Lecture 9:The While Loop + Math methods

AP Computer Science Principles

`while` loops

The `while` loop

- **`while` loop**: Repeatedly executes its body as long as a logical test is true.

```
while (test) {
    statement(s);
}
```

- Example:

```
int num = 1;                          // initialization
while (num <= 200) {                      // test
    print(num + " ");
    num = num * 2;                        // update
}
// output:  1 2 4 8 16 32 64 128
```

Example `while` loop

```
// finds the first factor of 91, other than 1
int n = 91;
int factor = 2;
while (n % factor != 0) {
    factor++;
}
println("First factor is " + factor);

// output:  First factor is 7
```

- `while` is better than `for` because we don't know how many times we will need to increment to find the factor.

Curly braces {}

- Curly braces mark the body of methods, for/while loops and conditional blocks. They are not necessary if the body or the block consists of only one statement.

- The following are equivalent.

```
for (int i = 5; i <= 25; i+=1) {
       print(i + ” “);
}
```

```
for (int i = 5; i <= 25; i+=1)
       print(i + ” “);
```

Curly braces {}

- The following are also equivalent.

```
if (x <= 1) {
        print(x + ” “);
}
```

```
if (x <= 1)
        print(x + ” “);
```

Curly braces {}

- The following are NOT equivalent.

```
int sum = 0;
for (int i = 1; i <= 10; i+=1) {
      print(i + ” “);
      sum += i;
}

int sum = 0;
for (int i = 5; i <= 25; i+=1)
      print(i + ” “);
      sum += i;    //error, why?
```

An example

```
int count = 10;

while(count > 9)
{
  count++;
}
```

– Infinite loop!!

Another Example

```
int count = 10;

while(count < 13)
  println(count + " ");
  count++;
```

– Still infinite loop!!

One More

```
int count=10;

while(count<13)
{
  println(count + " ");
  count++;
}
```

– Correct!

float vs double

- **float**: 32-bit data type representing real numbers to about 7 decimal places.

- **double:** 64-bit data type representing real numbers to about 16 decimal places.

Since Processing is graphics intensive, it is recommended that floats are used instead of doubles. In fact, float is the default type for decimal numbers in Processing. All of the built-in math functions in Processing returns floats. The math functions from the standard Math library in Java returns doubles.

```
double a = Math.sqrt(9); // a = 3.0 but is a double
float b = sqrt(9); // b = 3.0 but is a float
```

Processing's `Math` methods

Method name	Description
`abs(`*value*`)`	absolute value(int or float)
`ceil(`*value*`)`	rounds up(int)
`exp(`*value*`)`	exponential, base e
`log(`*value*`)`	logarithm, base e
`map(x,x1,x2,y1,y2)`	linear map x from [x1,x2] to [y1,y2]
`pow(`*base*`, `*exp*`)`	*base* to the *exp* power(float)
`random(`*value*`)`	random `float` from [0,value)
`random(`*value1*`, `*value2*`)`	random `float` from [value1,value2)
`round(`*value*`)`	nearest whole number(int)
`sqrt(`*value*`)`	square root
`sin(`*value*`)`, `cos(`*value*`)` `tan(`*value*`)`	sine/cosine/tangent of an angle in radians
`atan(value)` `atan2(dy,dx)`	inverse tangent (-PI/2, PI/2) inverse tangent (-PI, PI)
`degrees(`*value*`)` `radians(`*value*`)`	convert degrees to radians and back

Constant	Description
`PI`	3.1415926...

Calling `Math` methods

- Examples:

```
float squareRoot = sqrt(121.0);
println(squareRoot);      // 11.0

int absoluteValue = abs(-50);
println(absoluteValue);          // 50

println(pow(2,-3));    // 0.125
```

- The `Math` methods do not print to the console.
  - Each method produces ("returns") a numeric result.
  - The results are used as expressions (printed, stored, etc.).

Type casting(Processing)

- **type cast**: A conversion from one type to another.
  - To promote an `int` into a `float` to get exact division from `/`
  - To truncate a `float` from a real number to an integer

Syntax:

float(expression) or int(expression)

```
float result = float(19) / 5;      // 3.8
int result2 = int(result);           // 3
int x = int(pow(10, 3));         // 1000
```

Note: This casting method is unique to Processing and only works with float and int.

