International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-4 Issue-13 December-2013

Mathematical model of various statements of C-type Language

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

Abstract

Some of the important components of high level languages are statements, keywords, variable declarations, arrays, user defined functions etc. In case of object oriented programming language we use class, object, inheritance, operator overloading, function overloading, polymorphism etc. There are some common category of statements such as control statement, loop statements etc. Pointers are also one important concept in C-language. User defined functions, function subprograms or subroutines are also important concepts in different programming languages. The language like ALGOL was developed using Chomsky context free grammar. The similar concept used in C-type languages. The high level languages are now based on mathematical derivations and logic. Most of the components of any high level language can be obtained from simple mathematical logic and derivations. In the present study the authors have tried to give some unified mathematical model of few statements, arrays, user defined functions of C-language. However, the present method may further be extended to any other high level language.

Keywords

C-language, function, pointers, Chomsky, high level language.

1. Introduction

In high level languages there are several common features such as control statements, loop statements, input and output functions or statements, relational operators, logical operators, different data types, arrays, pointer type variable, string operations, functions, subprograms or subroutines etc. In object oriented language there is something extra such as class, object, inheritance, polymorphism, encapsulation etc. The syntax of a particular statement may be different in two different languages but the purpose of the statement is the same. In the present paper the authors tried to explain the meaning of a statement of high level language from mathematical concepts like set, integers, real number, functional approach etc. There is a general view from the user that C-language [6] is the mother language of all the subsequent languages like C++,C#,JAVA etc. Generally C, C++,C# and JAVA are called as C-type language as the programming structures are almost same in all the four languages. In the present study the authors tried to explore C-language from very simple mathematical models or explanation. The present study will help the user to understand a language better. It is possible to give complete mathematical explanation of the entire language but in the present study the authors have taken few selective statements, functions etc. The mathematical modelling of any programming language will help the user to understand a high level language much better and also will help to construct some new language. Here the authors propose that the same explanation may be given to any other language with minor revision.

2. Variables and Data types in C-language

A. Definition of Set: A set is any well defined collection of objects. For an example a collection, class and aggregate are used synonymously for the term Set. Here "well defined" means that it is possible to determine readily whether an object is member of a given set or not. The object that belongs to a set is called its element (or point or member). Now let us describe the two important methods in connection to a set[1-5]:

(1) Tabulation Method:-The tabulation method enumerates or list individual elements separated by commas and enclosed in braces. Example: English alphabet is written as{a, e, i, o, u}

(2) Defining property Method: This method is often more compact and convenient. A defining property of a set is property which

Manoj Kumar Srivastav, IGNOU, St. Xavier's College(Autonomous), Kolkata, India.
Asoke Nath, Department of Computer Science, St. Xavier's College(Autonomous), Kolkata, India.

is satisfied by each element of that set and nothing by else. A set can be expressed as:
{x|defing property} Or
{x: defing property}. x is a dummy symbols.
Examples of set:
N=set of natural numbers.
Z=set of integers.
Q=set of rational numbers.
R=set of real numbers

Table-1: Comparative study of different type set members

Set of natural number (N)	**Set of Integers (Z)**	**Set of rational number (Q)**	**Set of real Number (R)**	**Set of complex number (C)**
Algebraic properties	Algebraic properties	Algebraic properties	Algebraic properties	Algebraic properties
Ordered properties	Ordered properties	Ordered properties	Ordered properties	—
—	—	Density property	Density property	—
—	—	—	Completeness property	—
—	—	—	Archimedean Property	—

B. Data Types in C-language:

In C-language to define integer type variable one has to define the variable as int. To define character type variable one has to define the variable as char which also some integer (-128 to 128). Similarly to define any fractional numbers one can use float, double. So therefore int and char can be considered as Integer. Float, double can be considered as Real numbers [6].

Table-2: Range of values of data to completeness property

Data type	**Range of values**
Char	-128 to 128
unsigned char	0-255
Int	-32768 to 32,767
Float	3.4e-38 to 3.4e+38
Double	1.7e-308 to 1.7e+308

Complex number is not taken as variable declaration in C-language. Question-1: Why complex number is not taken as variable declaration? Answer : complex number does not possesses the order axiom properties as real number possesses . In table-3 the meaning of order property of integer and real numbers are shown but for any complex number that is not possible.

