

# Programming Errors in C

Errors are the problems or faults that occur in the program, which make the program's behavior abnormal. Programming errors are also known as bugs or defects, and the process of removing these bugs is known as **debugging**.

These errors are detected either during the time of compilation or execution.

There are five types of errors exist in C programming:

- Syntax error
- Run-time error
- Linker error
- Logical error
- Semantic error

Types of  
errors

1

Syntax error

2

Run-time error

3

Linker error

4

Logical error

5

Semantic error

## Syntax error

These errors mainly occur due to mistakes while typing or do not follow the syntax of the specified programming language. These errors can be easily debugged.

### For example:

If we want to declare the variable of type integer,

`int a; // this is the correct form.`

`Int a; // this is an incorrect form.`

### Commonly occurred syntax errors are:

- If we miss the parenthesis (}) while writing the code.
- Displaying the value of a variable without its declaration.
- If we miss the semicolon (;) at the end of the statement.

### Let's understand through an example.

```
#include <stdio.h>
int main()
{
    a = 10;
    printf("The value of a is : %d", a);
    return 0;
}
```

## Output

Compilation failed due to following error(s).

```
main.c: In function 'main':
main.c:13:5: error: 'a' undeclared (first use in this function)
    a = 10;
    ^
main.c:13:5: note: each undeclared identifier is reported only once for each function it appears in
```

In the above output, we observe that the code throws the error that 'a' is undeclared. This error is nothing but the syntax error only.

## Run-time error

Sometimes the errors exist during the execution time even after the successful compilation known as run-time errors. When the program is running, and it is not able to perform the operation is the main cause of the run-time error. The division by zero is a common example of the run-time error. These errors are very difficult to find, as the compiler does not point to these errors.

**Let's understand through an example.**

```
#include <stdio.h>

int main()
{
    int a=2;
    int b=2/0;
    printf("The value of b is : %d", b);
    return 0;
}
```

## Output

```
main.c:14:12: warning: division by zero [-Wdiv-by-zero]
Floating point exception

...Program finished with exit code 136
Press ENTER to exit console. □
```

## Linker error

This can happen either due to the wrong function prototyping or the usage of the wrong header file. For example, the main.c file contains the sub() function whose declaration and definition are done in some other file such as func. c. During the compilation, the compiler finds the sub() function in func.c file, so it generates two object files, i.e., main. o and func. o. At the execution time, if the definition of sub() function is not found in the function.o file, then the linker error will be thrown. The most common linker error that occurs is that we use Main() instead of main().

**Let's understand through a simple example.**

```
#include <stdio.h>
int Main()
{
    int a=78;
    printf("The value of a is : %d", a);
    return 0;
}
```

## Output

```
(.text+0x20): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

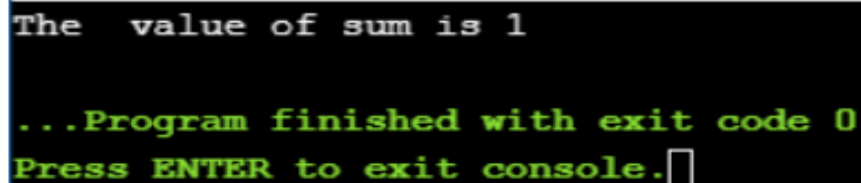
## Logical error

A logical error is an error that leads to an undesired output. These errors produce incorrect output, but they are error-free, known as logical errors. The occurrence of these errors mainly depends upon the logical thinking of the developer.

**Let's understand through an example.**

```
#include <stdio.h>
int main()
{
    int sum=0; // variable initialization
    int k=1;
    for(int i=1;i<=10;i++); // logical error, as we put the semicolon after loop
    {
        sum=sum+k;
        k++;
    }
    printf("The value of sum is %d", sum);
    return 0;
}
```

## Output



```
The value of sum is 1

...Program finished with exit code 0
Press ENTER to exit console.□
```

## Semantic error

Semantic errors are the errors that occur when the statements are not understandable by the compiler.

**The following can be the cases for the semantic error:**

- Use of a un-initialized variable.

```
int i;  
i=i+2;
```

- Type compatibility

```
int b = "javatpoint";
```

- Errors in expressions

```
int a, b, c;  
a+b = c;
```

- Array index out of bound

```
int a[10];  
a[10] = 34;
```

**Let's understand through an example.**

```
#include <stdio.h>  
int main()  
{  
    int a,b,c;  
    a=2;  
    b=3;  
    c=1;  
    a+b=c; // semantic error  
    return 0;  
}
```

In the above code, we use the statement `a+b=c`, which is incorrect as we cannot use the two operands on the left-side.

## Output

Compilation failed due to following error(s).

```
main.c: In function 'main':
```

```
main.c:17:6: error: lvalue required as left operand of assignment
```

```
    a+b=c;
```

```
        ^
```