CONTROL STRUCTURES

LIKE ALL LANGUAGES, PHP HAS CODE CONTROL STRUCTURES

CONTROL STRUCTURES CONTROL THE FLOW OF EXECUTION OF A PROGRAM.

1. CONDITIONAL STATAMENTS

Conditional statements allow a program to execute different code pieces, depending on one or more conditions

2. LOOPING STRUCTURES

Looping structures allow a program to repeat the execution of some code pieces

CONTROL STRUCTURES: IF

```
if(condition)
{
    Instructions;
}

// optional

else
{
    Instructions;
}
```

The if statement only executes if the condition inside the parentheses is evaluated to true. The condition can include any of the comparison and logical operators
For handling other cases, you can use else clause, which executes if the previous condition is false

CONTROL STRUCTURES: IF

```
if(condition)
{
    Instructions;
}

// optional

elseif(condition)
{
    Instructions;
}
elseif(condition)
{
…
}
else
{
}
```

In addition, the ternary conditional operator can be used in simple true/false statements.

```
echo $condition ? true_statement : false_statement;
```

For handling all other cases, there can be one else clause at the end, which executes if all previous conditions are false

CONTROL STRUCTURES: SWITCH

```
switch(expression)
{
    case value1:
                      //
Instructions;
     break;
    case value2:
     // Instructions;
    …
    // optional
    default:
     // Instructions
}
```

The switch statement checks for equality between an integer, float, or string and a series of case labels. It then passes execution to the matching case. The statement can contain any number of case clauses and may end with a default label for handling all other cases.
The statements end with the break keyword. Without it, the execution falls through to the next case

UNLIKE OTHER LANGUAGES, EXPRESSION CAN BE STRING TYPE

CONTROL STRUCTURES: WHILE LOOP

```
while(condition)
{
    // instructions;
}
```

The while loop runs through the code block only if its condition is true.

```
do
{
    // instructions;
}
while(condition);
```

The do-while loop works in the same way as the while loop, except that it checks the condition after the code block. Therefore, it always runs through the code block at least once.

CONTROL STRUCTURES: FOR LOOP

The for loop is used to go through a code block a specific number of times. It uses three parameters. The first parameter initializes a counter and is always executed once, before the loop. The second parameter holds the condition for the loop and is checked before each iteration. The third parameter contains the increment of the counter and is executed at the end of each iteration

```
for(counter=initial_value;condition;increment)
{
    // instructions;
}
```

```
for($i=0;$i<10;$i++)
{
    // instructions;
}
```

CONTROL STRUCTURES: FOR LOOP

ALTHOUGH IT IS NOT VERY WELL RECOMMENDED, WE CAN USE SOME KEYWORDS TO CHANGE THE LOOPING FLOW:

break: FORCES THE END OF THE LOOP

continue: FORCES THE NEXT ITERATION

```
for(counter=initial_value;condition;increment) {
    // instructions;
    if(condition) { break; }  // ends the loop
    // instructions;
    if(condition) { continue; } // goes to the next iteration
    // instructions;
}
```

break AND continue WORK IN WHILE LOOPS AS WELL.

EXIT AND RETURN

THE exit STATEMENT ENDS THE SCRIPT'S EXECUTION AT ANY MOMENT IS REACHED

THE return STATEMENT ENDS THE FUNCTION RETURNING (OR NOT) A VALUE. WE WILL STUDY IN THIS UNIT “BASIC PHP - FUNCTIONS”

IF THE return STATEMENT IS REACHED IN THE MAIN PROGRAM, IT ENDS ITS THE EXECUTION AS WELL