Table-3: Order Properties of Real Numbers

(i) If a,b $\in R$,then exactly one of the following statement hold- a<b,or a=b, or b<a
(ii) a<b and b<c => a<c for a,b,c$\in$ R
(iii) a<b => a+c < b+c for a,b,c$\in$ R
(iv) a<b and b<c => ac<bc for a,b,c$\in$R

Question 2: How one can have different range of value for different data types? Answer: This is because real number possesses completeness property but the complex number does not satisfy the completeness property [7-10].

Table-4: Completeness Property

(i) Every non–empty set of real numbers which is bounded above has the supremum or the least upper bound in R.
(ii) Every non–empty set of real numbers which is bounded below has the infimum or the greatest lower bound in R.

Question 3 How one can define character type variable/data? Answer: it is necessary to make one to one correspondence with some numbers .Therefore ASCII codes (0-255) are taken and make one to one correspondence with each character.

3. Decision Control Structure vs. Venn diagram

Decision control structure and its correlation with set/venn diagram:

(i) If statement :The format of if statement is

```
If(expression)
  Statement;
```

This is the simplest form of 'if ' statement .The expression is to be placed in the parenthesis. It can be a logical expression. The statement can be simple are compound statement.
This can represented in set for as follows-

- Set S={expression :statement}

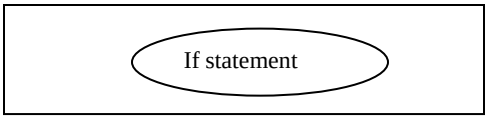


Fig-1: Diagrammatic representation of if-statement

(ii) If-else statement:-

The general format of if-else statement is

```
If(expression)
 Statment1;
   else
  Statement2;
```

If expression evaluates to true ,statement1 is executed ,while if expression evaluates to be false statement2 will be evaluated. Statement1 and statement2 can again be single or compound statement.

This can be written in venn-diagram as $x \varepsilon A \backslash B$

(iii) If-else statement:-

The general format of if-else statement is

```
If(expression)
 Statment1;
   else
  Statement2;
```

If expression evaluates to true, statement1 is executed, while if expression evaluates to be false statement2 will be evaluated. Statement1 and statement2 can again be single or compound statement.

This can be written in venn-diagram as $x \varepsilon A \backslash B$

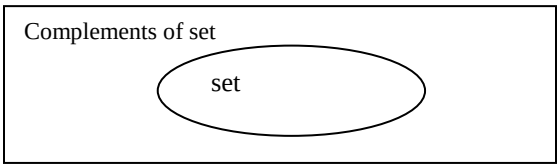


Fig-2: Diagrammatic representation of If-else

(iv) Nested if–else statement:-

It is possible to nest if-else statement within one another if-else. The inner if-else is said to be nested in the outer if-else. Nesting can go upto any level. There are several forms that nested if-else statement can take.

```
if(expression1)
{
if(expression2)
Statement1;
else
Statement2;
}
else
{
if(expression3)
statement3;
else
Statement4;
}
```

Important result based on this property:

Cantor's Intersection theorem:

Let F1, F2,F3,……be a countable collection of non-empty closed and bounded subset of R such that $F1 \supset F2 \supset F3$……

Then the intersection $\cap$ Fi is non empty.

Break : Break is a piecewise discontinuous .If the input condition are different then output will be different.

For example:

Let f(x) $=x+1 \quad 0<x\leq 1$

$=x+3 \quad 2<x\leq 3$

$=x+8 \quad 4<x\leq 8$

Table-5: Example for break statement

Input condition	Output result
$0<x\leq 1$ $2<x\leq 3$ $4<x\leq 8$	f(x)=x+1 =x+3 =x+8

The *break* is used normally to exit from any loop statement any language. In C-language the break statement is also used in switch-case statement which is actually compressed form of if-then-else structure. The use of *break* in switch-case is as follows –

The syntax for switch is

```
switch(expression)
{
case    value-1:
         Block-1
         break;
case   value-2:
        Block-2
        break;

…….
…….
```

```
default :
        Default-block
break;
}
Statement-x;
```

Table-6: Action of switch-case statement

Input condition	Output result
case value-1 case value-2 …… …….. …… If all cases are false then execute Default-block under default	Block-1 Block-2 …… …… …… Default-block

Here the condition is piecewise discontinuous.
Switch-case may be considered.
Continue: Here the input condition is continuous.

