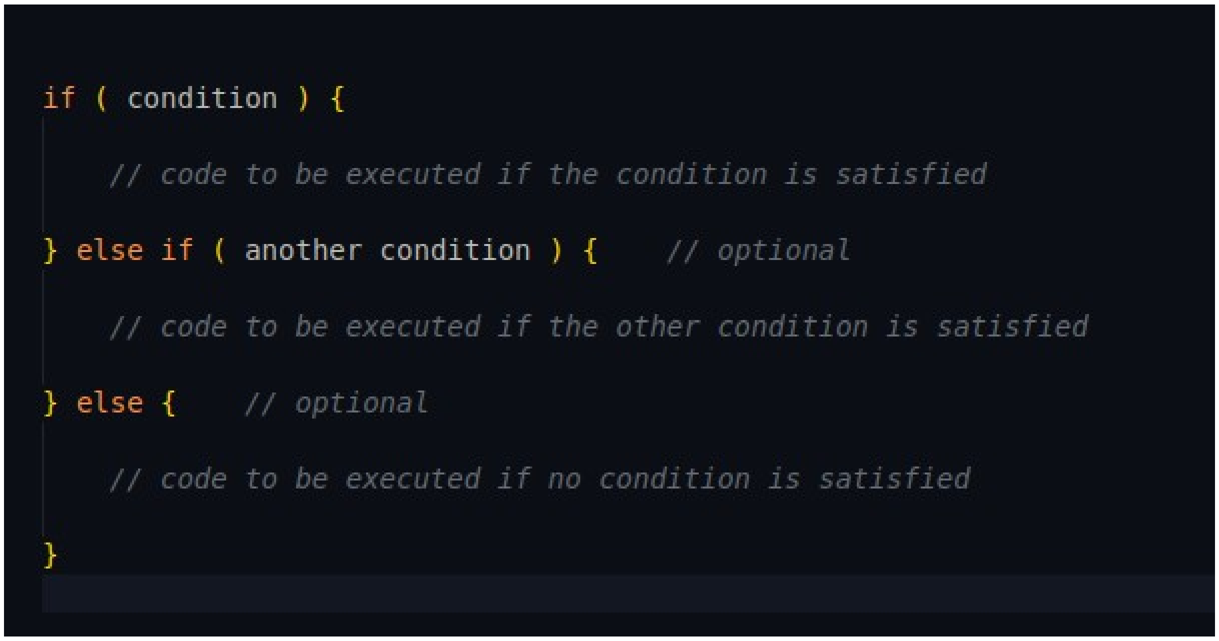
Lecture 9: loops and conditions & the Stack

Conditions:

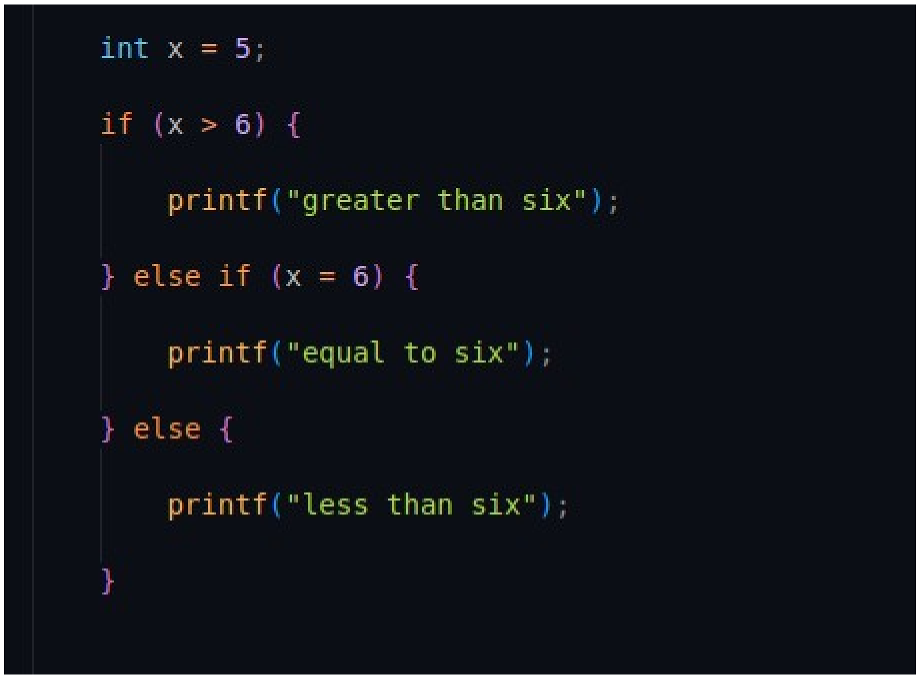
Conditions are one of the very basic and fundamental ways of controlling the flow of a program by checking if the execution satisfies a certain condition or not. In C we have two ways of doing so:

