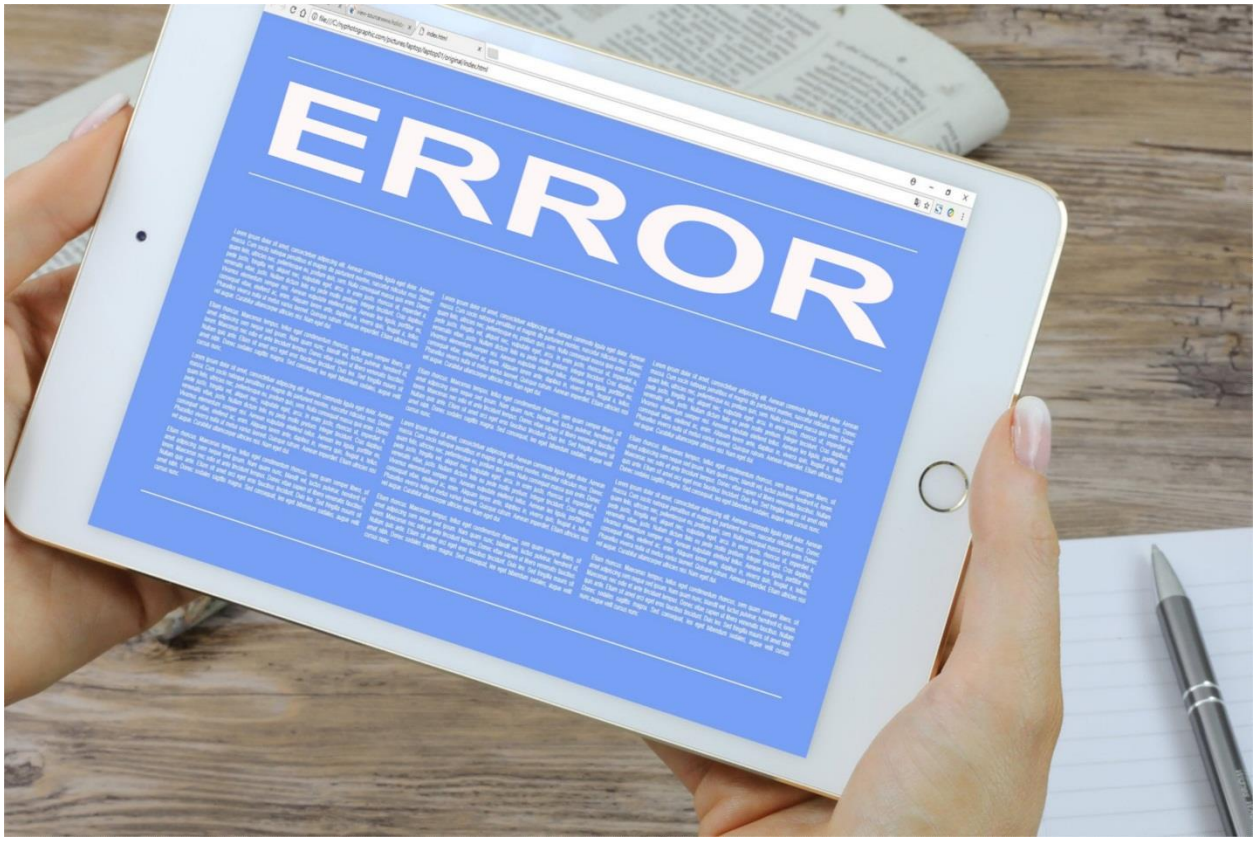
Errors in C



BY / SOHAIB DAR

An error in the C language is an issue that arises in a program, making the program not work in the way it was supposed to work or may stop from compiling.

If an error appears in a program, the program can do one of the following three things:

- the code will not compile, the program will stop working during execution,

➢ or

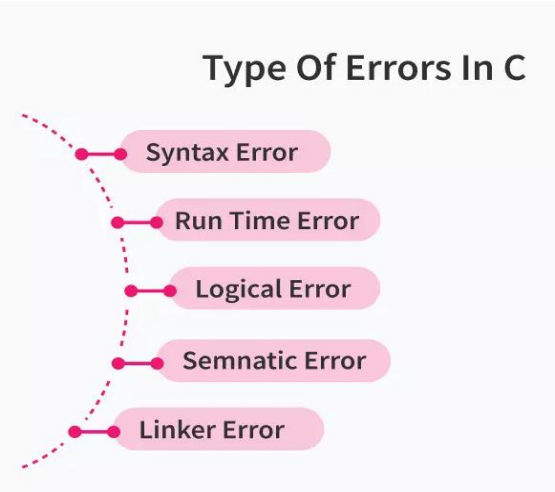
- the program will generate garbage values

➢ or

- an incorrect output.