4. Loop Statements in C-language

Interval : A subset of the real line is called an interval if it contains at least two numbers and contains all the real numbers lying between any two of its elements.
Let F be an ordered field, a and b are two element of F with a<=b. Each of the following subset of F is bounded ,with 'a' serving as a lower bound and 'b' as upper bound.

$[a,b]= \{x \in F : a \leq x \leq b\}$
$(a,b) = \{x \in F : a < x < b\}$
$[a,b)= \{x \in F : a \leq x < b\}$
$(a,b] = \{x \in F : a < x \leq b\}$

(i) Interval [a,b] of the real number partitioned into the following way :

x0=a	or	x0=a
x1=a+h	or	x1=a-h
x2=a+2h	or	x2=a-2h
…….	or	.....
…….	or	……..
…….	or	……
xn=a+nh	or	xn=a-nh

A for loop is generated on this interval. The syntax of for loop in C-language is as follows;
for(loop_var=intial value or statement; testing final value or checking some condition; increment/decrement of loop_var or a statement) <actions;>

Which is equivalent to interval like[a,b] with its n point partition and in increment/decrement form.
This is an example of finite interval[2-3].

(ii)while loop : The general form of the while loop is as the following :

```
while(condition)
{
Statement_block;
}
Statement_outside_while;
```

Here statement_block will be executed repeatedly till the condition becomes false, execution jumps to the statement outside the while loop.
Here the condition will be in the inequalities form and statement_block will depend on the condition whether it is true or false. Here the condition will be in the form as while(variable>=<value) :

(a)variable is equal to a value. (variable=value)That means here we are getting a singletone set.
(b)variable is greater than a value. (variable>value)That means here we are getting an interval in this form (value, ∞).
(c)variable is less than a value. (variable <value)
That means here we are getting an interval of the form (∞,value).
Here the statement_block is dependent on the condition of the given variable.

(iii)Do-while loop : It is same as while loop but statement_block is independent of the condition of the given variable.
Note : when the number of iteration is one ,the condition will be similar to if ,if-else etc.

5. Introduction to Array

An array can be called as collection of data items. All the data items should necessarily be of the same type. All the data items in an array have the same name. Members of an array one-to-one to correspondence with set of positive integers including zero. Since array consists of finite number of terms. Hence they are connectable. The diagram for array in the mathematical way:

Index	Variable
1	V1
2	V2

……..	…….
n-1	V(n-1)
N	Vn

Fig-3: Variable as Function

Variable [1] = Value 1
Variable [2] = Value 2
……

Variable [n] = Value n
Actually there is linear operator between two sets

6. Introduction to FUNCTION()

Functions are the major tools for describing the real world in mathematical terms.

In computer science in the programming language C, function plays an important role. The set of all possible input values is called the domain of the function.The set of all output values is called the range of the function. In calculus , Leohard Euler invented (1707-1783) invented a symbolic way to say "y is function of x" i.e. y=f(x) (" y equals f of x").In this notation, the symbol f represent the function.The letter x,called the independent variable,represent an input value from the domain of f, and y,the dependent variable,represent the corresponding output values f(x) in the range of f.

