible by 3
then
      print "Fizz";
      set print_number to
false;
    If i is divisible by 5
then
      print "Buzz";
      set print_number to
false;
    If print_number, print i;
    print a newline;
  end
```

Which to Use When Planning a Program?

We can use either flowcharts or pseudo code to plan out an application, or parts of an application. They are useful in understanding the types of decisions and processes that we need to code into our programs. Either/or or both could be employed and both have their positive and negative traits.

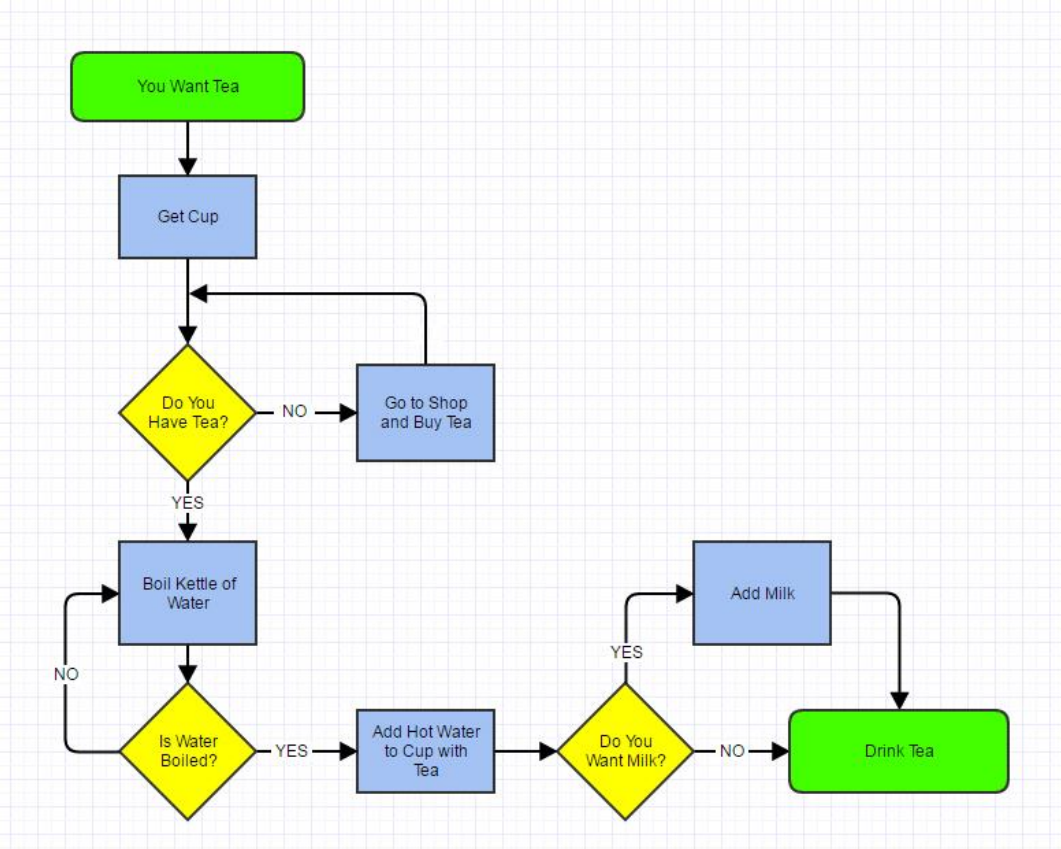
Flowchart	Pseudocode
Visual/Symbol based	**Text based that resembles actual code of any language**
Good for high-level overviews of a problem or smaller problems	**Ability to explain problems in detail**
Easily understandable by non-developers	**Can be re-used as comments in code**
Too much detail can create confusion	**Can be time-consuming**

Flowcharts can be made with online applications such as: https://www.gliffy.com/ or https://www.draw.io/

Pseudocode should be made with a simple word document

Planning a Cup of Tea

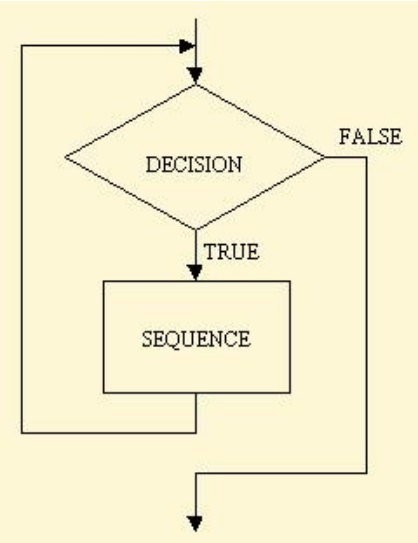
Flowchart: Start and End points are denoted by a **rounded edge box**.
Actions are denoted with **rectangles**.
Decisions are represented by a **diamond**.



The pointed arrows show the flow of the program illustrating where the next step is. You'll notice that decisions can lead to jumps in the chart often skipping or adding an action.

```
1   Making Tea
2   //////////
3
4   Get a cup from kitchen cabinet
5
6   if haveTea == true
7       Put tea bag in cup
8   else
9       Go to shop and buy tea bags
10      Then put them in cup
11
12  Boil kettle of water
13  When water is boiled, add hot water to cup
14
15  if wantMilk == true
16      Add milk to tea
17  else
18      Do not add milk
19
20  Drink Tea
```

Loops and Iterations



In order to enact some form of iteration (a while or for loop) that will repeat an action for a set period of time, you will have to place a decision in front of an action. This decision should keep linking back to the same action until the true or false statement has switched. When this happens the program will flow along a different course.

In pseudocode it is good practice to start a loop or iteration with the word “while” followed by the condition to be met. This is for a while loop of course. Underneath this should be the action that needs to be carried out and following this the while loop should be closed off with an “End While” to make it clear and distinct.

