IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3, May 2014.
www.ijiset.com
ISSN 2348 - 7968

Analysis of Compilation Errors, Runtime Errors, Reliability and Validity of a Program: A Case Study on C-language

Manoj Kumar Srivastav [1] , Dr.Asoke Nath[2]

[1]Indira Gandhi National Open University,St.Xavier's college(Autonomous)
Kolkata,West Bengal, India

[2]Department of computer Science,St.Xavier's college(Autonomous)
Kolkata,West Bengal, India

Abstract

When a program compiled using any compiler then it may display some error messages if the program has some syntax or grammatical mistakes. If there is no syntax error then also the program may not produce any output on the other hand as the compiler may display some run time error which are also called logical error. Apart from that the program produce some unwanted or unreliable output. The output may not be as par problem specification or in other words the output may not be valid. In the present paper the authors made a through study on these aspects that is reliability of a program, validity of the output and also different type of errors. As a case study the authors have chosen C-programming language .

Keywords: ***[compiler, syntax error, runtime error, logical error, reliable]***

1. Introduction

Srivastav et al already published papers on mathematical modelling of procedural language and object oriented language [1,2,3]. The authors tried to correlate simple mathematical models to describe procedural language like 'C' and object oriented language like Java. In the present study the authors made thorough study to run a program and how one can interpret syntax errors, reliability of a program and to verify the results whether they are as per question or not. The process of computation of a problem can be obtained using different methods in different programming languages. While writing a program using any programming, the following cases can occur :

Case-1 : The program is working and produces some results.

Case-2 : During compilation of the program it shows some syntax errors and unable to produce executable/object file.

Case-3 : The program is compiled without any error but showing some runtime errors while running the final program.

So therefore, a program written in a programming language will depends on the some events. The program written in a programming language may carry some syntax error/ runtime error/ no error.

So the probability of running a program depends on the cases as stated above. A program written in either procedural language or in object oriented language follows the above three cases. The programmer has to write the program in such a way which will produce consistent result.

2. Errors in a programming language

Any computer language is associated with some context free grammar, and the program written in that language has to follow the rule of that grammar. For example in English language a sentence such as "Ram and Aayush , are playing, with a ball". This sentence is syntactically incorrect because commas should not come the way they are in the sentence. Likewise, C also follows certain syntax rule. When a C program is compiled with some C-compiler then the compiler will check that whether the program is syntactically correct or not. If there are any syntax errors in the program , those will be displayed on the screen with the corresponding line number.

Example-1: Write a program to print a message on the screen

```
/* Program to print a message on the screen*/
#include <stdio.h
```

ISSN 2348 - 7968

```
main()
{
printf("Hello, how are you\n")
```

Let the name of the program be **test.c.** If we compile the above program as it is we will get the following errors:

Error test.c 1 :No file name ending
Error test.c 5: Statement missing ;
Error test.c 6 : Compound statement missing }

Edit the program again, correct the errors mentioned and the corrected version appears as follows:

Edit the program again, correct the errors mentioned and the corrected version appears as follows:

```
#include <stdio.h>
main()
{
printf("Hello, how are you\n");
}
```

Apart from syntax errors, another type of errors that are shown while compilation are semantic errors. These errors are displayed as warnings. These errors are shown if a particular statement has no meaning. The program does compile with these errors, but it is always advised to correct them also, since those lines may create problems during execution. The example of such an error is that say you have declared a variable but have not used it, and then you get a warning "code has no effect". These variables are unnecessarily occupying the memory.

Example-2: Linker Errors:

If a program contains syntax errors then the program does not compile, but it may happen that the program compiles successfully but we are unable to get the executable file, this happens when there are certain linker errors in the program. For example, the object code of certain standard library function is not present in the standard C library; the definition for this function is present in the header file that is why we do not get a compiler error. Such kinds of errors are called linker errors. The executable file would be created successfully only if these linker errors are corrected.

Example-3 : Logical and Runtime Errors:

After the program is compiled and linked successfully we execute the program. Now there are three possibilities:

1) The program runs successfully but produces wrong results,
2) The program runs successfully but produces wrong results, and
3) The program abort the job while encountering some logical problem/mistake which was not traceable during compilation stage.

The first case simply means that the program is correct. In the second case, we get wrong results; it means that there is some logical mistake in the program. This kind of error is known as **logical error.** To rectify logical error is a bit difficult task. The logical errors may be corrected using different debugging method. Debugging is the process of removing the errors from the program. This means manually checking the program step by step and verifying the results at each step. Debugging can be made easier by a tracer provided in Turbo C environment.

