ISSN 0975 3303

MJS, Vol.1, No. 2, Dec. 2002 - April 2003 pp.22-56

https://doi.org/10.12725/mjs.2.5

Some Features of C, C++ Language Illustrated Through Examples from Chemistry

K.V. RAMAN *

ABSTRACT

Computer programming has been used effectively by theoretical chemists and organic chemists to solve various types of problems in chemistry . Initially the languages used for computations in chemistry were FORTRANand BASIC.Later the Pascal language was used for solving problems in chemistry and physics. Recently the languages C and C++ and Java have been used to solve problems in chemistryIn this paper I will illustrate features of C,C++ choosing examples from chemistry.

Some examples presented in these languages are

Program to calculate reduced mass of homo diatomic or hetero diatomic

Program to calculate the molecular weight of a tetra atomic system ABCD

Program to calculate the Arrhenius parameters Ea and A

Program to compute NMR frequencies of spin ½ nuclei only

Program to calculate NMR and ESR frequencies

The examples presented in Java 2 are

Program to calculate unit cell dimension of a crystal

Program to generate the chair form and boat form of cyclohexane.

The examples presented in this monograph will help researchers in theoretical chemistry and organic chemistry to develop their own software.

* *Professor, (Reteired) Department of Chemistry St.Joseph's College (Autonomous) Trichy-620002 Phone (O) 0431 721390 (R) 0431 458270, Cell no. 98431 58270 email: ramankv 1@yahoo.com, ush_ram@satyam.net.in*

Introduction:

Theoretical chemists and organic chemists have used Computer programming effectively to solve various types of problems in chemistry . Initially FORTRAN [1] and BASIC[2] languages were used for computations in chemistry .Later the Pascal language[2] was used for solving problems in chemistry and physics. Recently the languages C and C++ [3-5] have been used to solve problems in chemistry. Prior to the C language, programmers had to choose between languages that optimized one set of traits or the other. For example although FORTRAN could be used to write fairly efficient programs for scientific applications it was not good for systems code.

Another important fact was that the languages FORTRAN and BASIC were not designed around structured programming .In the1990's new versions of FORTRAN, namely FORTRAN 90 and FORTRAN 95 have been introduced to bring many features available in the presently popular languages C and C++. The language Pascal had the structural elements but failed to include certain features necessary to make it apply for a wide range of programs .In the 1970's C was invented by Dennis Ritchie and included the features of structured programming and also useful for system code .To solve programs with complexity the structural programming concept had to be coupled with object oriented programming .C++ was invented by Bjarne Straustrup in 1979.This programming language was successful in the 80's and early 90's. Then the internet

And world wide web were initiated. The aspects of structured programming, object oriented programming and platform independent programs destined for distribution in internet were conceived in the new language JAVA[6] .The authors of this language are James Gosling, Patrick Naughton , Chris Warth, Ed Frank and Mike Sheridan of sun Microsystems conceived in 1991.The language was designed finally in 1995.Java 1.0,Java1.1 and Java 2 are the three different versions.

In this monograph I will illustrate features of C,C++ choosing examples from chemistry. The examples presented in this monograph will help researchers in theoretical chemistry and organic chemistry to develop their own software.

C Language in Chemistry

The C language invented by Dennis Ritchie has many interesting features which are presented below:

*Question mark operator

- concise for–loop
- Recursion
- Break
- Continue
- Pointers
- Structures
- Unions
- Macros

To execute all C programs using TURBO C compiler the following files have to be included.

#include<stdio.h>

#include<math.h>

In this monograph some of these features are illustrated through examples from chemistry.

One of the very important features of C language is the question mark operator which is equivalent to the IF-THEN ELSE statement. The form of the statement with this operator is

Variable=(condition)?exp1:exp2;

If condition is true Variable is assigned to expression 1 and if condition is false variable is assigned Exp2.The following program calculates reduced mass of a homo diatomic or hetero diatomic. The formulas for reduced masses of homo and hetero diatomics are given below:

