Introduction to C++ Programming

K Prasanna Kumar

Contents

Author is from 5G Testbed, Dept. of ECE, IISc Bangalore, Email: kprasannakumar.iith@gmail.com, prasannakk@iisc.ac.in

***Abstract*—This module gives a quick recap of C++ Programming. C++ is required for the Radio Driver Development for RF Engineers.**

1 Introduction

1) It is a general purpose, high level, procedural & object oriented language
2) Programming file has an extension ".cpp" and the Linux compiler is "g++"

2 Fundamentals

Problem 1. Arithmetic operations of two numbers

```
/* Basic Arthmetic Operation using
    C++
Author : K. Prasanna Kumar */

#include<iostream>
using namespace std;
int main()
{
int a ;
int b;
cout<<"Enter the value of a : ";
cin>>a;
//"cin" is similar to scanf
cout<<"Enter the value of b : ";
cin>>b;
```

```
int add = a+b;
cout<<"Sum of a & b : "<<add<<endl
    ;
int sub= a-b;
cout<<"Difference of a & b : "<<
    sub<<endl;
int mul= a*b;
cout<<"Product of a & b : "<<mul
    <<endl;
float div=a/b;
cout<<"Fraction of a & b : "<< div
    <<endl;
return 0;
}
```

Problem 2. Arithmetic operations of two numbers using C header files

```
/*Basic Arthmetic Operations using
    C++
Author : K. Prasanna Kumar */

#include<stdio.h>
int main()
{
int a;
int b;
printf("Enter the value of a : ");
scanf("%d", &a);
printf("Enter the value of b : ");
scanf("%d", &b);

int add = a+b;
printf("Sum of a & b is %d \n",add
    );
int sub = a-b;
printf("Differnece of a & b is %d
    \n", sub );
int mul = a*b;
printf("Product of a & b is %d \n"
    , mul);
float div = a/b;
printf("Fraction of a & b is %
    float \n", div );

/* By this program we can know
    that C libs can be
called in c++ */
return 0;
}
```

Problem 3. Logical operations using C++

```
/*Basic Logical Operations in C++
Author : K. Prasanna kumar*/

#include<iostream>
using namespace std;
int main()
{
int a;
int b;
cout<<"---------- Input Should
    Ehter 1 or 0 ----------"<<endl;
cout<<"Enter logical value of a : "
    ;
cin>>a;
cout<<"Enter logical value of b : "
    ;
cin>>b;

int and_op = a&&b;
cout<<"And Operation of a , b : "
    <<and_op<<endl;
int or_op=a||b;
cout<<"OR Operation of a , b : "<<
    or_op<<endl;
int not_op=!a;
cout<<"NOT Operation of a : "<<
    not_op<<endl;
return 0;
}
```

Problem 4. Find the size of a datatype, including modifiers

```
/* Description: Program to find
    the size of DATATYPES including
    modifiers
Author : K. Prasanna Kumar */

#include<iostream>
// <iostream> is inherent of <
    stdio.h>

using namespace std;

int main()
{
```

```
cout<<"Size of Integer = "<<sizeof
    (int)<<endl;
//"cout" is like "printf"
//"endl" is like "\n" and "<<" is
    like ","
cout<<"Size of Unsigned Integer =
    "<<sizeof(unsigned int)<< endl;
cout<<"Size of Signed Integer = "
    <<sizeof(signed int)<<endl;
cout<<"Size of Shot Integer = "<<
    sizeof(short int)<<endl;
cout<<"Size of Unsigned Short
    Integer = "<<sizeof(unsigned
    short int)<<endl;
cout<<"Size of Signed Short
    Integer = "<<sizeof(signed short
     int)<<endl;
cout<<"Size of long Integer = "<<
    sizeof(long int)<<endl;
cout<<"Size of Unsigned long
    Integer = "<<sizeof(unsigned
    long int)<<endl;
cout<<"Size of Signed long Integer
    = "<<sizeof(signed long int)<<
    endl;
cout<<"Size of Charator = "<<
    sizeof(char)<<endl;
cout<<"Size of Unsigned Charator =
    "<<sizeof(unsigned char)<<endl;
cout<<"Size of Signed Charator = "
    <<sizeof(signed char)<<endl;
cout<<"Size of Float = "<<sizeof(
    float)<<endl;
cout<<"Size of Double = "<<sizeof(
    double)<<endl;
cout<<"Size of Long Double = "<<
    sizeof(long double)<<endl;
cout<<"Size of Boolen = "<<sizeof(
    bool)<<endl;
// Datatype "Boolen" is not
    avalible in c

return 0;
}
```

