**For loop**

**Writer:** Sayak Haldar

**1. The following code ‘for(;;)’ represents an infinite loop. It can be terminated by.**

a) break

b) exit(0)

c) abort()

d) all of the mentioned  
  
**Answer) a) break.**

The infinite loop itself can be terminated by break. Other two commands terminate the program.

**For instance, exit:**

The C library function void exit(int status) terminates the calling process immediately. Any open file descriptors belonging to the process are closed and any children of the process are inherited by process 1, init, and the process parent is sent a SIGCHLD signal.

Now,

**void exit ( int status );**

**2. The correct syntax for running two variable for loop simultaneously is.**

a) for (i = 0; i < n; i++)

for (j = 0; j < n; j += 5)

b) for (i = 0, j = 0;i < n, j < n; i++, j += 5)

c) for (i = 0; i < n;i++){}

for (j = 0; j < n;j += 5){}

d) None of the mentioned

Answer) b)

But, It will not work in the way you were thinking.

**Now, general syntax of for loop is:**

**for ( init; condition; increment ) {**

**statement(s);**

**}**

Now, the flow of the for loop is the following:

Here is the flow of control in a 'for' loop −

The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.

After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

Now, here the condition is:

i < n, j < n

Now, ‘,’ will act an operator. So, this loop will work on only j<n condition.

**Now, if both conditions are to be applied, you have to use logical operators.**

&& or || could make a good deal.

You need proof that , will act as operator and chooses the rightmost condition. See the following:

**#include<stdio.h>**

**int main()**

**{**

**int i=0;**

**int j=0;**

**int n=10;**

**for (i = 0, j = 0;i < n, j < n; i+=5, j += 1)**

**{**

**printf("i=%d j=%d\n",i,j);**

**}**

**return 0;**

**}**

**3. Which for loop has range of similar indexes of ‘i’ used in for (i = 0;i < n; i++)?**

a) for (i = n; i>0; i–)

b) for (i = n; i >= 0; i–)

c) for (i = n-1; i>0; i–)

d) for (i = n-1; i>-1; i–)

**Answer) d**

**4. Which of the following cannot be used as LHS of the expression in for (exp1;exp2; exp3) ?**

a) Variable

b) Function

c) typedef

d) macros

**Answer)**

**Are we talking about exp1?**

Exp1 is here the init condition.

As, the generic for loop syntax is   
  
**for ( init; condition; increment ) {**

**statement(s);**

**}**

**Now, macro would be used in the position of init condition**

**#include<stdio.h>**

**#define VAL 5**

**int main()**

**{**

**int i=0;**

**for(VAL;i<=5;i++)**

**{**

**printf("Printing here\n");**

**}**

**return 0;**

**}**

It will work fine.

A function call would work fine too.

**#include<stdio.h>**

**int foo()**

**{**

**return 5;**

**}**

**int main()**

**{**

**int i=0;**

**for(foo();i<=5;i++)**

**{**

**printf("Printing here\n");**

**}**

**return 0;**

**}**typedef directly wont work. As typedef defines a derived type variable. We need to define variable of the newly defined type and use the variable name.

**5. What is the output of this C code?**

#include <stdio.h>

int main()

{

short i;

for (i = 1; i >= 0; i++)

printf("%d\n", i);

}

a) The control won’t fall into the for loop

b) Numbers will be displayed until the signed limit of short and throw a runtime error

c) Numbers will be displayed until the signed limit of short and program will successfully terminate

d) This program will get into an infinite loop and keep printing numbers with no errors

**Answer) c) Numbers will be displayed until the signed limit of short and program will successfully terminate**

Now, it will be incremented until I’s value overflows for short range. Hence, the value will be -65536. (in 64 bit compiler, the sizeof(short) is 4)That time, loop control statement will return false and loop will terminate.

