**Flowchart Notes**

* The flowchart is a graphical tool describing program flow, facilitating both designing algorithms and communicating to others how algorithms work
* An Instruction Pointer (IP) indicates which instruction or command is currently being executed in a program
* Program flow is the various paths that an IP can follow when executing a program

**Flowchart Elements**

* Terminator Oval - The beginning and end of program execution
* Arrow - Arrows indicate the direction of program flow
* Decision Diamond - Program execution can go multiple ways
* Input and Output Parallelograms - Input and/or output
* Processing Rectangle - something is computed

**Terminators**

* Flow starts at the “start” and ends at the “end”
* Start terminators have no inflow arrow and exactly one outflow arrow
* End terminators can have many inflow arrows but no outflow arrows

**Arrows**

* Make the flowchart more understandable by making your arrows point in the same general direction
* Use a connector to connect symbols which are far from each other on the flowchart. As a rule, connectors should be avoided as much as possible

**Decisions**

* The strength of flowcharts as a way to represent algorithm designs comes from how decisions can be made understandable to the casual observer.
* Decision diamonds are the only symbol with more than one outflow arrow

**Input and Output**

* Usually the designer will put a note or two in the parallelogram indicating what kind of data are being inputted or what is displayed to the user
* Every I/O symbol has a single arrow leaving it.
* Program flow does not stop in an I/O symbol, and there is always exactly one direction for program flow to go once the input/output task is complete.
* Designers frequently combine input and output instructions into a single flowchart symbol.
* One can also use a single I/O symbol to represent very complex output involving hundreds of lines of code.

**Processing**

* The processing symbol is used to represent any type of work that the program is performing. This could be simple tasks such as adding numbers or large tasks such as sorting numbers.
* When a flowchart gets too large or complicated to fit on a single sheet, then it is often useful to break it up into many smaller flowcharts. This can be accomplished using the processing symbol.

**Variations on Flowchart**

* When working with assembly language (such as with embedded systems), a flow process chart may be a good alternative to the flowchart. It facilitates working with linear languages and gives plenty of room for annotations. In those cases, though, it may be useful to replace the flow process chart symbols with flowchart symbols so a wider audience can understand them
* Swimlane Charts were developed in the 1940s as a way to represent processes that cross organizational boundaries. Modern swimlane flowcharts use the same symbols as flowcharts but with vertical columns representing structural borders (like program logic, user settings, user interface)
* The most important difference between a flowchart and an activity diagram is that a single arrow can split into two without a decision. This enables two concurrent activities.
* A decision tree is a graph used to represent the many outcomes of a large or complex decision.
* Decision trees cannot represent complete algorithms; they can only represent decisions. In other words, there are many ideas represented in flowcharts that cannot be represented with a decision tree diagram. The opposite is not true. Every single decision tree can easily be represented with a flowchart. We thus say that decision tree diagrams are subsets of flowcharts.

**Designing with Flowcharts**

* Step 1 - Draft
* Step 2 - Refinement (Mistakes are more easily found in the refinement stage of design than they are in coding)
* Step 3 - Validation (The algorithm walkthrough process is much easier with flowcharts than with other programming drafting tools. Make sure problem description has been met)
* Using all three of these design steps – drafting, refining, and validating – can greatly reduce the code-writing and debugging time of any nontrivial software development task.

Flowchart for user authentication:  
  
A diagram of a program

Description automatically generated

Flowchart for program that adds up the numbers between one and ten:  
  
A diagram of a program

Description automatically generated