3 Bit-wise Operators

Problem 5. Bit-Wise Logical Operations

```
#include<iostream>
int main()
{
int a, b;
std::cout<<"Enter the value of a :
    ";
std::cin>>a;
std::cout<<"Enter the value of b :
    ";
std::cin>>b;

std::cout<<"Bitwise AND Operation
    : "<<(a&b)<<std::endl;
std::cout<<"BItwise OR Operation :
    "<<(a|b)<<std::endl;
std::cout<<"Bitwise NOT Operation
    of a : "<<(~a)<<std::endl;
std::cout<<"Bitwise NOT Operation
    of b : "<<(~b)<<std::endl;

return 0;
}
```

Problem 6. Bit-Wise XOR

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
cout<<"Enter the value of a : ";
cin>>a;
cout<<"Enter the value of b : ";
cin>>b;

cout<<"Bit-Wise a XOR b : "<<(a^b)
    <<endl;
return 0;
}
```

Problem 7. Left Shift

Problem 8. Right Shift

4 Storage Class

5 Conditional Statements

5.1 if-else

Problem 9. Finding positive or negative integer

```
#include<iostream>
int main()
```

```
{
int x;
std::cout<<"Enter any integer as
    the value of x, except zero ... :"
    ;
std::cin>>x;

if (x>0)
std::cout<<"x is a positive
    integer"<<std::endl;
else
std::cout<<"x is a negative
    integer"<<std::endl;

return 0;
}
```

5.2 if-elseif

Problem 10. Finding a number belongs to which set of integers

```
#include<iostream>
int main()
{
int x;
std::cout<<"Enter the value of x :
    ";
std::cin>>x;

if (x>0)
{
std::cout<<"x is a Positive
    Integer "<<std::endl;
std::cout<<"It is also a Natural
    number" <<std::endl;
}
else if (x<0)
std::cout<<"x is a Negative
    Integer"<<std::endl;
else
{
std::cout<<"x belongs to Non-
    negative integers"<<std::endl;
std::cout<<"x belongs to Whole
    Numbers "<<std::endl;
}


return 0;
}
```

5.3 switch case

6 Iteration Statements

6.1 While loop

Problem 11. Printing the multiplication table of nth number

```
#include<iostream>
int main()
{
int n;
std::cout<<"Enter the numbe : ";
std::cin>>n;

int i =1;
while(i<=20)
{
std::cout<<n<<" x "<<i<<" = "<<n*i
    <<std::endl;
i++;
}

return 0;
}
```

Problem 12. Printing the multiplication table of nth number in reverse order

```
#include<iostream>
int main()
{
int n;
std::cout<<"Enter the numbe : ";
std::cin>>n;

int i = 20;
while(i>0)
{
std::cout<<n<<" x "<<i<<" = "<<n*i
    <<std::endl;
i--;
}

return 0;
}
```

6.2 do while

Problem 13. Termination of Print by Entering ”q”

```
#include<iostream>
int main()
{
char a;

do{
std::cout<<"Hi.."<<std::endl<<"I␣
   am␣working␣with␣SoapySDR"<<std::
   endl;
std::cout<<"press␣any␣key␣␣to␣
   continue␣or␣Enter␣q␣to␣terminate
   ␣"<<std::endl;
std::cin>>a;
}while(a!='q');

return 0;
}
```

6.3 for loop

Problem 14. Sum of n natural numbers formula verification

```
#include<iostream>
int main()
{
int n, sum, sum_formula;
std::cout<<"Enter␣the␣value␣of␣n␣:
   ␣";
std::cin>>n;

// Calucalation of Sum of n
   natural numbers using formala
sum_formula = n*(n+1)/2;

//Calucalation of Sum of n natural
    numbers
sum =0;
for (int i = 1; i<=n; i++)
{
 sum = sum + i;
}

std::cout<<"Sum␣using␣Formula␣:␣"
   <<sum_formula<<std::endl;
std::cout<<"Sum␣using␣for_loop␣:␣"
   <<sum<<std::endl;
return 0;
}
```

7 Functions

8 Arrays

8.1 One-Dimensional Array

Problem 15. Printing a linear array whose range is [0 2π], with an interval of 0.1

```
/*
Desciption : Froming an array with
    an intervial 0.1
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
*/

#include<iostream>

using namespace std;

int main()
{
float pi = 22/7;
int div = (2*pi)/0.1;
// Deculration of array with an
   internal of 0.1 from 0 to 2*pi
float arr[div];


// Initialization of array
float j =0; //Initial value of
   array

cout<<"Array␣=␣{";
for(int i =0; i<=div; i++)
{
 arr[i] = j;
 cout<<arr[i]<<",";
 j=j+0.1;
}
cout<<"}"<<endl;

return 0;
}
```

Problem 16. Printing a linear array of sinusoidal function values in range [0 2π], with an interval of 0.1

