Do-While loop

While / For loop review

- In While/For loops, the terminating condition is checked at the *beginning* of each loop

```
int i = 0;
while (i < 10) {
    printf("%d\n", i);
    i++;
}
```

Do-While loop

- Performs in the exact opposite manner
- Checks the condition AFTER each repetition

```
do {
    /*Code goes here*/
} while (<terminating condition>);
```

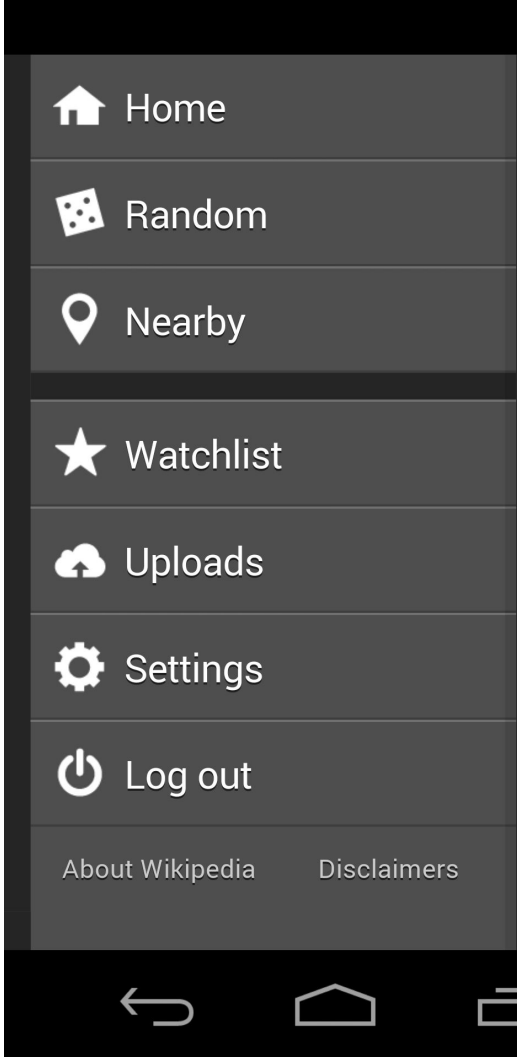
Do-While loop

- Performs in the exact opposite manner
- Checks the condition AFTER each repetition

```
int i = 0;
do  {
    printf("%d\n", i);
    i++;
} while (i < 10);
```

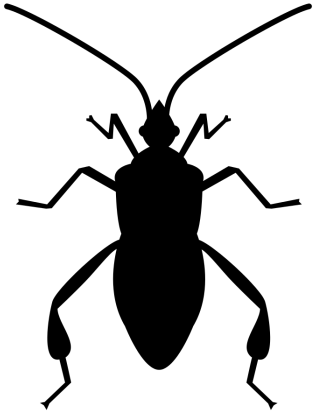
Do-While loop

- Advantage: the code will ALWAYS run at least once
  - This is helpful when you need that code to run once (then make decisions after that first execution)
  - Ex: Menu options

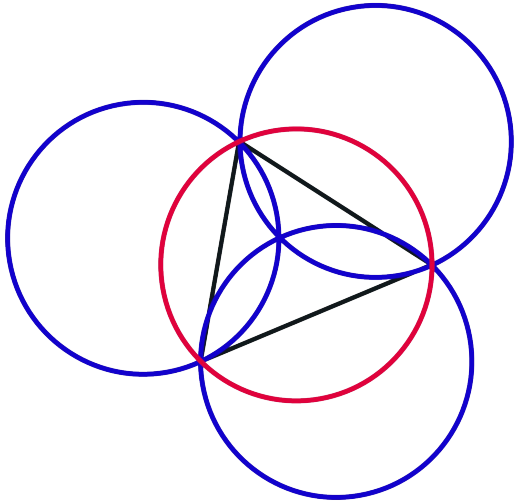


Do-While loop

- Disadvantages: Looks very similar to a while loop, but executes entirely differently
  - Can introduce bugs into your code if not properly executed



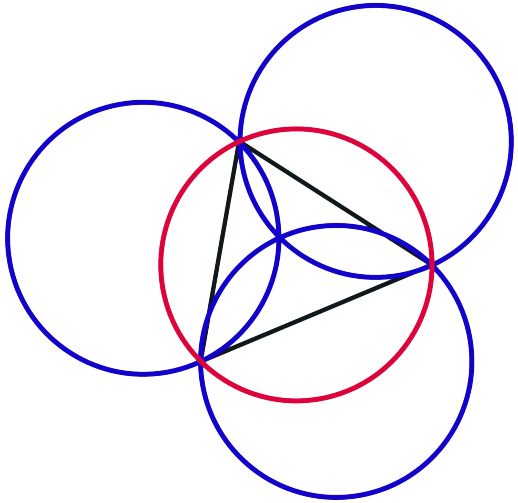
Structured Program Theorem



- Developed in 1966 by Corrado Bohm and Giuseppe Jacopini

- States that all logical statements (therefore, all computer programs) can be written as a sequence of while loops and if statements.

Structured Program Theorem



- Developed in 1966 by Corrado Bohm and Giuseppe Jacopini

- Therefore, do-while loops (and for loops) are unnecessary
- Also, argues against *break* and *continue* statements (since they execute out of sequence)

Do-While Coding Challenge: Bank

- Create a functional bank, using a do-while loop
  - Scan in an “initial deposit”
  - 5 Options:
    - 1) Deposit (add) money
    - 2) Withdraw (subtract) money
    - 3) Calculate interest, at 5%, on current amount
    - 4) Show current amount
    - 5) Quit program (use do-while loop)