Random Numbers

Random numbers

- random(value) produces a random float from 0(inclusive) to value exclusive.
  - `float x = random(5); // 0.0 <= x < 5.0`
  - `float y = 3 * random(4); // 0.0 <= y < 12.0`
  - `float z = random(4)+ 2; // 2.0 <= z < 6.0`

- random(value1, value2) produces a random float from value1(inclusive) to value2 exclusive.
  - `float x = random(5,10); // 5.0 <= x < 10.0`
  - `float y = random(-3,-1); // -3.0 <= y < -1.0`

Random Integers

How do we generate random integers?

```
int x = int(random(5));
// random integer 0 to 4 inclusive.
int y = int(random(3,9));
// random integer 3 to 8 inclusive.
int z = int(random(0,2));
// random integer 0 or 1
// useful for heads/tails
```

Categories of loops

- **definite loop**: Executes a known number of times.
  - The `for` loops we have seen are definite loops.
    - Print "hello" 10 times.
    - Find all the prime numbers up to an integer *n*.
    - Print each odd number between 5 and 127.

- **indefinite loop**: One where the number of times its body repeats is not known in advance.
  - The `while` loops are used for indefinite loops.
    - Prompt the user until they type a non-negative number.
    - Print random numbers until a prime number is printed.
    - Repeat until the user has types "q" to quit.

Example: Roll A Die

Write a method `rollDie` that simulates a die rolling experiment. The method repeatedly rolls a die and display the output, until there are a total of four aces (1). The method then prints out a success message that includes the total number of rolls.

```
rollDie();
```

1 2 2 5 1 6 4 1 2 1

Got 4 aces in 10 rolls!

```
rollDie();
```

1 1 1 5 1

Got 4 aces in 5 rolls!

Answer to rollDie

```
void rollDie(){
  int roll = 0;
  int aces = 0;

  while(aces < 4){
    int die = int(random(1,7)); //1 to 6
    if(die == 1)
      aces++;
    roll++;
    print(die + " ");
  }
  //found 4 aces
  println();
  println("Got 4 aces in " + roll + " rolls.");
}
```

Lab 1

Go to my codingbat homepage at(BOOKMARK THIS PAGE):

https://codingbat.com/home/lnguyen8@bostonpublicschools.org

And do all the problems from the ”while” page. This problems can be done with a for loop, but YOU MUST USE A WHILE LOOP!

Remember to login to save your work!

Lab 2

Write a method `tossCoin` that accepts an integer input `n` and simulates a coin toss experiment. The method repeatedly tosses a coin and display the output, H for heads and T for tails, until there are a total of `n` heads. The method then prints out a success message that includes the total number of toss.

```
tossCoin(4);
```

H T T H H T H
Got 4 heads in 7 tosses!

```
tossCoin(6);
```

H T H H T T T H H T H
Got 6 heads in 11 tosses!

Lab 2

Write a method `tossFour` that simulates a coin toss experiment. The method repeatedly tosses a coin and display the output, H for heads and T for tails, until there are a total of FOUR heads **in a row**. The method then prints out a success message that includes the total number of toss.

```
tossFour();
```

H T T H H T H H H H
Got 4 heads in a row in 10 tosses!

```
tossFour();
```

H H T H H H H

Got 4 heads in a row in 7 tosses!

Lab 2 Outline

Write the setup() method to test your methods.

```
void setup(){

}
void tossCoin(int n)
{

}
void tossFour()
{

}
```

Lab 3

In the lab from the previous lecture, you drew horizontal and vertical lines to form a grid.

Make a copy of that lab and modify it to use while loops instead of for loops.

Homework

1) Read and reread these lecture notes.

2) Complete the problem set on While Loops.

3) Complete the labs.

References

Part of this lecture is taken from the following book.

1) Stuart Reges and Marty Stepp. Building Java Programs: A Back to Basics Approach. Pearson Education. 2008.