Designing programs before coding…and a review of habits to have

CS 10A – DOCUMENTATION

Allen Zhao – SRJC CS 10A

A Review of Good Coding Habits

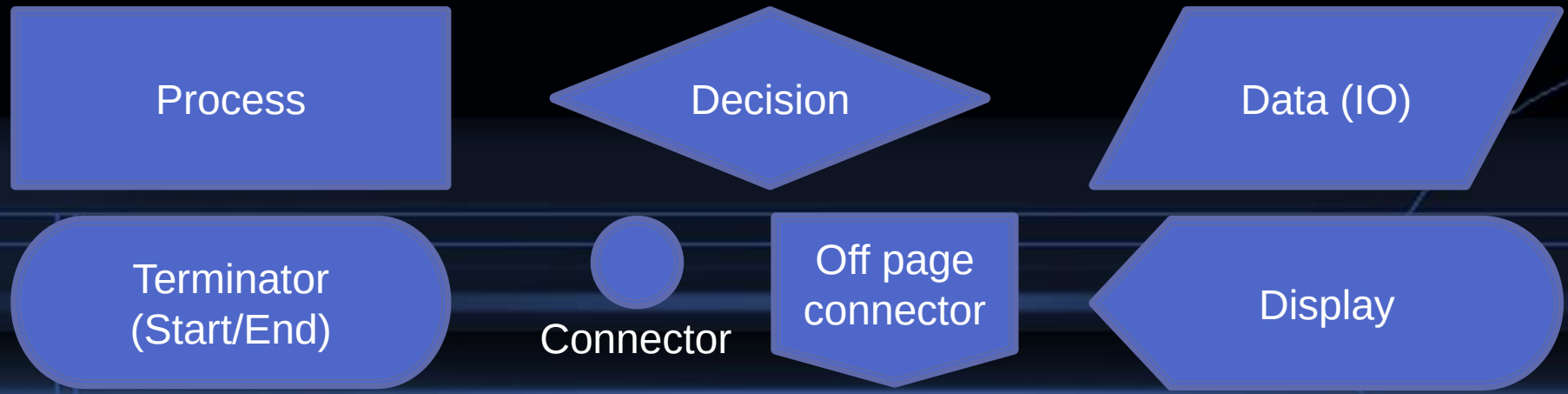
- Never hardcode. If a number has a definition and is used in multiple places, then use a variable in its place instead.
- Comment often. As your project gets bigger, you won't necessarily remember what your code even does at a certain part when you come back to it later. Also gives other people who need to look at your code fewer headaches.
- Utilize your space. Indent your code lines properly. Use new lines to break up long logic statements. This helps makes reading your code easier. Also, since most modern IDEs auto-tab for you, tabs happening out of place can be taken as a hint that something is missing in your syntax.
- Give good names for variables and functions. The names themselves are like comments, which makes code easier to read and comprehend.
  - Suggested: constants can be written in all caps
  - Suggested: functions and variables comprised of multiple words should be marked by capitalizing the first letter of each individual word, or spacing with an underscore. I also recommend choosing one style for functions and the other variables.
- Never delete code. Comment them out instead. This allows you to quickly revert back to your original code in case you can't find a bug that needs fixing.

Intro to Documentation

- Documentation is all about describing, in plain language, what the software does and how it works internally.
- There are many forms of software documentation, and they often involve visual aids and tables. However, there are some common standards.
- Another common method of documentation is known as the Source Code. This is basically an entire program translated into a bunch of comments, often one comment per command, while maintaining the original structure and formatting. Because there are many different programming languages out there, source code is useful in helping programmers translate programs between languages.
  - If you've ever asked why <some video game> cannot be remade for modern hardware, the most likely reason is because the original source code was lost, and it would be too expensive to recreate the whole thing from scratch if the intent is to mimic the original.
- For the most part, creating, maintaining, and updating documentation can be annoying. Programmers often forget that they exist. Nevertheless, having documentation at all is better than not having any. While smaller projects do not necessary need documentation, big projects certainly will.

