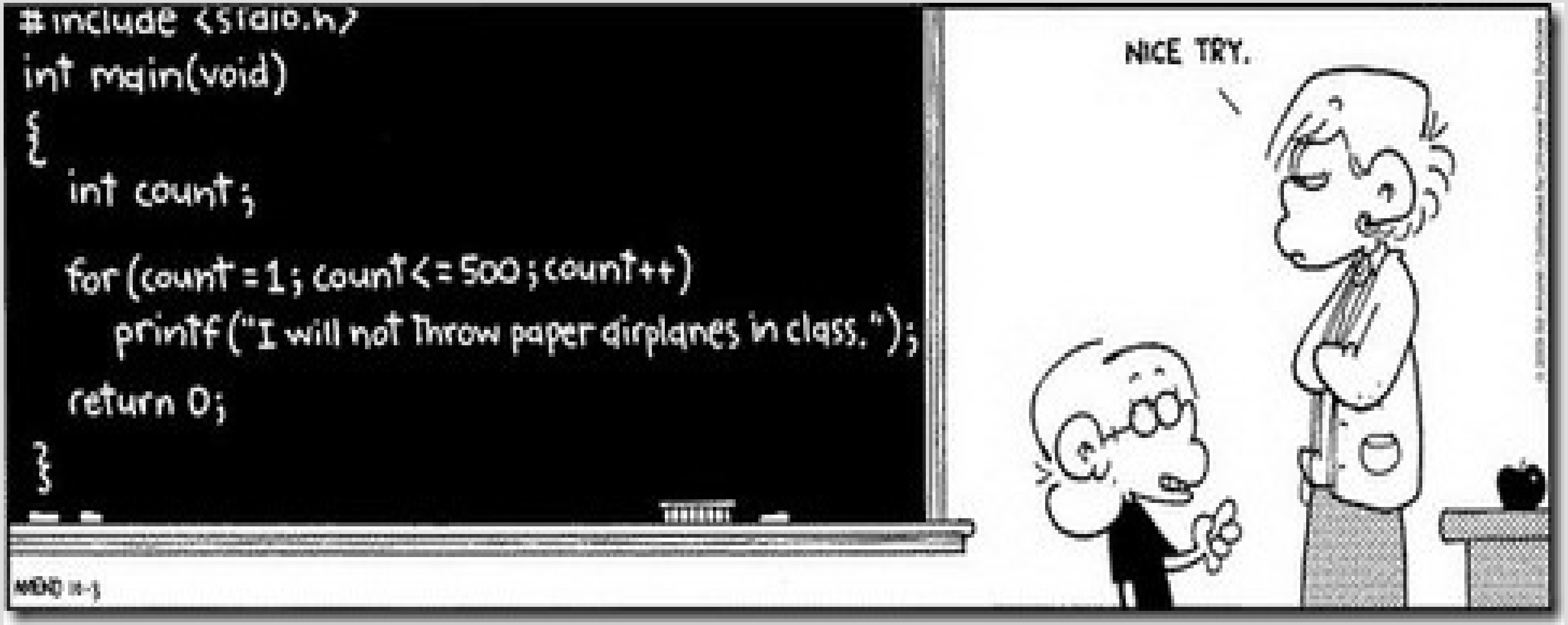
More loops

Ch 3.3-3.4



```
#include <stdio.h>
int main(void)
{
   int count;

   for(count = 1; count<= 500; count++)
      printf("I will not Throw paper airplanes in class.");

   return 0;
}
```

Loops

99 bottles of beer on the wall, 99 bottles of beer!
Take one down, pass it around, 98 bottles of beer on the wall!

98 bottles of beer on the wall, 98 bottles of beer!
Take one down, pass it around, 97 bottles of beer on the wall!

97 bottles of beer on the wall, 97 bottles of beer!
Take one down, pass it around, 96 bottles of beer on the wall!
...