Example-4 : Write a C program to compute the average of three numbers

```
/* Program to compute average of three numbers */
#include <stdio.h>
main()
{
int a,b,c, sum, avg:
a= 10;
b= 5;
c= 10;
sum = a + b + c;
avg = sum / 3;
printf("The average is %d\n",avg);
}
```

Output

The average is 8.

The exact value of average is 8.33 and the output we got is 8. So we are not getting the actual result, but a rounded off result. This is due to the logical error. We have declared variable **avg** as an integer but the average calculated is a real number, therefore only the integer part is stored in **avg.** Such kinds of errors which are not detected by the compiler or the linker are known as **Logical errors.**

The third kind of error is only detected during execution. Such errors are known are **run time errors.** These errors do not produce the result

at all, the program execution stops in between and the run time error message is flashed on the screen.

Example-5: Write a program to divide a sum of two numbers by their difference

```
/* Program to divide a sum of two numbers by their difference*/
#include <stdio.h>
main()
{
 int a,b;
float c;
 a= 10;
b= 10;
c= (a +b) / (a-b);
printf("The value of the result is %f\n",c);
}
```

The above program will compile and link successfully, it will execute till the first printf statement and we will get the message in this statement, as soon as the next statement is executed we get a runtime error of "Divide by zero" and the program halts. These type of errors are called **runtime errors.**

The similar type error also take place in the object Oriented language.

3.Correlation between Syntax error and Runtime error:

There is no relation between syntax error and runtime error. So, the correlation value between them is zero. In the syntax error we can not get the output and in the runtime error we can not get the actual result what we want to find. Hence to understand the cases mathematically for the output we have to define the following mathematical term related to set theory.

(i) Set: A set is a well defined collection of objects. In other words , an aggregate or class of object having specified property in common which enables us to tell whether any given objects belongs to it or not. The individual objects of the set are called members or elements of the set.

(ii) Characteristics function: A set is defined by a function, usually called a characteristic function, that declares which elements of X are members of the set and which are not. Set A is defined by its characteristics function

$X_A(x) = 1$ if x belongs to A

$= 0$ if x does not belongs to A

The characteristics function maps elements of X to elements of the set {0,1}, which is formally expressed by X_A :X->{0,1} .For each x belonging to X , when $X_A(x)=1$, x is declared to be as member of A; When $X_A(x)=0$, x is declared as a non-member of A.

(iii) fuzzy Set: A fuzzy set is a pair (U,m) where U is a set and m: U → [0,1].

For each x ε U the value m(x) is called the **grade** of membership of x in (U,m). For a finite set $U=\{x_1,x_2,\ldots,x_n\}$, the fuzzy set (U,m) is often denoted by $\{m(x_1)/x_1,\ldots,m(x_n)/x_n\}$.

Let x ε U Then x is called **not included** in the fuzzy set (U,m) if m(x)=0, x is called **fully included** if m(x)=1 and x is called a **fuzzy member** if $0 < m(x) < 1$. The set { x ε U | m(x) > 0}is called the **support** of (U,m) and the set { x ε U | m(x)=1} is called its **kernel**. The function m is called the **membership function** of the fuzzy set (U,m).

Sometimes, more general variants of the notion of fuzzy set are used, with membership functions taking values in a (fixed or variable) algebra or structure L of a given kind; usually it is required that L be at least a poset or lattice. These are usually called L-**fuzzy sets**, to distinguish them from those valued over the unit interval. The usual membership functions with values in [0, 1] are then called [0, 1]-valued membership functions.

Therefore, by taking fuzzy set theory we can define the function f : X →{0,1}, where X represents the set of program written by a programmer for a problem and is defined by

f(X)={0}, if the result is not found.

And f(X)={1}, if the exact result is found. It means the chances of running a program depends on the method of written program i.e. there exist one-one to mapping intitutively in the following way:

f(program written is logically right)=output will be all right.

But as shown previously that if someone want to find the result in float variable and if the programmer have taken integer variable we will get the output in the integer variable.

Hence we see that a program written logically may not give the exact result what it is required. The probability theory is also based on Aristotelian two-valued logic, When A is a fuzzy set and x is a relevant object, the proposition "x is a member of A" is not necessarily either true or false, as

ISSN 2348 - 7968

required by two-valued logic, but it may be true only to some degree, the degree to which x is actually a member of A.

4. Reliability and Validity of a Programming Language

A programmer can write a program in the different way but the written program can not give an optimal result .So there is need of some reliability and validity of a program.

4.1. Reliability

Reliability implies accuracy and consistency in program .Therefore, a program must be measured accurately and consistently. A reliable program should give the same output when it is compiled in different compiler in different time.