$m(homo)=M_1/2N_o$ $m(hetero)=M_1*M_2/((M_1+M_2)*N_o)$

Program 1

/* Program to calculate reduced mass of homo diatomic or hetero diatomic. The number of.*/

```
/* inversion centers in the molecule is input. If number of inversion centers is zero
*/

/* the calculation is for hetero diatomic and if number of inversion centers is one*/

 /*the calculation is for homo diatomic */

# define AVO 6.023e23

main()

{

      char molecule[20];

       int  nin;

      double redm,m1,m2;

      printf("enter name of molecule\n");

      gets(molecule);

      printf("enter no. of inv.centers and masses\n");

      scanf("%d%lf%lf ",&nin,&m1,&m2);

     redm=(nin==0)?m1*m2/((AVO*(m1+m2)):m1/(2.0*AVO);

     printf("the reduced mass for %s is %10.4e grams\n",molecule,redm);

}
```

Input and output data:

enter name of molecule

hydrogen

enter no.of inv.centers and masses

1 1.008 35.45

the reduced mass for hydrogen is 8.3679e-25 grams

enter name of molecule

hydrogen_chloride

enter no.of inv.centers and masses

0 1.008 35.45

the reduced mass for hydrogen_chloride is 1.6273e-24 grams

The advantage of C is that the 4 statements in FORTRAN or BASIC

30 IF NIN=0 THEN 40 ELSE 70

40 REDM=M1*M2/((AVO*(M1+M2))

50 GO TO 80

60 REDM=M1/(2*AVO)

80.....

are replaced by the single statement

redm=(nin==0)?m1*m2/((AVO*(m1+m2)):m1/(2.0*AVO);

in C language.

The next important feature of C language is the concise for loop.

Program 2

This program calculates the molecular weight of a tetra atomic system ABCD(KCNS) from data on atomic weights of the 4 atoms.

```
double awts[4]={39.0,12.01,14.008,32.0};

main()

{
```

```
        char molecule[20];
        int I;
        printf("enter name of molecule\n");
        gets(molecule);
        for (i=0;mw=0.0;i<=3;mw+=awt[i],++i);
        printf("the mol.wt. of %s is %7.3f grams\n",molecule,mw);
}
```

Input and Output

Enter name of molecule

KCNS

The mol.wt of KCNS is 97.018 grams

Unlike FORTRAN and BASIC the for loop in C can have multiple loop variables. The same program when written in BASIC requires 4 statements for the for loop as shown below:

```
30 MW=0.0
40 FOR I=1 TO 4
50 MW=MW+AWT(I)
60 NEXTI
70....
```

The next program introduces the global and local variables and illustrates the use of global variable. Similar to subroutines we have function subprograms in C. In such programs only one output variable is returned back to the main program. Let us see the following function subprogram which calculates the temperature in Celsius from Kelvin temperature and absolute zero value.

Program 3

```
/* main program */
main()
{
        int kelvin,ab_ze=273,celsius,z;
        printf("enter Kelvin temp.\n");
        scanf("%d",&Celsius);
        z=sub(kelvin,ab_ze);
        printf("the Celsius temperature is %d degrees\n",z);
}
/* Function program*/
sub(x,y)
int x,y;
{
        return(x-y);
}
```

The variable ab_ze is copied onto argument y and the variable kelvin is copied on to x and the function program sub(x,y) evaluates x-y and returns it to the main program. A return statement returns only one output vale to the main. If we have a problem involving simultaneous equations in two unknowns this function program cannot be used and we can make use of global variables. The following program to calculate the Arrhenius parameters Ea and A illustrates this point.