Software Flowcharts

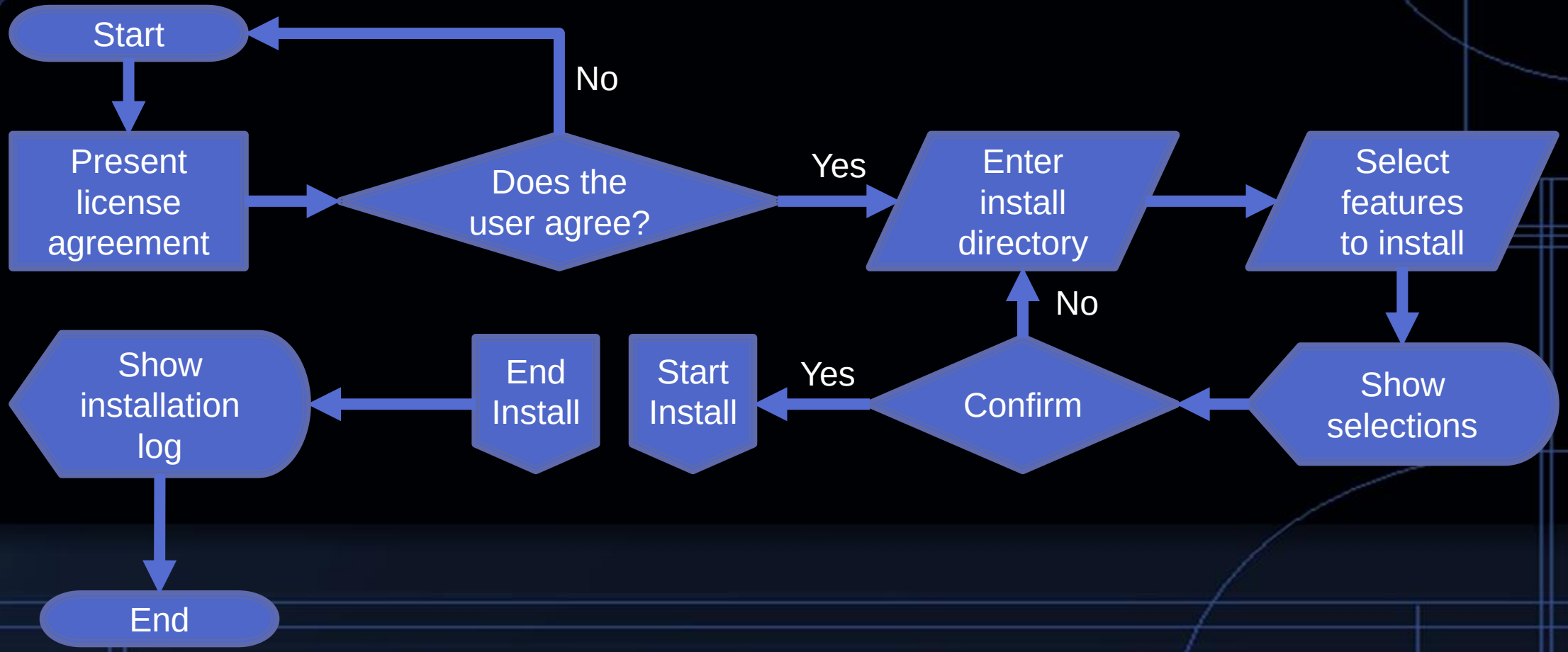
- Flowcharts are the most common defined standards in software documentation.
- A specific block shape denotes what kind of operation that's happening at any given point. Arrows point (in one direction only) from one block to the next.



How to Flowchart

- Start with the flowchart. Forget the code aspect for now. Map out the entire process of the program's objective in layman's terms. This is referred to as high-level detail.
- Each process and data block should then be translated into code independently, usually as functions.
- Connect the individual blocks together in your code with the appropriate loops and logic statements.
- Avoid crisscrossing your arrows. Use connectors (and name them) to visually link block diagrams together.
- I recommend using connectors to group sections of a flowchart dedicated to larger functions. Make sure that the use of on-page off-page connectors are consistent.
- Tip: A flowchart doesn't have to contain every detail, feature, or aspect of a program. It's just a general overview of how it should work.

Flowchart Example 1 – Install Software



Flowchart Example 2 – DigitSum.cpp

Summary for Proper Flowcharting

- Use the right shapes for the right context.
  - Some flexibility is allowed here, but the bare minimum is to use diamonds (decisions) where appropriate.
- Do not overlap shapes or arrows.
- Flow path must be clearly stated. Shapes with multiple outputs path need to specify the condition under which that specified path is taken.
- Start and end of a program must be clearly indicated.
- Do NOT include actual code in your flowchart text.
  - Text should be just a few words describing the process at hand.

Flowchart Tools

- I recommend getting a rough draft down on paper first just so you can scribble out your thoughts. Or any free drawing apps.
- You can use Powerpoint to draw your flowchart diagrams. If you go to Insert > Shapes > Flowchart, you'll find all the symbols you need to draw up a flowchart. If you want something less clunky, Microsoft Visio is the MS product designed for creating flowcharts.
- Since MS Visio is not free, I instead recommend one of the two free tools below to draw your flowcharts.
  - Google Drawings: On Google Drive, click New > More > Google Drawings
  - yEd Graph Editor: Requires an installation. This tool allows you to see where exactly you're currently zoomed in on the whole document.
- Most drawing tools contain a section solely dedicated to flowcharts within their Insert Shape commands. Look for those!