1

TYPES OF ERRORS

by
Dr. Sumit Srivastava
Dept. of Computer Science & Engineering

Unit -I

1

Programming Errors in C

2

- When we program, we have to deal with errors. Our most basic aim is correctness, but we must deal with incomplete problem specifications, incomplete programs, and our own errors.
- Here, we'll concentrate on a: how to deal with errors and techniques for finding errors in programs.

2

Errors

3

- *" ... I realized that from now on a large part of my life would be spent finding and correcting my own mistakes."*

  Maurice Wilkes, 1949
- When we write programs, errors are natural and unavoidable; the question is, how do we deal with them?
  - Organize software to minimize errors.
  - Eliminate most of the errors we made anyway.
  - Make sure the remaining errors are not serious.
- Solution is that avoiding, finding, and correcting errors.

3

Errors

4

- Errors, in general, are referred to as an action or fault or problem which is incorrect or makes the program behavior abnormal.
- In C programming language, the programming errors are bugs or faults that occur during runtime or compile time.
- 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 incorrect output.

4

Our Program

5

1. Should give the desired results for all legal inputs
2. Should give reasonable error messages for illegal inputs
3. Need not worry about misbehaving hardware
4. Need not worry about misbehaving system software
5. Is allowed to terminate after finding an error
   - 3, 4, and 5 are true for beginner's code; often, we have to worry about those in real software.

5

Sources of errors

6

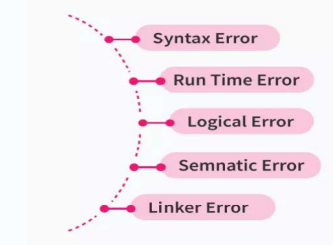
- Poor specification
  - "What's this supposed to do?"
- Incomplete programs
  - "but I'll not get around to doing that until tomorrow"
- Unexpected arguments
  - "but **sqrt()** isn't supposed to be called with **-1** as its argument"
- Unexpected input
  - "but the user was supposed to input an integer"
- Code that simply doesn't do what it was supposed to do
  - "so fix it!"

6

Types of Errors

7

- There are five different types of errors in C Programming:



7

Syntax Error

8

- Syntax errors occur when a programmer makes mistakes in typing the code's syntax correctly or makes typos.
- It occur when a programmer does not follow the set of rules defined for the syntax of C language.
- Syntax errors are sometimes also called compilation errors because they are always detected by the compiler. Generally, these errors can be easily identified and rectified by programmers.

8

Syntax Error

9

- This kind of errors are occurred, when it violates the rule of C writing techniques or syntaxes.
- This kind of errors are generally indicated by the compiler before compilation.
- Sometimes these are known as compile time error.

9

Syntax Error

10

- The most commonly occurring syntax errors in C language are:
  - Missing semi-colon (;)
  - Missing parenthesis ({})
  - Assigning value to a variable without declaring it

10

Syntax Error (Example)

11

- Let us take an example to understand syntax errors:

```
#include <stdio.h>

void main() {
    var = 5;        // We did not declare the data type of variable

    printf("The variable is: %d", var);
}
```

- Output:

```
error: 'var' undeclared (first use in this function)
```

- *If the user assigns any value to a variable without defining the data type of the variable, the compiler throws a syntax error.*

11

Syntax Error (Example)

12

Code:

```
#include <stdio.h>
intmain()
Int 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:4:5: error: unknown type name 'Int'
    Int a = 10;
    ^~~
```


12

Syntax Error (Example)

13

- Code:

```
#include<stdio.h>
main()
{
printf("Hello World")
}
```

- Output:

  expected ';' before '}' token

13

Runtime Error

14

- This kind of errors are occurred, when the program is executing.
- As this is not compilation error, so the compilation will be successfully done.
- This error occurs mainly when the program is still running, and it will not be able to perform some particular operation in the main which may lead to memory leakage.

14

Runtime Error

15

- Runtime errors can be a little tricky to identify because the compiler can not detect these errors.
- They can only be identified once the program is running.
- Some of the most common run time errors are: number not divisible by zero, array index out of bounds, string index out of bounds, etc.

15

Runtime Error

16

- Runtime errors can occur because of various reasons. Some of the reasons are:

1. Mistakes in the Code: Let us say during the execution of a while loop, the programmer 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.
4. Undefined Variables: If a programmer forgets to define a variable in the code, the program will generate a run time error.