PROGRAM 4

The Arrhenius equation for temperature dependence of rate constant is

$k = \exp(-E_a/RT)$

For two temperatures this equation is written as

$\ln k_1 = \ln A - E_a/RT_1$

$\ln k_2 = \ln A - E_a/RT_2$

These two simultaneous equations in the unknowns Ea and lnA can be solved and the program is written using global variables.

```
/* main program */

#define R=8.314

double xx,yy,k1,k2,t1,t2;/* global variables */

main()

{

        char reaction[30];

        double eact,pref;

        printf("give the name of the reaction\n");

        gets(reaction);

        printf("give k1,k2,t1 and t2\n");

        scanf("%lf%lf%lf%lf",&k1,&k2,&t1,&t2);

        sim_eq();

        pref=exp(xx);

        eact=yy;

        printf("the Arrhenius Parameters for %s are:\n", reaction);

        printf("-----------------------------------\n");

        printf(" the energy of activation is %10.4e joules \n",eact);

        printf("the preexponential factor is %10.4e per sec\n",pref);

        printf("-----------------------------------\n");
```

```
/* function program*/
/* The simultaneous equations ax+by=c; and dx +ey=f have the solutions */
/* x= ce-bf/(ae-bd) ; y=af-cd/(ae-bd) ;*/
sim_eq()  /* function definition*/
{
        double a1,b1,c1,d1,e1,f1,z1;
        a1=1.0;
        b1= -1.0/(R*t1);
        c1=logk1;
        d1=1.0;
        e1= -1.0/(R*t2);
        f1=logk2;
        z1= a1*e1-b1*d1;
        xx=(c1*e1-b1*f1)/z1;
        yy=(a1*f1-c1*d1)/z1;
}
```

Input and Output

Give the name of the reaction

Decomposition_of_benzaldehyde

Give k1,k2,t1 and t2

0.011 145.0 700.0 1000.0

[The Arrhenius Parameters for Decomposition of benzaldehyde are the energy of activation is 1.8403 e5 joules and the pre exponential factor is 5.951 e11 per sec

In this program xx and yy are global variables which can be used in main program as well as function program. Thus two output variables are taken to the main program using the technique of global variables. Another way to transfer more than one output variable to the main is by using pointers. The next program illustrates the use of pointers to obtain Arrhenius parameters.]

Program 5

```
#define R=8.314

main()

{

        char reaction[30];

        double eact,pref,a,b,c,d,e,f,*a1,*b1,*c1,*d1,*e1,*f1;

/* *a1,*b1,*c1,*d1,*e1 and *f1 arepointers to the variables a,b,c,d,e and f */

        doublek1,k2,t1,t2,x,y,*x1,*y1,eact,pref;

        a1=*a;b1=*b;c1=*c;d1=*d;e1=*e;f1=*f;x1=*x;y1=*y;

        printf("give the name of the reaction\n");

        gets(reaction);

        printf("give k1,k2,t1 and t2\n");

        scanf("%lf%lf%lf%lf",&k1,&k2,&t1,&t2);

        a=1.0;

        b= -1.0/(R*t1);

        c=logk1;
```

```
        d=1.0;
        e= -1.0/(R*t2);
        f=logk2;
        sim_eq(a1,b1,c1,d1,e1,f1,x1,y1);
        pref=exp(xx);
        eact=yy;
        printf("the ArrheniusParameters for %s are:\n", reaction);
        printf("-------------------------------------------\n");
        printf(" the energy of activation is %10.4e joules \n",eact);
        printf("the preexponential factor is %10.4e per sec\n",pref);
        printf("\n\n");
}
/* function program*/
sim_eq(aa1,bb1,cc1,dd1,ee1,ff1,xx1,yy1) /* function definition*/
double *aa1,*bb1,*cc1,*dd1,*ee1,*ff1,*xx1,*yy1;
{
        double z1;
        a1=1.0;
        b1= -1.0/(R*t1);
        c1=logk1;
        d1=1.0;
        e1= -1.0/(R*t2);
```

```
        f1 =logk2;
        z1 = (*aa1)*(*ee1)-(*bb1)*(*dd1);
        *xx1 =((*cc1)*(*ee1)-(*bb1)*(*ff1))/z1;
        *yy1 =((*aa1)*(*ff1)-(*cc1*(*dd1))/z1;
    }
```

Input and Output

Give the name of the reaction

Decomposition_of_benzaldehyde

Give k1,k2,t1 and t2

0.011 145.0 700.0 1000.0

the ArrheniusParameters for Decomposition_0f_benzaldehydeare

the energy of activation is 1.8403e5 joules

the preexponential factor is 5.9518e11 persec

Two more important features of C language are the break and continue statements. The break statement is used to quit from the loop based on a condition and execute statements following the loop. The break is used to quit from innermost loop only in nested loops.. The continue statement is used to skip a particular iteration in a loop. The program computes NMR frequencies of spin ½ nuclei only using the formula

