Structured Programming Development in C

Deitel 8th Edition, Chapter 3

TOPICS

Software Development Method Case Study

Control Structures

Sequential and Compound Control Structures

Selection Control

Repetition Control

Formulating Algorithms – Three Case Studies

Secure C Programming Notes

SOFTWARE DEVELOPMENT METHOD CASE STUDY

Software Development Method

Specify

Analyze

Design

Implement

Test and Verify

Maintain and Update

CASE STUDY

Write a program that computes the area and the circumference of a circle. Ask the user for the circle's radius. Display the output to three decimal places, with appropriate labels.

1. Case Study – Specify Problem

Type the text of the problem into your source file and use it as a starting point.

“Write a program that computes the area and the circumference of a circle. Ask the user for the circle’s radius. Display the output with appropriate labels, and to three decimal places.”

Do you understand the core task?

2. Case Study – Analyze

"Write a program that computes the area and the circumference of a circle. Ask the user for the circle's radius. Display the output with appropriate labels, and to three decimal places."

What are the nouns, verbs, inputs, outputs?

Other data needed?

Other constraints – what other requirements are there?

2. Case Study – Analyze – Find Nouns & Verbs

"Write a program that computes the area and the circumference of a circle. Ask the user for the circle's radius. Display the output with appropriate labels, and to three decimal places."

Nouns:

Verbs:

2. Case Study – Analyze – Nouns & Verbs

"Write a program that computes the area and the circumference of a circle. Ask the user for the circle's radius. Display the output with appropriate labels, and to three decimal places."

Nouns:

- program, area, circumference, circle, user, radius, output, labels, decimal places

Verbs:

- write, compute, ask, display

2. Case Study – Analyze Purpose of Each Noun

Noun	Purpose
program	
area	
circumference	
circle	
user	
radius	
output	
labels	
decimal places	

2. Case Study – Purpose of Each Noun

Noun	Purpose
program	Indicates overall task (not useful in this context)
area	Output data
circumference	Output data
circle	Indicates the types of calculations
user	Source of input data (keyboard)
radius	Input data
output	Not useful in this context
labels	How to do display output
decimal places	How to do display output

2. Case Study – Analyze Verbs

Verbs: write, compute, ask, display

What do the verbs tell you?

2. Case Study – Analyze Purpose of Each Verb

Noun	Purpose
write	
compute	
ask	
display	

2. Case Study – Purpose of Each Verb

Noun	Purpose
write	Indicates overall task (not useful in this context)
compute	What to do with input
ask	Get input from user
display	Show results to screen

2. Case Study – Summary of Analysis Step

Inputs:

- radius

Outputs:

- area of circle
- circumference of circle

Constraints & Other Notes

- Complete program
- Input comes from user
- Need formulas for area and circumference of a circle
- Output goes to screen, and is formatted with labels and 3 decimal places

3. Case Study – Design Algorithm

Step 1

- Step 1.1
- Step 1.2

Step 2

- Step 2.1
- Step 2.2

Step 3

Step 4

3. Case Study – Algorithm

Step 1 – Get input data

Step 2 – Compute results

Step 3 – Display results

Keep breaking each step down into component steps until you can't break it down any further.

3. Case Study – Design Algorithm – Break it Down into Smaller Steps

Step 1 – Get input data

- Step 1.1 – get radius

Step 2 – Compute results

- Step 2.1 – compute area
- Step 2.2 – compute circumference

Step 3 – Display results

- Step 3.1 – Display area
- Step 3.2 – Display circumference

Keep breaking each step down into component steps until you can't break it down any further.

4. Case Study – Implement

Write the steps out in C syntax

Test and Verify:

- Review for syntax and logic errors
- Use different inputs to test the accuracy of the output

Is there anything else that should be added?

5. Case Study – Maintain / Update

Remove code that isn't needed (if necessary)

Comments are present and complete

- What should be included in the comments?

Clean up code:

- Align neatly
- Add white space
- Remove unused code or comments
- Follow C programming best practices

CONTROL STRUCTURES

What is a Control Structure?