```
/*
Desciption : Froming an array with
    an intervial 0.1
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
*/

#include<iostream>
#include<cmath>
using namespace std;

int main()
{
float pi = 22/7;
int div = (2*pi)/0.1;
// Deculration of array with an
   internal of 0.1 from 0 to 2*pi
float arr[div];
float sin_arr[div];

// Initialization of array
float j =0; //Initial value of
   array

cout<<"Array = {";
for(int i =0; i<=div; i++)
{
 arr[i] = j;
 sin_arr[i]=sin(arr[i]);
 cout<<sin_arr[i]<<",";
 j=j+0.1;
}
cout<<"}"<<endl;

return 0;
}
```

8.2 Multi-Dimensional Array

Problem 17. Printing a matrix

```
/*
Description : Printing a Martrix
   Initilized at Compile Time
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
*/
#include<iostream>
using namespace std;

int main()
{
// Decularation and  Compile Time
   Initilization
int mat[3][2] = {{3, 2},{6,
   4},{9,8}};

// Printing a matrix
cout<<"A = " <<endl<<"[";
for (int i = 0; i<3; i++)
{
 for(int j=0; j<2; j++)
 {
        cout<<mat[i][j]<<"\t";
 }
 cout<<endl;
}
cout<<"]"<<endl;

return 0;
}
```

Problem 18. Printing 3D Array

```
/*
Description : Runtime
   Initilization and Printing of 3D
    Array
Programmer : K. Prasanna Kumar
Last Modified Date : 14th May 2021
*/

#include<iostream>
using namespace std;

int main()
{
// Decularation
int three_D[3][2][4];

// Run Time Initilization
cout<<"----------Initlization of 3
   D Array -------"<<endl;
for(int i=0 ; i<3; i++)
{
 for(int j=0; j<2; j++ )
 {
  for(int k=0; k<4; k++)
  {
   cout<<"Enter the value of
```

```
        element in ("<<i<<","<<j<<","
        <<k<<") : ";
     cin>>three_D[i][j][k];
    }
  }
}

// Printing 3D Array
cout<<"------- Three Dimensional
    Array ----------"<<endl<<"L = {"
    <<endl;
for(int i=0 ; i<3; i++)
{
 cout<<"{";
 for(int j=0; j<2; j++ )
 {
  cout<<"{";
  for(int k=0; k<4; k++)
  {
   cout<<three_D[i][j][k]<<"\t";
  }
  cout<<"}"<<",";
 }
 cout<<"}"<<endl;
}
cout<<"}"<<endl;

cout<<endl<<"Multi Dimentional
    Array is the array of arrys"<<
    endl;
return 0;
}
```

9 Pointer

Problem 19. Introduction to Pointer

```
#include<iostream>
int main()
{
int a =1;
float b = 2.0;
char c = 'd';

int *A; float *B; char *C;
A = &a;
B = &b;
C = &c;

std::cout<<"Address of Integer
    variable a : "<<A<<std::endl;
std::cout<<"Address of float
    Variable b : "<<B<<std::endl;
std::cout<<"Address of Charactor
    Variable c : "<<C<<std::endl;

return 0;
}
```

Problem 20. Size of a Pointer

```
#include<iostream>
int main()
{
int a = 1;
int *p =&a;
std::cout<<"Address of Integer
    Variable a : "<<p<<std::endl;
std::cout<<"Size of Integer
    Pointer Variable : "<<sizeof(p)
    <<std::endl;

float b = 2.0;
float *p1;
std::cout<<"Address of float
    Variable b : "<<p1<<std::endl;
std::cout<<"Size of float Pointer
    Variable : "<<sizeof(p1)<<std::
    endl;

char c = 'd';
char *p2;
std::cout<<"Address of Charactor
    Variable "<<p2<<std::endl;
std::cout<<"Size of Char Pointer
    Variable "<<sizeof(p2)<<std::
    endl;

std::cout<<std::endl<<"Analysis"<<
    std::endl;
std::cout<<"Size of the Pointer
    Variable of any data_type is 8
    Bytes"<<std::endl;

return 0;
}
```

Note : Size of the pointer depends up platform.

Problem 21. Void Pointer

```
/*
Description : Void Pointer Example
```

```
Programmer : K. Prasanna Kumar
Last Modified Date : 20th August
   2021

Note :
1. Void Pointer can hold the
   address of any type
2. Can be type casted to any type
*/

#include<iostream>
using namespace std;

int main()
{
int a = 10;
char b = 'x';

