|  |  |  |
| --- | --- | --- |
| **SR.NO** | **while loop** | **do-while loop** |
| 1. | While the loop is an entry control loop because firstly, the condition is checked, then the loop's body is executed. | The do-while loop is an exit control loop because in this, first of all, the body of the loop is executed then the condition is checked true or false. |
| 2. | The statement of while loop may not be executed at all. | The statement of the do-while loop must be executed at least once. |
| 3. | The while loop terminates when the condition becomes false. | As long as the condition is true, the compiler keeps executing the loop in the do-while loop. |
| 4. | In a while loop, the test condition variable must be initialized first to check the test condition in the loop. | In a do-while loop, the variable of test condition Initialized in the loop also. |
| 5. | In a while loop, at the end of the condition, there is no semicolon. **Syntax:**  while (condition) | In this, at the end of the condition, there is a semicolon. **Syntax:**  while (condition); |
| 6. | While loop is not used for creating menu-driven programs. | It is mostly used for creating menu-driven programs because at least one time; the loop is executed whether the condition is true or false. |
| 7. | In a while loop, the number of executions depends on the condition defined in the while block. | In a do-while loop, irrespective of the condition mentioned, a minimum of 1 execution occurs. |
| 8. | **Syntax of while loop:**  while (condition)  {  Block of statements;  }  Statement-x; | **Syntax of do-while loop:**  do  {  statements;  }  while (condition);  Statement-x; |
| 9. | **Program of while loop:**  Program of while loop:  #include  #include  Void main()  {  int i;  clrscr();  i = 1;  while(i<=10)  {  printf("hello");  i = i + 1;  }  getch();  } | **Program of do-while loop:**  #include  #include  Void main()  {  int i;  clrscr();  i = 1;  do  {  printf("hello");  i = i + 1;  }  while(i<=10);  getch();  } |
| 10. | **Flowchart of while loop:** while loop vs do-while loop in C | **Flowchart of do-while loop:** while loop vs do-while loop in C |

| **For loop** |  | **Do-While loop** |
| --- | --- | --- |
| Statement(s) is executed once the condition is checked. |  | Condition is checked after the statement(s) is executed. |
| It might be that statement(s) gets executed zero times. |  | Statement(s) is executed at least once. |
| For the single statement, bracket is not compulsory. |  | Brackets are always compulsory. |
| Initialization may be outside or in condition box. |  | Initialization may be outside or within the loop. |
| for loop is entry controlled loop. |  | do-while is exit controlled loop. |
| for ( init ; condition ; iteration ) { statement (s); } |  | do { statement(s); } while (condition); |
| #include <stdio.h>    int main()  {        int i = 0;        for (i = 5; i < 10; i++) {          printf("GFG\n");      }        return 0;  } |  | #include <stdio.h>    int main()  {        int i = 5;        do {          printf("GFG\n");          i++;      } while (i < 10);        return 0;  } |

| **while** | **do-while** |
| --- | --- |
| Condition is checked first then statement(s) is executed. | Statement(s) is executed atleast once, thereafter condition is checked. |
| It might occur statement(s) is executed zero times, If condition is false. | At least once the statement(s) is executed. |
| No semicolon at the end of while. while(condition) | Semicolon at the end of while. while(condition); |
| If there is a single statement, brackets are not required. | Brackets are always required. |
| Variable in condition is initialized before the execution of loop. | variable may be initialized before or within the loop. |
| while loop is entry controlled loop. | do-while loop is exit controlled loop. |
| while(condition) { statement(s); } | do { statement(s); } while(condition); |
| #include <stdio.h>    int main()  {        int i = 5;        while (i < 10) {          printf("GFG\n");          i++;      }        return 0;  } | #include <stdio.h>    int main()  {        int i = 5;        do {          printf("GFG\n");          i++;      } while (i < 10);        return 0;  } |