Lecture 5:

Loop Structures: *while* statement

Prof. Shervin Shirmohammadi
University of Ottawa

Loop Basics

- Most programs and algorithms require repetition structures in order to execute one or more instructions a certain number of times.
- A loop can be:
  - a definite loop if we know in advance the number of repetitions; or
  - an indefinite loop if we do not know in advance the number of repetitions.
- The number of repetitions:
  - in a definite repetition loop, this is controlled by a **counter** (usually an integer variable),
  - in an indefinite repetition loop, this is controlled by a **sentinel** (usually an integer variable).

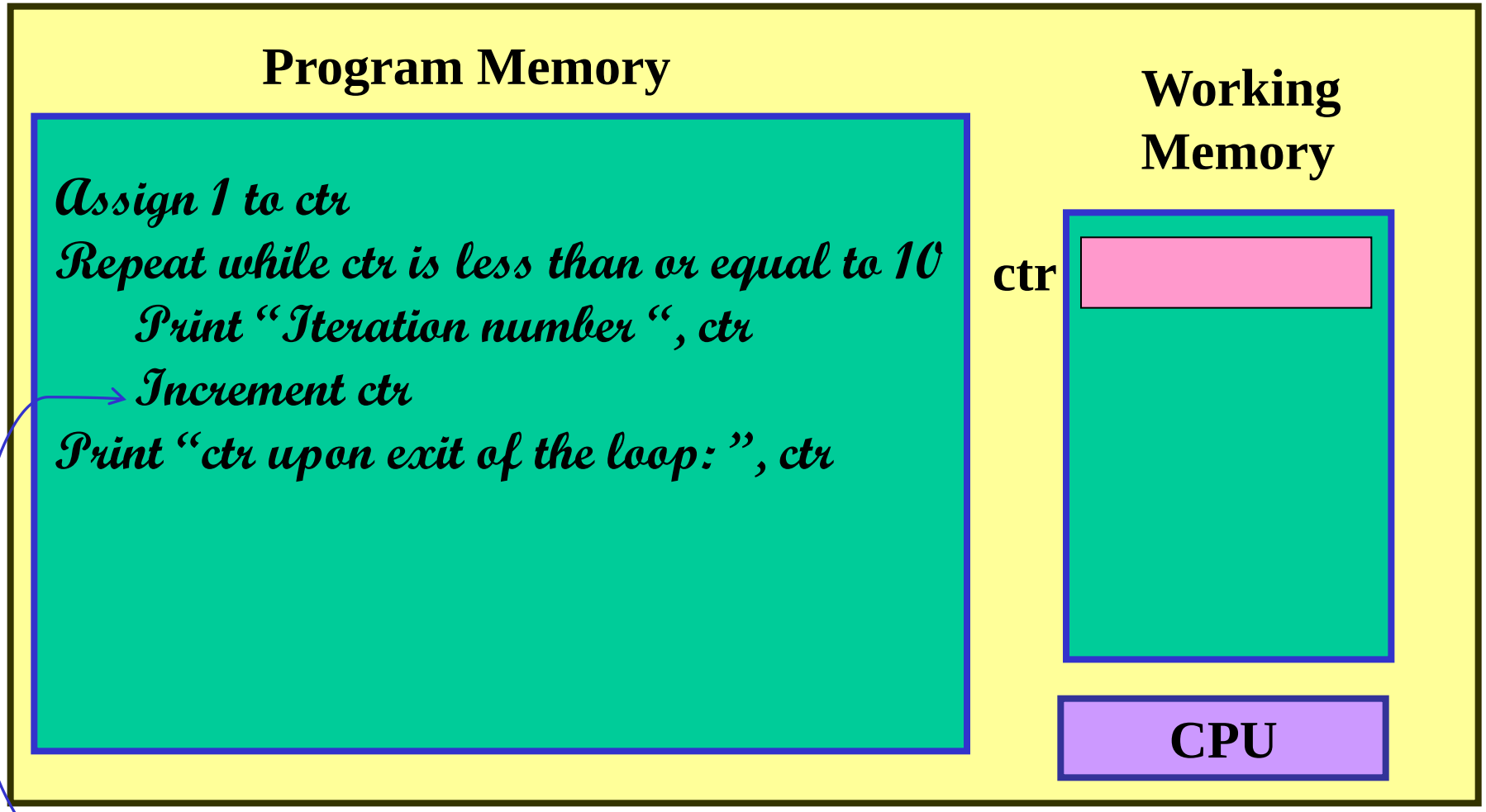
Loop structure expressed in pseudo-code

- A logical expression is evaluated repeatedly, and as long as the logical expression is TRUE, an instruction bloc is executed.