Write a program to output the above song
(See 99beer.cpp)

continue

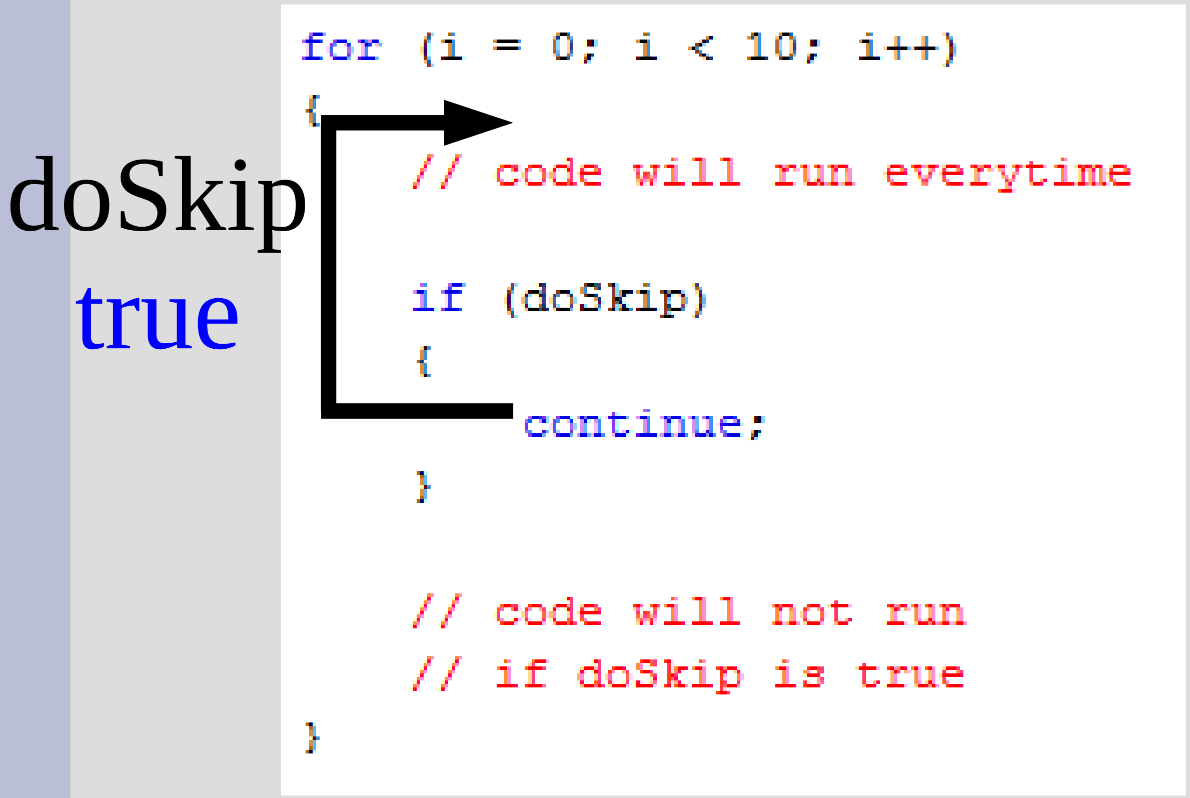
There are two commands that help control loops:

