comp10002 Week 3 Workshop: welcome back!

Last Week	In the lectures: loops, loops, and … looping.
Today	`while (current_time < 3.20) chat();` Discussions → Quiz → exercises together → group/individual work
Coming soon	• Next workshop in one week ☺ • Online Quiz 1: Thursday Week 5, 2:15PM

- *In my first driving lesson, my instructor asked me to loop around this roundabout for 20 times ☺*
- *And mine made me to do that until we ran out of petrol ☺*

B a C Pro!

Compare:

`a= 5;` `b= 5;`	`a= (b= 5);` // `b=5` is also an expression! `a= b = 5;` // `=` evaluated right to left

`a= a * b;`
`a= a+b;`
`n= n+1;`
`m= m-1;`
	`a *= b;`
	`a += b;`
	`n++;  n += 1;`
	`m--;`

`scanf("%d%d", &a, &b);`
//rest of the program
 	
`//` a loop for reading and processing a number of pairs `(a,b)`
	`if (scanf(%d%d",&a,&b) != 2) {`
	`  printf("invalid input\n");`
	`  exit(EXIT_FAILURE);`
	`}`
	// rest of the program, OR:
	`while (scanf(%d%d",&a,&b) == 2) {`
	`  //` do something with the new value of `a` and `b`
	`}`
	…

The `while` loop, the `for` loop.

`while` loop: It's just the same as Python's `while`!

Also, `break` and `continue` are just the same.

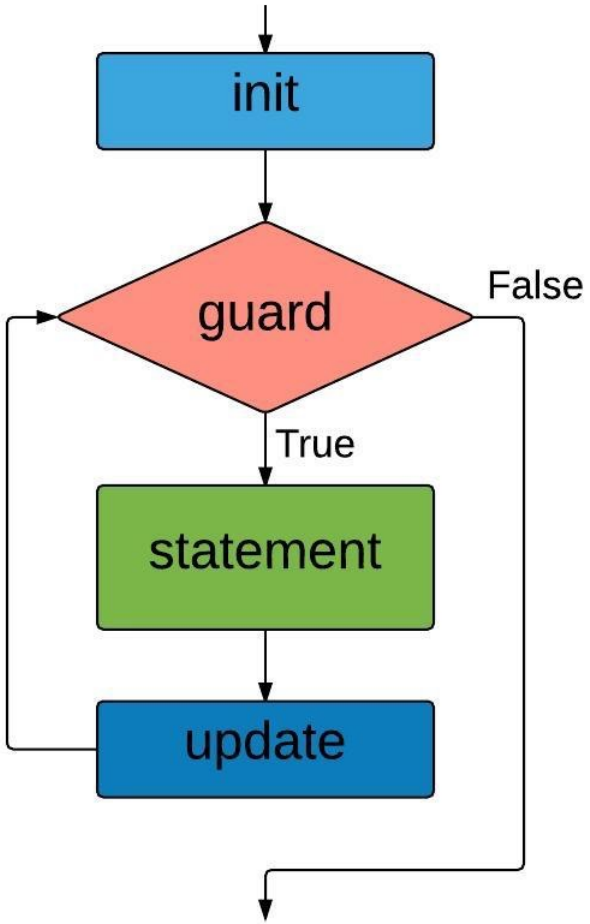
So, we have enough tools to implement any algorithms…

The `for` loop is different from that of Python!

It's an extended version of the `while` loop.

The `for` loop

```
for (init ; guard ; update ) {
    statement;
}
```




```
int s=0, i;
for ( i=0; i<10 ; i++ ) {
    s= s + i;
}
printf("i= %d, sum= %d\n",
        i, s);
```

