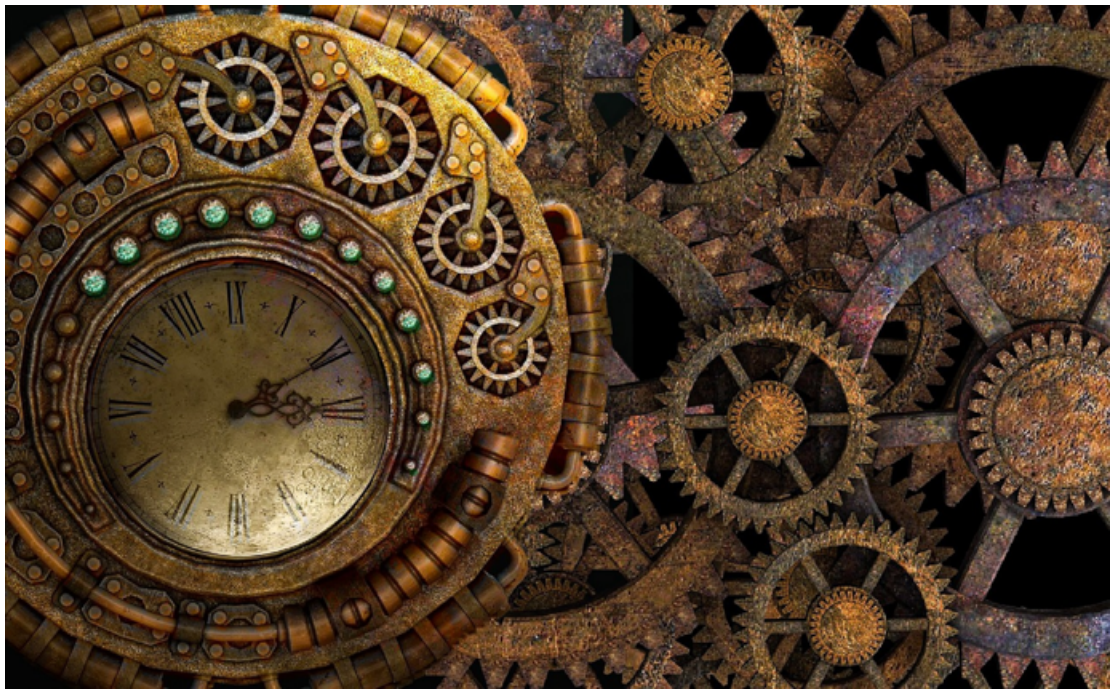
CS 161
Introduction to CS I
Lecture 6

- How can we represent words?
- Repeating the same behavior using loops
- How can we track down bugs in our programs?



Warning!

- A very sad tale – has this happened to you?
  - `g++ -o assignment2.cpp assignment2`
- Workarounds
  - Use a very different executable name (like `my_program`)
    - `g++ -o my_program assignment2.cpp`
    - Then this mistake would not overwrite your C++ file:
    - `g++ -o my_program.cpp assignment2`
    - (As long as `assignment2` does not exist)
  - Inside vim, use <ctrl>-Z (in command mode) to "suspend" it
    - Then compile and run your program
    - If you accidentally overwrite your .cpp, go back into vim (with "`fg`") and re-save it

Also…

- Do not sign up for multiple demo slots (one per assignment)
- If you will miss a lab, attend another lab that week to complete the activities and receive credit
- If you submit a Revision Plan for Assignment 1, you must email the TA who graded your assignment to have your Revision Plan considered (else nothing happens)
- Questions about Assignment 2?

Finally… letters and words!

