Tutorial Link https://codequotient.com/tutorials/Iteration - do while loop/5a22a7b3c66cfe38f29622a6

TUTORIAL

Iteration - do while loop

Chapter

1. Iteration - do while loop

The third kind of loop is do-while loop. It is a different kind of loop than for and while loops. The general form of do-while loop is:

```
do
{
   // set of statements, loop body
} while(condition);
```

The working of do-while loop is similar to while except the condition place. The do-while loop will execute the loop body once, after executing the loop body it will check the condition. If condition is true, it will execute the body again otherwise the statement after the do-while loop will be executed. So in this sense, do-while loop is known as exit-controlled loop, as it check the condition at the exit point of the loop. For example, the below program uses do-while loop:

```c
#include <stdio.h>

int main()
{
  int var1;
  var1 = 1;
  do
```

```
  {
    printf("var1 = %d\n", var1);
    var1++;
  } while(var1 <= 3);

  printf("This is after do-while loop.");
  return 0;
}

```

All these will do the almost same thing in different ways. The major difference between entry-controlled loops (for, while) and exit-controlled loops is that entry-control loops will check the condition before entering the loop body, and if condition evaluates to false the body will never execute. Whereas the exit-controlled loop will execute the loop body at least once, then check the condition to re-enter the loop or not. So in do-while loop one execution of loop body is guaranteed, whereas in other two it depends on condition.

Generally, do-while loops are used for making menu selections, where to make the decisions the loop body needs to be executed and depending on user's input further actions will be taken.

```
#include <stdio.h>

int main()
{
   int num1, num2, choice;
   num1 = 4;
   num2 = 6;
   printf("num1 = %d, \tnum2 = %d\n", num1, num2);
   do
   {
       printf("Please make a choice from below :\n");
       printf("1. To see num1 + num2.\n");
       printf("2. To see num1 - num2.\n");
       printf("3. To see num1 * num2.\n");
       printf("4. To exit the program.\n");
       scanf ("%d", &choice);

       if(choice == 1) printf("num1 + num2 = %d\n",
num1+num2);
```

```
       if(choice == 2) printf("num1 - num2 = %d\n", num1-
num2);
       if(choice == 3) printf("num1 * num2 = %d\n",
num1*num2);

   } while (choice != 4);
   return 0;
}
```

Can we do the same thing, with for loop or while loop? Yes we can. But the thing is that they are not usually used to do those things. Hence we have variety of loops, It's the programmer's decision which construct to use where in code. For example, the above behavior can be achieved from for loop also as below: -

```
#include <stdio.h>

int main()
{
   int num1, num2, choice;
   num1 = 4;
   num2 = 6;
   printf("num1 = %d, \tnum2 = %d\n", num1, num2);
   for(choice = 5; choice != 4; )
   {
       printf("Please make a choice from below :\n");
       printf("1. To see num1 + num2.\n");
       printf("2. To see num1 - num2.\n");
       printf("3. To see num1 * num2.\n");
       printf("4. To exit the program.\n");
       scanf ("%d", &choice);

       if(choice == 1) printf("num1 + num2 = %d\n",
num1+num2);
       if(choice == 2) printf("num1 - num2 = %d\n", num1-
num2);
       if(choice == 3) printf("num1 * num2 = %d\n",
num1*num2);
   }

   return 0;
}
```

In this case, we forcefully try to execute the for loop body once, then the rest depends as usual.