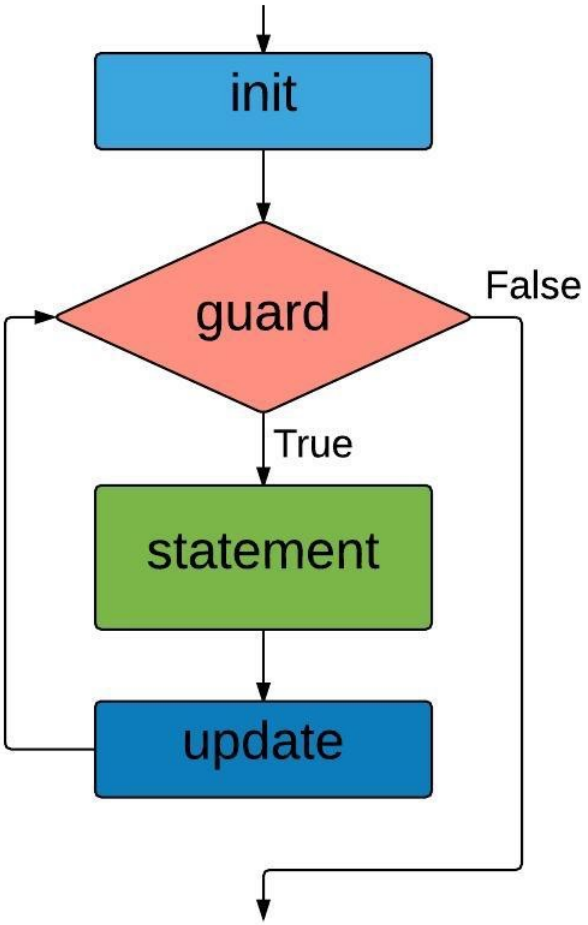
***Note**: Any of the components can be **empty**. The empty guard is equivalent to 1 (TRUE).*

The `for` and `while` loops

Compare:

```
for (init ; guard ; update) {
    statement;
}
```

```
init ;
while ( guard ) {
    statement;
    update ;
}
```



Note: *the* guard in `while` *cannot be* ***empty***.

`for (;;)` *is equivalent to* `while()`

Discussion on grok Playground: loops, `while` and `for`

Outlook:

- loops: from `Python`'s `while` → `C`'s `while` → `for`
- `break` & `continue`
- Exercise 4.1 (`grok.W03`)

When training on `grok's Playground,` write code segment to compute:

- $S = 1^2 + 2^2 + \ldots + 100^2$
- $1 + 1/2! + 1/3! + \ldots + 1/k!$

where k is not given, and the sum should include only elements i such as $1/i! \geq 10^{-6}$

- $1 + x + x^2/2! + x^3/3! + \ldots + x^k/k!$

where x is given, k is not given, and the computation should stop *after adding* the first member that satisfies $|x^i/i!| < 10^{-6}$

Understanding the for loop: Exercise 4.1 on grok.

Sample question: Trace the action of the loop, and determine the values printed out by the `printf` *statement. Assume that all variables have been declared to be of type* `int`

```
1   for  (i=0; i<20; i= i+3) {
2      printf ("%2d\n", i);
3   }
```

Exercise 4.2

Give a general construction that shows how any `do` *statement can be converted into an equivalent* `while` *statement.*

Quiz 1

What **XXX** *should be in the following fragment:*

```
printf("Enter value for a and b : ");
if ( scanf("%d%d",&a,&b) XXX  ) {
  printf("Please enter 2 integers\n");
  exit( EXIT_FAILURE );
}
```

A: **!= 0**	**B:** **!= 2**
C: **== 1**	**D:** **== 2**

E: Nothing is possible, as the fragment has errors somewhere

Quiz 2

What **XXX** *should be in the following fragment:*

```
int n, sum= 0;
printf("Enter a sequence of integers: ");
while ( scanf("%d",&n) XXX ) {
  sum += n;
}
printf("Sum of numbers is %d\n", sum);
```

A: **!= 0**	**B:** **!= 1**
C: **== 1**	**D:** **!= EOF**

E: Nothing is possible, as the fragment has errors somewhere

Quiz 3

What are the values of `i`, `s`, and `c` after the following statements:

```
int s=0, i, c=0;
for (i=0; i<3; i++) {
   s += i;
   c++;
}
```

A	`s= 10, i= 5, c= 5`
B	`s= 10, i= 5, c= 4`
C	`s= 15, i= 6, c= 6`
D	`none of the above, and no syntax error`
E	`the fragment has some syntax errors`

Quiz 4

Q3: How many lines and numbers are printed by the following segment:

```
int i,j;
for (i=0; i<10; i++) {
  if (i==3) break;
  for (j=0; j<3; j++) {
    printf("%d ", i*j);
  }
  printf("\n");
}
```

A	9 lines, 9 numbers
B	10 lines, 30 numbers
C	9 lines, 27 numbers
D	3 lines, 9 numbers
E	none of above, or syntax error

Quiz 5

Supposing that all variables are pre-declared as `int`.