**6. What is the output of this C code?**

#include <stdio.h>

void main()

{

int k = 0;

for (k)

printf("Hello");

}

a) Compile time error

b) hello

c) Nothing

d) Varies

**Answer) a) Compile time error.**

A minimalist for loop will be for(;;). here, if k’s value is used as condition statement, the loop statement should be like: for(;k;)

**7. What is the output of this C code?**

#include <stdio.h>

void main()

{

int k = 0;

for (k < 3; k++)

printf("Hello");

}

a) Compile time error

b) Hello is printed thrice

c) Nothing

d) Varies

Answer) a) Compile time error.

To make hello printed thrice, the program should be changed to:

#include <stdio.h>

void main()

{

int k = 0;

for (;k < 3; k++)

printf("Hello");

}

**8. What is the output of this C code?**

#include <stdio.h>

void main()

{

double k = 0;

for (k = 0.0; k < 3.0; k++)

printf("Hello");

}

a) Run time error

b) Hello is printed thrice

c) Hello is printed twice

d) Hello is printed infinitely

**Answer) b)**

1. **What is the output of the following code:**

#include <stdio.h>

void main()

{

double k = 0;

for (k = 0.0; k < 3.0; k++);

printf("%lf", k);

}

a) 2.000000

b) 4.000000

c) 3.000000

d) Run time error

**Answer) c) 3.000000. Because, the loop will terminated when k=3.000000**

**10. What is the output of this C code?**

#include <stdio.h>

void main()

{

int k;

for (k = -3; k < -5; k++)

printf("Hello");

}

a) Hello

b) Infinite hello

c) Run time error

d) Nothing

**Answer) d) Nothing.**

**11. What is the output of this C code?**

#include <stdio.h>

int main()

{

int i = 0;

for (; ; ;)

printf("In for loop\n");

printf("After loop\n");

}

a) Compile time error

b) Infinite loop

c) After loop

d) Undefined behaviour

**Answer) a) Compile Time error.**

for(;;) is valid. For(;;;) is not.

**12. What is the output of this C code?**

#include <stdio.h>

int main()

{

int i = 0;

for (i++; i == 1; i = 2)

printf("In for loop ");

printf("After loop\n");

}

a) In for loop after loop

b) After loop

c) Compile time error

d) Undefined behaviour

**Answer) a)In for loop After loop.**

Now, every semicolon in for loop statement separating init, cond and incrementwill act as a sequence point. Now, i is post incremented. But, I’s modified value is available in the memory or cache before condition statement executes. So, when condition statement is executed I is already 1. Hence, conditional statement returns true. And control will enter for loop. Printing **In for loop** once. Now, the increment statement will make i as 2. Hence, i==1 (the conditional statement) will return false. So, control wont enter loop again. Now, outside loop **After loop** is printed.

**13. What is the output of this C code?**

#include <stdio.h>

int main()

{

int i = 0;

for (foo(); i == 1; i = 2)

printf("In for loop\n");

printf("After loop\n");

}

int foo()

{

return 1;

}

a) After loop

b) In for loop after loop

c) Compile time error

d) Infinite loop

**Answer) is a) After loop**

**15. What is the output of this C code?**

#include <stdio.h>

int main()

{

int \*p = NULL;

for (foo(); p; p = 0)

printf("In for loop\n");

printf("After loop\n");

}

a) In for loop after loop

b) Compile time error

c) Infinite loop

d) Depends on the value of NULL

**Answer) b) Compile time error.**

**Because, of undefined reference to foo().**

**Whereas, the following wont invoke compilation error.**

#include <stdio.h>

int main()

{

int \*p = NULL;

for (foo(); p; p = 0)

printf("In for loop\n");

printf("After loop\n");

}

int foo()

{

return 5;

}

And, it will print **After loop.**

**16. What is the output of this C code?**

#include <stdio.h>

int main()

{

for (int i = 0;i < 1; i++)

printf("In for loop\n");

}

a) Compile time error

b) In for loop

c) Depends on the standard compiler implements

d) Depends on the compiler

**Answer) c) Depends on the standard compiler implements.**

‘for’ loop initial declarations are only allowed in C99 mode

note: use option -std=c99 or -std=gnu99 to compile your code