There are five different types of errors in C.

1. Syntax Error
2. Run Time Error
3. Logical Error
4. Semantic Error
5. Linker Error



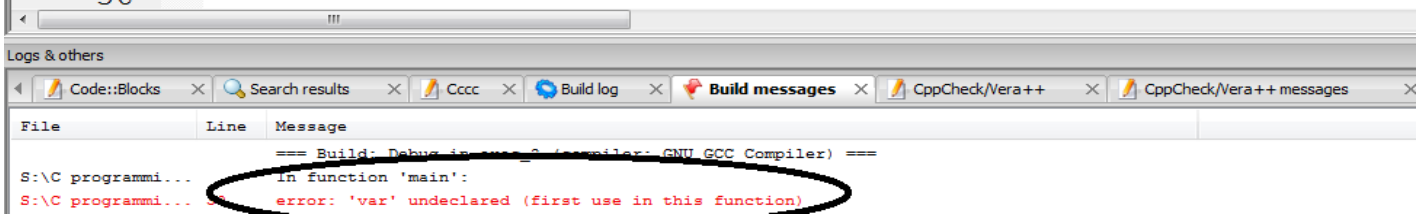
1. Syntax Error

- Syntax errors occur when a programmer **makes mistakes in typing the code's syntax correctly or makes typos.**
- Syntax errors occur when a programmer **does not follow the set of rules defined for the syntax of C language.**
- Syntax errors are also **called compilation errors** because they are always **detected by the compiler.**
- These errors can be easily **debugged** or **corrected.**

The most commonly occurring syntax errors in C language are:

- **Missing semi-colon (;)**
- **Missing parenthesis ({})**
- **Assigning value to a variable without declaring it.**

```
28  int main()
29  {
30      var = 5;     // we did not declare the data type of variable
31
32      printf("The variable is: %d", var);
33      return 0;
34  }
35
36
```



```
28  int main()
29  {
30      int var = 5;
31      printf("The variable is: %d", var)
32      return 0;
33  }
34
35
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | Build messages | Cp

File	Line	Message
		=== Build: Debug in exec_2 (compiler: GNU GCC Compiler) ===
S:\C programmi...		In function 'main':
S:\C programmi...	32	error: expected ';' before 'return'

```
28  int main()
29  {
30      Int va_1 = 5;
31      return 0;
32  }
33
34
```

Logs & others

Code::Blocks | Search results | Cccc | Build log

File	Line	Message
		=== Build: Debug in exec_2 (compiler
S:\C programmi...		In function 'main':
S:\C programmi...	30	error: unknown type name 'Int'

2. Run Time Error

- **Errors that occur during the execution (or running) of a programare.**
- **These errors occur after the the successful compilation.**
- **When a program is running, and it is not able to perform any particular operation, it means that we have encountered a run time error.**
- **Run time errors can be a little tricky to identify because the compiler can not detect these errors**.
- **Some of the most common run time errors are**:
  - number not divisible by zero,
  - array index out of bounds,
  - string index out of bounds.

Run time errors can occur because of various reasons. Some of the reasons are:

1. **Mistakes in the Code: during the execution of a *while* loop,**

   forgets to enter a break statement. This will lead the program to run infinite times, hence resulting in a run time error.

2. **Memory Leaks: If a programmer creates an array in the heap but**

   forgets to delete the array's data, the program might start leaking memory, resulting in a run time error.

3. **Mathematically Incorrect Operations: Dividing a number by zero, or calculating the square root of -1 will also result in a run time error.**

```
2.000000
1.732051
1.414214
1.000000
0.000000
-nan
-nan
```

```c
int main()
{
    for (int i = 4; i >= -2; i--){
        printf("%f", sqrt(i));
        printf("\n");
    }
    return 0;
}
```

```
"S:\C programming\c sheets\Solution of sheet 2\...
2.000000
1.732051
1.414214
1.000000
0.000000
-1.#IND00
-1.#IND00

Process returned 0 (0x0)   execution time : 0.0
Press any key to continue.
```

- **In the above example, we used a for loop to calculate the square root of six integers. But because we also tried to calculate the square root of two negative numbers, the program generated two errors.**

- **(the** IND **written above stands for "Ideterminate"). These errors are the run time errors.** -nan **is similar to IND.**

