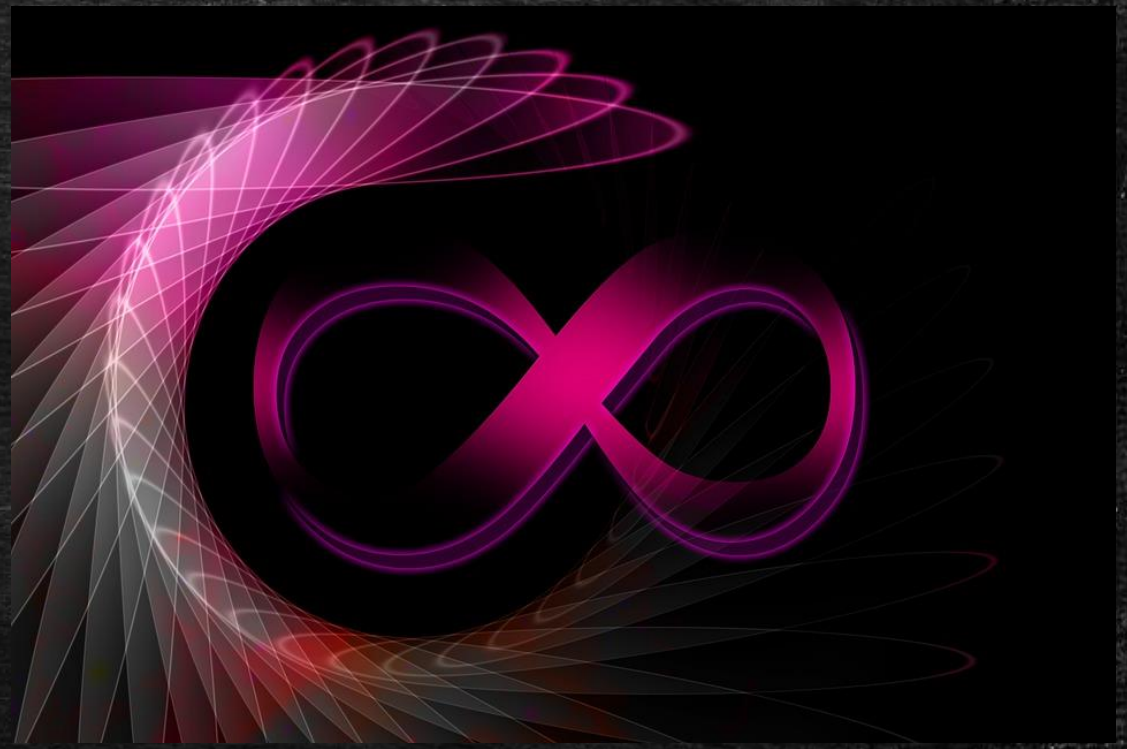
Programming with Loops



CS 150 – C++ Programming I
Lecture 6

Different Types of Loops

- C++ has four loops: `while`, `do-while`, and `for`
- Five ways to control the number of repetitions
  - Range iteration: elements of a container
  - Counter-controlled: fixed repetition
  - Sentinel: examine content of input
  - Data (aka EOF): process remaining data
  - Limit: examine result of calculation

Review: Range Loops

- C++11 introduced a new, simplified range-based *for* loop
  - Includes *string*, *vector* and the built-in array types
  - *for (type e : collection) . . .*
- Three variations
  - *for (auto e : col)* *// value*
  - *for (auto& e: col)* *// reference*
  - *for (const auto& e: col)* *// const ref*

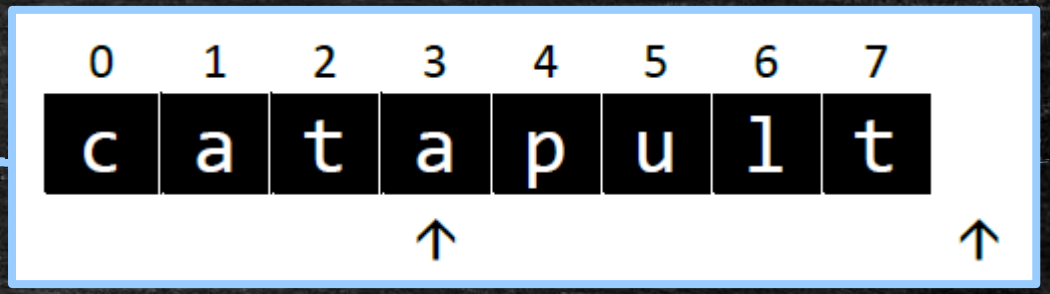
Counter-Controlled Loops

- Use a loop control variable to count number of repetitions
- Asymmetric bounds (string, array indexes)
  - `for (int i = 0; i < 10; ++i) . . .`
    - `i`, `j` common names for loop control variables
    - Initialized before loop entered; scope is loop body
    - Tested before loop entered (guarded loop)
    - Updated after loop actions are finished
- Generate sequences of data with symmetric bounds
  - `for (int i = 1; i <= 10; ++i) ..`

Strings & Counter Controlled Loops

- Memorize this pattern for processing all `char` in `string`
  - 1. Initialize both upper & lower bounds in initialization section
  - 2. Use `size_t` as the control variable type (not `int`)
  - 3. Initialize the upper bounds with `str.size()`
  - 4. Make sure control variable must never goes `< 0`
- Using the building a string idiom
  - Input is `const string&` (not modified)
  - `result` is the empty `string`
  - Concatenate each time through the loop
- Exercise: `toReverse`

Processing Substrings



- Suppose we want to count the `"cat"`s in a `string`
  - Must look at three characters at a time
- This algorithm & technique is worth memorizing
  - 1. Let `i`, `slen` be size of substring, `len` size of `str`
  - 2. Let upper bounds be `i <= len`
  - 3. Extract substring with `s.substr(i - slen, slen)`
  - 4. Use conditionals to examine `subs`
- Exercise: `Count`

Indefinite Loops

- The simplest loop (syntactically) uses *while*
  - *while (condition) { statements to repeat }*
- So, how many times will this loop repeat?

```
char c = randChar(); // returns random char
while (c != 'Q')
{
   cout << c;
   c = randChar();
}
```

- You can't tell. That's why it's called an indefinite loop

"Primed" Sentinel Loops

- A sentinel loop stops when it reads a special value from input (the sentinel)
- The primed loop is one technique for writing a sentinel loop
  - Named after the old-fashioned hand pumps that had to be primed before they could suction water
  -
```
read a value into a variable
while the variable is not the sentinel
    process the variable
    read the next value
```

Flag-Controlled Sentinel Loop

- Instead of two read statements you can use a Boolean flag to signal if you have found the sentinel

```
bool finished{false}; // control flag
while not finished
    read the value
    if value is the sentinel
        set finished to true
    else
        process the variable
```

The Loop-and-a-Half Idiom

- Normally a loop has only one exit (at the loop condition)
  - Some languages also allow you to exit from inside a loop
  - Ada does this with its *exit when* construct
- In C++ you can do the same thing with *if* and *break*
  -
```
while there is more to process
    read the value
    if value is the sentinel then break
    process the variable
```
  - This called the loop-and-a-half idiom
  - Loop Exits and Structured Programming