- Letters/symbols: `char`
- Words: `string`
  - This is not a C++ primitive type, but instead a special class of objects
- Escape sequences
  - \n: newline
  - \t: tab
  - \”: double quote
  - More: https://en.cppreference.com/w/cpp/language/escape

```
string favorite_color = "red";

/* Query the user */
cout << "What is your favorite color? " << endl;
cin >> favorite_color;

cout << "You like " << favorite_color << endl;

/* You can use \n instead of endl */
cout << "You like " << favorite_color << "\n";
cout << "Longer code" << endl;
cout << "Shorter code\n";
```

String operations: concatenation

```
string favorite_color = "red";

/* Query the user */
cout << "What is your favorite color? " << endl;
cin >> favorite_color;
cout << "You like " << favorite_color << endl;

/* String concatenation, version 1 */
string two_colors = favorite_color + " and pink";
cout << "How about " << two_colors << "?" << endl;

/* String concatenation, version 2 */
favorite_color += " and green";
cout << "How about " << favorite_color << "?" << endl;
```

vs.

Decision making with random chance

```
int dice_roll = 0;

/* Roll the dice */
dice_roll = rand()%6 + 1;
cout << "You rolled " << dice_roll << endl;
if (dice_roll < 3)
  {
    cout << "You lose..." << endl;
  }
else /* Rolled 3 or higher */
  {
    cout << "You win!" << endl;
  }
```

Seeding the random number generator

```
#include <iostream>
#include <cstdlib> /* include to allow rand() to be used */
#include <ctime>   /* include to allow time() to be used */

using namespace std;

int main()
{
  int dice_roll = 0;

  /* Seed the generator with the current time,
   * so it's different each time */
  srand(time(NULL));

  /* Roll the dice a few times */
  dice_roll = rand()%6 + 1;
  cout << "You rolled " << dice_roll << endl;
```

Only once

Switch statement

- Improves readability when there are many choices (**menus**)

Must be an expression

If/then/else if/then/else

```
if ((user_feeling == 'y') ||
    (user_feeling == 'Y'))
  {
    cout << "Today is a great day!" << endl;
  }
else if ((user_feeling == 'n') ||
         (user_feeling == 'N'))
  {
    cout << "Today is not going well." << endl;
  }
else /* user_feeling is not 'y'/'Y' or 'n'/'N' */
  {
    cout << "Invalid choice." << endl;
  }
```

Can only check equality

Switch

```
switch (user_feeling)
  {
  case 'y': /* fall through to case 'Y' */
  case 'Y':
    cout << "Today is a great day!" << endl;
    break;
  case 'n': /* fall through to case 'N' */
  case 'N':
    cout << "Today is not going well." << endl;
    break;
  default: /* user_feeling is not 'y'/'Y' or 'n'/'N' */
    cout << "Invalid choice." << endl;
  }
```

If-then "short-circuiting"

- What does this evaluate to?
  - `(3 > 4) && (27 > 3)`
  - Computer: `(3 > 4)` is **`false`** so the entire expression is **`false`**. Stop here!
- Likewise:
  - `(3 < 4) || (27 < 3)`
  - Computer: `(3 < 4)` is **`true`** so the entire expression is **`true`**. Stop here!

Loops

- `if-then` and `switch` enable **selection** between outcomes
- We will now move on to **repetition**

