More on Loops

Based on slides © McGraw-Hill
Modified by Diana Palsetia

Another loop: Do-While

do {
    \textit{statement(s)}
} while (\textit{condition});

- Do the statement/block at least once
- Evaluate the \textit{condition}. If it is
  - true: re-execute \textit{statement(s)}; repeat step 2
  - false: we’re done with the loop

```c
int x = 0;
do {
    x = x + 1;
} while (x < 3);
```

Nested loops

- Loop inside a loop
- Just like nested if statement
- Classic usage: row-col problem

```c
#include <stdio.h>
int main(){
    int i;
    int j;
    for (i = 1; i <= 10; i++) {
        for (j = 1; j <= 10; j++){
            printf("%d \t", i * j);
        }
        printf("\n");
    }
    return 0;
}
```

Break and Continue Statements

- \textit{break} and \textit{continue} are Java statements
- Are also “flow control” statements
  - if, while, do-while, for, return
  - A break “breaks you out” of the closest enclosing loop or switch statement
  - A continue is a shortcut to the next iteration of the loop
- A loop may have
  - Zero or more break statements
  - Zero or more continue statements
while-loop with break, continue

while (condition1){
    ... if (condition2) continue; // go up and re-evaluate condition1
    if (condition3) break; // exit the loop
    ...
    ...
} // after a break statement, execution resumes here

while-loop with break, continue example

int x = 1;
while (x <= 10){
    if (x % 2 == 0){
        printf("%d\n",x); break;
    }
    x = x + 1;
}

What will happens with break vs. continue ?

for-loop with break, continue

for (expr1; condition1; expr2){
    ...
    if (condition2) continue; // evaluate expr2, then condition1
    if (condition3) break; // exit the loop
    ...
    ...
} // after a break statement, execution resumes here