//Declaration of void pointer
void *p;

p = &a;
cout<<"Address of the integer type
   stored in p : "<<p<<endl;

p= &b;
cout<<"Address of the char type
   stored in p : "<<p<<endl;

cout<<endl<<"Analysis :"<<endl<<"
   Void Pointer Variable can store
   address of any type"<<endl;

cout<<"Size of the Void Pointer is
   : "<<sizeof(p)<<endl;
return 0;
}
```

Problem 22. Null Pointer

```
#include<iostream>
int main()
{

int *p = NULL;
std::cout<<"Value of Null Pointer
   : "<<p<<std::endl;
std::cout<<"Size of Null Pointer :
   "<<sizeof(p)<<std::endl;
return 0;
}
```

Problem 23. Pointer to pointer

```
#include<iostream>
using namespace std;
int main()
{
float A[5] = {10, 20, 30, 40, 50};
float *ptr;
ptr = A;
cout<<"Base address of array A : "
   <<A<<endl;
cout<<"Address stored in Pointer
   Variable (ptr) : "<<ptr<<endl;
cout<<"Element at ptr : "<<*ptr<<
   endl<<endl;

float **ptptr;
ptptr = &ptr;
cout<<"Address of a Pointer
   Variable stored in Pointer
   Variable (ptptr) : "<<ptptr<<
   endl;
cout<<"Address stored in Pointer
   Variable (ptr) : "<<*ptptr<<endl
   ;
cout<<"Element at the address
   stored in Pointer Variable (ptr)
   : "<<**ptptr<<endl;

return 0;
}
```

10 Address & Pointer Arithmetic's

Problem 24. Fundamental Parameters of 1D array

```
#include<iostream>
int main()
{
int a[5] = {1, 2, 3, 4, 5};
int size_of_a = sizeof(a);
int length_of_a = sizeof(a)/sizeof
   (int);

std::cout<<"Size of the array : "
   <<size_of_a<<std::endl;
std::cout<<"Length of the array :
   "<<length_of_a<<std::endl;
```

```
std::cout<<"Base Address of the
    array : a = "<<a<<std::endl;
std::cout<<"Element of an array at
    index 0 : a[0] = "<<a[0]<<std::
    endl;
std::cout<<"Element of an array at
    the base address : *a = "<<*a
    <<std::endl;

return 0;
}
```

Problem 25. Printing 1D array elements using address arithmetic's

```
#include<iostream>
int main()
{
int a[5] = {1, 2, 3, 4, 5};
std::cout<<"Base Address of the
    array : a = "<<a<<std::endl;
std::cout<<"Element of an array
    with index 0 : a[0] = "<<a[0]<<
    std::endl;
std::cout<<"Element of an array
    with base address as pointer : *
    a = "<<*a<<std::endl;

std::cout<<std::endl;
std::cout<<"Pinting the address of
    array element using Address
    Arthematic : "<<std::endl;
std::cout<<" Addr_a = {";
for(int i = 0; i<5; i++)
{std::cout<<a + i <<" ";};
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing the array
    elements using Address
    Arthematic : "<<std::endl<<"a =
    {";
for (int i = 0; i<5; i++)
{std::cout<<*(a + i)<<" ";};
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing the array
    elements using index : "<<std::
    endl<<"a = { ";
for(int i = 0; i<5; i++)
{std::cout<<a[i]<<" "; }
std::cout<<"}"<<std::endl;
return 0;
}
```

Problem 26. Fundamentals of 2D array

```
#include<iostream>
int main()
{
int a[5][2] = {{01,11}, {02, 12},
    {03, 13}, {04, 14}, {05, 15}};

std::cout<<"Base Address of the 2D
    Array : a = "<<a<<std::endl;
std::cout<<"Size of 2D array ... :"
    <<sizeof(a)<<std::endl;

std::cout<<std::endl;
std::cout<<"Base Address of the
    first 1D array in 2D array : *a
    = "<<*a<<std::endl;
std::cout<<"Size of the 1D array
    in 2D array : "<<sizeof(*a)<<std
    ::endl;

std::cout<<std::endl;
std::cout<<"No. of 1D arrays in 2D
    array : "<<sizeof(a)/sizeof(*a)
    <<std::endl;
std::cout<<"Lenght of 1D array in
    2D array : "<<sizeof(*a)/sizeof(
    int)<<std::endl;

std::cout<<std::endl;
std::cout<<"Element at base
    address of 2D array : "<<**a<<
    std::endl;
std::cout<<"Element at index (0,
    0) of 2D array : "<<a[0][0]<<std
    ::endl;
return 0;
}
```

Problem 27. Printing 2D array elements using address arithmetic's

