APS 105 Lecture 6 Notes

Last lecture: math library and random number generation

Today: random number generation between a range and if statements.

Recall:

```
0 % 5 = 0
1 % 5 = 1
2 % 5 = 2
3 % 5 = 3
4 % 5 = 4
5 % 5 = 0
```

repeats again!

Therefore, %5 on any number will return a number between 0 & 4.

and %j on any other number will produce a number between 0 & j-1

To produce random # between 0 and 5, we do this

```
rand() % 6
```

To produce a random # between 1 and 6 we do this

```
rand() % 6 + 1
```

To produce a random # between [MIN, MAX]

```
rand() % (MAX - MIN + 1) + MIN
```

Decision-making with if-statements

A key capability of a computer is to take decisions based on condition

if condition is true → do something

if condition not true → do something else

In C, general form of if statements:

```
if ( condition )
        statement to evaluate if condition is true;
else
        statement to evaluate if condition is false;
```

condition must be evaluated to either true or false.

1] Recall a bool variable takes either true or false

2] Also, if condition was int 1 it means true and 0 means false (or any non-zero number)

Note:

```
bool x = true;
bool y = false;

printf("%d %d", x, y); // prints 1 0
```

3] condition can be "relational expression" that is evaluated as true or false

E.g. x > 0 , height >= width

List of relational operators, <, <=, >=, >, ==, !=

E.g.
```
if (height == width)
        printf("This is a square \n");
else
        printf("This is a rectangle \n");
```

What can you do with "relational operators"?

(i) compare int to double, e.g. (7.0 >= 2)

(ii) mix arithmetic and relational operators, but arithmetic operators have higher precedence, e.g.

x <= y + 5

① y + 5

② x <= (y + 5)

(iii) compare characters (or the ASCII code of characters)

' ' < '0' < '1' < ..... '9' < 'A' < 'B' < .... < 'Z' < 'a' < 'b' < .... 'z'

i.e. ASCII code of all upper case letters is smaller than ASCII of small case letters.

(iv) can compare char to int

e.g. '0' == 0 evaluates to false

(v) CANNOT DO 68 <= x <= 74

↓

2 relational operators on 1 variable is not permitted

↓

we need "logical operators" or "boolean operators"

These are AND &&, OR ||, NOT !

So 68 <= x <= 74 translates to 2 expressions between a boolean operator, ((x >= 68) && (x <= 74))

or (x >= 68 && x <= 74) as logical operators have lower precedence compared to relational.

E.g. Check if letterA is either 'A' or 'a' and set isletterA to true

-DEMO-

```
char letterA = 'A';
bool isletterA = false;
if ( letterA == 'A' || letterA == 'a' ) {
    printf("%c is letter a", letterA);
    isletterA = true;
}
```

① If we have more than 2 statements, need to contain them in {}

② no need for else statement if you don't want

E.g. !

```
if (done == false)  <=>  if (!done)
```

(done: bool variable)

Lazy evaluation

For example, I want to evaluate if x/y is <10 but before that, I want to make sure y is not 0.

Option 1:

```
if (y != 0) {
    if (x / y < 10) {
    }
}
```

Option 2 (elegant & neat):

```
if ( y != 0  &&  x / y < 10 )
     (1)         (2)
```

in this case, 1st y!=0 is checked, 2nd && is observed

if y!=0 is false, x/y <10 will not be evaluated, since result of && is false automatically, when 1 argument is false

The rule:

<LHS> || <RHS> : <RHS> will NOT be evaluated if <LHS> is true

<LHS> && <RHS> : <RHS> will NOT be evaluated if <LHS> is false

Roulette Example - DEMO -

```
// User needs to guess that a random# between 1 and
// 36 will be even or odd. If the user guess
// correctly, they win, else they loose.
#include <time.h>  -> to use time() function
#include <stdio.h>  -> to use printf and scanf
#include <stdlib.h>  -> to use rand() and srand()
#include <stdbool.h>  -> to use bool, true, false
int main (void) {

    int guess;

    srand(time (NULL));

    while (true) {

        printf("Enter your bet, 0 for even and
                1 for odd: ");
        scanf("%d", &guess);

        int spin = rand() % 36 + 1;

        if (spin % 2 == guess)
                printf ("You win!");
        else
                printf("You lose :(");

    }
    return 0;
}
```