# CSE 142
# Computer Programming I

**Loops**

K-1

---

# Loop Design

**The first question is, "What is the loop body?"**

**The next question is, "What is the termination test?**

**The third question is, "Should the termination test come at the top of the loop or at the bottom?"**

**The final problem is translating those answers into C.**

K-2

---

# Some Examples

Just type back (exactly) the line of input the user typed.

**Loop body?**
Read one character; print that character

**Termination test?**
The character read is '\n'

Test Location?
I want to print the '\n', so I want to enter the loop body at least once ⇒At the bottom

K-3

---

# Writing That in C

```
do {
   scanf ("%c", &nextChar);
   Printf ("%c", nextChar);
} while (nextChar != '\n');
```

K-4

---

# Example 2

Flush (throw away) the rest of the input line.

**Loop body?**
Read one character

**Termination test?**
The character read is '\n'

**Test Location?**
Read the character then test ⇒At the bottom

K-5

---

# Writing That in C

```
do {
   scanf ("%c", &nextChar);
} while (nextChar != '\n');
```
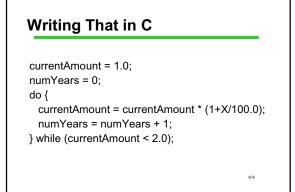
K-6

---

## Example 3

A rule of thumb is that money invested at X% will double in about 72 / X years.  Is that (roughly) right?

**Loop body?**
> Simulate one year passing:  Multiply current amount by (1+X/100); increment counter of number of years by 1.

**Termination test?**
> Stop as soon as the money has doubled

**Test Location?**
> Ambiguous:  "stop after n years" (bottom) and "stop before going another year" (year) are the same

## Writing That in C

```
currentAmount = 1.0;
numYears = 0;
do {
   currentAmount = currentAmount * (1+X/100.0);
   numYears = numYears + 1;
} while (currentAmount < 2.0);
```

## Writing That in C (Again)

```
currentAmount = 1.0;
numYears = 0;
while (currentAmount < 2.0);{
   currentAmount = currentAmount * (1+X/100.0);
   numYears = numYears + 1;
}
```

## Example 4: Simple Adding Machine

Evaluate expressions like "238 + 18 – 10 + 18"

**Loop body?**
> Tricky:    238 + 18 – 10 + 18
>
> Read an operator; read an operand; update expression value

**Termination test?**
> Stop when no more [operator, value] pairs left in put

Test Location?
> Top: There may not be any iterations of the loop body

## Writing That in C

```
nItemsRead = scanf("%d", &currentValue);
// test for input error
while (/* still more input */) {
   opChar = GetNextOpchar();
   nItemsRead = scanf ("%d", &nextOperand);
   // test for input error
   currentValue = Operate (currentValue, opChar, nextOperand);
}
```

## Example 5:
### *Event-Driven Programming*

Modern programs tend to be "event-driven"
- Program starts, sets itself up.
- Program enters a loop, waiting for some event or command to happen:
  - mouse click, key click, timer, menu selection, etc.
- Program performs operation ("handles" the event or command)
- Program goes back to its wait loop

## Simple Command Interpreter

**Repeatedly read in "commands" and handle them.**

**Input (symbolized by single characters)**

    **a -- execute command A by calling *process_A( )***

    **b -- execute command B by calling *process_B( )***

    **q -- quit**

**Pseudocode for main loop:**

    **get next command**

    **if a, execute command A**

    **if b, execute command B**

    **if q, signal quit**

K-13

---

# Command Interpreter Loop Control Schema

**repeat until quit signal**

**use variable "done" to indicate when done**

```
set done to false
do {
    body statements
    if quit command, set done to true
} while done is false
```

K-14

---

**Command Interpreter**

```c
int main(void) {
    int     FALSE=0;
    Int     TRUE=1;
    char    command;
    int     done;

    done = FALSE;
    do {
        command = ReadCommand( ); /* get command from user */

        if (command == 'A'){
            processA ( ); /* execute command A */
        } else if (command == 'B') {
            processB ( ); /* execute command B */
        } else if (command == 'Q') {
            done = TRUE;  /* quit */
        } else {
            printf("Unrecognized command\n");
        }

    } while (! done);

    return 0;
}
```

K-15

---

## Example 6

Print a line with n asterisks (e.g., "******")

**Loop body?**
    Print one asterisk
    Update count of number printed so far

**Termination test?**
    Stop when number printed == n

**Test Location?**
    Top: Allows n=0 (and n<0 in a way)

K-16

---

## Writing That in C

```c
for (starCount=0; starCount<n; starCount = starCount+1) {
    printf ("*");
}
```

*Loop body executes for values of iteration variable (starCount) 0, 1, 2, …, n-1*

K-17

---

## Example 6

Print the following:

```
N     **********
N-1   *********
N-2   ********
      *******
      ******
      *****
      ****
3     ***
2     **
1     *
```

**Loop body?**
    Print one row of n stars

**Termination test?**
    Stop when n == 0

**Test Location?**
    Top (allows for 0 rows total)

K-18

## Writing That in C

```
for (row=N; row>0; row = row-1) {
    // print "row" asterisks
}
```

*Loop body executes for values of iteration variable (row)
N, N-1, N-2, … , 1*

K-19

## Writing That in C

Print the following:

```
N      **********
N-1    *********
N-2    ********
       *******
       *******
       ******
       *****
       ****
3      ***
2      **
1      *
```

```
for (row=N; row>0; row = row-1) {
    for (stars=0; stars<row; stars=stars+1) {
        printf ("*");
    }
}
```

"Nest Loops"
   Outer loop executes once for N iterations
   Inner loop executes N times, for "row"
   iterations each time.

K-20

## Example 7: Printing a sin Wave

Print the following:

```
*
   *
     *
       *
        *
        *
       *
     *
   *
  *
    *
      *
```

**Loop body?**
   Print one row

**Termination test?**
   Stop when you've printed enough rows

**Test Location?**
   Top (allows for 0 rows total)

K-21

## Writing That in C

Online demo

K-22