American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-6, Issue-11, pp-121-127
www.ajer.org
Research Paper Open Access

Algorithm and C Language Programming Concepts in Algebraic Structures and their Ideals

Ahmad Sarosh.
(Research Scholar) Kolhan University, Chaibasa, Jharkhand, India.

***Abstract*:** *In this paper, We Introduce a Concept of Algorithm and Programmingof CLanguagein Algebraic Structures and their Ideals, Sub-Programs have been Introduced to Some of the Algebraic Structures and their Idealswith Examples and Executed Programs.*

Keywords: *Algorithm, Programming SYNTAX of C Language, Keywords of C-Languages, Algebraic Structures , Ideals, Prime Ideals.*

Date of Submission: 02-11-2017 Date of acceptance: 17-11-2017

I. INTRODUCTION

C-Language [1]

The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis Ritchie [1].The C language is belonging to middle level programming language [1].
Operating system programs such as Windows, Unix, Linux are written in C language [1].
C has been written in assembly language [1].

C is a general-purpose programming language, and is used for writing programs in many different domains, such as operating systems, numerical computing, graphical applications, etc. It is a small language, with just 32 keywords .It provides "high-level" structured programming constructs such as statement grouping, decision making, and looping, as well as "low-level" capabilities such as the ability to manipulate bytes and addresses[2]."

C-Program

Syntax[1]

```
#include <stdio.h>

int main()
{
    printf("Algorithm And C Language Programming Concepts in Algebraic Structures and their Ideals");

getch 0;
}
```

Group Theory

Definition:A Set G is Said to be a Group with an Operation 'o' which Satisfies the Following Postulates.
(i) aob $\in$ G$\forall$ $a, b \in G$ *Closure Axiom.*
(ii) aoboc = aoboc $\forall$ a, b *and* $c \in G$ *Associative Axiom.*
(iii) aoe = eoa = a$\forall$ $a, \in G$ *Identity Law.*
(iv) aoa^{-1} = a^{-1}oa = e $\forall$ $a, \in$ *Inverse Law.*

Algorithm / Sub Program in C Language

Step 1: Start
Step 2: Read a, b, c ,e from a SET .

```
scanf("%d%d%d%d",a,b,c,e);
```

Step 3: Read Operator 'o' as $+ \; or \; *$ and Calculate aob.

```
printf("The Closure of aob =%d", aob);
```

Step 4: if $aob \in G$ Closure Law Holds

```
printf("Enter the range of integers (only integers): \t");
fflush(stdin);
scanf("%d",&n);
printf("\nRange of integers n=%d: \t",n);
printf("\nEnter any two numbers:\t");
scanf("%d%d",&x,&y);
printf("\nNumber entered are a=%d b=%d:\t",a,b);
sum=a+b;
mul=a*b;
printf("\nsum=%d, Multiply= %d :\t",sum,mul);
if (sum<=n &&mul<=n)
printf("\Closure Law Holds .");
else
printf("\nClosure Law doesnot holds.");
```

Step 5: Check for Associativity Axiom.

```
printf("Enter the range of integers (only integers): \t");
fflush(stdin);
scanf("%d",&n);
printf("\nRange of integers n=%d: \t",n);
printf("\nEnter any three numbers:\t");
scanf("%d%d",&a,&b, &c);
printf("\nNumber entered are  a=%d b=%d:\t",a,b);
sum1=a+(b+c);
   sum2=(a+b)+c;
printf("\nsum1=%d, sum2= %d :\t",sum1,sum2);
if (sum1==sum2)
printf("\Associative Law Holds .");
else
printf("\Associative Law Doesn't Holds.");*/
```

Step 6: Check for Identity.
Read $e = 1 \; or \; e = 0$as per the Operation

```
printf("Enter the range of integers (only integers): \t");
fflush(stdin);
scanf("%d",&e);
printf("\nRange of integers n=%d: \t",n);
printf("\nEnter any number:\t");
scanf("%d ",&a);
printf("\nNumber entered is  a=%d t",a,b);
sum=a+e;
mul=a*e;
if (sum==a&&mul==a)
printf("\Identity Holds .");
else
printf("\Identity Axiom is not fulfilled.");
```

Step 7: Check for Inverse, when Identity ($e = 0$ or o==+)

```
printf("Enter the range of integers (only integers): \t");
fflush(stdin);
e==0;
```

```
printf("\nEnter any number:\t");
scanf("%d ",&a);
printf("\nEnter other number:\t");
scanf("%d ",&b);
if (a+b==e)
printf("\Inverse Exists.");
else
printf("\Inverse Axiom fails.");*/
```

Check for Inverse, when Identity ($e = 1$ or o==*)

```
printf("Enter the range of integers (only integers): \t");
fflush(stdin);
e=1;
b=(1%n);
printf("\nEnter any number:\t");
scanf("%d ",&a);
printf("\nEnter other number:\t");
scanf("%d ",&n);
if (a*b==e)
printf("\Inverse Exists.");
else
printf("\Inverse Axiom fails.");*/
```

Step 8: Stop.