n=mH/lh

Program 6

```
/* Use of break statement in loops */
/* It is possible to quit from the loop and transfer the control*/
/* to the statement following the loop */
 /* using a condition. In this program the NMR frequencies of nuclei with */
```

```
/* spin=1/2 only are calculated. If the nucleus has quadrupole moment then*/
/* its NMR frequency is not computed since such nuclei have spin */
/* greater than 1/2. */
#define H 6.626e-34
#define BN 5.05e-27
#define CF 1.0e06
main()
{
        char nucleus[20];
        int i;
        double field=1.0,moment,spin,qmom,fre;
        for(i=1;i<=5;++i){
        printf("enter name of nucleus\n");
        gets(nucleus);
        printf("enter value of quadrupole moment\n");
        scanf("%lf",&qmom);
        if(qmom!=0.0){
        printf("the nucleus has quadrupole moment and control exits theloop\n");
        break;}
        printf("enter mag.moment and spin values\n");
        scanf("%lf%lf",&moment,&spin);
        fre=fabs(moment)*field/(spin*H*CF);
        printf("the NMR frequency for %s is %7.3f MHz\n",nucleus,fre);
        }
```

}

Input and Output

Enter name of nucleus

Hydrogen

Enter value of quadrupole moment

0.0

enter magnetic moment and spin values

2.79285 0.5

the NMR frequency for hydrogen is 42.571MHz

enter name of nucleus

Gallium_69

enter value of quadrupole moment

0.2318

the nucleus has quadrupole moment and control exits the loop

Program 7

```
/* The program uses continue statement */

/*When this statement is used the specific iteration is skipped if condition is true
and */

/* the control is transferred to carry out the next iteration of loop*/

/* This program counts the total number of fermions between atomic numbers 1
and 6 */

/* A fermion has odd number of nucleons and a Boson has even number of
nucleons */

main()

{

        char atom[20];
```

```
int atnr, nenr, massnumber, count=0;
for(atnr=1;atnr<=6;++atnr){
printf("enter name of atom\n");
gets(atom);
printf("enter no. of neutrons\n");
scanf("%d",&nenr);
massnumber=atnr+nenr;
if(massnumber%2==0){
printf("the number of nucleons is even and ");
printf("nucleus is a Boson. control skips this iteration\n");
continue;
}
++count;
printf("the total number of nucleons between atomic numbers 1 and 6 is %d\n",count);
}
```

Input and Output

```
Enter name of atom
Hydrogen
Enter no.of neutrons
0
enter name of atom
Helium
Enter no.of neutrons
2
```

the number of nucleons is even and nucleus is a Boson. Control skips this iteration

enter name of atom

lithium

enter no. of neutrons

4

enter name of atom

beryllium

enter no. of neutrons

5

enter name of atom

boron

enter no. of neutrons

6

enter name of atom

carbon

enter no. of neutrons

6

the number of nucleons is even and nucleus is a Boson. Control skips this iteration

the total number of fermions between atomic numbers 1 and 6 is 4

A very important feature of C, C++, Pascal ,FORTRAN 90 and Java is recursion. In recursion a function can call itself. Recursion is applicable to calculations when something can be defined in terms of itself. Examples are summation of numbers or finding factorials of numbers. The factorial of a number is written as

!n= n.(!(n-1)). For example !4 = 4 .!3.The following program computes the number of valence diagrams for a conjugated cyclic system.

Program 8

```
/* Computing no. of valence diagrams for cyclic conjugated systems */

/* using the formula  res=!n/((!n/2) x (!(n/2 +1)) */

main()
```

```
{
        char system[25];
        int res,npi, m, p, j;
        float fact();
        for(j=1|j<=3;++j){
        printf("enter no. of pi electrons\n");
        scanf("%d",&npi);
        printf("enter name of system\n");
        gets(system);
        m=npi/2;
        p=m+1;
        res=fact(npi)/((fact(m)*fact(p));
printf("the number of valence diagrams for %s is %d\n", system, res);}
}
float fact (i)
int i;
{
      if(i==0)
        return(1);
          else
              return(i*fact(i-1));
}
```

Input and Output

Enter no. of pi electrons

6

enter name of system

benzene

the number of valence diagrams for benzene is 5

Enter no. of pi electrons

10

enter name of system

naphthalene

the number of valence diagrams for naphthalene is 42

Enter no. of pi electrons

14

enter name of system

anthracene

the number of valence diagrams for anthracene is 429

The next program calculates the empirical and molecular formula of an organic compound containing carbon, hydrogen, oxygen and nitrogen or a hydrocarbon. The program uses the break statement to quit from inner loop based on a condition.