4.2. Validity

A program is said to be valid if it actually measures what is supposed to be measured to give the actual output. Validity is the importance characteristics of a good programming.

Reliability is the necessary condition for validity, but validity is not the necessary condition for reliability .In other words a program will be valid if it will be reliable also.

4.3 Example of showing reliability and validity in procedural language: (in case of C-language)

Write a program to find the biggest of three numbers:-

```
#include<stdio.h>
#include<conio.h>
main()
{
int a,b,c,big;
clrscr();
printf("\n Enter three numbers :");
scanf("%d%d%d",&a,&b,&c);
if(a>b)
        if(a>c)
        big=a;
    else
        big=c;
else
        if(b>c)
            big=b;
     else
  big=c;
printf("\n Biggest number is %d',big);
getch();
}
```

Output:

```
Enter three numbers :18,-5,13
Biggest number is 18
```

Remarks : When this program is executed , the user has to enter three numbers and in each case user will get the appropriate , consistent and accurate result. Therefore this is reliable program. This is also a valid program as the program is consistent of only those logic which is required to be found from the given problem.

4.4 Example showing of a program is reliable but is not valid.

(i) Write a program to calculate (a) sum of n numbers and (b) standard deviation

```
#include<stdio.h>
#include<math.h>
int  sum(int a[],int n);
float average(int a[],int n);
int large(int a[],int n);
float  s_dev(int a[],int n);
int  readdata(inta[]);
void main()
{
int a[20], n;
int s,mx;
float avg,sd;
char ch;
do
{
clrscr();
printf("\n Enter the list--------.\n");
n=readdata(a);
s=sum(a,n);
avg=average(a,n);
mx=large(a,n);
sd=s_dev(a,n);
printf("\n sum=%d\n", s);
printf("Average=%6.2f\n",avg);
printf("largest number=%d\n", mx);
printf(standard deviation=%7.4f\n",sd);
printf("\ n Do you want to continue(y/n):");
scanf("%c",&ch);
} while(ch=='y' ||ch=='Y');
}
/*int readdata(int a[]):function to input a list*/
int readdata(int a[])
{
int i, n;
printf("\n Enter size of your list(1-20):")
while(1)
{
printf("\ n Enter size(1-20):");
scanf("%d",&n);
```

```
if(n<1||n>20)
printf("\n  ****Invalid Data ****\n");
else
break;
}
printf("\n Enter %d elements one by one---
→\n",n);
for(i=0; i<n; i++)
{
printf("a[%d]=",i);
scanf("%d",&a[i]);
}
return n;
}

/*int sum(inta[],int n): function to calculate sum of
n elements*/
int sum(int a[], intn)
{
int s, i;
for(i=0; i<n; i++)
s=s+a[i];
return s;
}
/*float average(int a[], int n):function to calculate
average of 'n' elements*/
float average(int a[],int n)
{
float avg;
avg=(float) sum(a,n)/n;
return avg;
}
/* int large(int a[],int n): function to calculate
largest number*/
Int large(int a[], int n)
{
 int i, mx;
mx=a[0];
for(i=1;i<n; i++)
if(a[i]>mx)
mx=a[i];
return mx;
}
/*float  s_dev(int a[], int n): function to calculate
standard deviation*/
float  s_dev(int a[], int n)
{
int i;
float avg,sd;
avg=average(a,n);
sd=0;
for(i=0;i<n;i++)
sd=sd+(a[i]-avg)*(a[i]-avg);
sd= sqrt(sd/n);
return sd;
}
```

Output:
Enter size of your list(1-20):
Enter Size(1-20): 5
Enter 5 elements one by one…….
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a [4]=5
sum=15
average=3.00
largest number=5
standard deviation=1.4142
Remarks : The given problem ask only about sum of n numbers and standard deviation but here no need of calculate largest number so this is a reliable solution but not a valid solution
Example (ii) :-Write a C program to find factorial of a given integer k using while loop:

```
#incldue<stdio.h>
#include<conio.h>
main()
{
 int k, kfact, I;
clrscr();
printf("\ nEnter the number :");
scanf("%d",&k);
kfact=1;
for(i=1; i<=k; i++)
kfact=kfact*I;
printf("\n %d factorial is %d", k,fact);
getch();
}
```

Output: Enter the number :4
4 factorial is 24
Remarks: This is a reliable problem but this is not a valid program because the problem is asked in the while loop.