Repeat while *logical_expression*
    Instruction bloc that is executed while *logical_expression* is TRUE
OR
Repeat
    Instruction bloc that is executed while *logical_expression* is TRUE
While *logical_expression*

  - **Reminder**: In pseudo-code, all instructions at the same level of indentation constitute an instruction bloc
  - **Important**: the instruction bloc must modify one or more variables in the *logical_expresssion*.
- Nested loop structures
  - The loop structure is considered a single instruction
  - So it is possible to include another loop structure as part of the set of instructions within a loop structure

Example of a definite loop

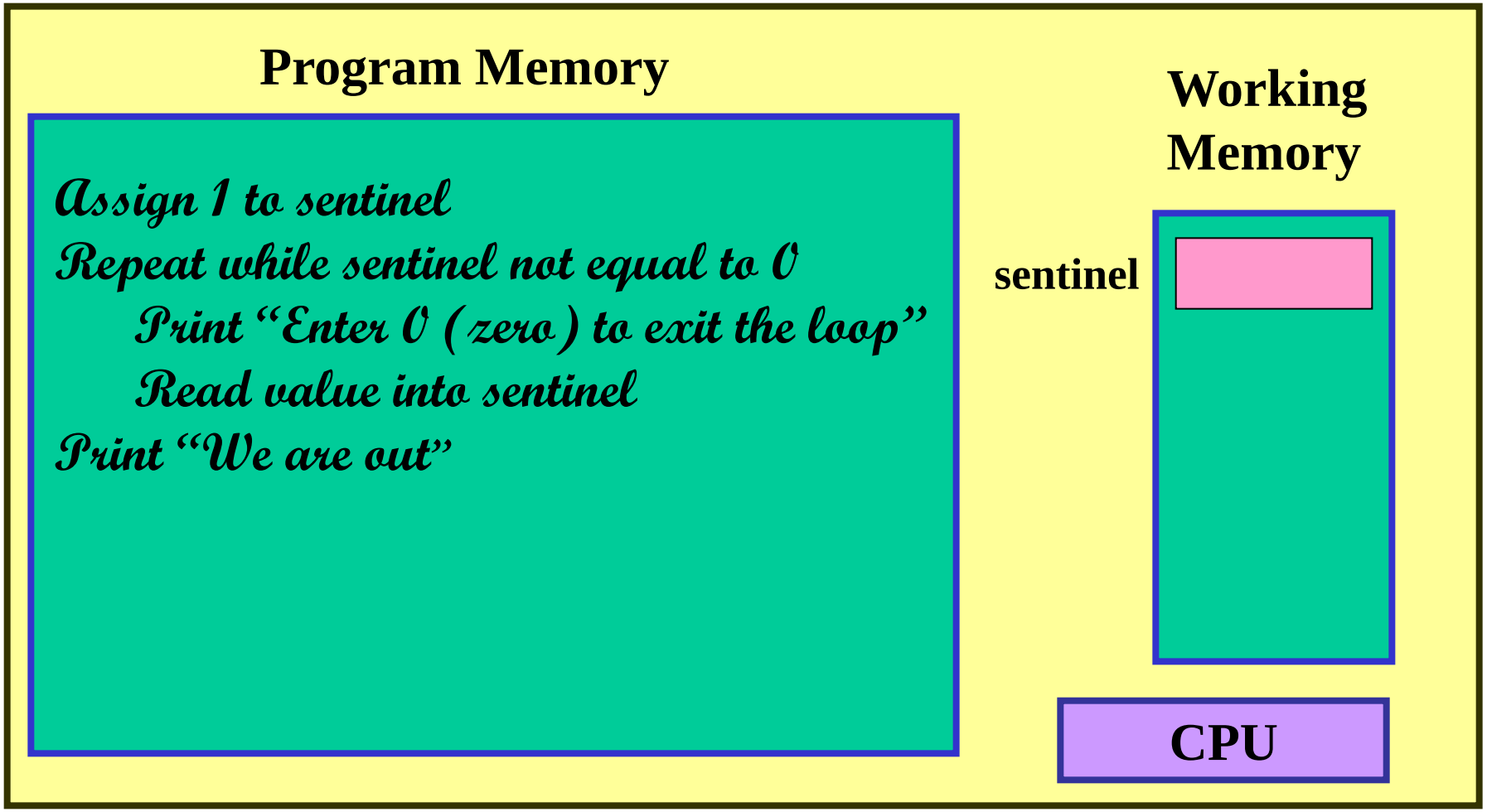


What will be the consequence of not having this line in the program?

Example

- Write a program that prints all integers from 10 to 20 on separate lines.