```c
int main()
{
    int var = 2147483649;

    printf("%d\n", var);
    return 0;
}
```

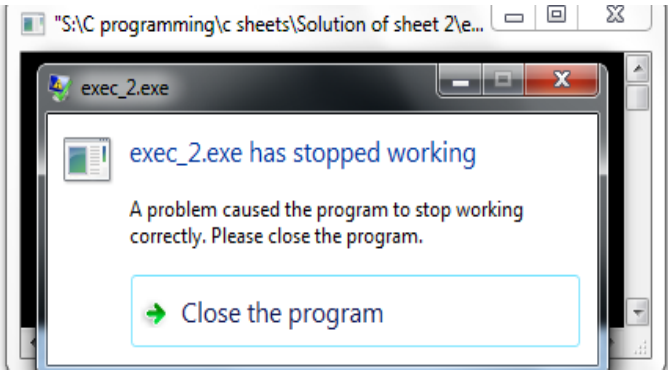
```
"S:\C programming\c sheets\Solution of sheet 2\exec_2\bi...
-2147483647

Process returned 0 (0x0)   execution time : 0.018 s
```

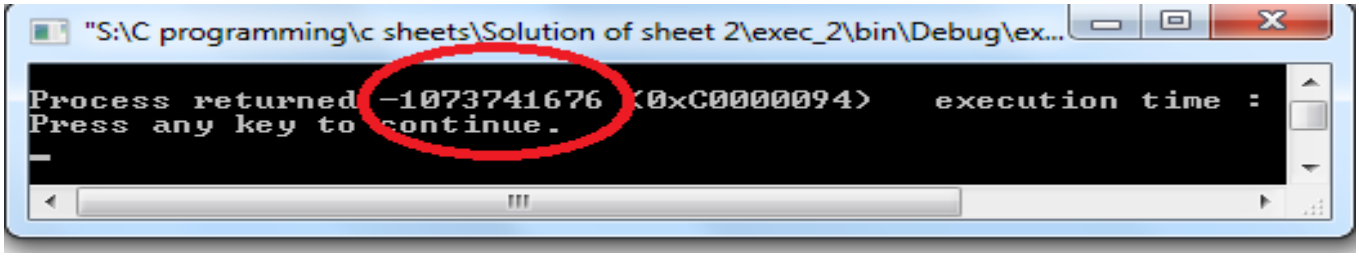
This is an integer overflow error. The maximum value an integer can hold in C is 2147483647. Since in the above example, we assigned 2147483649 to the variable var, the variable overflows, and we get -2147483647 as the output.

---

crashed : divided by zero (run time error)

```
27
28  int main()
29  {
30      int x = 6, y = 0, z;
31      z = x / y;
32       printf("%d\n", z);
33       return 0;
34  }
35
```





3. Logical Error

- **Logical errors are those errors in which we think that our code is correct, the code compiles without any error and gives no error while it is running, but the output we get is different from the output we expected.**
- **The logical error is an error that leads to an undesired output.**
- **The occurrence of these errors mainly depends upon the logical thinking of the developer.**
- **In 1999, NASA lost a spacecraft due to a logical error. This happened because of some miscalculations between the English and the American Units. The software was coded to work for one system but was used with the other.**

Logic errors can be caused by:-

- **incorrectly using logical operators, eg expecting a program to stop when the value of a variable reaches 5, but using <5 instead of <=5**
- **incorrectly using Boolean operators**
- **unintentionally creating a situation where an infinite loop may occur**
- **incorrectly using brackets in calculations**
- **unintentionally using the same variable name at different points in the program for different purposes.**
- **using incorrect program design.**
  - **Unlike a syntax error, a logic error does not usually stop a program from running. The program will run, but not function as expected.**

```c
int main()
{
  float a = 10;
    float b = 5;

    if (b = 0) {   // we wrote = instead of ==
        printf("Division by zero is not possible");
    } else {
        printf("The output is: %f", a/b);
    }
    return 0;
}
```

```
"S:\C programming\c sheets\Solution of sheet 2\exec_2\...
The output is: 1.#INF00
Process returned 0 (0x0)   execution time : 0.020 s
Press any key to continue.
```

INF signifies a division by zero error.

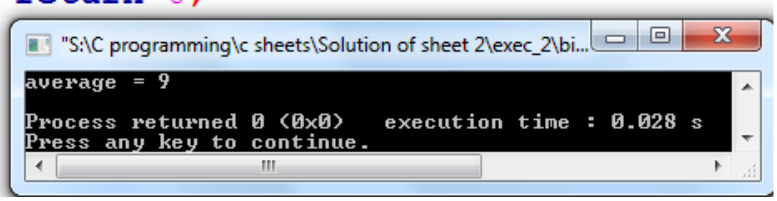
In the above example, we wanted to check whether the variable b was equal to zero. But instead of using the equal to comparison operator (==), we use the assignment operator (=). Because of this, the if statement became false and the value of b became 0. Finally, the else clause got executed.