4.5 Example showing of a program which is not reliable and not valid.

```
/* Program to compute average of three
numbers */
#include <stdio.h>
 main()
{
Int a,b,c, sum, avg:
```

```
a= 10;
b= 5;
c= 10;
sum = a + b + c;
avg = sum / 3;
printf("The average is %d\n",avg);
}
```

Output

The average is 8.

Remarks :The exact value of average is 8.33 and the output we got is 8. So we are not getting the actual result, but a rounded off result. Therefore this is not a reliable and valid result.

4.6Example showing reliability and validity in object Oriented programming language.

(i) Write a program to display whether a year is "leap year " or not.

```
Import java.io.*;
class q1
{
public static void main(String args[])throws IOException
{
int yyyy;
Buffered Reader br= new BufferedReader(new InputStreamReader(System.in));
System.out.print("\n Enter the year=");
yyyy=Integer.parseInt(br.readLine());
if((yyyy%4==0 && yyyy%100!=0)||(yyyyy%400==0))
System.out.println(yyyy +"is a leap Year");
else
System.out.println(yyyy +"is  not a leap Year");
}
}
```

Output: Enter the year= 2003

2003 is not a leap year.

Remarks :Here the program is written in actual requirements. This is a reliable and valid program.

4.7 Example showing program which is reliable but not valid.

(i) Write a program to input two numbers and calculate hcf of two numbers

```
import java.io.*;
class HCF_LCM
{
Int hcf(int a, int b)
{
int r;
while((r=a%b)!=0)
{
a=b;
b=r;
}
return b;
}
int lcm(int a, int b)
{
int h, lc;
h=hcf(a,b);
lc=(a*b)/h;
}
return lc;
}
class q2
{
public static void main(String args[])throws IOException
{
int  a,b, h,lc ;
Buffered Reader br= new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter 1st number=");
a=Integer.parseInt(br.readLine());
System.out.print("Enter 2nd number=");
b=Integer.parseInt(br.readLine());
HCF_LCM H=new HCF_LCM();
h=H.hcf(a,b);
lc=H.lcm(a,b);
System.out.println("\na=  "+a+"b="+b"  HCF ="+h+"lcm="+lc)
}
}
```

Output:

Enter 1st number= 20

Enter 2nd number=25

A=20 b=25

HCF=5 lcm=100

Remarks: The above problem is reliable but not valid since the problem is related to solve only hcf but in the above program hcf and lcm both are solved.

5. Reliability and Validity between Procedural language vs Object Oriented Language

Both the programming language follows the reliability and validity on a program but the object oriented language gives an optimal solution with respect to procedural language .So, a programmer should be more reliable on object –oriented language rather than the procedural language.

6.Method of checking Reliability and Validity of Programming Language

ISSN 2348 - 7968

There may be different method for checking the reliability and validity of a program. We can take two method as follows:

(i) By observation method.

(ii) By Test- reTest method

(iii) Checking of reliability and validity of a program on the basis of memory space

(i) **Observation Method:** A programmer can learn the reliability and validity of a program by observation method. In the observation method the programmer can observe the syntax ,method and required logic for a problem in a programming language . The programmer can observe the process of method of writing a particular problem in different programming language without any error. For Example, the syntax used for procedural language is different from object –oriented language. A programmer should observe the different process of writing of a program in different programming language so that the output will be optimal.

(ii) **Test-ReTest Method:** To check the reliability and validity of a program a user has to test the output of a program by entering different types of valid input value and so that the optimal result can be obtained. If the user on executing the program gets some error like compile error/runtime error , he has to debug the program and again he retest the program by entering the different input value. Hence the input is taken as sample space for testing the reliability and validity of a program.

(iii) **Checking of reliability and validity of a program on the basis of memory space:**

The computation of a problem is consists of two things.

(i) A program being written for that problem, and

(ii) Output of the program.

So there is need to occupying memory space for two things said above . Since reliability implies accuracy and consistency in program i.e the output of the program should be consistent and give the same output when it is compiled in different compiler in different time. Validity of a program implies to actual need from a problem with the required method as demands from a problem to solve that problem.

To understand reliability and validity let us take practical Example from daily life situation:

(i) The human being has an intelligent brain. So, therefore there are some memory and intelligence quotient in the brain. Suppose a student of class LKG has given a problem of class four(iv) to solve .Then this type of problem is not valid for the student of class LKG because this type of problem is not match with his age and level of intelligence Quotient.

(ii) Suppose a student have given a problem to solve it in particular method but he solve the problem in another method or solve the problem with adding some extra thing. Then this type of solution is not valid as here it may be considered that there is need of some extra work.

(iii)Suppose a formative test is taken from a student of class vii on the basis of his standard and qualification and consistently he is doing good performance then we can say that the student is reliable.

