The comma operator in c comes with the lowest precedence in the C language. The comma operator is basically a binary operator that initially operates the first available operand, discards the obtained result from it, evaluates the operands present after this, and then returns the result/value accordingly.

In this article, we will take a look at comma operators in C according to the

Why Do We Use Comma Operators in C?

The comma sign is used for mainly two different purposes in the C language – as an operator and as a separator. Thus, its behavior is very different according to where we place/use it in a program.

The Comma as a Separator

When we want to declare multiple numbers of variables in any program and provide different arguments in any function, we use the comma in the form of a separator in the program.

For instance,

int x, y, z;

In the statement mentioned above, the comma acts as a separator and informs the compiler that the x, y, and z variables are three different types of variables.

Example

#include<stdio.h>

int main() {

int m = 5, n = 10;

void function(m, n);

}

Here, the program will use both m and n as function parameters. Take a look at the following program to understand how we use comma operators in the C language.

#include<stdio.h>

main() {

int p = 35;

int q = (p++, ++p);

printf(“%d”, q);

}

### The Comma as an Operator

We use the comma in the form of an operator when we want to assign multiple numbers of values to any variable in a program with the help of a comma.

For instance,

p = 40, 50, 60, 70, 80, 90;

q = (40, 50, 60, 70, 80, 90);

In the very first statement mentioned above, the value assigned to the variable p will be equal to 40. It is because the (=) equal to assignment operator has a higher priority as compared to the (,) comma operator. Thus, the program will assign the variable x with a value of 40.

In the case of the second statement, the value of q will be equivalent to 90. It is because the values 40, 50, 60, 70, 80 and 90 are enclosed in the form of braces () and these braces have a higher priority, as compared to the equal to (=) assignment operator. Thus, when we provide the variable q with multiple values (using the comma operator and the braces), then the program will consider the right-most value as the result/output of this expression. Thus, the program variable q will be assigned with the value of 90.

### Example

#include<stdio.h>

int main() {

int m = (10, 20);

int n = (func1(), func2());

}

In this case, the program will assign the value 20 to the variable m. Then, for the next statement in the program, the execution of func1() will occur first. Only after that will the execution of the second one occur.

Let us consider another program in C,

#include <stdio.h>

int main()

{

int x,y;

x = 40,50,60;

y = (40,50,60);

//asking the compiler to print the values assigned

printf(“x= %d, y= %d\n”,x,y);

return 0;

}

Output

x= 40, y= 60

## Let us take a look at a confusing program.

Read the set of code given below very carefully before we predict the output for it:

#include <stdio.h>

int main()

{

int x= 70,80,90;

int y;

y= (70,80,90);

//asking the compiler to print the values assigned

printf(“x= %d, y= %d\n”,x,y);

return 0;

}

Here, if you think that the output generated here would be equal to x= 70, y= 90, then you are absolutely wrong!

For easier understanding, consider the statement that says:

int s= 50,70,90;

This statement is basically a declaration that also contains an initialisation statement. The comma is used in the form of a separator here, and thus, we cannot use such values in a program.

The actual output of the code would be equal to a compile-time error since the comma is invalid in such a given circumstance.

## Practice Problems on Comma Operators in C

**1.**What would be the output obtained out of the program mentioned below?

#include<stdio.h>

main() {

int g = 83;

int h = (g++, ++g);

printf(“%d”, h);

}

**A.**g = 83

**B.**g = 85

**C.**h = 85

**D.**h = 83

**Answer : C.**h = 85

**2.**What would be the output obtained out of the program mentioned below?

#include <stdio.h>

int main()

{

int j,k;

j = 25,45,65;

k = (25,45,65);

printf(“j= %d, k= %d\n”,j,k);

return 0;

}

**A.**j= 25, k= 45

**B.**j= 25, k= 65

**C.**Compile-time Error

**D.**j= 45, k= 65

**Answer : B.**j= 25, k= 65

**3.**What would be the output obtained out of the program mentioned below?

#include <stdio.h>

int main()

{

int v= 12,13,14;

int z;

z= (12,13,14);

printf(“v= %d, z= %d\n”,v,z);

return 0;

}

**A.**v= 13, z= 14

**B.**v= 12, z=13

**C.**v= 12, z= 14

**D.**Compile-time Error

**Answer: D.**Compile-time Error