```
#include<iostream>
int main()
{
```

```
int a[5][3] = {{01,11,21}, {02,
    12, 22}, {03, 13, 23}, {04, 14,
    24}, {05, 15, 25}};

std::cout<<"Base Address of 2D
    array : a "<<a<<std::endl;
std::cout<<"Base Address of 1st 1D
    array of 2D array : "<<*a<<std
    ::endl;
int sz_of_2D_ary_A = sizeof(a);
int sz_of_1D_ary = sizeof(*a);
int num_1d_arys = sz_of_2D_ary_A/
    sz_of_1D_ary;
int len_1d_ary = sz_of_1D_ary/
    sizeof(int);
std::cout<<std::endl;

std::cout<<"Base addresses of 1D
    arrays in 2D array : "<<std::
    endl;
std::cout<<"{ ";
for(int i = 0; i<num_1d_arys; i++)
{std::cout<<a + i<<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Addresses of Elements
    in 1st 1D array of 2D array :"<<
    std::endl;
std::cout<<"{ ";
for(int i = 0; i<len_1d_ary; i++)
{std::cout<<*a + i<<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Address of all the
    Elements in 2D array A : "<<std
    ::endl;
std::cout<<"{ ";
for (int i = 0; i<num_1d_arys; i
    ++)
{
 std::cout<<"{ ";
 for(int j =0; j<len_1d_ary; j++)
 {std::cout<<*(a + i) +j<<" ";}
 std::cout<<"}"<<std::endl;
}
std::cout<<"}"<<std::endl<<std::
    endl;
std::cout<<"Elements in the 1st 1D
    array of 2D array "<<std::endl;
std::cout<<"{ ";
for(int i=0; i<len_1d_ary; i++)
{std::cout<< *(*a + i) + 0 <<" ";}
std::cout<<"}"<<std::endl;
std::cout<<"Elements in the 2nd 1D
    array of 2D array "<<std::endl;
std::cout<<"{ ";
for(int i=0; i<len_1d_ary; i++)
{std::cout<< *(*a + i) + 1 <<" ";}
std::cout<<"}"<<std::endl;
std::cout<<"Above arrays are
    printed using address arithmetic
    "<<std::endl;

std::cout<<std::endl;
std::cout<<"Printing elements of 2
    D array using Address Arithmetic
    "<<std::endl;
std::cout<<"{ ";
for (int i = 0; i<num_1d_arys; i
    ++)
{
 std::cout<<"{ ";
 for(int j =0; j<len_1d_ary; j++)
 {std::cout<<*(*(a + i) +j)<<" ";}
 std::cout<<"}"<<std::endl;
}
std::cout<<"}"<<std::endl;
return 0;
}
```

Problem 28. Fundamentals of 3D array

Problem 29. Printing 3D array elements using address arithmetic's

Problem 30. Pointer Array : array which contains the address of arrays or variables

```
// Array of Pointers
#include<iostream>
int main()
{
int A[5] = {1, 2, 3, 4, 5};
int B[5] = {11, 12, 13, 14, 15};
int C[5] = {21, 22, 23, 24, 25};
int *buffs[] = {A, B, C};
```

```
std::cout<<"Base Address of array
    A : "<<A<<std::endl;
std::cout<<"Base Address of array
    B : "<<B<<std::endl;
std::cout<<"Base Address of array
    B : "<<C<<std::endl;

std::cout<<std::endl;
std::cout<<"Base address of
    Pointer Array (*buffs[]) : "<<
    buffs<<std::endl;

std::cout<<std::endl<<"Printing
    address using Address
    arithematic"<<std::endl;
for (int i = 0; i<3; i++)
{std::cout<<"Address in index "<<
    i <<" of buffs : "<<*(buffs + i)
    <<std::endl;}

std::cout<<std::endl<<"Printing
    address using index"<<std::endl
    ;;
for (int i = 0; i<3; i++)
{std::cout<<"buffs["<<i<<"] = "<<
    buffs[i]<<std::endl;}
std::cout<<std::endl;

std::cout<<"Printing address of
    elements in A : "<<std::endl;
std::cout<<"{";
for(int i = 0; i<5; i++)
{std::cout<<buffs[0] + i <<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing elements in A
    using *buffs[] "<<std::endl;
std::cout<<"{";
for(int i = 0; i<5 ; i++)
{std::cout<<*(buffs[0] + i)<<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing elements in B
    using *buffs[] "<<std::endl;
std::cout<<"{";
for(int i = 0; i<5 ; i++)
{std::cout<<*(buffs[1] + i)<<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing elements in C
    using *buffs[] "<<std::endl;
std::cout<<"{";
for(int i = 0; i<5 ; i++)
{std::cout<<*(buffs[2] + i)<<" ";}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing address of
    elements in A, B & C as a matrix
    "<<std::endl;
std::cout<<"{";
for(int i = 0; i<3; i++)
{
 std::cout<<"{";
 for(int j = 0; j<5; j++)
 { std::cout<<buffs[i] + j<<" "; }
 std::cout<<"}"<<std::endl;
}
std::cout<<"}"<<std::endl<<std::
    endl;

std::cout<<"Printing elements in A
    , B & C as a matrix "<<std::endl
    ;
std::cout<<"{";
for(int j = 0; j<3; j++)
{
std::cout<<"{";
for(int i = 0; i<5 ; i++)
{std::cout<<*(buffs[j] + i)<<" ";}
std::cout<<"}"<<std::endl;
}
std::cout<<"}"<<std::endl;
return 0;
}
```

11 Memory Management

11.1 Address Mapping

Problem 31. Copying data from one memory address to another memory address using "memcpy"

