A Brief Problem

Read a number from the user.

Print every number from 1 up to the user's number.

Did you use a for loop or a while loop?
Why might you use a for loop on this problem?
So… there's a familiar pattern to the problem that tells you how to solve it!

A-1
4/22/2001

CSE142 Computer Programming I

Loop Development and Program Schemas

or… Secrets of CSE142 Revealed!

4/22/2001

Goals for Loop Development

Getting from problem statement to working code

Systematic loop design and development

Recognizing and reusing code patterns

A-3
4/22/2001

Example: Rainfall Data

General task: **Read daily rainfall amounts and print some interesting information about them.**

Input data: Zero or more numbers giving daily rainfall followed by a negative number (sentinel).

Example input data:

0.2 0.0 0.0 1.5 0.3 0.0 0.1 -1.0

Empty input sequence:

-1.0

Given this problem statement…
Is there just one way to approach the problem?

A-4
4/22/2001

Seattle Rainfall Program

```
#include <stdio.h>

int main(void) {
  /* Print out the daily rain prediction for
     tomorrow. */
  printf("yes.\n");
}
```

A-5
4/22/2001

Elsewhere… Rainfall Analysis

Some possibilities:

- Just print the data for each day
- Compute and print the answer to:
  - How many days worth of data are there?
  - How much rain fell on the day with the most rain?
  - On how many days was there no rainfall?
  - What was the average rainfall over the period?
  - What was the median rainfall (half the days have more, half less)?
  - On how many days was the rainfall above average?

What's similar about these? Different?
Which one is right?

A-6
4/22/2001

Example: Print Rainfall Data

```
#include <stdio.h>
int main (void) {
   double rain;        /* current rainfall from input */
   /* read rainfall amounts and print until sentinel (<0) */
   scanf("%lf", &rain);
   while (rain >= 0.0) {
      printf("%f  ", rain);
      scanf("%lf", &rain);
   }
   return 0;
}
```

Example: # Days in Input

```
#include <stdio.h>
int main (void) {
   double rain;          /* current rainfall from input */
   int ndays;            /* number of days of input */
   /* read rainfall amounts and count number of days */
   ndays = 0;
   scanf("%lf", &rain);
   while (rain >= 0.0) {
      ndays = ndays + 1;
      scanf("%lf", &rain);
   }
   printf("# of days input = %d.\n", ndays);
   return 0;
}
```

Is There a *Pattern* Here?

```
#include <stdio.h>
int main (void) {
   double rain;

   /* read rainfall amounts */

   scanf("%lf", &rain);
   while (rain >= 0.0) {
      printf("%f  ", rain);
      scanf("%lf", &rain);
   }

   return 0;
}
```

```
#include <stdio.h>
int main (void) {
   double rain;
   int ndays;
         /* read rainfall amounts */
   ndays = 0;
   scanf("%lf", &rain);
   while (rain >= 0.0) {
      ndays = ndays + 1;
      scanf("%lf", &rain);
   }
   printf("# of days %d.\n", ndays);
   return 0;
}
```

Pattern: “Read until Sentinel”

```
#include <stdio.h>
int main (void) {
   double variable;        /* current input */
   declarations;
   initial;
   scanf("%lf", &variable);
   while (variable is not sentinel) {
      process;
      scanf("%lf", &variable);
   }
   final;
   return 0;
}
```

We can reuse this pattern…so, we only have to solve this problem once!

Program Patterns

A design pattern is a pattern of code that solves a general problem.
- Also called a “program schema”

Learn patterns through experience, observation.

Write down your pattern’s name, context, advantages/disadvantages, and an example or two.

If you encounter a similar problem, you can reuse the pattern!

Tips For Problem Solving

Given a problem to solve, look for a familiar pattern.

Work the problem by hand to gain insight into possible solutions.

Ask yourself “what am I doing?”

Check your code by hand-tracing on simple test data.

Secrets of 142 Finally Revealed

Why is this class so hard?

The problems you solve in HW are interesting and would be difficult for an experienced programmer…

…except that the experienced programmer has seen the various pieces of these problems before!

This class is so hard because you are building up *your* repository of patterns!

A-13
4/22/2001

Back to the Pattern: "Read until Sentinel"

```
#include <stdio.h>
int main (void) {
   double variable;          /* current input */
   declarations;
   initial;
   scanf("%lf", &variable);
   while (variable is not sentinel) {
      process;
      scanf("%lf", &variable);
   }
   final;
   return 0;
}
```

We can reuse this pattern…so, we only have to solve this problem once!

A-14
4/22/2001

Pattern Placeholders

In this pattern, variable, declarations, sentinel, initial, process, and final are placeholders.

variable: holds the current data from input
- Replace every occurrence with your variable.

sentinel: the value that signals end of input

declarations: any additional variables needed

initial: statements needed to initialize variables before any processing is done

process: the actual work done for each input value

final: operations needed after all input has been processed

A-15
4/22/2001

Print Rainfall Data

```
decls:      double rain;          /* current rainfall */

initial:

            scanf("%lf", &rain);
            while (rain >= 0.0) {
process:         printf("%f ", rain);

                 scanf("%lf", &rain);
            }
final:
```