1. "if { . . . } else if { . . . } else { . . . }" statement:
   the syntax for the "if" statement is as follows:



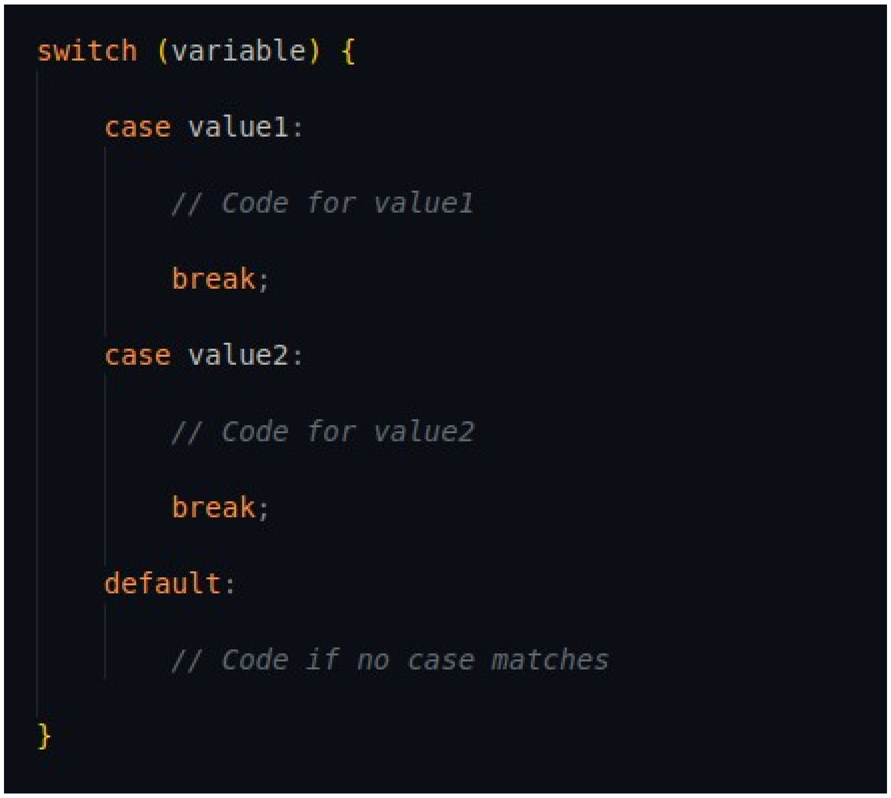
```
if ( condition ) {
    // code to be executed if the condition is satisfied
} else if ( another condition ) {    // optional
    // code to be executed if the other condition is satisfied
} else {    // optional
    // code to be executed if no condition is satisfied
}
```

Here is an example:



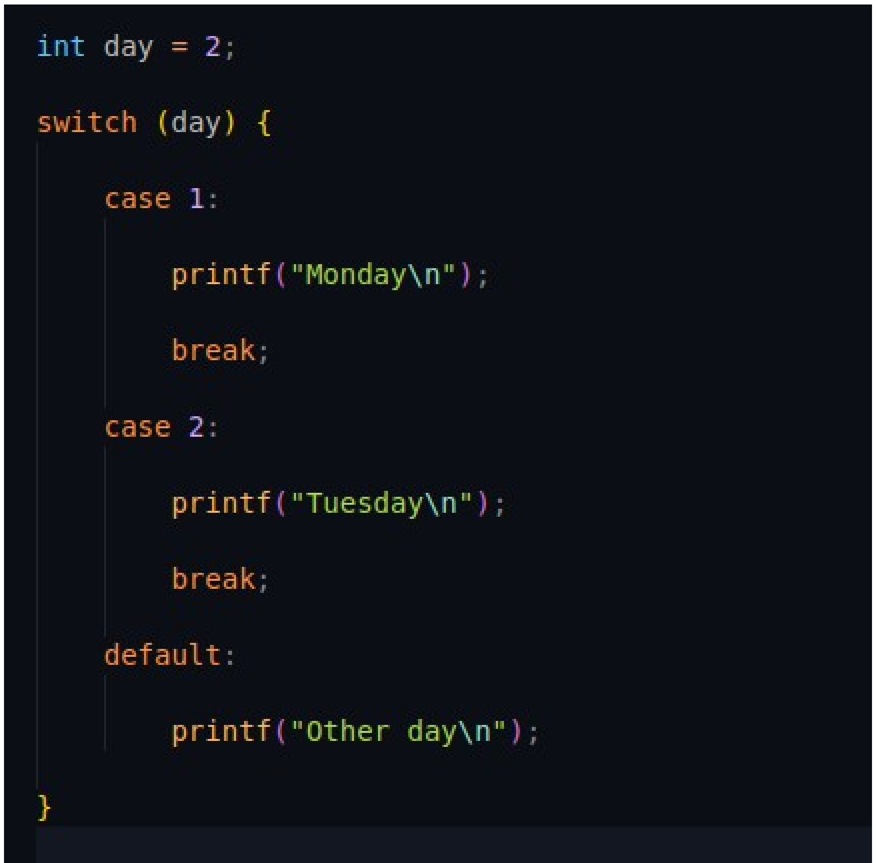
```
int x = 5;
if (x > 6) {
    printf("greater than six");
} else if (x = 6) {
    printf("equal to six");
} else {
    printf("less than six");
}
```