Ring

Definition: A Set $(R, +, .)$ is said to be a Ring if the Set Satisfies Certain Axioms.

(i) $R$is an Abelian Group(Group with Commutative Axiom) with Respect to the Operation $'+'$.

(ii)Closure Law Holds for

$$a.b \in R \forall\ a, b\ \in R$$
$$(a.b).c = a.(b.c)\ \ \forall\ a, b, c \in R$$

(iii)Distributive Law Holds

$$a.(b + c) = (a.b) + (a.c)$$
$$(b + c).a = (b.a) + (c.a)$$

Ideals in a Ring[4]

Definition [4]: Let a Set(R, o_1, o_2)be a Ring. A Subset$I$ of R is Said to be a Left Ideal of Rif

(i) $ro_1a \in I$ And $ro_2a \in I$ for all $r \in R$ and $a \in I$

Algorithm / Sub Program in C Language

Step 1: Start.
Step 2: Read a Range for a Set RCalled Ring n(Axioms Fulfilled).
Step 3: Choose a Subset I of R, and Read a range $for\ I$ $(m <= n)$.
Step 4: take any two Numbers$from\ I\ and\ R$.
Step 5: Calculate their Compositions $(ro_1a\ \ and\ \ ro_2a)$.
Step 6: Check the Compositions within the Set I.
Step 7: Display the Result.
Step 8: Stop

```
Syntax
#include<stdio.h>
#include<math.h>
void main()
{
int m, n;
printf("Enter the Range of a Set Ring  R: \t");
fflush(stdin);
scanf("%d",&n);
```

```
printf("\nRange for a Subset I of R(m<=n) =%d: \t",m);
scanf("%d",&m);
printf("Tthe Composition for the Operator + is %d",m+n);
if(m+n<=m)
{
printf("Tthe Composition Law for the Operator' +' Holds %d\n" ,m+n);
}
else
{
printf("The Composition Doesn't Holds for '+' ");
return 0;
}
if(m*n<=m)
{
printf("Tthe Composition Law for the Operator' *' Holds %d \n" ,m*n);
}
else
{
printf("The Composition Doesn't Holds for '*' ");
return 0;
}
getch();
}
```

Let a Set(R, o_1, o_2) be a Ring. A Subset I of R is Said to be a Right Ideal of R if
(ii) $ao_1r \in I$ and $ao_2r \in I$ for all $r \in R$ and $a \in I$

Syntax:

```
}
if(m<=n)
printf("the Composition for the Operator + is %d",m+n);
else
printf("The Composition Doesn't Holds");
return 0;
}
```

Example [3]: Modulo Divisor of 2,{ Z_{10}} is an Ideal of the SET of Integers.

Syntax:

```
void main()
{
int count;
int n;
intm,k,r,d;

printf("Enter that Range Integes");
scanf("%d",&n);

for(count = 0; count <= n-1; count++)// Display the numbers 0 -9//
printf("%d ", count);
printf("\n");
printf("Please enter any the values for Modulo Z10, = \n");
scanf("%d %d",&m,&k);
d=m/k;
r=m%k;
```

```
if(r<=d)
{
printf("Integer division: %i divided by %i is %i\n",m,k,d);
printf("The remainder when %i is divided by %i is %i\n",m,k,r);
return 0;
}
```

Example [3]: Let R be a Ring of 2×2 Matrices over Integers

Let $A = \{\begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix} \| \ x, y \ integers\}$ Then A is a Right Ideal of R.

Let $p = \begin{pmatrix} u & v \\ 0 & 0 \end{pmatrix} \in A$ and $q = \begin{pmatrix} w & x \\ 0 & 0 \end{pmatrix} \in A$

$p - q = \begin{pmatrix} u & v \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} w & x \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} u-w & v-x \\ 0 & 0 \end{pmatrix} \in A.$

Let $r = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} \in R$

Then $pr = \begin{pmatrix} u & v \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} = \begin{pmatrix} u\alpha + v\beta & u\gamma + v\delta \\ 0 & 0 \end{pmatrix} \in A$Thus A is a Right ideal of R.