Which fragment compute `s=` $1^2 + 2^2 + \ldots + n^2$?

A	`s= 1*1 + 2*2 + … + n*n  X`
B	`for (i=1; i<=n; i++) s += i*i;`
C	`for (s=0; n > 0; n--) s += n*n;`
D	`for (i=1, s=0; i<=n; i++) s = s + i^2;` x
E	`for (i=1, s=0; i<=n; i++) s += i**2;` x

4.5 – Design (Discussion, view exercise in `grok.W03`)

Design and implement a program `grapher.c` *that reads integers and draw a simple graph. Assume that all of the values read are between 1 and 70. Example:*

```
H: grapher
Enter integers between 1 and 70 inclusive: 3 7 11
  3 |***
  7 |*******
 11 |***********
```

4.6, 4.7 – Design (Discussion, view exercise in `grok.W03`)

Design a program `my_wc` *that count the number of characters, words, and lines in the input. Example of execution:*

```
H: my_wc
Enter text:
Mary      has a      little lamb,
Little lamb, 123 little            lamb;
^D    (or ^Z if using MinGW/Windows)

Lines: 2
Words: = 10
Chars: ??
```

Understand the problem first! Make clear:

- What is a character, how to get it?
- What is a line, how to recognize it?
- What is a word, how to recognize it?

Lab (minimal requirements): Implement 4.5, 4.6+4.7

4.5: *Design a program* `grapher.c` *that reads integers and draw a simple graph. Graph example:*

```
  3 |***
 11 |***********
```

4.6+4.7: *Design a program* `my_wc` *that count the number of characters, words, and lines in the input. Use* `mary.txt` *to test your program, that is:*

```
bash $ my_wc < mary.txt
```

But first, of course, create file `mary.txt` (you can copy the file content from `grok.W03.Exercise 4.6`).

Wrap-Up

Today's topics:

- loops (the `while` loop is a special case of the `for` loop),
- `scanf`: how to read
  - a single datum, or a single tuple of data
  - a series of numbers, or a series of data tuples

Minimal Implementation:

- Exercise 4.5 (grok W03): Simple character graph
- Exercise 4.6 (grok W03): Character and line counting
- Exercise 4.7 (grok W03): Counting characters, words, and lines

Extra Implementation in grok W3X

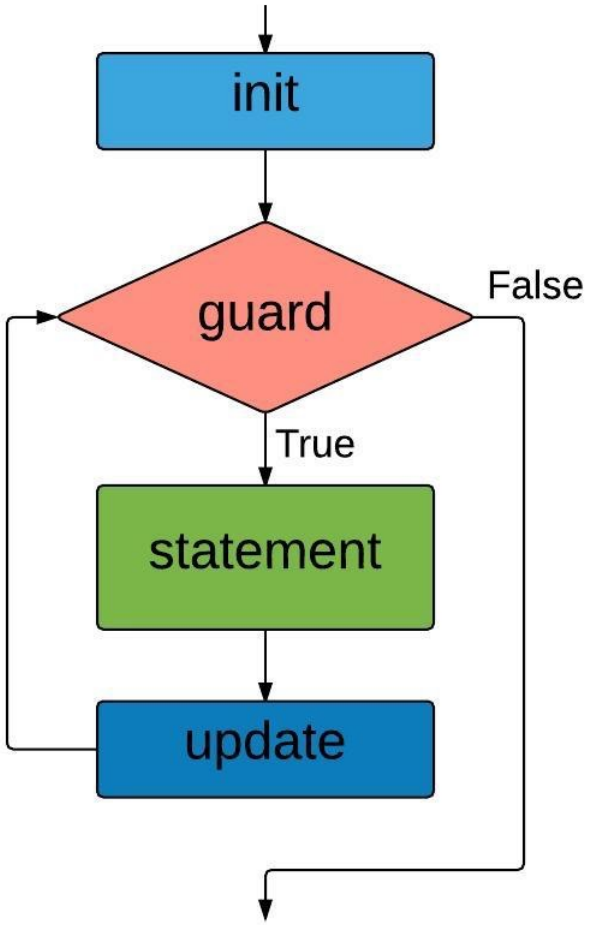
- Exercise 4.9: Computing the next prime number
- Exercise 4.4: Printing (a part of) the ASCII table
- Exercise 4.3: Computing Fibonacci numbers
- Exercise 4.8: Extending the 3n problem

