Iterative Statements (Loops)

Hour 7

Objectives

- for loop
- comma operator
- null statement
- while loop
- do-while loop
- break and continue

for Loop

Loop controls in one location

- Test at top
  - May not execute
- Any expression may be omitted
- Expression 1 is the initializer
  - Executed only once
- Expression 2 is the loop test
  - Loops while expression 2 is true
  - Tested after expr 1
  - Tested after expr 3
- Expression 3 is the update
  - for (expr-1; expr-2; expr-3)
  - for (i = 0; i < 10; i++)
  - printf("%d", i);

for Loop Examples

Simple counting

```c
int i, j;
for (i = 0; i < 10; i++)
  printf("%d
", i); /* prints 0 - 9 */
for (i = 0; i < 10; i += 2)
  printf("%d
", i); /* prints 0, 2, 4, 6, 8 */
for (i = 0, j = 0; i < 10 && j < 5; i += 2, j++) /* comma operator */
  printf("%d,%d
", i, j);
for (i = 79; i >= 0 && s[i] == ' '; i--)
  /* null statement */
```

for Loop Variations

Non-"standard" loops

```c
for (;;) /* infinite loop */
  statement;
for (i = 0; s[i] == ' '; i++) /* initializes i */
  statement;
for (; s[i] != ' '; i++) /* empty init */
  statement;
List L;
for (L = root->next; L != root; L = L->next) /* have to work */
  statement; /* with integers */
```

while Loop

Controlled repetition

- Test at the top
  - May not be executed
- Loops while expression is true
  - while (expression)
    - statement;

```c
int n = 100;
while (n > 0)
  printf("%d
", n--);
while((c = getchar()) != EOF)
{
  ...
}
```

do-while Loop

Less common loop

- Test at the bottom
  - Executed at least once
- Loops while expression is true
  - Opposite of Pascal's repeat-until
- Useful when the test expression is initialized by a statement in the loop body

```c
do
{
  statements;
  } while (expression);
do
{
  scanf("%d", &n);
  } while (n != 0);
```
Loop Interruption

Used to simplify program structure

- **break**
  - Terminates the inner most loop (execution resumes with the statement following the loop)

- **continue**
  - Skips remaining code in inner loop (from the continue statement to the end of the loop)
  - starts next loop iteration
    - for loops resume at the update expression following the test expression
    - do and do-while loops resume at the test expression

- Usually in an if-statement

---

Loop Interruption Examples

Some common idioms

```c
for (avi = optind; avi < argc; avi++)
    { if ((fp = fopen(argv[avi], "r")) == NULL)
        { printf("ERROR: unable to open "", argv[avi]);
            continue; /* try to open the next file */
        }
        
```

```c
for (;;) /* from qsort partition */
    { while (a[++i] < v); /* null statement!! */
        while (a[--j] > v);
        if (j <= i) /* loop until i and j cross */
            break;
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
```