Errors

Errors

An error can be defined as a mistake done by the programmer which results in abnormal working of a program.

Syntax error

Errors that occur when rules of writing C programming syntax are violated. Syntax refers to the rules that define the structure of a language. The syntax error is an incorrect construction of the source code.

These errors are detected by the compiler and also called as **compile time errors.**

Example: missing; after a statement, unmatched braces or incorrect variable declarations.

Syntax error

```
//missing semi colon
#include <stdio.h>
int main() {
    int a, b, c;
    a = 2; b = 3
    c = a + b;     printf("The sum of a and b is %d", c);
    return 0;   }
```

Logical error

These are the type of error which provide incorrect output but appears to be error free.

On compilation and execution of program, desired output is not obtained.

These types of errors totally depend upon logical thinking of a program.

Logical error

//example: a-b instead of a+b, addition of 2 numbers

```
#include <stdio.h>
int main() {
    int a, b, c;
    a = 2;     b = 3;     c = a - b;
    printf("The sum of a and b is %d", c);
    return 0;   }
```

Semantic error

Errors due to an improper use of program statements.

Semantic errors refer to logical errors or incorrect program logic that leads to undesired output.

A semantic occurs due to the wrong use of the variables. The syntax of the programming language would be correct but the problem may be caused due to the use of wrong variables, wrong operations or in a wrong order. The compiler can’t catch this error.

Semantic error

For example, if you are using an uninitialized variable as the code given below:

```
#include<stdio.h>

void main()

{

int j;

 j++;

printf("%d",j);

}
```

In the code snipped given above the variable j is uninitialized but it is post-incremented.

Semantic error

Example 2: Type incompatibility:

```
int a = "hello";
```

Run time errors:

The errors that occur during the program execution (runtime) that is after successful compilation.

These types of errors are difficult to find as the compiler does not point to the line at which error occur. A program with runtime errors can compile successfully, but unexpected events or situations may cause the program to crash or result in incorrect output when executed.

eg. Division by zero, Array index out of bounds, Null pointer dereferencing

Run time errors:

```
 //division
#include <stdio.h>
int main() {
    int a, b;
    a = 2;     b = 0;     c=a/b ;
    printf("division  %d", c);
    return 0;   }
```

Linker Error

These type of error occurs when the linker is not able to find the function definition of the given function.

Ex. We have used pow() without including math.h header file.

Linker Error

```
 //addition of 2 numbers
#include <studio.h>    // studio instead of stdio
int main() {
    int a, b, c;     a = 2;
    b = 3;     c = a - b;
    printf("The sum of a and b is %d", c);
    return 0;   }
```

Error

From the point of view of when errors are detected.

Compile time errors: syntax errors and static semantic errors indicated by the compiler.

Runtime errors: dynamic semantic errors, and logical errors, that cannot be detected by the compiler (debugging).

Difference between compile time and runtime error

Parameter	Compile-time Errors	Runtime Errors
Definition	Occur during compilation	Occur during program execution
Detection	Detected by the compiler	Detected by the program
Occurrence	Occur when syntax or semantic rules are violated	Occur when the program encounters unexpected or erroneous conditions during execution
Causes	Syntax errors, semantic errors, linker errors	Division by zero, null pointer dereference, out-of-bounds array access, memory leaks, and other runtime-specific issues
Debugging	Debugging symbols are used to identify and fix errors	Debugging tools such as exception handling and tracing are used to identify and fix errors
Impact	Prevent the program from running or generating an executable file	May cause the program to crash or produce unexpected results during runtime

Thank You