2. “switch( ) { case: . . . ; case: . . . ; }” statement:
the syntax of the “switch” statement is as follows:



```
switch (variable) {

    case value1:

        // Code for value1

        break;

    case value2:

        // Code for value2

        break;

    default:

        // Code if no case matches

}
```

Here is an example:



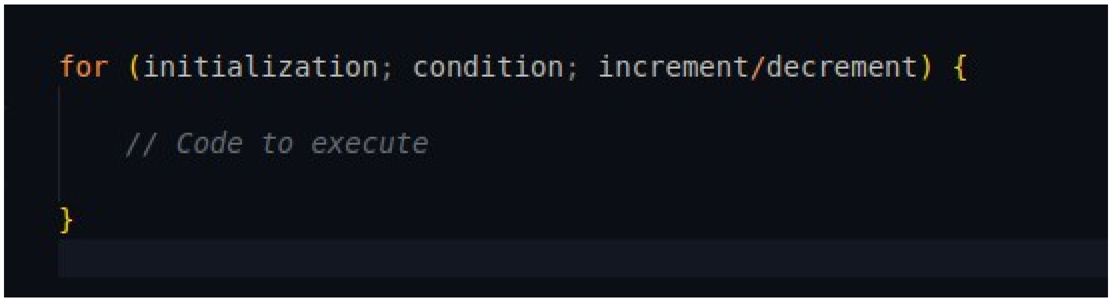
```
int day = 2;

switch (day) {

    case 1:

        printf("Monday\n");

        break;

    case 2:

        printf("Tuesday\n");

        break;

    default:

        printf("Other day\n");

}
```

Loops:

Loops are another way of controlling the flow of the execution of a code. Loops are a great way of determining how many times an action needs to be done based of a certain condition. Here are some ways of how we can apply the concept of looping in C:

1. “for (. . .) { . . . }” statement:

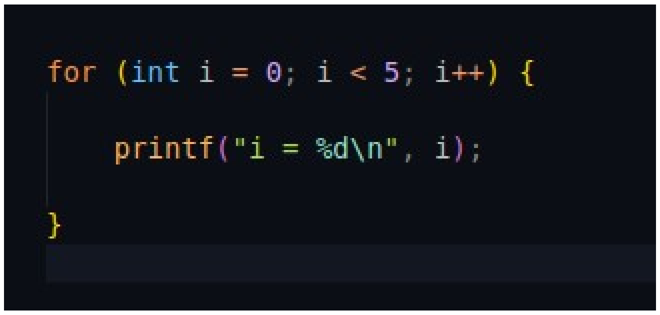
   Here is the syntax of the “for” loop:



```c
for (initialization; condition; increment/decrement) {
    // Code to execute
}
```

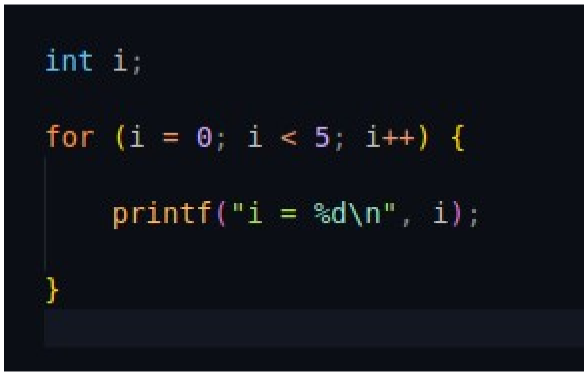
   Here is the meaning of each of the above:
   - initialization: this is the value that the loop will start based on, and will modify along the execution until it reaches the desired value.
   - Condition: this is the value of the condition that, after each execution of the loop, gets checked to see if the “initialization” has reached the “condition” yet or not.
   - increment/decrement: this is the operation that will be done on the “initialization” after each loop execution to modify it in order for it to get closer and closer to the “condition” that we want.