Program 9

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
main()
{
        char system[20];
        int j,n,nc,nh,no,nn,nc1,nh1,no1,nn1,d5,d6,d7,d8;
        int p1,p2,p3,p4,nx,t,t1,t2,t3,t4,k,code;
        double atc,ath,ato,atn,emwt,mwt,x1,x2,x3,x4,d,d1,d2,d3,d4;
        double pc, ph, po, pn, f;
        printf("enter no. of system\n");
```

```
scanf("%d",&n);
for(j=1;j<=n;++j){
printf("enter name of system and at.wts of C,H,O,N\n");
scanf("%s%lf%lf%lf%lf",system,&atc,&ath,&ato,&atn);
printf("enter mol.wt.\n");
scanf("%lf",&mwt);
printf("enter percentages of elements\n");
scanf("%lf%lf%lf%lf",&pc,&ph,&po,&pn);
x1=pc/(100.0*atc);
x2=ph/(100.0*ath);
x3=po/(100.0*ato);
x4=pn/(100.0*atn);
d=x1;
for(k=1;k<=10;++k){
f=d/(float)k;
d1=x1/f; d2=x2/f ; d3=x3/f; d4=x4/f;
d5=(int)(d1*10.0+0.5);
d6=(int)(d2*10.0+0.5);
d7=(int)(d3*10.0+0.5);
d8=(int)(d4*10.0+0.5);
p1=(int)(d5/10);
p2=(int)(d6/10);
p3=(int)(d7/10);
p4=(int)(d8/10);
t1=d5-p1*10;
t2=d6-p2*10;
t3=d7-p3*10;
```

```
t4=d8-p4*10;
t=t1+t2+t3+t4;
if(t==0)
break;
}
        nc=p1;nh=p2;no=p3;nn=p4;
emwt=nc*atc+nh*ath+no*ato+no*ato;
nx=(int)(mwt/emwt);
nc1=nx*nc;
        nh1=nx*nh;
        no1=nx*no;
        nn1=nx*nn;
        printf("enter code value\n");
        scanf("%d",&code);
        if(code==1){
        printf(" the emp. formula for the org. compd. is \n");
        printf("C%dH%dO%dN%d\n",nc,nh,no,nn);
        printf("the Mol. formula is\n");
        printf("C%dH%dO%dN%d\n",nc1,nh1,no1,nn1);
        }
        else{
        printf(" the emp. formula for the hydrocarbon is\n");
        printf("C%dH%d\n",nc,nh);
        printf("the Mol. formula is\n");
        printf("C%dH%d\n",nc1,nh1);
```

```
        }
        }
        getch();
        return;
}
```

Input and Output

Enter no. of systems

2

enter name of system and atwts of C,H,O and N

nitrosopropane 12.01 1.008 16.0 14.008

enter mol.wt

73.094

enter percentages of C,H,O and N

49.32 9.59 21.91 19.18

enter code value

1

the empirical formula of the org. compd. is C3H7O1N1

the Mol. formula is C3H7O1N1

enter name of system and atwts of C,H,O and N

methane 12.01 1.008 16.0 14.008

enter mol.wt

16.0

enter percentages of C,H,O and N

75.0 25.0 0.0 0.0

enter code value

2

the empirical formula of the org.compd. is C1H4

the Mol. formula is C1H4

Another important feature of C is macro definition. This creates simple functions. The following program calculates the maximum among rms velocity and average velocity of a gas using macro definition.

Program 10

```
/*Program using macros to obtain the maximum among rms and average velocities*/
/* of oxygen gas*/
#define MAX_VEL(c1,c2) (c1>c2)?c1:c2)
main()
{
        char molecule[25];
        double gcon=8.314e7;pi=3.14159,temp=298.0;
        double mwt=32.0,p,z1,z2,vmax;
        printf("enter name of molecule\n");
        gets(molecule);
        p=gcon*temp/mwt;
        z1=sqrt(8.0*p/pi);
        z2=sqrt(3.0*p);
        vmax=MAX_VEL(z1,z2);
        printf("the maximum velocity for%s is\n",molecule);
        printf("equal to%10.4e cm/sec\n",vmax);
}
```

Input and Output

Enter name of molecule

Oxygen

The maximum velocity for oxygen is 4.8195e04 cm per sec

The next program uses nested macros to compute the maximum among three velocities, namely rms velocity average velocity and most probable velocity of a gas.

Program 11