- **For loop:** repeat statements a fixed number of times

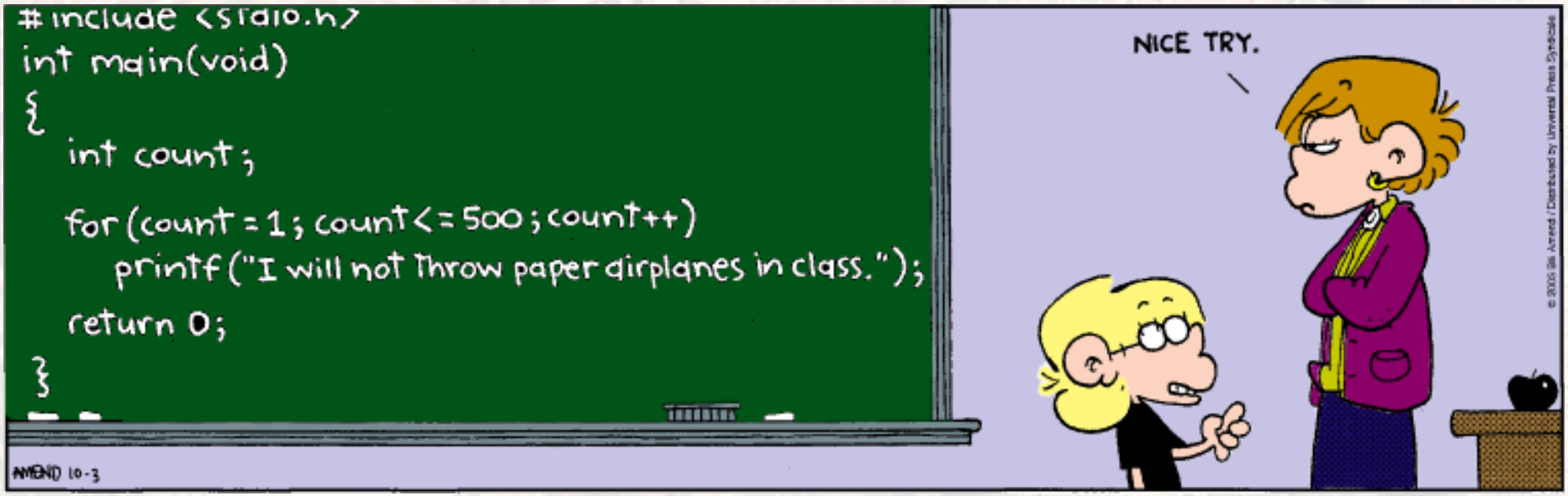
Oregon State University
College of Engineering

Common `for` loop patterns

```
for (<var> = low;  <var> <= high; <var>++)
{
  <statement>;
  …
}
for (<var> = high; <var> >= low;  <var>--)
{
  <statement>;
  …
}
```

Oregon State University
College of Engineering

For loops save programming effort!



Oregon State University
College of Engineering

Predict the output

A

```
int x;
for (x = 0; x <= 100; x++)
   cout << "I will not throw paper airplanes\n";
```

B

```
for (x = 0; x < 100; x++)
   cout << "I will not throw paper airplanes\n";
```

C

```
for (x = -100; x <= -1; x++)
   cout << "I will not throw paper airplanes\n";
```

D

```
for (x = 100; x >= -100; x--)
   cout << "I will not throw paper airplanes\n";
```

Predict the output (more challenging)

E

```
int x;
for (x = 0; x > 100; x++)
   cout << "I will not throw paper airplanes\n";
```

F

```
for (x = 0; x <= 10; x += 5)
   cout << "I will not throw paper airplanes\n";
```

G

```
for (x = -10; x <= -1; x = x / 2)
   cout << "I will not throw paper airplanes\n";
```

H

```
for (x = 0; x < 100; y++)
   cout << "I will not throw paper airplanes\n";
```

Oregon State University
College of Engineering

For loops: easy automation

Change to 10, or 100, or 10000000

```
/* Roll the dice a few times */
for (int x = 0; x < 3; x++)
  {
    dice_roll = rand()%6 + 1;
    cout << "You rolled " << dice_roll << endl;
  }
```

What if we don't know how many times in advance?

- **`While` loop:** repeat statements while test is true

```
/* Roll the dice */
dice_roll = rand()%6 + 1;
cout << "You rolled " << dice_roll << endl;

/* Keep rolling until you get a 3 */
while (dice_roll != 3)
  {
    dice_roll = rand()%6 + 1;
    cout << "You rolled " << dice_roll << endl;
  }
cout << "You win!" << endl;
```

A do-while loop can reduce duplicated code

while loop

```
/* Roll the dice */
dice_roll = rand()%6 + 1;
cout << "You rolled " << dice_roll << endl;

/* Keep rolling until you get a 3 */
while (dice_roll != 3)
  {
    dice_roll = rand()%6 + 1;
    cout << "You rolled " << dice_roll << endl;
  }
cout << "You win!" << endl;
```

Test before

do-while loop

```
/* Keep rolling until you get a 3 */
do
  {
    dice_roll = rand()%6 + 1;
    cout << "You rolled " << dice_roll << endl;
  } while (dice_roll != 3);
cout << "You win!" << endl;
```

Test after

Semi-colon required!

