# Computer Science 1: Starting Computing
# CSCI 1300

University of Colorado Boulder

# Computer Science 1: Starting Computing CSCI 1300

Dr. Ioana Fleming
Spring 2019
Lecture 13

University of Colorado
Boulder

# Reminders

Submissions:

- Project I: due Sat 2/17 at 6pm
- Homework 5: due 2/24 at 6pm

*Disabilities forms*

Readings:

- Ch. 4 - this week - loops
- Ch. 2 - strings
- Ch. 6 - next week - arrays

**Practicum I: Wed 2/20 at 5:30 pm**

*Conflict? Contact your TA*



# Chapter Four: Loops

# Chapter Goals

- To implement **while, for and do...while** loops
- To avoid infinite loops and off-by-one errors
- To understand nested loops
- To implement programs that read and process data sets
- To use a computer for simulations

# Topic 1

1. The while loop
2. Problem solving: hand-tracing
3. The for loop
4. The do loop
5. Processing input
6. Problem solving: storyboards
7. Common loop algorithms
8. Nested loops
9. Problem solving: solve a simpler problem first
10. Random numbers and simulations
11. Chapter summary

# The Three Loops in C++

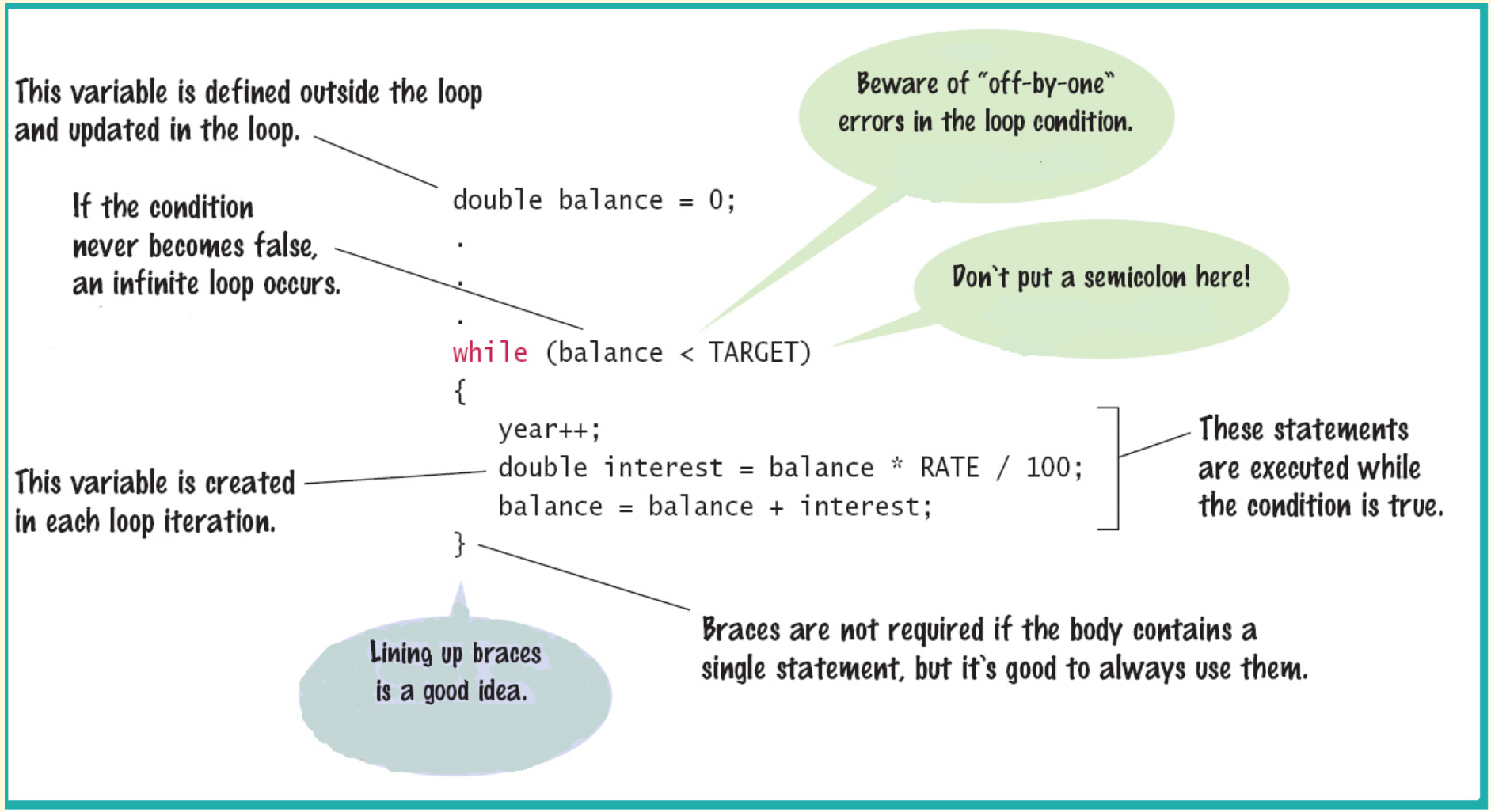
C++ has three looping statements:

**while()**

**for()**

**do {} while()**

# The `while` Statement



# while Loop Examples: Table 1

| Loop (all preceded by i=5; ) | Output | Explanation |
|---|---|---|
| while (i > 0)<br>{ cout << i << " "; i--; } | 5 4 3 2 1 | When i is 0, the loop condition is false, and the loop ends. |
| while (i > 0)<br>{ cout << i << " "; i++; } | 5 6 7 8 9 10 11<br>... | The i++ statement is an error causing an "infinite loop" (see Common Error 4.1). |
| while (i > 5)<br>{ cout << i << " "; i--; } | (No output) | The statement i > 5 is false, and the loop is never executed. |
| while (i < 0)<br>{<br>cout << i << " ";<br>i--;<br>} | (No output) | The programmer probably thought, "Stop when i is less than 0". However, the loop condition controls when the loop is executed, not when it ends (see Common Error 4.2). |
| while (i > 0);<br>{<br>cout << i << " ";<br>i--;<br>} | (No output, program does not terminate) | Note the semicolon before the {. This loop has an empty body. It runs forever, checking whether i > 0 and doing nothing in the body. |

# Example of Normal Execution

**`while`** loop to hand-trace

What is the output?

```
i = 5;
while (i > 0)
{
    cout << i << " ";
    i--;
}
```

# Example 2

- `i` is set to 5
- The `i++;` statement makes `i` get bigger and bigger
- The condition will never become `false`
- You have an *infinite loop*

```
i = 5;
while (i > 0)
{
    cout << i << " ";
    i++;
}
```

The output never ends

**5 6 7 8 9 10 11**…

# Another Programmer Error

What is the output?
How many times does the loop execute?

```
i = 5;
while (i < 0)
{
   cout << i << " ";
   i--;
}
```

# A Very Difficult Error to Find

(especially after looking for hours and hours!)

What is the output?

How many times does the loop execute?

```
i = 5;

while (i > 0);
{
   cout << i << " ";
   i--;
}
```

# A Very Difficult Error to Find

(especially after looking for hours and hours!)

That semicolon causes the **while** loop to have an "empty body" which is executed forever.

The **i** in **(i > 0)** is never changed.

There is no output!

```
i = 5;
while (i > 0);
{
   cout << i << " ";
   i--;
}
```

# Using a Loop to Solve an Investment Problem

An investment problem:

*Starting with $10,000, how many years until we have at least $20,000, at 5% interest?*

The algorithm:

1. Start with a year value of 0 and a balance of $10,000.
2. **Repeat** the following steps **while the balance is less than $20,000**:
   - Add 1 to the year value.
   - Compute the interest by multiplying the balance value by 0.05 (5 percent interest) (will be a **const**, of course).
   - Add the interest to the balance.
3. Report the final year value as the answer.

