PPL 4/27

Reagan Shirk

April 27, 2020

Final

- This is the first lecture I've shown up to in weeks, oops
- Our final will be take-home, we'll get it the Friday of dead week and it will be due during the final period on Wednesday (no later than 6:30pm on Wednesday)
- We can work in groups up to 3, they don't have to be the same as our homework groups

In-Line Expansion

- Some subroutines can be expanded in line at the point of call
- Essentially just copy and pasting the called function into the caller- it replicates the code
- Pros:
  - Avoids ooverhead with space allocation, the rest were missed
- Cons:
  - Missed

Parameter Passing

- Formal parameters vs actual parameters
  - Formal: used in function definition
  - Actual: parameters that are passed into the function
- Passing modes:
  - By value
    - We have definitely use this one, a new copy of the actual variable is being passed
    - Copies all contents from the caller stack frame to the callee stack frame
  - By reference
    - We have probably used this one, the actual address to the actual parameter are being passed
  - By value/result
    - Probs haven't heard of this one
    - Pass by value and pass by reference hybrid
    - Similarity with by value: copies the value of the actual parameter into the formal parameter, local changes mostly don't affect the actual parameter
    - Similarity with By reference: the final value/result is copied back into the actual parameter, so the actual parameter *is changed*
  - Read only
  - By name
    - boils down to textual replacement and implies re-evaluation of the passed parameter
  - Named parameters (not the same as passing by name): arguments are usually positional and requires explicity stating the name of the formal argument in the calling sequence
  - Default and others
    - No need to actually pass them, they have a default value
- when implementing, the main thing to think about is to pass the address of the original parameter or to copy it