

Control Structures

- In the programming we need to sometimes control the flow of operation other than just the sequential statements. In this case we need to control statements
- Control statements are classified in 2 types viz. the **iterative statements** and **conditional statements**
- **Iterative Statements** are use to perform certain operation repetitively for certain number of times. In C we have 3 iterative statements viz. **for loop, while loop and do-while loop**. Iterative statements are also called as repetitive statements as they repeat a set of statements for a given number of times
- **Conditional statements** are use to perform the operations based on a particular condition i.e if a condition is true perform one task else perform another task. In C we have two conditional statement namely **if-else statements** and **switch case statements**. Conditional statements are also called as selective statements i.e these statement select a particular statement to be executed based on the condition

for Loop

- For is a iterative statement. It is used to repeat a set of statements number of times. The syntax (method) of writing for statement is given below:

```
for(initializations; conditions; increment/decrement/updating) {  
    -  
    -  
    statements;  
    -  
    -  
}
```
- The sequence of execution of the for loop is such that the **initialization statements are executed first**. These initialization statements are executed only once. They are used to initialize the values of the variables
- The **second step** if **checking the condition** specified. There can be only one condition. If more than one conditions are required they can be combined using the logical AND, OR operators. If the condition is false the execution will go outside the braces of for loop, if the condition is not true.
- The **third step** is to **execute all the statements** inside the curly braces. These statements will be executed sequentially. The number of statements can be of any count. There can be another control statement is required inside one control statement.
- The **fourth step** is the **increment/decrement or updating operations**. These operations are not necessarily increment or decrement operations, but mostly these are increment decrement and hence called so. We can update the variables over here before starting the next iteration of the iterative statements
- **Finally the control goes back to the second step** as said the first step is executed only once, the steps that are repeated continuously are the second, third and fourth steps. After the fourth step the condition is checked again, If the condition is true then execution continues, else the control goes outside the for loop i.e curly braces

[Please refer the link to know the programs](#)

Nested for Loop

- A for loop inside another for loop is called as nested for loop
- When a particular operation has two references, we require nested for loop. For example if we want to keep a reference of row number and column number, then we use a nested for loop

[Please refer the link to know the programs](#)

While loops and do-while loops

- While and do-while loops are also for repetitive operations
- The operations are slightly different than the for loop, but the same operations can be implemented by for, while or do-while loops. Although there is one major difference in a do-while loop wherein one particular operation cannot be implemented.

- Syntax of while loop

```
while(condition) {  
    -  
    -  
    statements;  
    -  
    -  
}
```

- The operations of the while loop is such that, first the condition is checked. If the condition is true, then the statements are executed. Once the statements are executed, the condition is again checked and this keeps on repeating, until the condition is false. If the condition is false, the statements inside the loop are not executed, instead the control directly comes out of the loop

- Syntax of do-while loop

```
do {  
    -  
    -  
    statements;  
    -  
    -  
}while(conditions);
```

- In this case the operations is slightly different i.e first the statment are executed and then the condition is checked. If the condition is true the statement are executed again. If the condition is false, the statements are not executed again.
- One major point to be noted is that in case of while loop, the statement are executed atleast once even if the condition is not true for the first statement. On the other hand, in case of for loop and while loop, even for the first time the statements are executed only if the condition is true.

[Please refer the link to know the programs](#)

C-Control Structures for Selection

If-else Selective Statement

- This is very important statement used to check the condition and accordingly execute a set of statements, based on whether the condition is true or false. Hence it is called as selective statement. It selectively executes some statements and doesn't executes some.
- These syntax of the If-else condition is as given below

```

if(condition) {
    -
    -
    -
    statements1;
    -
}
else {
    -
    -
    -
    statements2;
    -
}

```

- The set of statements names as statements1 in the above syntax are executed if the condition given with the if statement is true. The set of statements statements2 are not executed in this case.
- If the condition specifies in the if statement is false then the statements named as statements2 in the above syntax are executed. The set of statements statements1 are not executed in this case.
- **Note: The else part is optional i.e we can have a if statement without the else statement. As seen in the above given syntax, only the first half i.e we can have if statement as shown below.**

```

if(condition) {
    -
    -
    -
    -
    statements1;
    -
    -
}

```

[Please refer the link to know the programs](#)

If-else ladder or if else if

- In some cases we have to check multiple cases of a particular condition. In such cases we have to use an if-else ladder. A set of if-else statements as shown below is called as if-else ladder
- Syntax is given below

```

if(condition) {
    statements;
}
else {
    if(condition) {
        statements;
    }
    else {
        if(condition) {
            statements;
        }
        else {
            statements;
        }
    }
}

```

- In this case the first condition is checked, if it is true the statements inside the if statement are executed. But if the condition is false it goes to the else statement. Again there is a condition with if; the statements are executed if this second condition is true. Else it again goes to the else and again checks the condition associated with this if statement. Thus if one condition satisfies no other else is checked thereafter.

[Please refer the link to know the programs](#)

Switch-Case Selective Statement

- In the previous section we have seen the if-else ladder. A better solution of this is switch-case. Using switch-case the if-else ladder can be implemented in a much better way.
- The syntax of switch-case is given below:

```
switch (expression) {  
    case constant1:  
        // Code to execute if expression == constant1  
        break;  
  
    case constant2:  
        // Code to execute if expression == constant2  
        break;  
  
    case constant3:  
        // Code to execute if expression == constant3  
        break;  
  
    default:  
        // Code to execute if none of the cases match the expression
```

Note: Break statement transfers the control outside the current loop. We will see some more cases of break statement along with the for / while / do-while statement.