Example of an indefinite loop

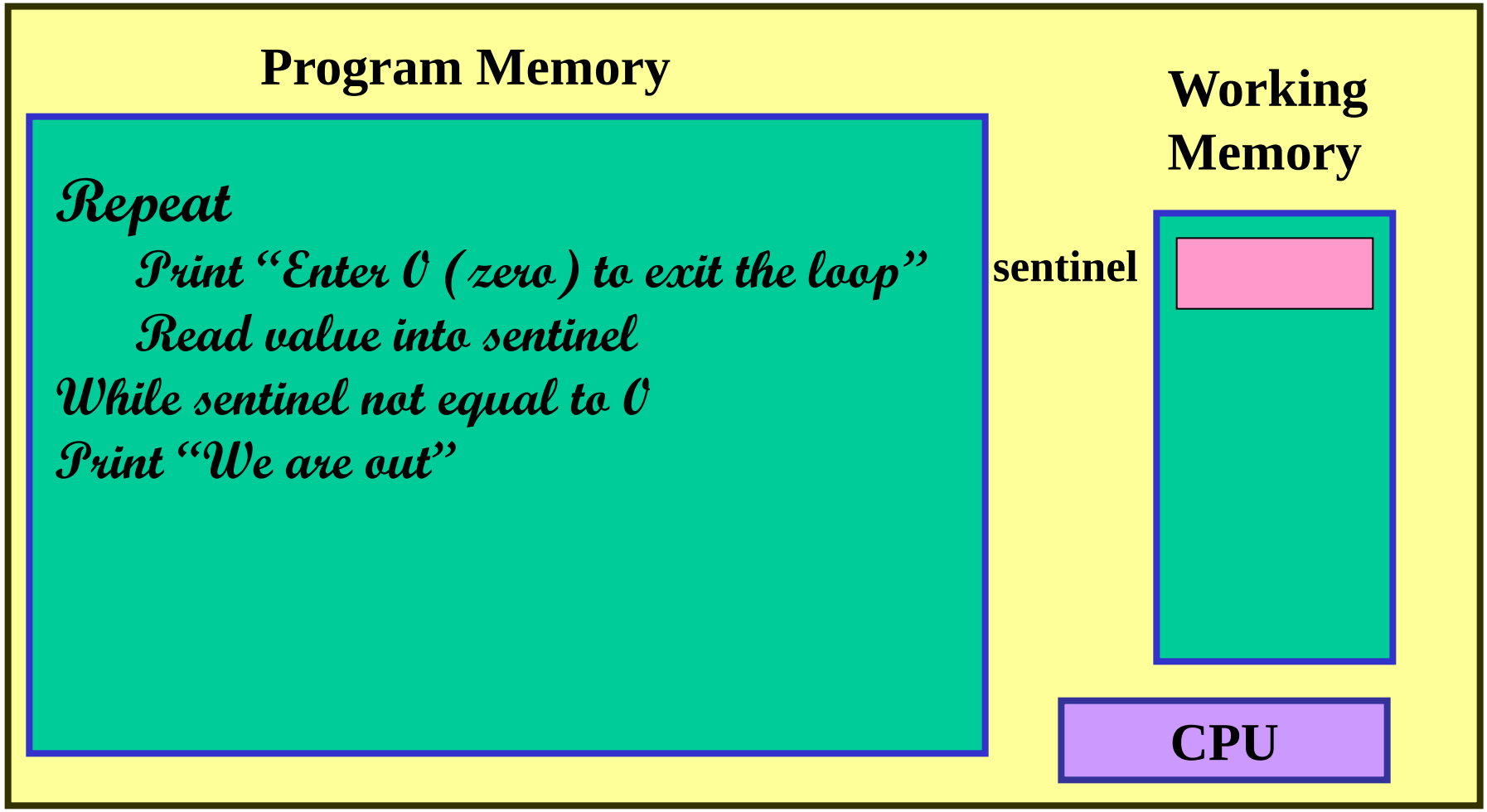


Example

- Write a program that asks the user for an integer input and exits if the input is 0.

Example of another indefinite loop

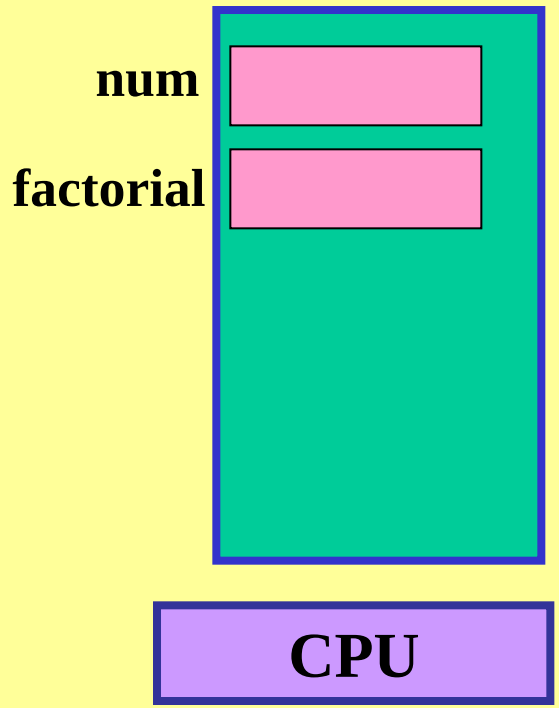
- What are the differences with the previous one?



Example of a nested loop

Program Memory

```
Print "Enter a positive integer, or -1 to quit"
Read value into num
Repeat while num not equal to -1
    Assign 1 to factorial
    Assign num to ctr
    Repeat while ctr larger or equal to 1
        Assign factorial*ctr to factorial
        Decrement ctr
    Print "The factorial of ", num, "is ", factorial
    Print "Enter positive integer, or -1 to quit"
    Read value into num
Print "You have chosen to quit"
```



Example

- Write a program that computes the factorial of an integer entered by the user.

Loop Structures in C

- There exist three repetition structures in C:
  - the `while` structure;
  - the `do/while` structure; and
  - the `for` structure.

The *while* Repetition Structure

- Allows the execution of one or more C commands while a logical expression remains true.
- Syntax for repeating a single C command:

```
while (logical expression)
    command;
```

Careful: there is no `;` here.

The C command directly below the `while ()` is executed as long as `logical expression` remains true.