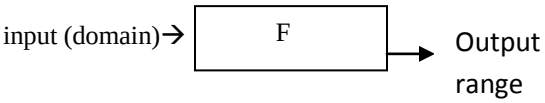


Fig-4: function call

Function or sequence of function have the properties of convergence/ divergence, uniform convergence, point wise convergence, limit/continuity. In the programming language function have the properties to return value and their types. The return type is optional; the function returns int type data. The return type must be void if no value is returned. Introduction to if-else statement: If two or more events associated with a random experiments are said to be mutually exclusive or incompatible events if the occurrence of any one of them prevents the occurrence of all other i.e., if no two or more of them can occur simultaneously in the same trial. Suppose we toss a coin and the outcome will be either a head or a tail. Similarly in if-else statement the output will be either in the if part or in the else part. The true part can be taken as head and false part can be taken as tail. Function: - A function from a set R is a rule that assigns a unique elements f(x) in R to each element x in D.

```
#include<stdio.h>
void main()
{
int a,b,c;
printf("Enter two integers=");
scanf("%d%d",&a,&b);
c=a+b;
printf("\n the result after addition is %d",c);
getch();
}
```

Composite function :-Composition is another method for combining function.

If f,g are two functions, the composite function f0g is f○g defined by
(f○g)(x)=f(g(x)
f o g

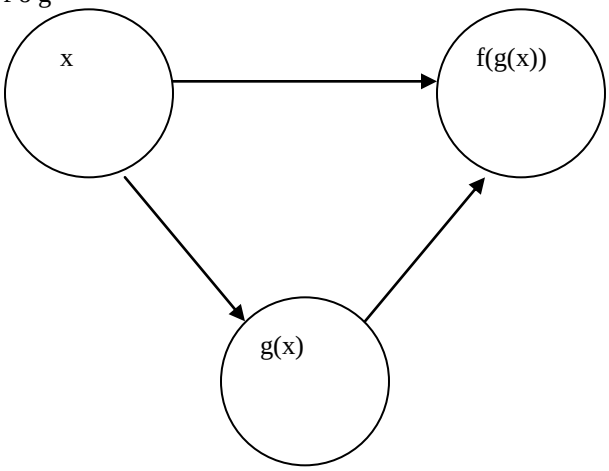


Fig-5: Multiple function calls

The domain of f○g consist of two numbers x in the domain of g for g(x) lies in the domain of f.

A. User Defined Function:

Programmers can write their own functions for performing any specific task of the program. These type of functions are called user-defined function. To create and to use these functions, the following three steps are required:

1. Function prototype definition
2. Function declaration
3. Function call

Example of function call:

```
main ()
{
-    - - - - - -
-    - - - - - - -
function_1();
-    - - - - - -
-    - - - - - -
function_2();
-    - - - - - - -
-    - - - - - - -
function_1 ();
-    - - - - - - -
}
function_1 ()
{
-    - - - - - - -
-    - - - - - - - -
}
function_2()
{
-    - - - - - - -
function_3();
-    - - - - - - - -
}
function_3 ()
{
-    - - - - - - - - -
-    - - - - - - - - -
}
```

Flow a control in a multipurpose function:
Actually, user defined function are composite type function in the following way
(i) F(g) or f (f1 (f2 (f3 (f4 - - - - - (fn)))))
(ii) F(f1, f2, f3,- - - - - - - - - (fn))))))

<u>Case I:-</u> f (f1(f2(f3(f4(- - - - - - (fn)))))))
In this case, first of all function fn then fn-1 is evaluated and after that fn-2 evaluated and so in. At last f is evaluated.
Here f is dependent on f1,f2,f3,……,fn
f1 is dependent on f2,f3,………...,fn
f2 is dependent on f3,…………….,fn
f3 is dependent on f4,f5,……………,fn
Only here fn is independent.
Example :

```
-    - -- - - - - - - - - - - - -
-    ---- ----- -- -- -- - - - - -
-    - - - - - - - - - -- - - -- -
     float ratio(int x, int y, int z);
     int difference(int x,int y );
main()
{
 int a, b,c;
scanf("%d%d%d",&a, &b,&c);
printf("%f\n",ratio(a,b,c));
}
float ratio (int x, int y, int z)
{
if(difference(y,z))
return(x/(y-z));
  else
return (0.0);
}
int difference( int p, int q)
{
if (p!=q)
return (1);
else
return(0);
}
```