16

Runtime Error

17

- Let us consider an array of length 5 i.e. array[5], but during runtime, if we try to access elements i.e. array[10] then we get segmentation fault errors called runtime errors. Giving only an array length of 5

```
// C program to demonstrate
// a runtime error
#include <stdio.h>

// Driver code
int main()
{
  int array[5];
  printf("%d", array[10]);
  return 0;
}
```

- Output: -621007737

But in output trying to access more than 5 i.e. if we try to access array[10] during runtime then the program will throw an error or will show an abnormal behavior and print any garbage value.

17

Runtime Error (Example)

18

we used a for loop to calculate the square root of six integers. But because we also tried calculating the square root of two negative numbers, the program generated two errors (the IND written above stands for "Indeterminate")

```
// A program that calculates the square root of integers
#include <stdio.h>
#include <math.h>

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

```
2.000000
1.732051
1.414214
1.000000
0.000000
-1.#IND00
-1.#IND00
```

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


18

Runtime Error (Example)

19

```c
#include<stdio.h>

void main() {
    int var = 2147483649;

    printf("%d", var);
}
```

Output:

```
-2147483647
```

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 (because of the circular property)

19

Runtime Error (Example)

20

Code:

```c
#include<stdio.h>
main() {
   int x = 52;
   int y = 0;
   printf("Div : %f", x/y);
}
```

Output:

Program crashes during runtime.

20

Logical Errors

21

- Sometimes, we do not get the output we expected after the compilation and execution of a program. Even though the code seems error free, the output generated is different from the expected one. These types of errors are called Logical Errors.
- 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.

21

Logical Errors

22

- Even if the syntax and other factors are correct, we may not get the desired results due to logical issues. But they seem to be error-free. These are referred to as logical errors.
- We sometimes put a semicolon after a loop, which is syntactically correct but results in one blank loop. In that case, it will display the desired output.

22

Logical Errors

23

- *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.*

23

Logical Errors (Example)

24

```c
#include <stdio.h>

void 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);
    }
}
```

Output:

```
The output is: inf
```

INF signifies a division by zero error. In the above example, at line 8, 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.

24

Logical Errors (Example)

25

```
#include<stdio.h>
main() {
   int i;
   for(i = 0; i<5; i++); {
      printf("Hello World");
   }
}
```

- Output:

  Hello World

In this example, the for loop iterates 5 times but the output will be displayed only one time due to the semicolon at the end of for loop.

25

Logical Errors (Example)

26

```
#include <stdio.h>
intmain()
{
int sum=0;
int a=1;
for(inti=1;i<=20;i++);
{
sum=sum+a;
a++;
}
printf("The sum of the numbers is %d", sum);
return 0;
}
```

- Output:

```
The sum of the numbers is 1
```

The above program is to print the output as the sum of numbers less than 20, but as we have specified semicolon after the for loop statement, so it gives the wrong output where statements inside for loop are not executed.

26

Semantic Error

27

- This kind of error occurs when it is syntactically correct but has no meaning. This is like grammatical mistakes.
- These errors have occurred when the program statements are not correctly written which will make the compiler difficult to understand such as the use of the uninitialized variable, type compatibility, errors in writing expressions, etc.
- The most commonly occurring semantic errors are: use initialized variables, errors in writing expressions, type compatibility, and array index out of bounds.

27

Semantic Error

28

- 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.
- 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.

28

Semantic Error (Example)

29

```
#include<stdio.h>

void main() {
    int a, b, c;

    a * b = c;
    // This will generate a semantic error
}
```

Output:

```
error: lvalue required as left operand of assignment
```

When we have an expression on the left-hand side of an assignment operator (=), the program generates a semantic error. Even though the code is syntactically correct, the compiler does not understand the code.

29

Semantic Error (Example)

30

```
#include<stdio.h>

main() {
    int x, y, z;
    x = 10;
    y = 20;
    x + y = z;
}
```

- Output:

  [Error] lvalue required as left operand of assignment

If some expression is given at the left side of assignment operator, this may generate semantic error.

30

Linker error

31

- This kind of errors are occurred, when the program is compiled successfully, and trying to link the different object file with the main object file.
- When this error is occurred, the executable is not generated.
- This could be due to incorrect function prototyping, an incorrect header file, or other factors

31

Linker error (Example)

32

```c
#include <stdio.h>

void Main() {
    int var = 10;
    printf("%d", var);
}
```

- Output:

```
undefined reference to `main'
```

In the above code, as 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. This is one of the most common type of linker error.

32

Conclusion

33

- There are 5 different types of errors in C programming language: Syntax error, Runtime 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.

33

End of Today's Lecture

34

Doubts && Queries?

34

35

THANK YOU

35