Challenge: Re-write this for loop as a while loop

```
/* Roll the dice a few times */
for (int x = 0; x < 3; x++)
  {
    dice_roll = rand()%6 + 1;
    cout << "You rolled " << dice_roll << endl;
  }
```

What vocabulary did we learn today?

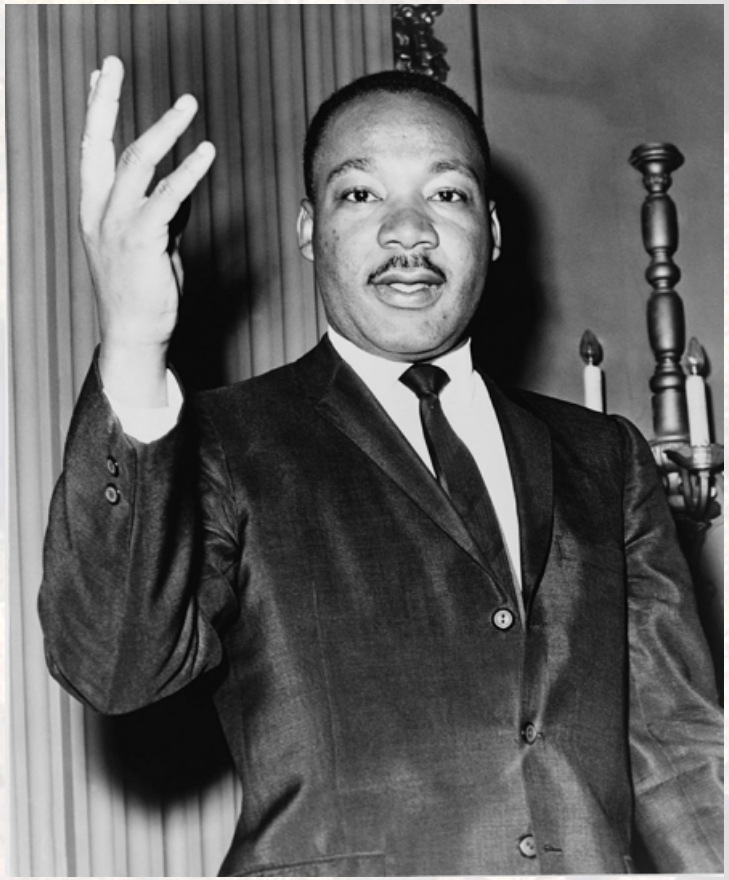
- String
  - Concatenation
- Random number generator
  - Seed (the random number generator)
- Conditional statement: `switch`
- Short-circuit for `if/then`
- Loops: `for, while, do/while`
  - Loop counter

What ideas and skills did we learn today?

- Seed the generator only once, unless you want to get the same sequence of numbers
- Control structures
  - Selection
    - `if-then`
    - `switch`
  - Repetition – shorten the code
    - `for`
    - `while`
    - `do-while`

MLK Jr. Day Extra Credit

- Write a profile of a **non-Caucasian pioneer** in Computer Science, to include:
  - Biographical sketch
  - Contributions to CS
  - References/sources
- Well written profiles will earn extra credit towards Midterm 1
- See Canvas for more details



Week 2 nearly done!

- ❑ Attend lab (laptop required)
- ❑ Read **Rao Lesson 6** (pp. 128-142) – loops
  Rao pp. 79-81 - strings
- ❑ Finish **your Assignment 2 design** (due **Sunday, Jan. 19**)

No class Monday! See you on **Wednesday**!