Programming, Problem Solving, and Abstraction

Chapter Four

Loops

Chapter 4

Chapter 4 – Concepts

- Loops.
- Loop progress.
- Loop termination.
- Loop design.
- Loops that iterate over a data stream.

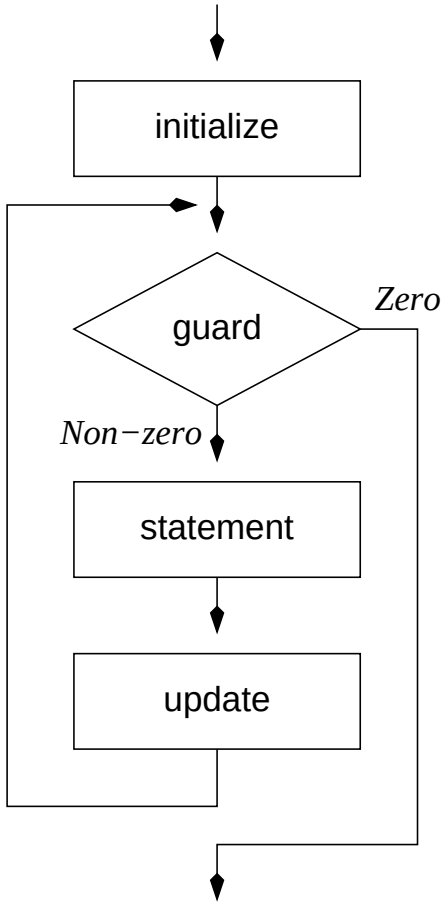
4.1 Controlled iteration

The `for` statement is important in C.

`for` (*initialize* ; *guard* ; *update*)
statement

A simple example:

- `forloop1.c`



4.1 Controlled iteration

The guard is only tested once per iteration.

If it becomes true part way through an iteration, that iteration still continues through to the end of the block.

Write a `for` loop that calculates the sum of the numbers from one to a hundred.

4.1 Controlled iteration

- `daynumber.c`
- `forloop2.c`
- `forloop3.c`

In a nested loop the inner loop completes all iterations before the update statement in the outer loop is executed. Then, if the outer guard is still non-zero, the inner loop is commenced again, starting with the initialize component.

The ++ postincrement operator is common in loop control.

4.1 Exercise 1

Write a nested `for` loop that calculates and prints the sum of the factors of n (not including n itself), for each value n from one to a hundred.

4.1 Controlled iteration

If a loop fails to make progress at each iteration, it is infinite.

An extreme example: "`for(;;);`" is a valid loop that does nothing for ever.

Be alert to the possibility that you may create infinite loops. Check loops by inserting a `printf` into each one.

4.2 Case study

Write a program that shows how, for interest rates of 2%, 3%, 4%, 5%, 6%, and 7%, a regular savings amount of $100 per month grows over periods of 1 to 7 years.

- `savings.c`

4.2 Case study

Points to note:

- All significant constants are set using `#define`
- The table is two-dimensional, so the loop structure is also two dimensional
- The inner loop is there to simply calculate one value, and is not part of the table's structure
- In-line comments and blank lines are used to separate the main components of the program

4.3 Program layout and style

What does this program fragment do?

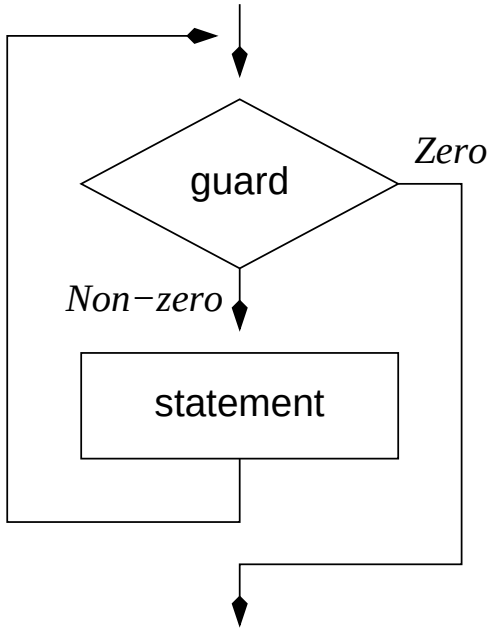
- `daynumber-squash.c`

For humans, the layout of a program is important. You will make fewer mistakes in programs that are presented tidily.