```c
int main()
{
    int number;

    for (number = 1; number <=7; number++);
    {
        printf("Number = %i\n",number);
    }
    return 0;
}
```

```
"S:\C programming\c sheets\Solution of sheet 2\exec_2\bin...
Number = 8
Process returned 0 (0x0)   execution time : 0.019 s
```

```
28  int main()
29  {
30      int x=6, y = 7, avg;
31      avg = x + y /2;
32         printf("average = %i\n",avg);
33      return 0;
34  }
35
36
37
38
```

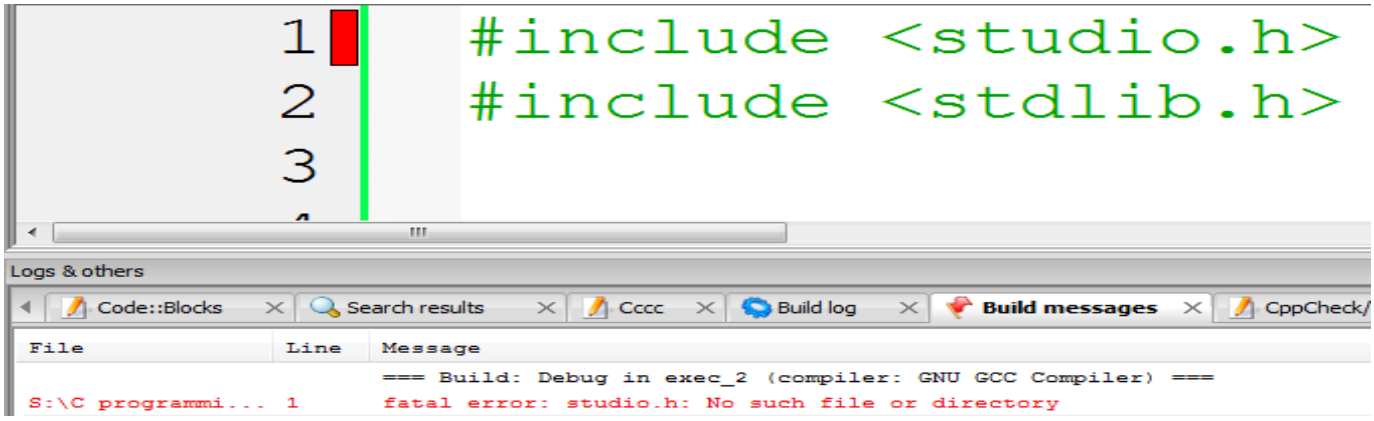
error:bracket is not used formula



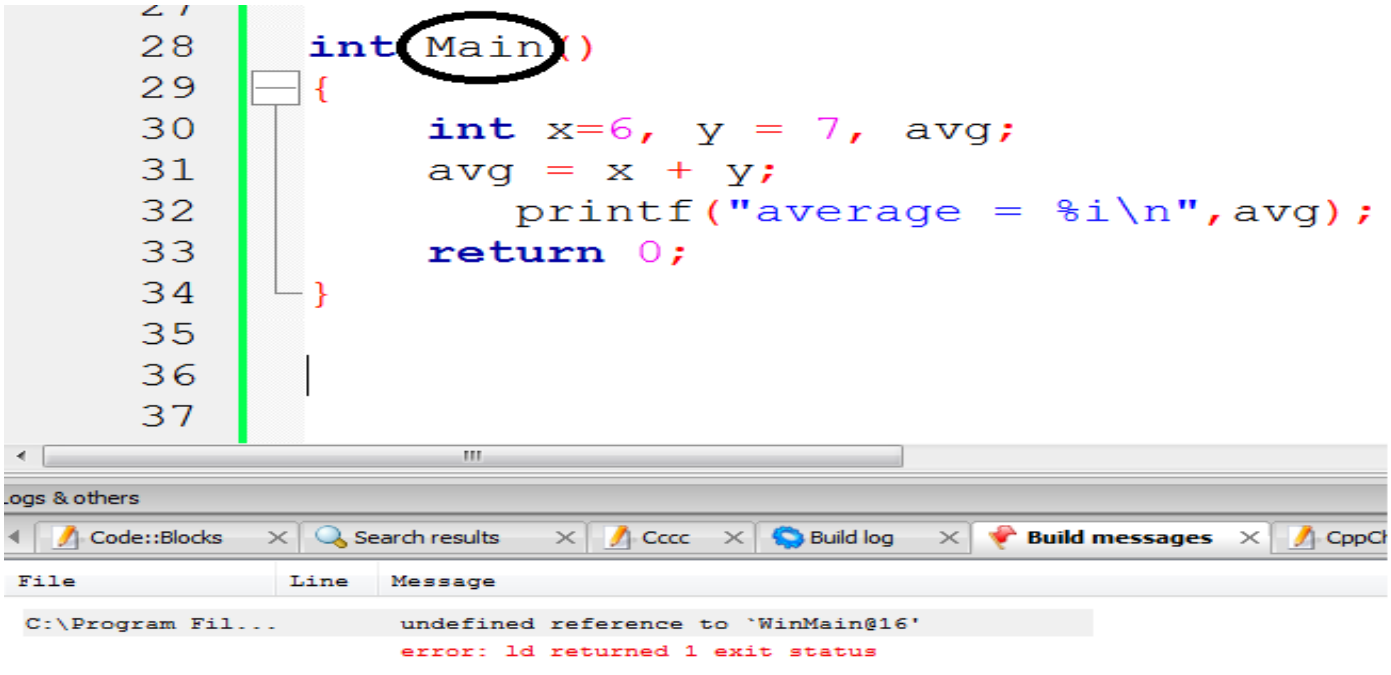
4. Linker Error

Linker is a program that takes the object files generated by the compiler and combines them into a single executable file.

- Linker errors are the errors encountered when the executable file of the code can not be generated even though the code gets compiled successfully.
- This error is generated when a different object file is unable to link with the main object file.
- This can be happened either due to the wrong function prototyping or usage of the wrong header file.



- Error: No such header file or directly



- **We wrote Main() instead of main(), the program generated a linker error.**
- **This happens because every file in the C language must have a main() function.**
- **As in the above program, we did not have a main() function, the program was unable to run the code, and we got an error.**

5. Semantic Error

- **Errors that occur because the compiler is unable to understand the written code are called Semantic Errors.**
- **A semantic error will be generated if the code makes no sense to the compiler, even though it is syntactically correct.**
- **Semantic errors are different from syntax errors, as syntax errors signify that the structure of a program is incorrect without considering its meaning. On the other hand, semantic errors signify the incorrect implementation of a program by considering the meaning of the program.**
  - **It is like using the wrong word in the wrong place in the English language.**
    For example, adding a string to an integer will generate a semantic error.

The most commonly occurring semantic errors are:

- **Use of un-initialized variables.**
- **Type compatibility.**
- **Array index out of bounds.**
- **Errors in expressions.**

- type compatibility