<u>Series Connection for user defined function</u>

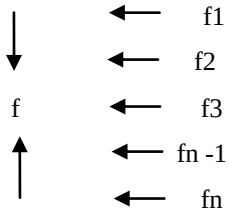
f ⟵ f1 ⟵ f2 ⟵ f3 ⟵ f4……⟵ fn-1 ⟵ fn

<u>Case II:-</u> f (f1, f2, f3, - - - - fn)
Here each f1, f2, - - - - - ,fn evaluated separately and then the function f is evaluated.
f is dependent on f1,f2,f3,………,fn
f1,f2,f3,f4…….fn are independent.
Example:

```
………………………
……………………….
void printline(char c);
void value (float,float,int);
main()
{
float principal, inrate;
int period;
printf("Enter principal amount,interest=");
scanf("%f,%f",&principal,&inrate);
printf("Period=");
scanf("%d",&period);
printline('Z');
value(principal,inrate,period);
printline('C');
}
void printline( char ch)
{
int i;
for(i=1; i<=52; i++)
printf("%c",ch);
printf("\n");
}
void value(float p,float r,int n)
```

```
{
int year;
float sum;
sum=p;
year=1;
while(year<=n)
{
sum=sum*(1+r);
year=year+1;
}
printf("%f\t%f\t%f\t%d\t%f\n",p, r, n, sum);
}
```

Parallel Connection for user defined function:-



B. Recursive function:

Recursion Theorem :

Recursion theorem guaranteeing that recursively defined function exist. Given a set 'x',an element a of 'x' and a function f: X-> X,the theorem states that there is a unique function F: N ->X(where N denotes the set of natural number including zero) such that

f(0)=a
F(n+1)=f(F(n)) for every natural number n.
The Recursion theorem : let Y be a non empty set, and suppose that x0$\mathcal{E}$ Y. suppose also that H:Y$\rightarrow$Y is a function. Then there exist a unique function R : N$\rightarrow$Y such that R(0)=x0 and such that for all n$\in$N,

R(n+1) =H○R(n)

Or

R(n+1)=H(R(n))

Programme:
Write a C program to find sum of first n natural number using recursion

```
#include<stdio.h>
int sum(int n)
int main()
{
 int  num,add;
printf("Enter a positive integer :\n");
scanf("%d",&num);
add=sum(num);
printf("sum=%d",add);
}
int sum(int n)
{
if(n==0)
return n;
else
return n+sum(n-1);
}
```

Output:
Enter a positive integer :4
10
Explanation:-
Sum(4)=(4)+sum(3)
=((4)+(3))+sum(2)
=(((4)+(3))+(2))+sum(1)
=((((4)+(3))+(2))+(1))+sum(0)
=(((((4)+(3))+(2))+(1))+(0))
=((((4)+(3))+(2))+(1))
=(4)+((3)+(3))
=((4)+(6))
=(10)

Step1 :taking function R:N$\rightarrow$ Y equivalent to sum function.Here
Sum function is in this form

sum: N$\rightarrow$N

Step2:in recursion we are taking R(0)=x0.here we are taking sum(0)=0. This is the case when n==0 in the above programming where it is written that
if(n==0)
return n;
Step3:
In recursion theorem we are taking the function as R(n+1)=H(R(n)).
Here we are returning n + sum(n-1) when n$\in$ N i.e.
Sum(n)=n + sum(n-1) which is equivalent to R(n)=H(R(n-1))=n+R(n-1) if function H:N$\rightarrow$ N is defined as H(n)=n+R(n-1)

Table-7: Recursive function call: calculating sum of 'n' numbers

Step	**Function returning result**	**Condition**
1.taking R=sum	-	-
2.taking R(0)=sum(0)	return n	n=0

3.taking R(n)=H(R(n-1)) =n+R(n-1) Which is equivalent to Sum(n)=n+sum(n-1)	Return n+sum(n-1)	$n \in N$

Example: function to evaluate factorial of n as follows:

```
---------- ----
----------------
----------------
  factorial (int n)
{
int fact;
if (n==1)
return (1);
else
fact=n*factorial(n-1);
return(fact);
}
```

Table-8: Recursive Function call: calculating factorial of 'n'