A-16
4/22/2001

Print # Days of No Rain

```
           double rain;       /* current rainfall */
decls:     int nDryDays;      /* days without rain */

initial:   nDryDays = 0;

           scanf("%lf", &rain);
           while (rain >= 0.0) {
process:        if (rain == 0.0)
                    nDryDays = nDryDays + 1;
                scanf("%lf", &rain);
           }
final:     printf ("Dry days: %d\n",nDryDays);
```

A-17
4/22/2001

Print Largest Daily Rainfall

```
           double rain;         /* current rainfall */
decls:     double maxRain;      /* Largest amount
                                   seen so far */
initial:   maxRain = 0.0;

           scanf("%lf", &rain);
           while (rain >= 0.0) {
process:        if (rain > maxRain)
                    maxRain = rain;
                scanf("%lf", &rain);
           }
final:     printf ("Largest rainfall: %f\n",
                     maxRain);
```

A-18
4/22/2001

Print Average Daily Rainfall

```
         double rain;          /* current rainfall */
decls:   double totalRain;     /* rain amount */
         int    nRain;         /* days */
initial: totalRain = 0;
         nRain = 0;
         scanf("%lf", &rain);
         while (rain >= 0.0) {
process:     totalRain = totalRain + rain;
             nRain = nRain + 1;

             scanf("%lf", &rain);
         }
final:   printf ("average rainfall is %f\n",
                   totalRain / nRain);
```

A-19
4/22/2001

Print Average Daily Rainfall (2)

```
         double rain;          /* current rainfall */
decls:   double totalRain;     /* rain amount */
         int    nRain;         /* days */
initial: totalRain = 0;
         nRain = 0;
         scanf("%lf", &rain);
         while (rain >= 0.0) {
process:     totalRain = totalRain + rain;
             nRain = nRain + 1;

             scanf("%lf", &rain);
         }
final:   if (nRain > 0)
           printf ("avg: %f\n", totalRain / nRain);
         else printf("No data");
```

A-20
4/22/2001

Summary

Loop design is not always a top-to-bottom process

Sometimes "process"/ "init"/ "final" is useful, with "decls" as needed

A program schema is a pattern of code that solves a general problem

We looked at just one, "Read Until Sentinel."

Look for other general patterns as you get more experience

A-21
4/22/2001

Bonus! A Whirlwind Tour of Loops and Patterns

We're now done with in-class work on iteration.

But… you should make absolutely *sure* that you understand this incredibly important control flow structure.

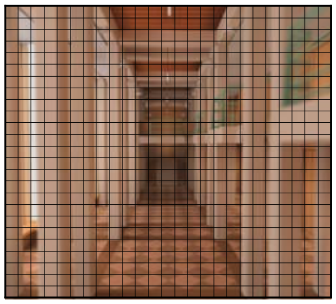
It's at the heart of many vital patterns…

A-22
4/22/2001

Simulation: Time Ticks

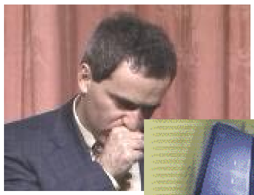
4/22/2001

"Ray Tracing": Iterating over a grid



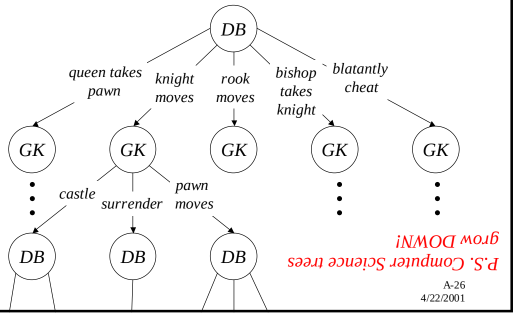
A-24
4/22/2001

Artificial Intelligence: "Tree Search"



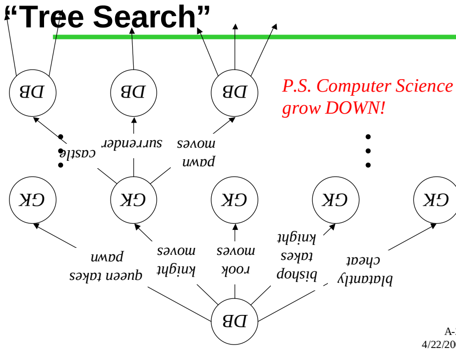
Pictures from ibm.com

Artificial Intelligence: "Tree Search"



A-26
4/22/2001

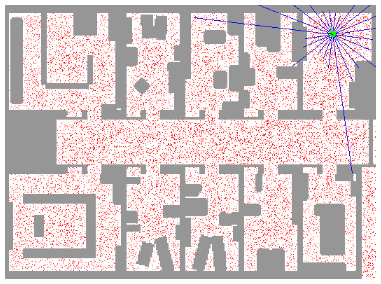
Artificial Intelligence: "Tree Search"



P.S. Computer Science trees grow DOWN!

A-27
4/22/2001

Robotics: Random Sets and Time Tick



A-28
4/22/2001

QOTD: Time Keeps Slipping…

Imagine your job was to write a simulation that predicted a value over time…

Say: How much money will Monty lose in *n* days if he plays one game of LMAD every day **and** 4 out of 5 contestants "keep"?

Describe just the "time tick" pattern that you would use to write this.

Sketch out each of the elements: name, context, advantages/disadvantages, and *short* example.

And, label the parts of the pattern that can be replaced for any given problem.

A-29
4/22/2001