Additional slides

The `for` loop

```
for (init ; guard ; update) {
    statement;
}
```

```
int s=0, i;
for ( i=0; i<10 ; i++ ) {
    s= s + i;
}
printf("i= %d, sum= %d\n",
        i, s);
```



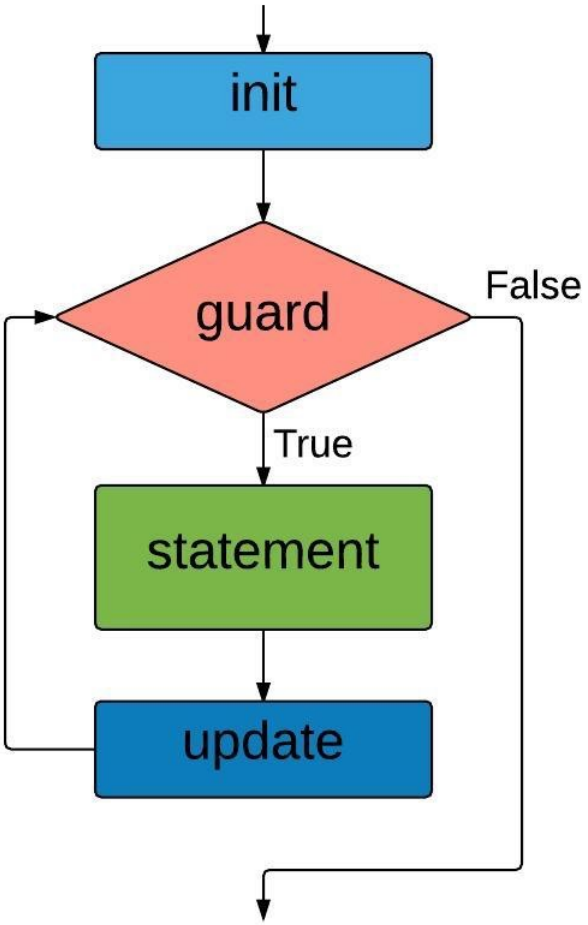
***Note**: Any of the components can be **empty**. The empty guard is equivalent to 1 (TRUE).*

The `for` and `while` loops

Compare:

```
for (init ; guard ; update) {
    statement;
}
```

```
init ;
while ( guard ) {
    statement;
    update ;
}
```



Note: *the* guard in `while` *cannot be* ***empty***.

`for (;;)` *is equivalent to* `while(1)`

Lab: Implement 4.5, 4.6+4.7, 5.6

4.5: *Design a program* `grapher.c` *that reads integers and draw a simple graph. Graph example:*

```
 3 |***
11 |***********
```

4.6+4.7: *Design a program* `my_wc` *that count the number of characters, words, and lines in the input. Use* `mary.txt` *to test your program, that is:*

```
bash $ my_wc < mary.txt
```

5.6: Two numbers are an amicable pair if their factors (excluding themselves) add up to each other. The first such pair is 220, which has the factors [1, 2, 4, 5, 10, 11, 20, 22, 44,55, 110], adding to 284; and 284, which has the factors [1, 2, 4, 71, 142], the sum of which is 220. The next pairs are 1,184 and 1,210; and then 2,620 and 2,924.

Write a function that takes two int *arguments and return true if they are an amicable pair. Test the function using an appropriate scaffolding [so now you also need main(), of course]. Use* `e56.data` *to test your program.*

CHALLENGE: write a program that search for amicable pairs and print them!