continue tells the loop to start over again

break stops the loop

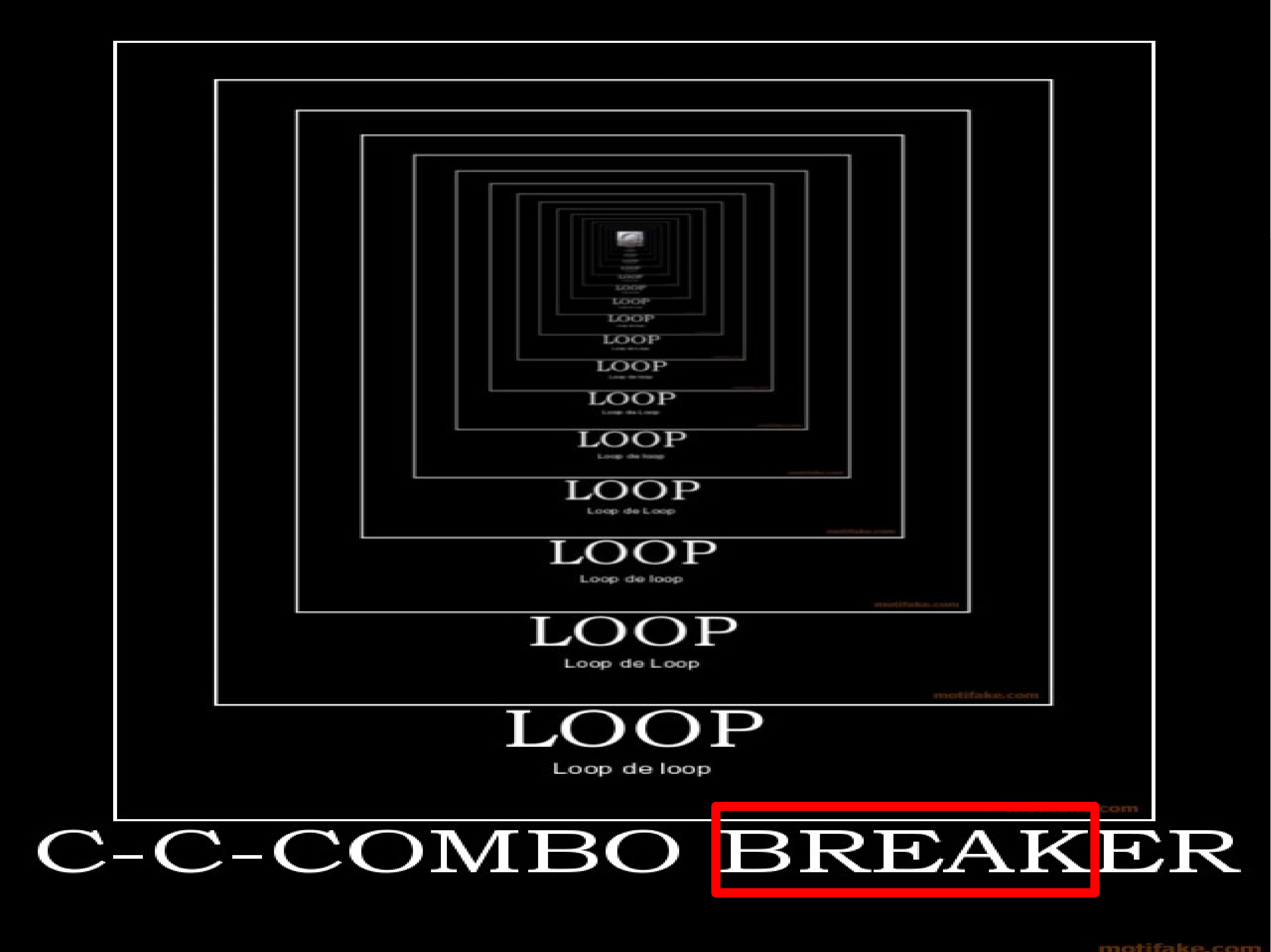
continue

continue command can be issued to start at the next iteration of a loop



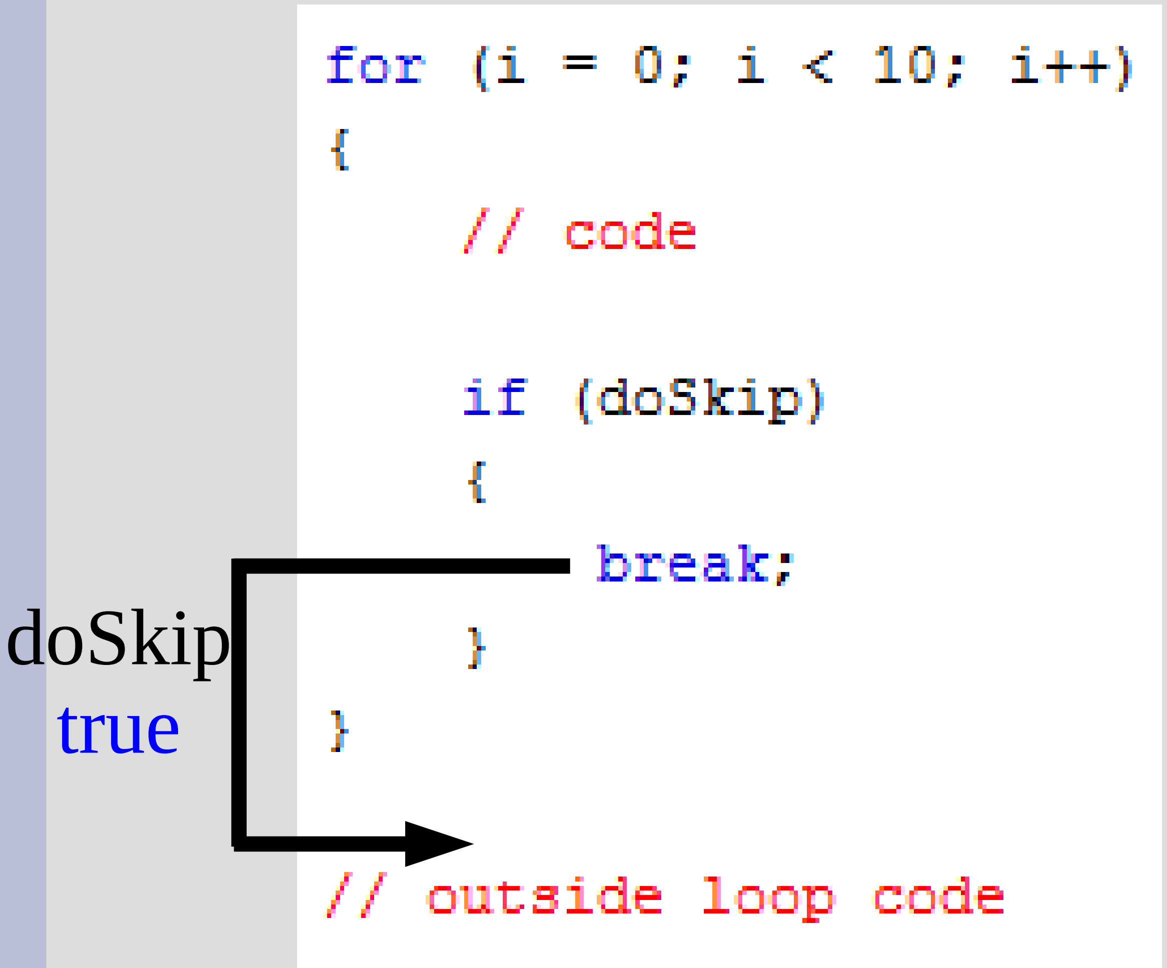
```
for (i = 0; i < 10; i++)
{
    // code will run everytime

    if (doSkip)
    {
        continue;
    }

    // code will not run
    // if doSkip is true
}
```

(See: continue.cpp)


LOOP
Loop de Loop
LOOP
Loop de loop
LOOP
Loop de Loop
LOOP
Loop de loop
LOOP
Loop de Loop
LOOP
Loop de loop
motifake.com
C-C-COMBO BREAKER
motifake.com