# The Complete Investment Program

sec01/doublinv.cpp

```
#include <iostream>
using namespace std;

int main()
{
   const double RATE = 5;
   const double INITIAL_BALANCE = 10000;
   const double TARGET = 2 * INITIAL_BALANCE;

   double balance = INITIAL_BALANCE;
   int year = 0;

   while (balance < TARGET)
   {
      year++;
      double interest = balance * RATE / 100;
      balance = balance + interest;
   }

   cout << "The investment doubled after "
        << year << " years." << endl;

   return 0;
}
```

Program Run

```
The investment doubled after 15 years.
```

# Program Run

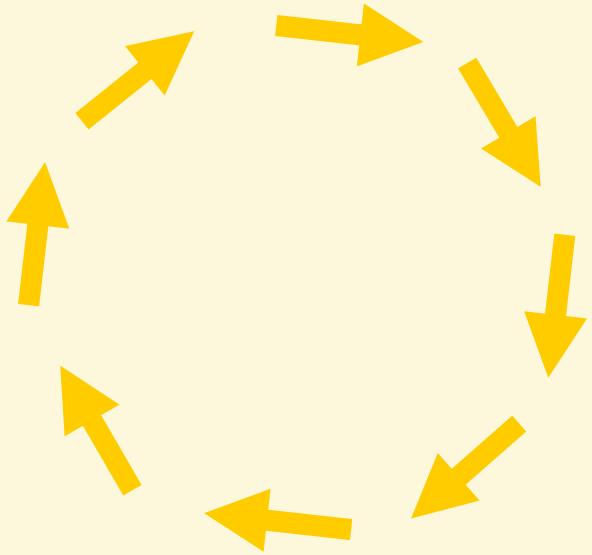
…the values are updated for 15 iterations…

…until the **`balance`** is finally(!) over $20,000 and the **`while()`** test becomes **`false`**.

| before entering while's body | | at the end of while's body | | |
|---|---|---|---|---|
| balance | year | interest | balance | year |
| 10000.00 | 0 | 500.00 | 10500.00 | 1 |
| 10500.00 | 1 | 525.00 | 11025.00 | 2 |
| 11025.00 | 2 | 551.25 | 11576.25 | 3 |
| 11576.25 | 3 | 578.81 | 12155.06 | 4 |
| 12155.06 | 4 | 607.75 | 12762.82 | 5 |
| 12762.82 | 5 | 638.14 | 13400.96 | 6 |
| 13400.96 | 6 | 670.05 | 14071.00 | 7 |
| 14071.00 | 7 | 703.55 | 14774.55 | 8 |
| 14774.55 | 8 | 738.73 | 15513.28 | 9 |
| 15513.28 | 9 | 775.66 | 16288.95 | 10 |
| 16288.95 | 10 | 814.45 | 17103.39 | 11 |
| 17103.39 | 11 | 855.17 | 17958.56 | 12 |
| 17958.56 | 12 | 897.93 | 18856.49 | 13 |
| 18856.49 | 13 | 942.82 | 19799.32 | 14 |
| 19799.32 | 14 | 989.97 | 20789.28 | 15 |
| 20789.28 | 15 | while statement is over | | |

# Common Error – Infinite Loops

- Forgetting to update the variable used in the condition is common.
- In the investment program, it might look like this:



```
year = 1;
while (year <= 20)
{
   balance = balance * (1 + RATE / 100);
}
```

- The variable **year** is not updated in the body

# Common Error – Off-by-One Errors

In the code to find when we have doubled our investment:

Do we start the variable for the years at 0 or 1 years?

Do we test for `< TARGET`

or for `<= TARGET`?

# Think to Decide!

- Start with a small example, that you can easily verify on paper
- Consider starting with \$100 and a **RATE** of 50%.
  - We want \$200 (or more).
  - At the end of the first year,
    the balance is \$150 – not done yet
  - At the end of the second year,
    the balance is \$225 – definitely over **TARGET**
    and we are done.
- We made two increments.

What must the original value be so that we end up with 2?

Zero, of course.

# Other while loop examples

- doubleinv.cpp
- doubleinv_v2.cpp
- multTable_w1.cpp
- multTable_w2.cpp
- multTable_w3.cpp
- liftoff.cpp