CSE 142
Computer Programming I

Loop Development and Program Schemas

Goals for Loop Development

Getting from problem statement to working code

Systematic loop design and development

Recognizing and reusing code patterns

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: 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 raw data, what sort of information might we want to print?

Rainfall Analysis

Some possibilities:

Just print the data for each day

Compute and print the answer to one of these questions

- 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 of the days have more, half less)?
- On how many days was the rainfall above average?

What's similar about these? Different?

Rainfall Analysis

Some possibilities:

Just print the data for each day

Compute and print the answer a question like:

- 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's similar about these? Different?

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;
}
```

H2-7

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;
}
```

H2-8

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;
}
```

H2-9

Program Schema

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

Learn patterns through experience, observation.

If you encounter a similar problem, try to reuse the pattern

H2-10

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.

H2-11

Schema: “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;
}
```

H2-12

Schema Placeholders (1)

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

variable holds the current data from input. It should be replaced each place it occurs with the same appropriately named variable.

sentinel is the value that signals end of input.

declarations are any additional variables needed.

H2-13

Schema Placeholders (2)

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

process is the "processing step" - work done for each input value.

final is any necessary operations needed after all input has been processed.

H2-14

Schema for Rainfall

```
#include <stdio.h>
int main (void) {
  double rain;          /* current rainfall */
  declarations;
  initial;
  scanf("%lf", &rain);
  while (rain >= 0.0) {
    process;
    scanf("%lf", &rain);
  }
  final;
  return 0;
}
```

H2-15

Loop Development Tips

Often helps to start with

- What has to be done to *process* one more input value?
- What information is needed for *final*?

Declare variables as you discover you need them.

- When you create a variable, write a comment describing what's in it!

Often easiest to write *initial* last

- *initial* is "what's needed so the loop works the 1st time"

H2-16

Loop Development Examples

We will fill in the "Read Until Sentinel" program schema to solve a couple of problems

To save room on the slide, we will leave out this boilerplate:

```
#include <stdio.h>
int main(void) {
    Loop Schema
    return 0;
}
```

H2-17

Print Rainfall Data

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

initial:

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

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

H2-18

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);
```

H2-19

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);
```

H2-20

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);
```

H2-21

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");
```

H2-22

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

H2-23