```
/*Program using macros to obtain the maximum among rms and average and
most */

/* probable velocities of oxygen gas at 298 K*/

#define MAX_VEL(c1,c2) (c1>c2)?c1:c2)

main()

{
        char molecule[25];

        double gcon=8.314e7;pi=3.142,temp=298.0;

        double mwt=32.0,p,z1,z2,vmax,z3;

        printf("enter name of molecule\n");

        gets(molecule);

        z1=sqrt(8.0*p/pi);

        z2=sqrt(3.0*p);

        z3=sqrt(2*p);

        printf("the average velocity is %10.4e cm/sec\n",z1);

        printf("the rms velocity is %10.4e cm/sec\n",z2);
```

```
        printf("the most probable velocity is %10.4e cm/sec\n",z3);
        vmax=MAX_VEL(z1,MAX_VEL(z2,z3)));/*use of nested macros */
        printf("the maximum velocity for%s is\n",molecule);
        printf("equal to%10.4e cm/sec\n",vmax);
}
```

Input and Output

Enter name of molecule

Oxygen

The average velocity is 4.4400e04 cm/sec

The rms velocity is 4.8195e04 cm/sec

The most probable velocity is3.9351e04 cm/sec

The maximum velocity for oxygen is equal to 4.8195e04 cm/sec

In C++ the inline functions are used instead of macro definition.

C++ IN CHEMISTRY

C++ language is an enhancement and extension of C and has been invented by Bjorne Strausstrup. It is an object oriented language and has many important features not available in C

Some of them are

*scope resolution operator

*classes

*inline functions

*function overloading

*inheritance

These features will be illustrated using examples from chemistry

The scope resolution operator is used in three important situations.

1) Suppose a program has the same variable name for global and local variables this operator(: :)can be used to access the global variable.

2) Functions consisting of several lines are defined outside what is called the class structure and form class members and can be invoked by using scope resolution operator.

3) The scope resolution operator is also used in inheritance which is supported by allowing one class to incorporate another class in its declaration.

The following programs illustrate these points.

The next program calculates nmr frequency and esr frequency using same variable name fre. The global variable fre(NMR) is also used to calculate chemical shift. The relevant formulas are

$n_{NMR}=g_n b_n H/h$ $n_{ESR}=g_e b_e H/h$ chemical shift =splitting $x10^6/n_{NMR}$

PROGRAM 12

```
#include<iostream.h>

#include<math.h>

#include<conio.h>

double fre;

const double plank =6.626e-34;

const double cf=1.0e06;

const double cf1=1.0e09;

const double bn=5.05e-27;

const double bm=9..274e-24;

main()

{

        char nucleus[20],radical[20];

        double fre, field, mom, chshi(), cs;

        cout<<" enter name of nucleus and name of radical\n";
```

```
        cin>>nucleus>>radical;
        cout<<"enter values of gn and ge and field\n";
        cin>>gn>>ge>>field;
        ::fre=gn*bn*field/(plank*cf);
        fre=ge*be*field/(plank*cf1);
        cs=chshi();
cout<<"the nmr frequency for <<nucleus<<"is="<<::fre<<"MHz\n";
cout <<"the esr frequency for "<<radical<<"is="<<::fre<<"GHz\n";
cout<<"the chem. shift for "<<nucleus<<"is="<<cs<<"ppm\n";
        return 0;
}
double chshi()
{
double shift,sepn=40.0;
        shift=sepn /::fre;
        return(shift);
}
```

Input and Output

Enter name of nucleus and name of radical

proton DPPH

Enter values of gn and ge and field

5.5 2.0 1.0

the nmr frequency for proton is=41.918201 MHz

the esr frequency for DPPH is=27.992756 GHz

the chem. shift for proton is= 0.954239 ppm

The next program deals with inline functions which are equivalent to the macro

definition in C. Normally we use function calls in dealing with function subprograms. but alternatively the code of functions can be expanded inline at the point of each invocation. For expanding a function the keyword inline is used and precedes the function definition. When a function is called and returned a significant amount of overhead is generated . By inline expansion programs run faster. But inline is generally used for small functions. The following program computes the maximum among three velocities of a gas using inline approach. This program has been earlier presented using macros in C.

Program 13