Algorithm/ Sub-Program :Example: Let R be a Ring of 2×2 Matrices over Integers

Step1: Start
Step2: Declare Array to Read Matrices of Order 2×2.
Step3: Read the Order of First Matrix $\boldsymbol{m} \times \boldsymbol{n}$and Second Matrix$\times \boldsymbol{b}$.
Step4:Enter the elements of First Matrix and make it a Loop Run to Form a Matrix.
Step4: Enter the elements of Second Matrix and make it a Loop Run to Form a Matrix.
Step5: Calculate the Sum of the two Matrices using **sum[c][d] = first[c][d] + second[c][d];**
Step6: Calculate the Multiplication of the two Matrices using **mul[c][d] = first[c][d] * second[c][d];**
Step7: Display the Calculated Matrices.
Step8 : Stop.

```
#include <stdio.h>

int main()
{
int m, n, c, d, first[6][6], second[6][6], sum[6][6];

printf("Enter the number of rows and columns of matrix\n");
scanf("%d%d", &m, &n);
printf("Enter the elements of first matrix\n");

for (c = 0; c < m; c++)
for (d = 0; d < n; d++)
scanf("%d", &first[c][d]);

printf("Enter the elements of second matrix\n");

for (c = 0; c < m; c++)
for (d = 0 ; d < n; d++)
scanf("%d", &second[c][d]);

printf("Sum of entered matrices:-\n");

for (c = 0; c < m; c++) {
for (d = 0 ; d < n; d++) {
sum[c][d] = first[c][d] + second[c][d];
printf("%d\t", sum[c][d]);
 }
printf("\n");
 }
```

```
return 0;
}
```

Prime Ideal
Definition: An Ideal ***P*** of a Ring $(\boldsymbol{R}, +, .)$ is said to be a Prime Ideal if
$\forall\, a, b \in R \;\Rightarrow\; aob \in P \;\Rightarrow either\ a \in P\ or\ b \in P.$ Where $'o'$ is a Binary Operation.

Example: Let Z be a Ring of Integers, Then$\{0\}$ is an Ideal which is a Prime Ideal of Z
$\forall\, m, n \in Z \;\ni mn \in \{0\} \Rightarrow mn = 0 \Rightarrow Either\ m = 0\ or\ n = 0.$
Sub-Program

Syntax:

```
#include<stdio.h>
#include<math.h>

void main()
{
int count;
intn,k , result,a,b;
printf("Enter that Range Integes = ");
scanf("%d",&n);
printf("Enter that starting Integer = ");
scanf("%d",&k);
for(count = -k; count <= n; count++)      // Display the numbers from Negative Integers to Positive
Integers//
printf("%d ", count);
printf("\n");
printf("Take two Integers a & b from the above Integer Set\n");
scanf("%d%d",&a,&b);
result=a*b;

if(result==0)
{
printf("The result a*b = %d , Either a or b is Zero\n Hence {0} is a Prime Ideal of Z",result );
}
else
{
printf("Both a & b zero/ Both are non-zero\n Hence {0} is not a Prime Ideal of Z");
 }
getch();
}
```

Cyclic Group
A group G is said to be Cyclic if there exists a ∈ G, Such that the Group Itself is generated by a.
Group can be denoted by$< a >$

Syntax:

```
#include <stdio.h>
int main()
{
int n, i;
printf("Enter the Considered Generator: ");/* Consider any Number as Generator*/
scanf("%d",&n);

for(i=1; i<=10; ++i)
```

```
{
printf("%d * %d = %d \n", n, i, n*i);
}
return 0;
}
```

II. CONCLUSION

This Research has a vast Advantages, as it contains Some Logical Programming Parts in C Language which can Further leads to the development of Applications in Programming Languages, and will Encourage Mathematical Logics to associate and Execute on the Front End of Computers in one go. In This Topics, we have Associated Algebraic Theory and Basic Concepts of Algorithm and Programming in C-Language with Some Executed Sub-Programs.

ACKNOWLEDGEMENT

I am Grateful to Dr.(Prof).SanatMandal,for his Co-operation and Guidance throughout this Research Topic.

REFRENCES

[1]. The C Programming Language Second Edition1988, ByBrian W. Kernighan, Dennis M. Ritchie, AT&T Bell Laboratories Murray Hill, New Jersey
[2]. An Introduction to the C Programming Language and Software Design, By Tim Bailey 2005.
[3]. Modern Abstract Algebra, By Lalji Prasad, Paramount Publications Patna, India.
[4]. Fermat's last theorem. A Genetic introduction to Algebraic Number theory (1977).By Harold M. Edwards p. 76.

Ahmad Sarosh. Algorithm and C Language Programming Concepts in Algebraic Structures and their Ideals." American Journal of Engineering Research (AJER), vol. 6, no. 11, 2017, pp. 121-127.