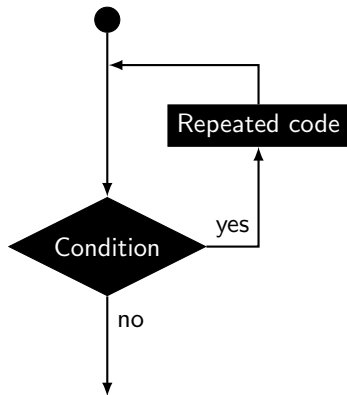


Lecture 5 - running in circles

Repeat instructions



- Block of code needs to be repeated
- Execution is sequential, e.g.: first instruction first, than second ...
- Loops allow to execute a statement, or a group of statements a number of times

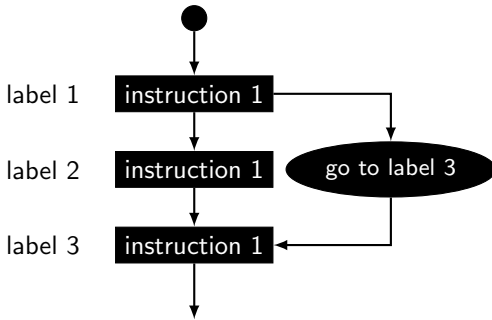
Example 1

Write 100 consecutive numbers ...

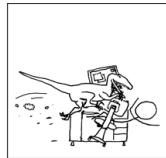
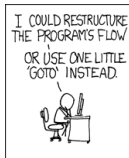
Bad idea:

```
...  
printf("%d\n", 0);  
printf("%d\n", 1);  
printf("%d\n", 2);  
...  
printf("%d\n", 100);
```

GO TO



- Provides a jump to from *goto* to a labeled statment
- Although available in many languages the use is **highly discouraged**, and is a mark of poor programming skills
- Makes program hard to follow and modify
- Any algorithm that uses *goto* can, and should be rewritten to avoid it!



Out first repeated statement

Better to forget this one!

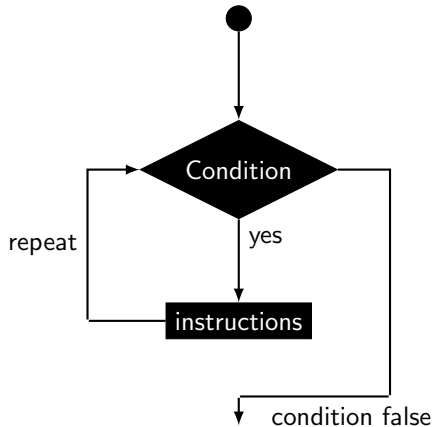
```
...  
int i=0;  
start:  
printf("%d\n",i);  
++i;  
if(i<=100) goto start;  
...
```

while loop

```
...  
while(condition)  
{  
    instructions;  
}  
...  
while(condition)  
    1 instruction;
```

- Executes as long as condition is *true*
- Condition is checked **before** execution
- ... might not execute at all if condition is *false*!
- When finished passes to the line immediately following the loop

while loop



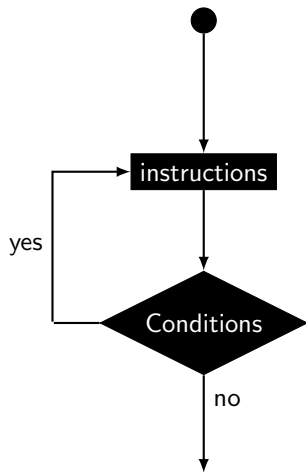
- Executes as long as condition is *true*
- Condition is checked **before** execution
- ... might not execute at all if condition is *false*!
- When finished passes to the line immediately following the loop

do ... while loop

```
...  
do  
{  
    instructions;  
} while(condition)  
...  
do  
    1 instruction;  
while(condition)
```

- Executes as long as condition is *true*
- Condition is checked **after** execution
- ... executes at least one time, even if condition is *false*!
- When finished passes to the line immediately following the loop

do ... while loop



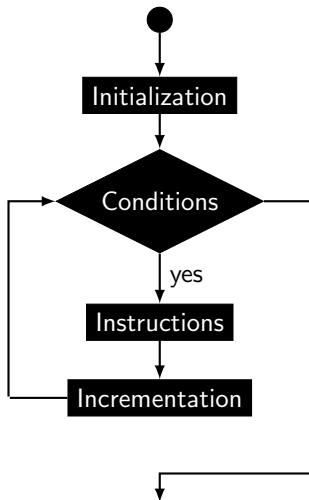
- Executes as long as condition is *true*
- Condition is checked **after** execution
- ... executes at least one time, even if condition is *false*!
- When finished passes to the line immediately following the loop

for

```
...  
for ( init; condition; incr ) {  
    instructions  
}  
...  
for ( init; condition; incr )  
    instructions  
...  
for(;;) {} //forever
```

- The **init** step is executed first, and only once.
- ... not required as long as ; is in
- Condition is checked **before** execution
- ... will not execute if initially condition is *false*!
- **incr** is performed after the instructions are executed, as a last step,
- ... not required as long as ; is in
- Condition is checked again
- ...

for



- The **init** step is executed first, and only once.
- ... not required as long as ; is in
- Condition is checked **before** execution
- ... will not execute if initially condition is *false*!
- **incr** is performed after the instructions are executed, as a last step,
- ... not required as long as ; is in
- Condition is checked again
- ...

Nested loops

Lopp in a loop in a loop in a ...

```
...  
for ( init; condition; incr ) {  
    for ( init; condition; incr ){  
        for ( init; condition; incr ){  
            instructions  
        }  
    }  
}  
...
```

- C allows to use loops inside another loops
- Can get tricky

Infinite Loop

Loop that runs forever ...

```
...  
for(;;) {} //forever  
...  
while (true){}  
...
```

- The program never ends

break

Terminates the loop

```
...  
break;  
...
```

- The loop is terminated and the code following the loop is executed

continue

Start the next run immediately

```
...  
continue;  
...
```

- The loop execution is stopped, and started from the beginning

Example with functions

Write a program, calculating the Fourier expansion of a square wave. Use a function to calculate the values.

$$f(x) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{n} \sin\left(\frac{n\pi x}{L}\right) \quad (1)$$