```
#include<iostream>
#include <cstring>
int main()
{
int ar1[5] =  {10,20,30,40,50};
```

```
int ar2[5];

std::memcpy(ar2, ar1, sizeof(ar1))
    ;
// function Syntax :
//void * memcpy(void * destination
    , const void * source, size_t
    num);

std::cout<<"Size of the array2 : "
    <<sizeof(ar2)<<std::endl;
std::cout<<"Length of the array2 :
    "<<sizeof(ar2)/sizeof(int)<<std
    ::endl;
std::cout<<"Elements in array 2 : "
    <<std::endl<<"{";
for(int i =0; i<sizeof(ar2)/sizeof
    (int); i++)
{std::cout<<ar2[i]<<" ";}
std::cout<<"}"<<std::endl;
return 0;
}
```

Problem 32. Copying data from one memory address to another memory address using "memmove"

```
#include<iostream>
#include<cstring>
int main()
{
int arr_0[5] =
    {1,10,100,1000,10000};
int arr_1[5];

std::memmove(arr_1, arr_0, sizeof(
    arr_0));
// function Syntax :
//void * memmove(void *
    destination, const void * source
    , size_t num);

std::cout<<"Size of array_1 : "<<
    sizeof(arr_1)<<std::endl;
std::cout<<"Lenght of the array_1
    : "<<sizeof(arr_1)/sizeof(int)<<
    std::endl;
std::cout<<"Elements of array_1 :
    "<<std::endl<<"{";
for(int i = 0; i<sizeof(arr_1)/
    sizeof(int); i++)
{std::cout<<arr_1[i]<<" ";}
std::cout<<"}"<<std::endl;

std::cout<<"Elements in array_0 :
    "<<std::endl<<"{";
for(int i = 0; i<sizeof(arr_0)/
    sizeof(int); i++)
{std::cout<<arr_0[i]<<" ";}
std::cout<<"}"<<std::endl;

return 0;
}
```

Problem 33. memset()

11.2 Dynamic Memory Allocation

12 Structures

Problem 34. Printing marks list log

```
#include<iostream>
// Defining a structure
struct student
{
 int id;
 char name[20];
 float percentage;
};

int main()
{
// Initialization of Stracture
    Variables
struct student s1 = {1, "Prasanna
    Kumar", 85};
struct student s2 = {2, "Shrey
    More", 90};

std::cout<<"Student Details : "<<
    std::endl;
std::cout<<s1.id<<" "<<s1.name<<"
    "<<s1.percentage<<std::endl;
std::cout<<s2.id<<" "<<s2.name<<"
    "<<s2.percentage<<std::endl;

return 0;
}
```

Problem 35. Size of structure element and structure

```
#include<iostream>
struct student {
 int id;
 char name[19];
 float percentage;
};
int main()
{
struct student s1;
std::cout<<"Size of Structure
   Elements : "<<std::endl;
std::cout<<"id  : "<<sizeof(s1.id)
   <<std::endl;
std::cout<<"name : "<<sizeof(s1.
   name)<<std::endl;
std::cout<<"percentage : "<<sizeof
   (s1.percentage)<<std::endl;

std::cout<<std::endl;
std::cout<<"Size of the Structure
   with above elements : "<<sizeof(
   s1)<<std::endl;

return 0;
}
```

Problem 36. Display a progress report of a student, by calculating percentage using member function of a structure

```
#include<iostream>
struct progress_report
{
 int S_No;
 std::string name;
 float maths;
 float physics;
 float chemistry;
 float  percentage;

 void percentage_calculation()
 {
   percentage =  ((physics/60) + (
      chemistry/60) + (maths/150))
      *100/3;
 }