break

<u>break</u> will exit the current loop

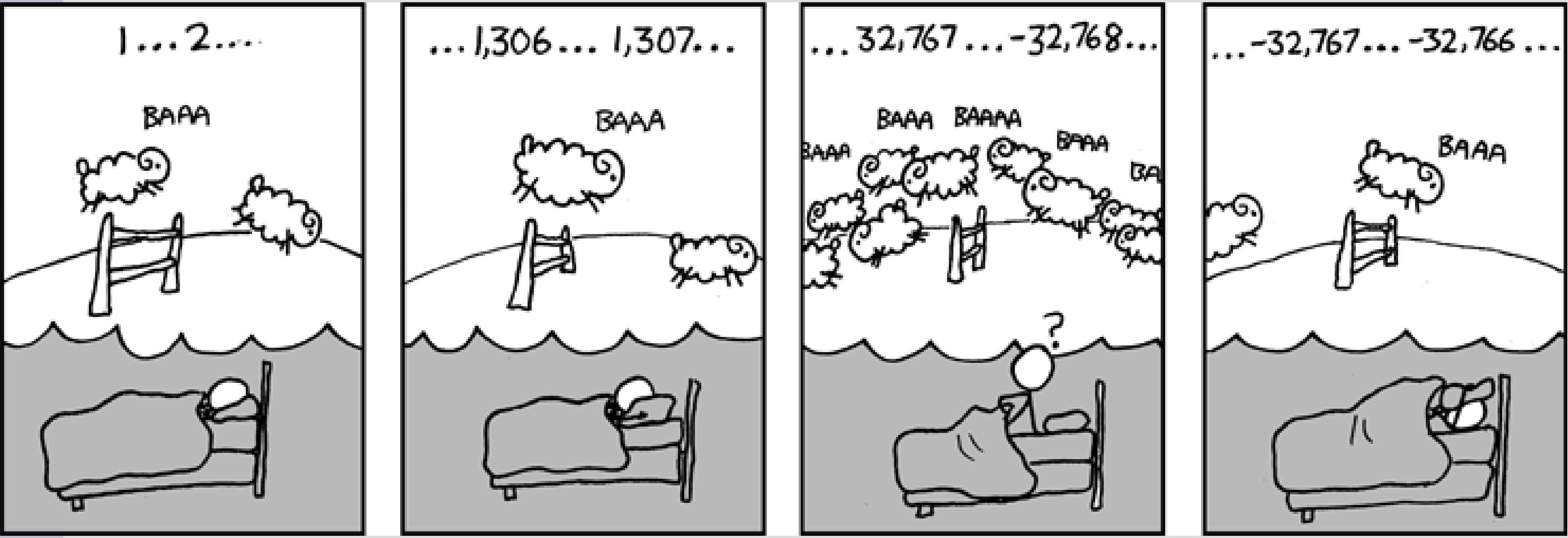


```
for (i = 0; i < 10; i++)
{
    // code

    if (doSkip)
    {
        break;
    }
}

// outside loop code
```

(See: break.cpp)

Infinite loops

(See: countingSheep.cpp)



while loop

```
While you == jerk do:
   today(again)
```

https://www.youtube.com/watch?v=7-Nl4JFDLOU

Loops to sum

Loops allow you to decide how many times
a piece of code should run on the fly
(i.e. at run time, not compile time)

You can either directly prompt the user
how many times or make a special value
to “exit” on

(See: sumLoop.cpp)

Debugging

When your program is not working, it is often helpful to add cout commands to find out what is going on

Normally displaying the value of your variables will help you solve the issue

Find up until the point where it works, then show all the values and see what is different than you expected

Loop practice

Write a program that asks the user to input a value, then show the sum from 1 to that value in the following format:

```
Find the sum from 1 to what
value?   5

1+2+3+4+5 = 15
```

(See: sumToN.cpp)

Nested for loop

Now modify the code so it shows all sums less than or equal to the entered values, as such:

```
Find the sum from 1 to what
value?   4
1 = 1
1+2 = 3
1+2+3 = 6
1+2+3+4 = 10
```

(See: sumAllToN.cpp)

Nested for loop

Like nested if statements, we can also make nested loops (which can cause headaches)

It might help to think of each loop as an added dimensions:

- 1 loop = 1 dimension (line/ruler)
- 2 loops = 2 dimensions (plane/square/area)
- 3 loops = 3 dimensions (volume/cube)
- ...