- The expression or variable can be given in the brackets associated with the switch statement. The values of this expression / variable are the labels associated with the cases inside the switch statement.
- The value if the expression / variable is first compared with the label1. If they are equal, then the statements followed by the corresponding case are executed. The break statement followed by these statements, transfers the control after the switch statement.
- If the expression / variable is not equal to label1, then it is directly compared to label2 without executing the statements followed by the label1.
- Hence the statements of only that case are executed with the correct label value of the expression / variable.

Note:


1. Break statement is not compulsory. But if the break statement is not written, everything will be executed after the case statement.
2. Default is not necessary. But in case if default is not written and some case occurs which is not considered then there will be no operation performed and the user will not be able to realize the problem in the program.
3. The label can only be value, it cannot be a condition or an expression.
4. The brackets associated with the switch statement can have either the variable or expression and no condition.

[Please refer the link to know the programs](#)

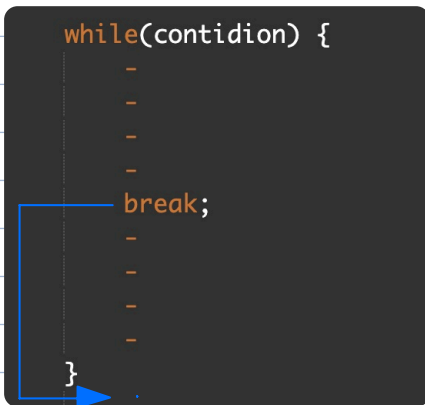
Branching Statements(Break, Continue, Goto)

- These statements transfer the control to a different place in the program. We will see the operation of each of these. We have already seen the use of break statement in switch case. Let us see the use of continue and break in other loop statements.
- The break statement neglects the statements after it in the loop and transfers the control outside the loop
- The figure shows the loop of break statement in each of the loop statements i.e for while and do-while statements.

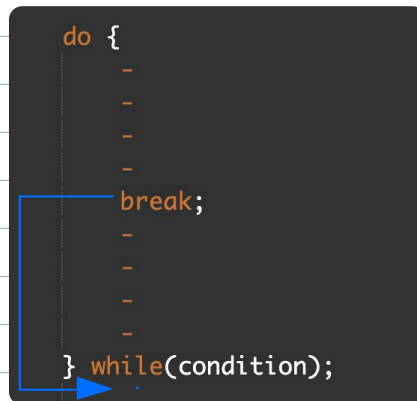
```
for(initialization; conditions; updation) {  
    -  
    -  
    -  
    break;  
    -  
    -  
    -  
}
```



```
while(contidion) {  
    -  
    -  
    -  
    break;  
    -  
    -  
    -  
}
```

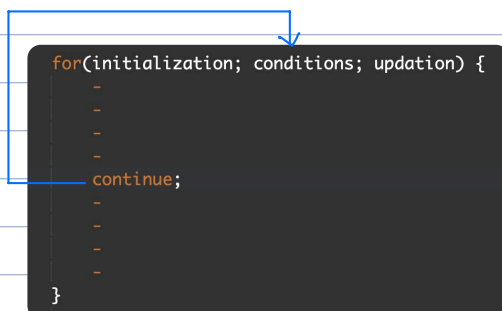


```
do {  
    -  
    -  
    -  
    break;  
    -  
    -  
    -  
} while(condition);
```

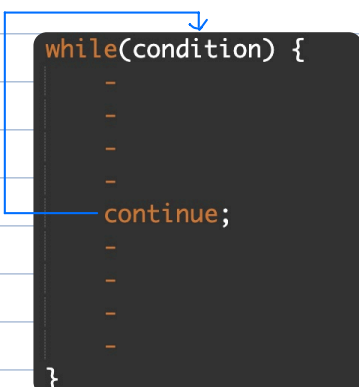


- The continue statement also neglects the statements after it in the loop and transfers the control back to the starting of the loop for next iteration. The starting of the loop for next iteration. The below figure shows the operation of continue statement in each of the loop statements i.e. for, while and do-while statements.

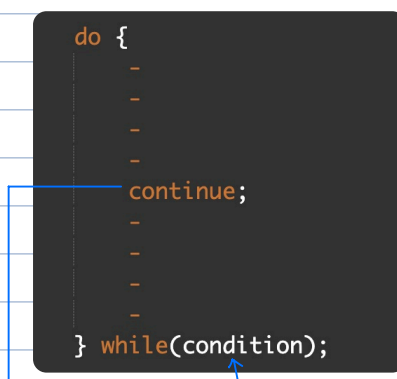
```
for(initialization; conditions; updation) {  
    -  
    -  
    continue;  
    -  
    -  
    -  
}
```



```
while(condition) {  
    -  
    -  
    -  
    continue;  
    -  
    -  
    -  
}
```



```
do {  
    -  
    -  
    -  
    continue;  
    -  
    -  
    -  
} while(condition);
```



- The goto statement transfers the control to the label specified with the goto statement. A label is any name given to a particular statement. A label is to be followed with a colon(:).
- The syntax of a goto statement is as shown below
-

```
-  
-  
-  
goto label1  
-  
-  
-  
label1:  
-  
-  
-
```

OR

```
-  
-  
-  
label2:  
-  
-  
-  
goto label2:  
-  
-  
-
```

- Thus it shows that the label can be before or after the goto statement and hence the control can be transferred to earlier or next statements using a goto statement.

Please refer the link to know the programs