};

int main()
{
 struct progress_report s1;
 s1.S_No = 1;
 s1.name = "Prasanna Kumar";
 s1.maths = 150;
 s1.physics = 56;
 s1.chemistry = 51;
 s1.percentage_calculation();
 std::cout<<"Students Progress
    Report : "<<std::endl;
 std::cout<<s1.S_No<<" "<<s1.name
    <<std::endl;
 std::cout<<"Mathematics : "<<s1.
    maths<<std::endl;
 std::cout<<"Physics : "<<s1.
    physics<<std::endl;
 std::cout<<"Chemistry : "<<s1.
    chemistry<<std::endl;
 std::cout<<"Percentage : "<<s1.
    percentage<<std::endl;

 struct progress_report s2;
 // Similarly we can do it for s2
}
```

13 Classes & Objects

Problem 37. Introduction to public class

```
/*
Description : Attendence of a
   student using Classes in cpp
Programmer : K. Prasanna Kumar
Last Modified Date : 24th April
   2021

Syntax :
Definition : --
class CLASS_NAME
{
        public:

        datatype variable_1;
        datatype variable_2;
        ...
        ...
// Note: varibles deculared in the
    class are called Artributes or
   features or Data
```

```
        return_type function_one(
           arg_1, arg_2)
        {

        }

        return_type function_two()
        {

        }
        -----
        ------

// Note : Functin defined in the
   class are also known methods
}

return_type main()
{
Decularation :--
class_name object_name ;

Initialization :--
object_name.varible_1 = ;
object_name.varible_2 = ;

object_name.function_one(
   actval_arg1, actual_arg2);
};

*/

#include<iostream>
using namespace std;

// Definition :
class student
{
public :

        int id;
        char name[20];

        void attendence()
        {
          cout<<"Present "<<endl<<
             endl;
        }

};

int main()
{
 // Deculration:
 student sandeep;

 // Runtime Initialization
 cout<<"Enter id of student : ";
 cin>>sandeep.id;
 cout<<"Enter name of student : ";
 cin>>sandeep.name;
 sandeep.attendence();
 cout<<sandeep.name<<endl;
return 0;
}

/*
Note :
1. We can declure multiple objects
    for a class
2. Classes are as similar as
   stractures
3. CLASS_NAME is similar as
   TAGLINE in stractures
4. Object_Name is sikilar as
   stracture_variable
5. Differnce between class and
   stracture is class contains both
    data & function, but stractures
    has only data
*/
```

14 Constructor & Destructor

Problem 38. Introduction to constructor (Default Constructor)

```
/*
Description : Default Constractor
Programmer : K. Prasanna Kumar
Last Modified Date : 29th June
   2021

Definition : Constractor is
   special type of member function.
*/


#include<iostream>
using namespace std;
```

```
class SoapySDR
{
 public :

 int Tx, Rx;
        SoapySDR()
        {
         cout<< "Welcome to
            SoapySDR"<<endl;
         Tx = 0; Rx = 1;
        }
};

int main()
{

 SoapySDR soapy;
 int *tx = &soapy.Tx, *rx = &soapy
    .Rx;

 cout<<"Tx Stream Setup Address : "
    <<tx<<endl;
 cout<<"Tx Stream Setup Value : "
    <<*tx<<endl<<endl;

 cout<<"Rx Stream Setup Value : "
    <<*rx<<endl;
 cout<<"Rx Stream Setup Address : "
    <<&soapy.Rx<<endl;
return 0;
}
```

Problem 39. Finding the Surface areas & Volume of a box using parameterized constructor

```
/*
Description : Example for
   Parameterized Constructors
Programmer : K. Prasanna kumar
Last Modified Date : 29th July
   2021

Note : This example tell us that
   the member functions are
to return multiple varibles.
*/


#include<iostream>
using namespace std;

class box
{
private :
int l, b, h;

public :
int L_Sur_Area;

 box(int length, int breadth, int
    height)
 {
   l = length; b = breadth; h =
      height;

   float T_Sur_Area = l*b + b*h +
      l*h;
   cout<<"Total Surface Area of
      Cubiod : "<<T_Sur_Area<<endl;

   L_Sur_Area = 2*h*(l + b);
 }

 int  Volume(){ return l*b*h;};

 float Sur_Area1, Sur_Area2,
    Sur_Area3;
 void Surface_Areas()
 {
  Sur_Area3 = l*b;
  Sur_Area1 = l*h;
  Sur_Area2 = b*h;
 }

};

int main()
{
 box cubiod(3, 2, 5);
 cout<<"Later Surface Area of
    Cubiod : "<<cubiod.L_Sur_Area<<
    endl;

 cout<<"Volume of Cubiod : "<<
    cubiod.Volume()<<endl;

 cout<<endl<<"Face Area's of
    Cubiod are :--- "<<endl;
 cubiod.Surface_Areas();
```

```
 cout<<"Area of Face 1 : "<<cubiod
     .Sur_Area1<<endl;
 cout<<"Area of Face 2 : "<<cubiod
     .Sur_Area2<<endl;
 cout<<"Area of Face 3 : "<<cubiod
     .Sur_Area3<<endl;

return 0;
}
```

15 Inheritance

15.1 Single Inheritance

```
#include<iostream>
class Base
{
 public :
 int x, y;