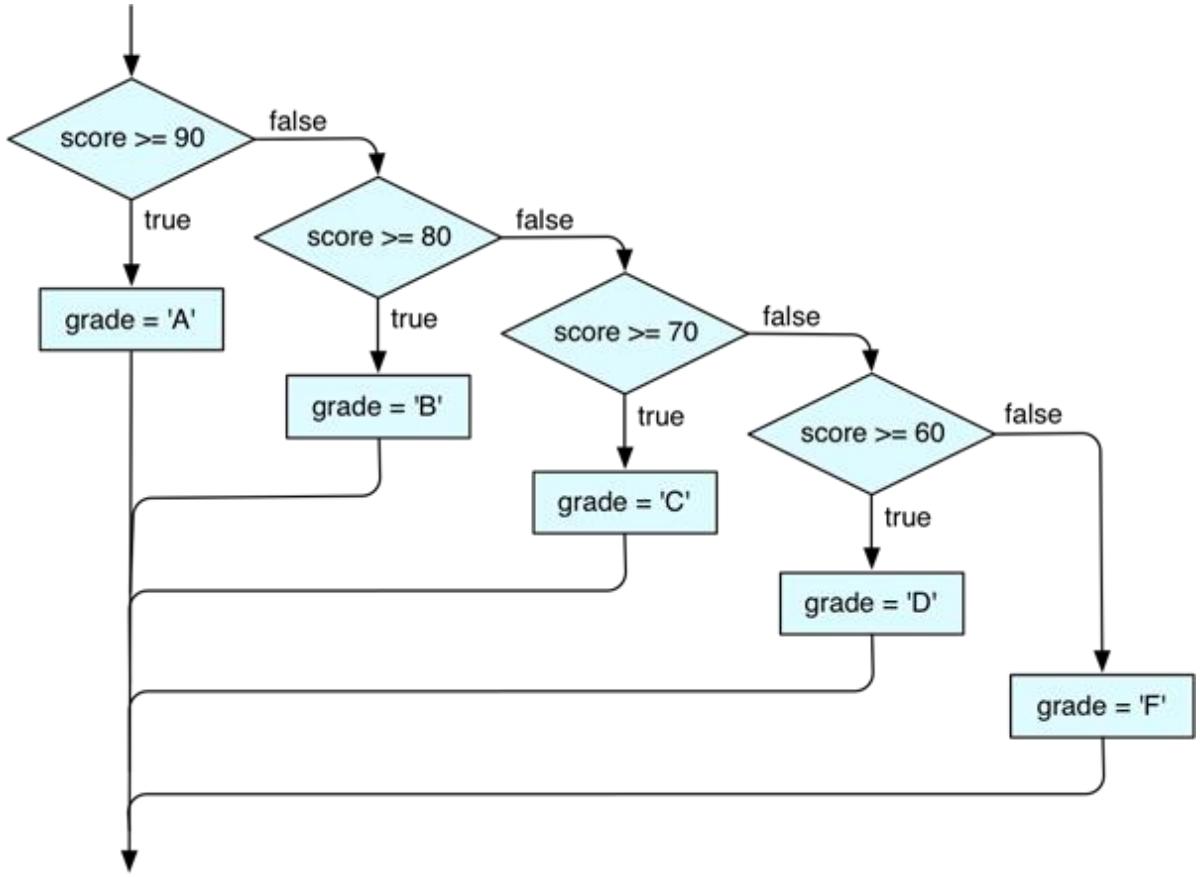
WHILE house of cards is still standing

    Add more cards to top

END WHILE

Multiple Decisions

If in a flowchart there are multiple decisions to be made on the same level you can just link the decision diamonds together in a nested format. The chart will read: is this true, is this true, is this true down along the group and when it reaches the correct statement it will flow in a new direction.



For more information on Flowcharts and Pseudocode check out this website:
http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl_flow.htm

Operators

Operators are used to perform operations on variables and values. There are a number of different types of operators which are broken down to the type of actions that they carry out. Such actions include subtracting integers from each other, concatenating (putting together) strings together, assigning values to variables and comparing variables.

There are 5 main types of operators:

1. **Assignment Operators:** These put values (or assign values) to variables.
2. **Arithmetic Operators:** These perform certain mathematical operations such as addition, subtraction, multiplication, division etc.
3. **String Operators:** Special operators designed specifically to deal with string values and variables. The most common version is Concatenation which sticks strings together.
4. **Comparison Operators:** These are used in conjunction with **if else** statements (SELECTIONS or conditional statements), in order to compare variables to values e.g. if myHeight > 1.5m
5. **Logical Operators:** These are used specifically in conditional statements to extend the conditions to be checked or compared.

In the following examples the variable **x** is **equal to 2**, and the variable **y** is **equal to 1**.

1. Assignment Operators

Assignment	Same as...	Description	Show it
x = y	x = y	The left operand gets set to the value of the expression on the right	x = 1

2. Arithmetic Operators

Operator	Name	Example	Result	Show it
+	Addition	$x + $y	Sum of $x and $y	3
-	Subtraction	$x - $y	Difference of $x and $y	1
*	Multiplication	$x * $y	Product of $x and $y	2
/	Division	$x / $y	Quotient of $x and $y	2

4. Comparison Operators

Operator	Name	Example	Result	Show it
==	Equal	$x == $y	Returns true if $x is equal to $y	False
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	False
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	True
>	Greater than	$x > $y	Returns true if $x is greater than $y	True
<	Less than	$x < $y	Returns true if $x is less than $y	False
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	True
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	False

5. Logical Operators Example

Operator	Name	Example	Result	Show it
and	And	$x and $y	True if both $x and $y are true	False
or	Or	$x or $y	True if either $x or $y is true	True