Step	Recursion theorem	Function return	Condition
Taking function $R \equiv$ fact	Function $R:N \rightarrow Y$ Here take Y=N	-	-
R(0)=1	R(0)=x0	Return 1	n=1
R(n)=H(R(n-1)) =n*R(n-1)if function H is defined as H(n)=n*H(n-1) and this is equivalent to fact(n)=n*fact(n-1)	Function $H:Y \rightarrow Y$ and R(n+1)=H(R(n)) For natural number $n \in N$.here we take Y=N	Return n*fact(n-1)	If n>1 and $n \in N$

Fibonacci series: Write a program to print n numbers

Fibonacci series by applying recursive call.

```
#include<stdio.h>
#include<time.h>
long double fib(int n);
 void main()
{
 int i,n ;
long int ts,td,t=NULL,te;
long double f;
char ch;
do
{
clrscr();
printf("\n Enter n(1-50) :");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
ts=time(t);
f=fib(i);
te =time(t);
td=te-ts;
printf("fib(%d)=%19.0Lf  Time=%ld sec,I,f,td);
}
printf("\n Do you want to continue(y/n?):");
scanf("%c",&ch);
} while(ch=='y' || ch=='Y');
}

/*long double fib(int n):function to calculate a number of fibonacci series using recursive call */
long double fib(int n)
{
if(n==1 || n==2)
return1;
else
return fib(n-1)+fib(n-2)
}
```

The Fibonacci series is a recurrence relation .The steps involves in it in the following way.

Table-9: Recursive Function call: calculating Fibonacci series

Steps	Recursion theorem	Function return	Condition
Taking function R=fib	Function $R:N \rightarrow Y$ here we are taking N=Y	-	-
R(0)=1 R(1)=1,intial condition	R(0)=x0-intial condition.here we are taking R(0)=1 and R(1)=1 as intial condition.	return 1	n==1|| n==2
R(n)=H(R(n-1)) =R(n-1)+R(n-2) if function H is defined as H(R(n-	Function $H:Y \rightarrow Y$ and R(n+1)=H(R(n)) For natural number $n \in N$.here we	return fib(n-1)+fib(n-2)	$n \geq 3$ and $n \in N$

1))=R(n-1)+R(n-2) which is equivalent to fib(n)=fib(n-1) +fib(n-2)	take Y=N.		

What will happen if initial condition is not given :
Let us take an example

---
main()
{
printf("This is an example of recursion\n")
main();
}
When executed, the program will produce an output something like this:
This is an example of recursion
This is an example of recursion
This is an example of recursion
This is an example of recursion
……..
Execution is terminated abruptly; otherwise the execution will continue indefinitely [1-5].
The process of calling a function is dependent on composition of mapping.

7. Conclusion and Future Scope

In the present paper the authors have studied how the various statements of C-language can be verified or compared with various mathematical models or functions. It has been shown every statement of C-language has a mathematical description. In the present paper the authors have concentrated only on C-language [6]. But the authors have already extended the idea to explore object oriented language or any scripting language. In the present paper the authors have focused on statements of C-language, data types, storage allocation, functions and arrays. In our future work we have already started to give complete mathematical description of all remaining statements of C-language and also to derive some unique mathematical formulation for any other programming language also.

References

[1] Real Analysis , S.K.Mapa, publication-Asoke Prakasan, year1998.
[2] Higher Algebra, S.K.Mapa,publication-Sarat Book Distribution,year 2000.
[3] Discerete Mathematics with Proof , Eric Gossett, Wiley , India .
[4] Discrete Mathematics, N.Chandrasekaran, M.Umaparvathi, PHI, 2010.
[5] Discrete Mathematics and its Applications, K.H.Rosen, Tata Mcgraw-Hill Publishing Company limited, 2003.
[6] Programming in ANSI C, E.Balagurusamy, Tata Mcgraw Hill Book Company, 2004.
[7] Calculus and analytic geometry-Thomas, Finney pearson eduation asia-2002.
[8] Mathematics Dr. R.D.Sharma-Dhanpat Rai publication (p)ltd-2010.
[9] Notes of constructive mathematics and computer programming-P.Martin Lof.
[10] Complex variable and application-churchill Brown. McGraw-Hill International publication1996.

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.

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 and he is working as a postgraduate teacher in Mathematics in one esteemed institution.