```
#include<iostream.h>

#include<math.h>

#include<conio.h>

//Program using inline code to obtain the maximum among rms and average and
most

// probable velocities of oxygen gas at 298 K

 const double gcon=8.314;

const pi=3.142,

const temp=298.0;

inline double max(double c1,double c2)

{

    return  (c1)>(c2)?(c1):(c2);

}

main()

{

        char gas[25];

        double mwt,m1,m2,m3;

        cout<<"enter name of gas and mol.wt.\n";
```

```
        cin>>gas>>mwt;
        m1=sqrt(8.0*gcon*temp/(pi*mwt));
        m2=sqrt(3.0*gcon*temp/mwt);
        m3=sqrt(2*gcon*temp/mwt);
        cout<<"the average velocity is" <<m1<<" m/sec\n";
        cout<<"the rms velocity is "<<m2<<" m/sec\n";
        cout<<"the most probable velocity is" <<m3<<" m/sec\n";
        cout<<"the max.velocityfor"<<gas<<"is=\n";
cout<<max(m1,max(m2,m3))<<"m/sec\n";
        getch();
        return 0;
}
```

Input and Output

enter name of gas and mol.wt.

oxygen 32.0e-03

the average velocity is 443.997021 m/sec

the rms velocity is 481.946444 m/sec

the most probable velocity is 393.507624 m/sec

the max. velocity for oxygen is=481.946444 m/sec

When functions contain more than one line inline code cannot be used and we can make use of scope resolution operator for such functions. The following program calculates the de Broglie wavelengths of an electron accelerated through different potentials. The concepts of modulus operator, integer division, division of float by integer and nested if else statements are also illustrated in this program. The de broglie wavelength is given by formula

lamda= $h/(2meV)^{1/2}$.

Using the values of mass of electron, charge of electron the formula is reduced as lamda= $(150/V)^{1/2}$ in angstrom units.

The listing of the program with input and output is presented below:

PROGRAM 14

```
#include<iostream.h>
#include<math.h>
#include<conio.h>
class w_len
{       private :
        int v[5],i,x,y;
        double lamda,x1;
        public:
        w_len();
        void calc();
}
        w.len::w_len() //use of scope resolution operator
        {for(i=1;i<=4;i++)
         {cout<<"value of accelerating potential:";
           cin>>v[i];
        }
}
        void w.len::calc()
        {
         for(i=1;i<=4;i++){
         y=150%v[i];
        x=150/v[i];
```

```
        x1=150.0/v[i];
        if(v[i]<=150){
                if(y==0)
                lamda=sqrt((double)x);
                else
                lamda=sqrt(x1);
                }
                else
                lamda=sqrt(x1);
        cout<<"\nThe de Broglie wavelength of electron";
        cout<<"at "<<v[i]<<" volts is="<<lamda<<"angstroms\n";}
}
void main()
{
      clrscr();
      w_len wl;
      wl.calc();
      getch();
}
```

Input and output

Value of acc.potential:1

Value of acc.potential:150

Value of acc.potential:34

Value of acc.potential:15000

The de Broglie wavelength of electron at 1volts is =12.247449angstroms

The de Broglie wavelength of electron at 150volts is = 1angstroms

The de Broglie wavelength of electron at 34 volts is =2.10042angstroms

The de Broglie wavelength of electron at 15000 volts is =0.1angstroms

Another important feature of C++ is function overloading. In the C library there are three functions ,namely abs(),labs() and fabs().These functions obtain the magnitude of a an integer or long integer or floating point quantity. Although these three functions perform identical actions, three different names are used. But in C++ we can use the same name for all the three functions. The functions that share the same name are said to overloaded. The following program uses the same name sum() for adding two integers or two floating point variables .The first sum() gives the sum of atomic number and neutron number and classifies whether the nuclei are Bosons or fermions. If mass number is even(then mass number%2 becomes 0) then the nucleus is a Boson. Otherwise the nucleus is a Fermion. The second sum() computes the mol.wts of 10 diatomics by adding at.wts. of individual atoms.. At.nr and neutron number are integer variables whereas at.wt and mol.wt. are floating point variables.

Program 15