(See: nestedLoop.cpp)

Nested for loop

Ask the user for a size of matrix, then show the identity matrix for that dimension:

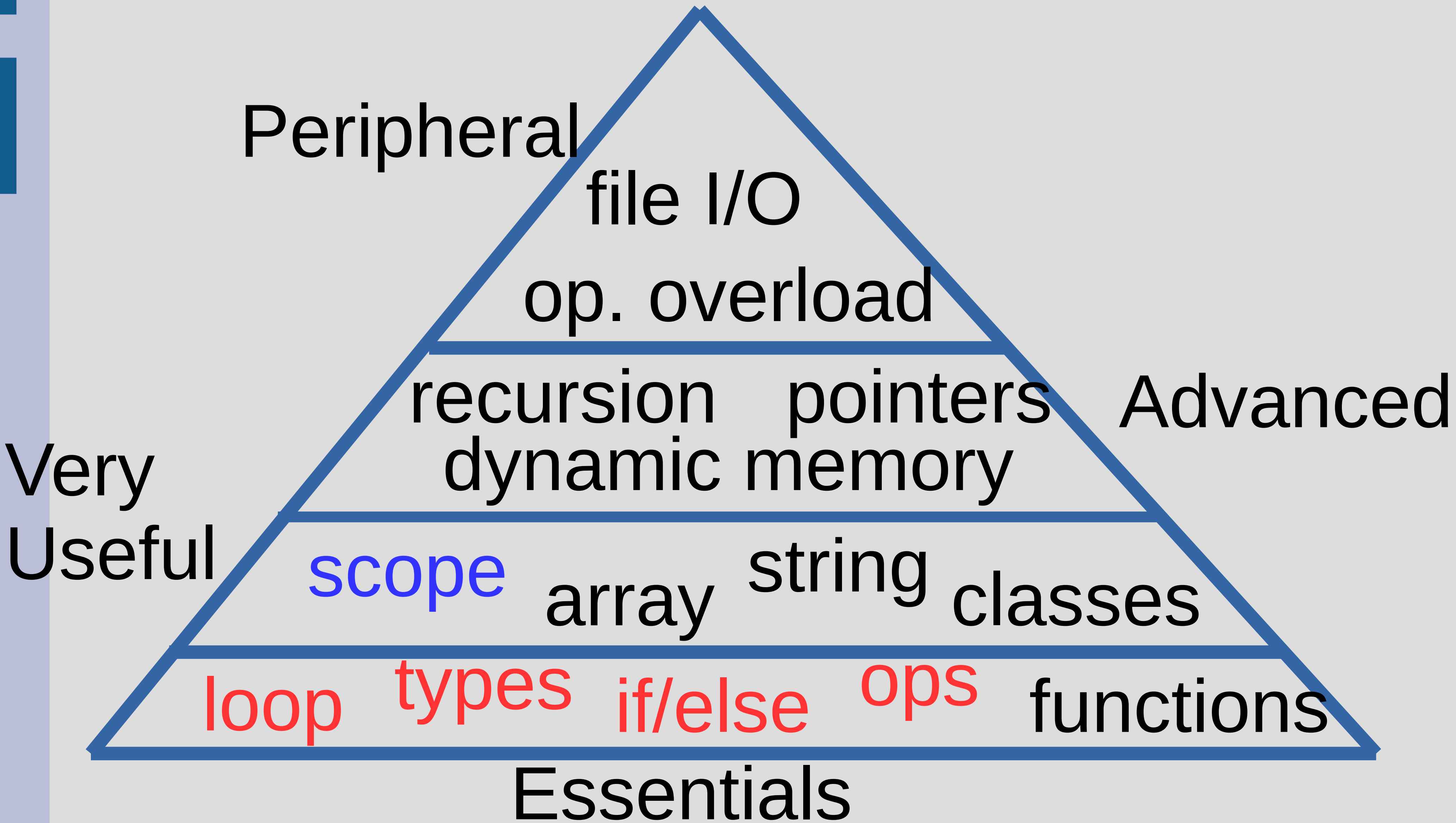
```
What size?  4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

(See: identityMatrix.cpp)

while vs do-while

(see: vendingV2.cpp)

Overview


Peripheral
file I/O
op. overload
recursion
pointers
dynamic memory
Advanced
Very Useful
scope
array
string
classes
loop
types
if/else
ops
functions
Essentials