```
26
27
28  int main()
29  {
30      unsigned int index = "Sohaib Dar";
31
32      return 0;
33  }
```

- Errors in expressions

```
26
27
28  int main()
29  {
30      unsigned int number1, number2, number3
31      number1 + number2  = number3;
32      return 0;
33  }
```

- use of un-initialized variables

```
26
27
28  int main()
29  {
30      unsigned int number;
31      number = number /4;
32      return 0;
33  }
```

- array index out of bounds.

```
26
27
28  int main()
29  {
30      unsigned int Data[10];
31      Data[20] = 25;
32      return 0;
33  }
```

Conclusion

- **There are 5 different types of errors in C programming language: Syntax error, Run Time error, Logical error, Semantic error, and Linker error.**

- **Syntax errors, linker errors, and semantic errors can be identified by the compiler during compilation. Logical errors and run time errors are encountered after the program is compiled and executed.**

- **Syntax errors, linker errors, and semantic errors are relatively easy to identify and rectify compared to the logical and run time errors. This is so because the compiler generates these 3 (syntax, linker, semantic) errors during compilation itself while the other 2 errors are generated during or after the execution.**

The source Code is on My Github

- **Sheet #2 : https://github.com/SohaibDar61/Promblem-Solving-in-C/tree/main/Sheet%20%232**
- **sheet #1 : https://github.com/SohaibDar61/Codeforces-promblem-solving-/tree/main/Sheet%20%231**

Data Structure & Algorithms

- **https://github.com/SohaibDar61/Data_Structures**

E-mail: eng.sohaibdar@gmail.com
Phone:+201018741441
Upwork: https://www.upwork.com/freelancers/~019e71f2adc499c4e8
LinkedIn: https://www.linkedin.com/in/sohaibdar61

- **Anyone can edit or optimize the code , and I hope that revision helps and gets better.**

Thank You

#S