```
#include<iostream.h>

#include<math.h>

#include<conio.h>

int  sum(int atnr,int nenr);

float sum(float awt1,float awt2);

main()

{

        int atnr,j;

        float awt1,awt2,mw[11];

        int sum1,count=0,count1=0;

        for(atnr=1;atnr<=10;++atnr){

        cout<<"enter neutron number\n";

        cin>>j;
```

```
        cout<<"enter atwts.\n";
        cin>>awt1>>awt2;
        clrscr();
        if((sum(atnr,i)%2)!=0){
        ++count;}
        else{
        ++count1;}
        mw[atnr]=sum(awt1,awt2);
        }
        for(atnr=1;atnr<=10;++atnr)
        cout<<"the mol.wt.of system"<<atnr<<" is ="<<mw[atnr]<<"
grams\n";
        cout<<"the number of Bosons is"<<count1<<"\n";
        cout<<"the number of Fermions is"<<count<<"\n";
        getch();
        return 0;
}
int sum(int atnr,int nenr)
{
 return atnr+nenr;
}
float sum(float awt1,float awt2)
{
 return awt1+awt2;

}
```

Input and output

```
enter neutron number
0
enter atwts
1.008 35.45
```

enter neutron number
2
enter atwts
1.008 127.0
enter neutron number
4 enter atwts
1.008 79.0
enter neutron number
5
enter atwts
1.008 1.008
enter neutron number
6
enter atwts
35.45 35.45
enter neutron number
6
enter atwts
1.008 2.016
enter neutron number
7
enter atwts
16.0 14.008
enter neutron number
8
enter atwts
39.0 35.45
enter neutron number
10
enter atwts
19.0 19.0
enter neutron number
10

enter atwts

35.45 19.0

the mol.wt of system 1 is 36.458 grams

the mol.wt of system 2 is 128.007996 grams

the mol.wt of system 3 is 80.008003 grams

the mol.wt of system 4 is 2.016grams

the mol.wt of system 5 is 70.900002 grams

the mol.wt of system 6 is 3.024 grams

the mol.wt of system 7 is 30.007999 grams

the mol.wt of system 8 is 74.449997 grams

the mol.wt of system 9 is 38.0 grams

the mol.wt of system 10 is 54.450001 grams

the number of Bosons is 5

the number of fermions is 5

Another salient feature of C++ is inheritance. In C++ inheritance is supported by allowing one class to incorporate another class in its declaration. Inheritance allows a hierarchy of classes to be built ,moving from the most general to the most specific. A simple example is given below.

Building class is declared and it has the features rooms, floors and area. Using this definition of building we can create derived classes for house and school.. They also have rooms ,floors and the building area. The following program includes molecule as the base class and atom y ,atom z as derived classes. The molecular weight of a tri atomic molecule ABC is calculated in this program

Here also scope resolution operator is used.

SUMMARY:

In this manuscript some features of C ,C++ have been illustrated using examples from chemistry.

EXPERIMENTAL: The C and C++ programs were run using Turbo C compiler in an IBM PC.

REFERENCES:

1. T.L.Isenhour and P.C. Jurs (1979) Introduction to Computer Programming in FORTRAN , London Allyn &Bacon Inc.,

2. K.Ebert ,L. Iderer and T.L. Isenhour (1989) Computer applications in chemistry New York VCH publishers.

3. K.V.Raman (1993)Computers in Chemistry New Delhi Tata McGraw-Hill Publishing company ,

4. K.V.Raman (1992)The salient features of C language illustrated through examples from chemistry Chemistry Education New Delhi Wiley International company

5. K.V.Raman (1996) Some unique features of BASIC, Pascal and C languages illustrated through examples from chemistry Chemistry Education .. New Delhi Wiley International company

6 Patrick Naughton and Herbert Schildt (1999)Java ™ 2 The Complete Reference 3rd edition New Delhi Tata McGraw Hill Publishing Co.,