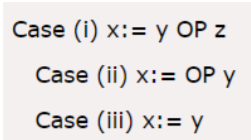
**Implementation of DAG**

**Aim:** To write a program to implement DAG

**Algorithm:**



• Step 1:

o If y operand is undefined then create node(y).

o If z operand is undefined then for case(i) create node(z).

• Step 2:

o For case(i), create node(OP) whose right child is node(z) and left child is node(y).

o For case(ii), check whether there is node(OP) with one child node(y).

o For case(iii), node n will be node(y).

• Output

o For node(x) delete x from the list of identifiers. Append x to attached identifiers list for the node n found in step 2. Finally set node(x) to n.

**Program:**

#include<stdio.h>

#include<string.h>

int i=1,j=0,no=0,tmpch=90;

char str[100],left[15],right[15];

void findopr();

void explore();

void fleft(int);

void fright(int);

struct exp

{

int pos;

char op;

}k[15];

int main()

{

printf("\t\tINTERMEDIATE CODE GENERATION OF DAG\n\n");

scanf("%s",str);

printf("The intermediate code:\t\tExpression\n");

findopr();

explore();

}

void findopr()

{

for(i=0;str[i]!='\0';i++)

if(str[i]==':')

{

k[j].pos=i;

k[j++].op=':';

}

for(i=0;str[i]!='\0';i++)

if(str[i]=='/')

{

k[j].pos=i;

k[j++].op='/';

}

for(i=0;str[i]!='\0';i++)

if(str[i]=='\*')

{

k[j].pos=i;

k[j++].op='\*';

}

for(i=0;str[i]!='\0';i++)

if(str[i]=='+')

{

k[j].pos=i;

k[j++].op='+';

}

for(i=0;str[i]!='\0';i++)

if(str[i]=='-')

{

k[j].pos=i;

k[j++].op='-';

}

}

void explore()

{

i=1;

while(k[i].op!='\0')

{

fleft(k[i].pos);

fright(k[i].pos);

str[k[i].pos]=tmpch--;

printf("\t%c := %s%c%s\t\t",str[k[i].pos],left,k[i].op,right);

for(j=0;j <strlen(str);j++)

if(str[j]!='$')

printf("%c",str[j]);

printf("\n");

i++;

}

fright(-1);

if(no==0)

{

fleft(strlen(str));

printf("\t%s := %s",right,left);

}

printf("\t%s := %c",right,str[k[--i].pos]);

}

void fleft(int x)

{

int w=0,flag=0;

x--;

while(x!= -1 &&str[x]!= '+' &&str[x]!='\*'&&str[x]!='='&&str[x]!='\0'&&str[x]!='-'&&str[x]!='/'&&str[x]!=':')

{

if(str[x]!='$'&& flag==0)

{

left[w++]=str[x];

left[w]='\0';

str[x]='$';

flag=1;

}

x--;

}

}

void fright(int x)

{

int w=0,flag=0;

x++;

while(x!= -1 && str[x]!= '+'&&str[x]!='\*'&&str[x]!='\0'&&str[x]!='='&&str[x]!=':'&&str[x]!='-'&&str[x]!='/')

{

if(str[x]!='$'&& flag==0)

{

right[w++]=str[x];

right[w]='\0';

str[x]='$';

flag=1;

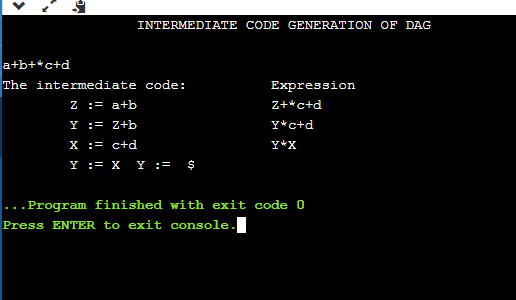
}

x++;

}

}

**Input / Output :**

****

**Result**: The program was successfully compiled and run.