   Here is an example:



```c
for (int i = 0; i < 5; i++) {
    printf("i = %d\n", i);
}
```

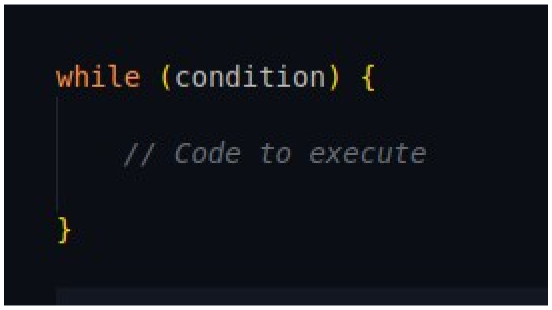
The “int i” can also be declared outside the loop and assigned inside of it as follows:



```
int i;

for (i = 0; i < 5; i++) {

    printf("i = %d\n", i);

}
```

2. “while (. . .) { . . . }” loop:

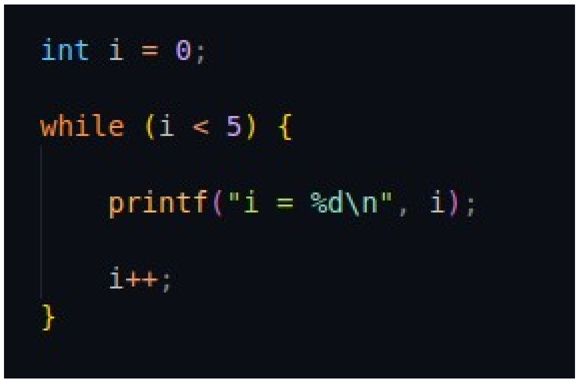
Here is the syntax of the “while” loop:



```
while (condition) {

    // Code to execute

}
```

In the “while” loop, we simply provide a condition that is related to the operation we want to do inside the loop, to be checked after each execution of the code in the loop. The loop will keep iterating until one of the checks fails to fulfill the condition, which will then stop the loop.

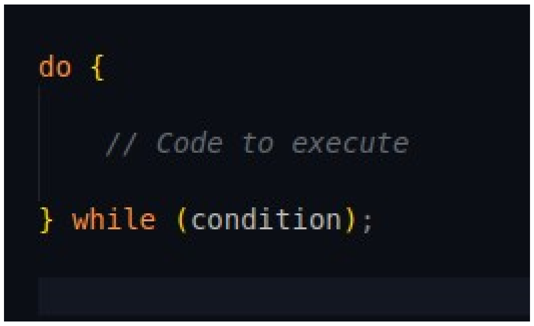
Here is an example on how this happens:



```
int i = 0;

while (i < 5) {

    printf("i = %d\n", i);

    i++;
}
```

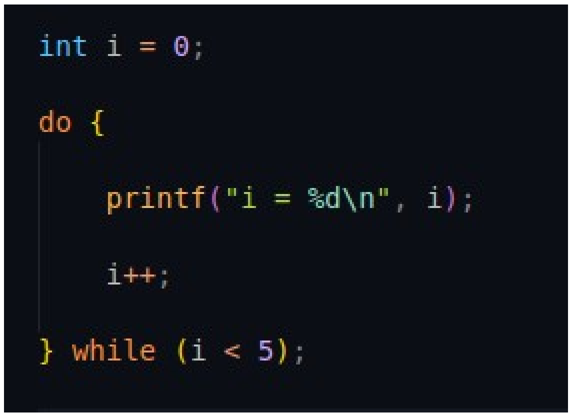
As we see, the code inside the loop is affecting the condition directly, incrementing it after each iteration by 1, until an iteration comes where the condition is not fulfilled, where then the loop will stop.

3. “do { . . . } while (. . .)” loop:
   g
   This is another version of the “while” loop, where the key difference between the two is that the regular “while” checks the condition before it starts to execute the code inside of it for the first iteration, and the “do while” one executes the code for the first time without checking the condition.
   So it is safe to say that the “do while” loop executes the code at least once.
   Here is the syntax of the “do while” loop:



```
do {

    // Code to execute

} while (condition);
```

as we see, the condition check comes after the code, which will lead the compiler to go through the code before it checks the condition.
Here is an example on this:



```
int i = 0;

do {

    printf("i = %d\n", i);

    i++;

} while (i < 5);
```

g

Best wishes, Eng. Ali Hassan