Now we are taking two cases in a programming of a problem as being taken to solve it:

Case 1: Some Extra work(program) is done with required program of the problem:

Let S_1= Memory space required for actual required program.

S_2= Memory space required for the program which is written with some extra work(program).

O_1 = Memory space required for actual required output of the program.

O_2= Memory space required for the output which is written with some extra work(program).

Percentage of difference of total memory space =T (say)=($(S_2-S_1)/S_1$)*100%+((O_2-O_1)/O_1)*100%

The program will be not valid if T>0 or T<0. The program will be valid only when T=0 and there may be some condition (or some extra property of compiler) with respect to memory space.

Case 2:Written program does not follows the required method but output is same:

Let S_1= Memory space required for actual method to write the required program.

S_2= Memory space required for the program which is written with another method.

Here output is same. Therefore difference between output is zero.

ISSN 2348 - 7968

Total memory space required= Memory space required for a program+ Memory space required for output

Percentage of difference of total memory space =T (say)=($(S_2-S_1)/S_1$)*100%

The program will be not valid if T>0 or T<0. The program will be valid only when T=0 and there may be some condition (or some extra property) with respect to memory space.

7. Conclusion and future scope

Reliability means a matter of stability of a program with its input and validity is the correlation of the program with given problem and with certain independent compiler criteria. A program possessing poor reliability will yield low validity.Therefore, a reliability is a prerequisite of validity. A program may be reliable but not valid. This is because it may yield consistent output, but those output need not representing what exactly we want to measure. If a program produces valid output then it is implies that the program is reliable. In future the authors are planning to introduce to calculate time complexity of a problem and to correlate with reliability factor and also the validity of a program. Suppose two programs are reliable and also the output is valid but the time complexity of the two programs are different. In that case how do we justify the validity of the output of two programs. In the next work the authors will try to incorporate the importance of time complexity along with validity.

References

[1] Mathematical modeling of various statements of C-type Language**,** Manoj Kumar Srivastav **,** Asoke Nath, International Journal of Advanced Computer Research(IJACR), Vol-3,Number-1, Issue-13, Page:79-87 Dec(2013).

[2] Mathematical Description of variables, pointers, structures, Unions used in C-type language, Manoj Kumar Srivastava, Asoke Nath, Journal of Global Research in Computer Science, Vol-5, No.2, Page: 24-29,Feb(2014).

[3] Manoj Kumar Srivastav, Asoke Nath, A Mathematical Modeling of Object Oriented Programming Language : a case study on Java programming language, Current Trends in Technology and Science(CTTS) Vol-3, Issue-3, Page 134-141,(2014).

[4] George J.Klir, B.Yuan , "Fuzzy Sets and Fuzzy Logic(Theory and Application)", Prentice Hall of India Private Limited, New Delhi

[5] S.C.Malik, Savita Arora, "Mathematical Analysis", New Age International(P) Limited Publishers(2006).

[6]Arun Kumar Kulshrestha, "Teaching of Mathematics", R. Lall Book Depot(2003).

[7] L. A. Zadeh (1965) "Fuzzy sets". *Information and Control* 8 (3) 338–353.

[8] Klaua, D. (1965) Über einen Ansatz zur mehrwertigen Mengenlehre. Monatsb. Deutsch. Akad. Wiss. Berlin 7, 859–876.

[9] Gottwald, S. , "An early approach toward graded identity and graded membership in set theory". *Fuzzy Sets and Systems* **161** (18): Page 2369–2379(2010).

[10] D. Dubois and H. Prade , Fuzzy Sets and Systems. Academic Press, New York(1988).

[11] E. Balagurusamy , Programming in ANSI C,Tata McGraw Hill Education Private Limited, 2012

[12] E.Balagurusamy, Programming with JAVA, Tata McGraw Hill Education Private Limited, 2011

[13] T Jeyapoovan,A First Course in Programming with C, Vikas Publishing House Pvt Ltd. 2013

Mr. Manoj Kumar Srivastav has been Post graduate in Pure Mathematics from University of Calcutta in year 2004 with Special paper in advanced functional analysis and category theory, universal algebra and lattice theory. At present he is doing MCA from IGNOU at St.Xavier's college as study center and he is working as a postgraduate teacher in Mathematics in an esteemed institution.

Dr. Asoke Nath is Associate Professor in the Department of Computer Science. At present he is busy with research work in Cryptography and Network Security, Steganography, Green Computing, e-learning, Mathematical formulation of computer Language. He has more than 100 published research papers in National and International Journals.