Decision Making Instructions

Conditional statements are used to perform operations based on some condition.

If Statement

```
if (condition) {
// This block of code will get executed, if the condition is True
}
```

If-else Statement

```
if (condition) {
// If condition is True then this block will get executed
} else {
// If condition is False then this block will get executed
}
```

if else-if Statement

```
if (condition) {
// Statements;
}
else if (condition){
// Statements;
}
else{
// Statements
}
```

Ternary Operator

It is shorthand of an if-else statement.

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Switch Case Statement

It allows a variable to be tested for equality against a list of values (cases).

```
switch (expression)
{
case constant-expression:
statement1;
statement2;
break;
case constant-expression:
statement;
break;
...
default:
statement;
}
```

Iterative Statements

Iterative statements facilitate programmers to execute any block of code lines repeatedly and can be controlled as per conditions added by the programmer.

while Loop

It iterates the block of code as long as a specified condition is True

```
while (/* condition */)
{
/* code block to be executed */
}
```

do-while loop

It is an exit controlled loop. It is very similar to the while loop with one difference, i.e., the body of the do-while loop is executed at least once even if the condition is False

```
do
{
/* code */
} while (/* condition */);
```

for loop

It is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

```
for (int i = 0; i < count; i++)
{
/* code */
}
```

Break Statement

break keyword inside the loop is used to terminate the loop

```
break;
```

Continue Statement

continue keyword skips the rest of the current iteration of the loop and returns to the starting point of the loop

```
continue;
```