4.4 Uncontrolled iteration


```
while ( guard )
    statement
```

is the same as a `for` loop without the *initialize* and *update* components.



4.4 Uncontrolled iteration

In general, a `for` is used when the number of iterations is known in advance; and `while` is used when the end is recognized only when it is arrived at.

Note that, as for a `for` loop, the `while` guard is only tested once per iteration.

- `threen.c`

4.4 Uncontrolled iteration

A `for` or `while` loop continues when the guard is true, and cannot terminate until it is false.

To choose a guard, decide what condition must be true at the end of the loop, and then negate it.

Every iteration must move one step closer to making the guard false.

4.4 Exercise 2

Write a `while` loop that calculates $\lceil \log_2 x \rceil$ for a `double` value $x \geq 1$, by counting how many times x can be halved before it becomes less than one.

4.4 Uncontrolled iteration

There is a second way to exit a loop: the `break` statement.

It causes immediate exit, with control transferred to the next statement after the loop.

It is appropriate to use `break` when there are two results of a loop, one of which represents "success" and the other "failure".

4.4 Uncontrolled iteration

Prior to the `break`, a flag variable is set to indicate that the second exit is being used.

After the loop, the next statement should test the flag.

If there is no need to test the flag, then `break` is probably being abused.

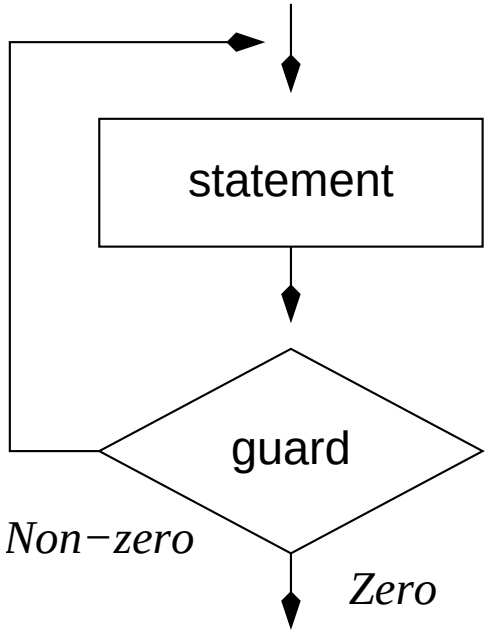
- `isprime.c`

4.4 Uncontrolled iteration

The do statement tests the guard after each iteration.

Hence the guard might be false the first time the body executes.

Best to avoid this type of loop.



4.5 Iterating over the input data

A loop can be controlled by a call to `scanf`, and continues as long as the `scanf` continues to receive valid input.

- `readloop1.c`

4.5 Iterating over the input data

Two further functions assist with character-by-character processing of the input: `getchar` and `putchar`.

They read/write a single character value, as an `int`.

The value `EOF` (usually -1) is returned if `getchar` attempts to read characters beyond the end of the file.

- `fortcomm.c`

4.5 Exercise 3

A self-assessment challenge to end the chapter:

Without looking at the program in Figure 1.2 on page 8, can you write a program that computes the sum and mean of a set of input numbers, supplied one per line?

Chapter 4 – Summary

- In C, the `for` loop is very powerful. Any/all all of the *initialize*, *guard*, and *update* parts can be altered, or omitted.
- The `while` loop is just a simplified variant of the `for` loop.
- Nested loops are used to create two-dimensional tables.
- Using `scanf` or `getchar`, a loop can process a stream of input data.