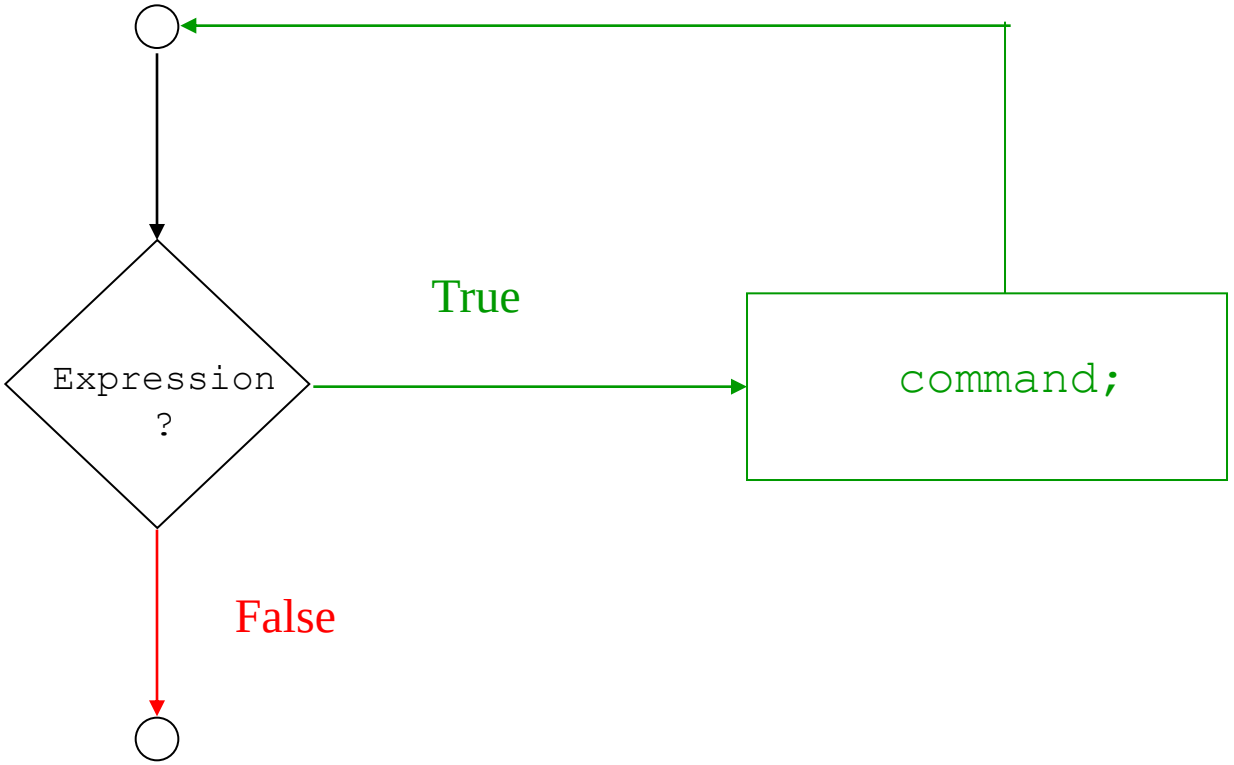
- Syntax for repeating multiple C commands:

```
while (logical expression)
{
    command 1;
    command 2;
    ⋮
    command n;
}
```

The C commands delimited by a pair of `{}` directly below the `while ()` are executed as long as `logical expression` remains true.

- The `while` structure is a pre-tested loop.
  - ➔ It is possible that the commands in the loop are never executed.

- Flowchart of the `while` structure for repeating a single command:



Components of a Loop

- Initial value of counter
- Condition (defines the final value)
- Body of statements
- Increment/decrement counter

```
int counter = 1;

While(counter<5)
   {
       printf("%d\n", counter);
       counter++;
   }
```

- A definite `while` loop is controlled by a counter, and the following are required if a definite loop is to execute correctly:
  - initialization of the counter before the loop,
  - a command in the loop that modifies the value of the counter,
  - a logical condition that tests the value of the counter against its expected final value such that the `logical expression` evaluates to false and the loop is exited.
- An indefinite `while` loop is controlled by a sentinel, and the following are required if an indefinite loop is to execute correctly:
  - initialization of the sentinel before the loop,
  - a command in the loop that modifies the sentinel,
  - a logical condition on the sentinel that eventually causes the `logical expression` to evaluate to false in order to exit the loop.
- It is very good practice to indent the instructions in the instruction block of a `while` loop in order to help visualize the structure of the program. Using `{}` to include a single command in a `while` loop is possible and can also help clarify the code.

- Example of a definite repetition loop using the `while` structure:

```
int ctr;
ctr = 1;
while (ctr <= 10)
{
      printf("Iteration number: %d\n", ctr);
      ctr++;
}
```

- Example of an indefinite repetition loop using the `while` structure:

```
int sentinel;
sentinel = 1;
while (sentinel != -1)
{
      printf("Enter -1 to exit the loop\n");
      scanf("%d", &sentinel);
}
```