Statement(s) that combine individual instructions into:

- a single logical unit
- with one entry point and
- one exit point

"Controls" the flow of execution

TRANSFER OF CONTROL

The order in which the statements execute

May be other than sequential

THREE KINDS OF CONTROL STRUCTURES

Sequential and compound statements

Selection control structure

Iteration (repetition) control structure

SEQUENTIAL AND COMPOUND CONTROL STRUCTURES

SEQUENTIAL CONTROL

A single statement

- May span more than one line

Each statement, or grouping of statements, is executed one after another in the order they appear

COMPOUND STATEMENT DEFINITION

Any group of statements surrounded by curly braces, aka a **block**

Executed sequentially

{ ← Control always starts here, at the first left curly brace

Statement 1;

Statement 2;

…

Statement n;

} ← Control ends here, at the next right curly brace

COMPOUND VS. SINGLE STATEMENT

A compound statement can be substituted for a single statement

And vice versa

Although such substitutions may change the logic of the program

SELECTION CONTROL

Selection Statements in C

Choose statement(s) to execute based on the value of a **condition**

Three types of selection:

- Single
- Double
- Multiple

THE IF SELECTION STATEMENT

Performs **single selection:**

- Performs a task if the condition is true
- Does nothing if condition is false

In C:

- "True" is **any nonzero numerical value (usually we just say 1)**
- "False" is **0**

No boolean data types in C

THE IF...ELSE SELECTION STATEMENT

Used for **double** and **multiple selection:**

- Performs a task if the condition is true
- Performs a different task if condition is false

NESTED IF...ELSE STATEMENTS

Tests for multiple cases by placing **if…else** inside of other **if…else** structures

- See two types of syntax, p. 76 & 77

Set of statements within curly braces is a compound statement (block)

REPETITION CONTROL

Repetition Definition and Types

Repeatedly executing the same statement(s)

Also known as **iteration**

Also known as **looping**

Two types of repetition

- Conditional
- Counter-controlled

Repetition (Iteration) Statements in C

Repetition can be implemented using a variety of statements and techniques

- Chapter 4

Three **repetition statements** in C:

- while
- do
- for

THE WHILE ITERATION STATEMENT

Continues to execute some statements (the body of the loop) while a condition **remains true**

Stops executing when condition becomes false

Formulating Algorithms
Case Study 1 –
Counter-Controlled Iteration

COUNTER-CONTROLLED ITERATION

Determine class average on a quiz, where we know the number of grades to average

What is the algorithm?

COUNTER-CONTROLLED ITERATION ALGORITHM

Get the grades:

    Get grade 1

    Get grade 2

    Get grade 3

    …

    Get grade n

Compute average

Display results

COUNTER-CONTROLLED ITERATION EXAMPLE

Fig03_06.c

Remember to initialize variables, especially ones used in calculations

Placement of Variable Definitions

Important!

Can group variable declarations at beginning of main

- And later at the beginning of functions

Can declare variables right before their first use

Formulating Algorithms
Case Study 2 –
Sentinel-Controlled Iteration

SENTINEL-CONTROLLED ITERATION

Uses a **sentinel value**

- Special value that indicates the end of the data

Sentinel-controlled iteration is a specific type of conditional iteration

Sentinel-Controlled Iteration to Determine Class Average

Determine class average, but the number of grades to average is not known in advance

What is the algorithm?

SENTINEL-CONTROLLED ITERATION ALGORITHM

Get the grades:

    Get grade until the sentinel value is entered

Compute average

Display results

Fig03_08.c

SENTINEL-CONTROLLED ITERATION EXAMPLE

Fig03_08.c

Converting Between Data Types

Data can be converted to a different type of data explicitly and implicitly

Explicit conversion

- Line 28 uses a **cast operator:** (float)
- Temporarily treats the variable **total** as if it was a float

Implicit conversion

- In an expression using different data types, some values are promoted
- In line 28, the variable **counter** is promoted

Integer Division

Remember – the result of dividing two integers is always another integer

Formatting Floating-Point Numbers

The printf in line 41 uses %.2f

.2 indicates the **precision** to use for displaying the value

- So, the number of values to display after the decimal point

If no precision is used in FCS, **default precision .6** is used

Values **displayed** may be rounded

- If the value in memory is larger than the display precision

Values **in memory** are not changed

- No matter how it is displayed

Formulating Algorithms
Case Study 3 –
Nested Control Statements

NESTED CONTROL STATEMENTS

Fig03_10.c

Counter-controlled loop to analyze 10 results to see if instructor should get a bonus (lines 15-30)

Each result analyzed to determine if it is a pass or fail

- Selection nested inside of iteration (lines 22-27)

Counts of passes and fails are used

Determine if 8 students passed in total (lines 37-39)

SECURE C PROGRAMMING

SECURE C PROGRAMMING NOTES

Arithmetic Overflow Error

- Occurs when a value is too large to store in its intended memory location
- Platform-specific min and max values that can be stored in an int are INT_MIN and INT_MAX, which are in limits.h

Unsigned Integers

- Used only for values that are nonnegative (so, zero and positive numbers)