CSE 142
Computer Programming I

Iteration

(and functions)

Overview

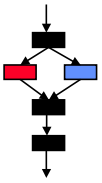
Concepts this lecture

- Iteration - repetitive execution
- Loops and nested loops
  - while statements
  - for statements
- Program development process
  - Stub/dummy routines
  - Unit Testing

Chapter 5

- Read Sections 5.1-5.8, 5.10
- Section 6.6 talks about development and testing, but is a little bit ahead of us

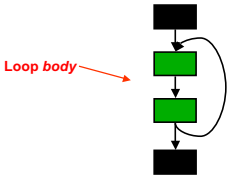
Preliminaries



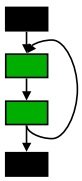
- Machine's execution speed
  - 10,000,000 lines of C per second
- Programmer speed
  - 12 lines of working C per day
- How many days will it take to write a program that runs for 1 second?

One More Type of Control Flow

Sometimes we want to repeat a block of code.
This is called a *loop*.



Loops



The loop body is a lot like a function:

- The body has a single, logical purpose
- There is often a single "loop variable" that *parameterizes* what it does

Additionally, there is a *termination test*

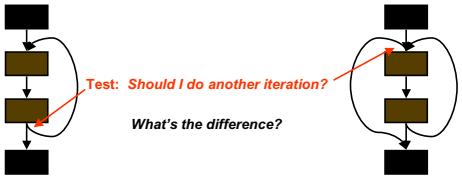
Example:

```
nItemsRead = scanf ("%d", nextInt);
if (nItemsRead != 1) {
    printf ("Input error.  Try again.\n");
}
```

Test: If it didn't work, try again

H1-7

Loop Structure



H1-8

Example:

Read integers until you get a zero, then print total

```
    nItemsRead = scanf ("%d", nextInt);
    if (nItemsRead != 1) {
Loop body →     printf ("Input error.  Try again.\n");
        return –1;
    }
    total = total + nextInt;
Loop test →  if nextInt isn't zero, repeat
    printf ("Total is %d\n", total);
```

H1-9

Example:

Sum digits of a number: e.g., 1234 ⇒ 10

```
    total = 0;
Loop test →  if theInt is zero, all done
        nextDigit = theInt % 10;
Loop body →  total = total + nextDigit;
        theInt = theInt / 10;
```

H1-10

Example:

Compute sin(x) for x = 0.0, 0.1, 0.2, …, 2.0

```
    x = 0;
Loop test →  if (x <= 2.0) continue
        result = sin(x);
Loop body →  printf ("%lf %lf\n", x, result);
    x = x + 0.1;
```

Iteration variable

H1-11

Example:

Print n asterisks

```
    index = 0;
Loop test →  if (index < n) continue…
Loop body →     printf ("*");
    n = n +1;
```

Iteration variable

H1-12

Loops in C

Three kinds:

- **Test At Top**
  while
- **Test At Bottom**
  do…while
- **Repeat as an iteration variable changes**
  for

H1-13

Test at top: *while*

Loop condition

```
while ( condition ) {
    statement1;
    statement2;
    ...
}
```

Loop body: Any statement, or a compound statement

H1-14

Test at top: *while*

Sum digits of a number: e.g., 1234 ⇒ 10

Loop test →
Loop body →

```
total = 0;
while (theInt > 0) {
    nextDigit = theInt % 10;
    total = total + nextDigit;
    theInt = theInt / 10;
}
printf ("sum of digits is %d\n", total);
```

H1-15

Test at bottom: *do…while*

```
do {
    statement1;
    statement2;
    ...
} while ( condition );
```

Loop body: Any statement, or a compound statement

Loop condition

H1-16

Test at bottom: *do…while*

Read integers until you get a zero, then print total

Loop body →
Loop test →

```
do {
    nItemsRead = scanf ("%d", nextInt);
    if (nItemsRead != 1) {
        printf ("Input error.  Try again.\n");
        return –1;
    }
    total = total + nextInt;
} while (nextInt != 0);
printf ("Total is %d\n", total);
```

H1-17

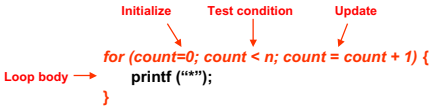
Iterate over variable: *for*

```
for ( initialize;
      test condition;
      update)  {
    statement1;
    statement2;
    ...
}
```

H1-18

Iterate over variable: *for*

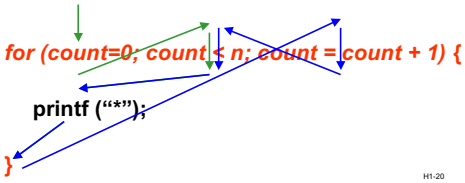
Print n asterisks



H1-19

Control Flow: *for*

Print n asterisks



H1-20

Live Example: *Arithmetic*

Input: 21.28 * 18.72

-4.7 + 0.8

12 / 3.0

Output: Computed result

H1-21