 Base()
 {
 std::cout<<"I am in Base Class"<<
     std::endl;
 x= 10; y=20;
 }

 int addition(int x1, int x2)
 {
  int x3 = x1+x2;
  return x3;
 }

};

class Derived : public Base
{
    public :
    float p, q;

    Derived()
    {
    std::cout<<"I am in Derived
        Class "<<std::endl;
    p = 56.99; q = 31.453;
    }

    void substraction( int x1, int
        x2)
    {
    int x3 = x1-x2;
    std::cout<<"Difference of two
        values is : "<<x3<<std::endl;
    }

};

int main()
{
 Derived d;
 std::cout<<"Member Variables of
     Base Class : "<<d.x<<" "<<d.y<<
     std::endl;
 std::cout<<"Member Variables of
     Derived Class : "<<d.p<<" "<<d.
     q<<std::endl;

 int A, B,C;
 std::cout<<"Enter the value of A
     : ";
 std::cin>>A;
 std::cout<<"Enter the value of B
     : ";
 std::cin>>B;
 C = d.addition(A, B);
 std::cout<<"Addition of A and B
     is : "<<C<<std::endl;
 d.substraction(A, B);
return 0;
}
```

Note : Base class object cannot access the constructor, variable, member functions of the derived class

15.2 Multilevel Inheritance

```
#include<iostream>
class Grand_Parent
{
 public :
  int a, b;
  Grand_Parent()
  {
   std::cout<<"I am in Grand
       Parent Class"<<std::endl;
   a = 23; b= 24;
  }

};
```

```
class Parent : public Grand_Parent
{
   public :
   int x, y;
   Parent()
   {
    std::cout<<"I am in Parent
       Class "<<std::endl;
    x = 2*a; y=x+b;
   }

};

class child : public Parent
{
   public :
   child()
   {
     std::cout<<"I am in child
        class"<<std::endl;
   }

   void det()
   {
     std::cout<<"If 2X2 matrix is
        formed as" <<std::endl;
     std::cout<<a<<" "<<b<<std::
        endl<<x<<" "<<y<<std::endl;
     std::cout<<"Det of the matrix
        is : "<<a*y-b*x<<std::endl
        ;
   }

};

int main()
{
 child c;
 std::cout<<"Varible value in
    Grant Parent Class : : "<<c.a<<"
    "<<c.b<<" "<<std::endl;
 std::cout<<"Varible value in
    Parent Class : "<<c.x<<" "<<c.y
    <<" "<<std::endl;
 c.det();
 return 0;
}
```

15.3 Multiple Inheritance

```
#include<iostream>
class Father
{
   public :
   int f1, f2;
   Father()
   {
    std::cout<<"I am in Father
       Class "<<std::endl;
    f1 = 10; f2 = 15;
   }
};

class Mother
{
   public :
   float m1, m2;
   Mother()
   {
    std::cout<<"I am in Mother
       Class "<<std::endl;
    m1 = 7.1; m2 = 0.8;
   }
};

class child : public Mother,
   public Father
{
 public :
  child()
  {
   std::cout<<"I am in child Class
      "<<std::endl;
  }

 void det()
 {
 std::cout<<"Matrix A = "<<std::
    endl;
 std::cout<<f1<<" "<<f2<<std::endl
    <<m1<<" "<<m2<<std::endl;
 std::cout<<"Det of A is : "<<f1*
    m2 - f2*m1<<std::endl;
 }
};

int main()
{
```

```
child c;
std::cout<<"Varibles in Father
    Class : "<<c.f1<<" "<<c.f2<<std
    ::endl;
std::cout<<"Varibles in Mother
    Class : "<<c.m1<<" "<<c.m2<<std
    ::endl;
c.det();
return 0;
}
```

Note: In all types of inheritance the default access specifier is "pravite". So, it is advised to mention access specifier during inheritance.

16 Polymorphism

16.1 Compile-Time Polymorphism

Problem 40. Constructor overloading

```
#include<iostream>
class Constructor
{
 public :
 int x, y, z;
 Constructor()
 {
    std::cout<<"Default Constructor
       "<<std::endl;
    x = 10; y = 20;
 }

 Constructor(int a)
 {
  std::cout<<"Parameterized
     Constructor "<<std::endl;
  z = a;
 }

};

int main()
{
Constructor obj1;
Constructor obj2(5);
std::cout<<"Value of x in above
    class : "<<obj1.x<<std::endl;
std::cout<<"Value of y in above
    class : "<<obj1.y<<std::endl;
std::cout<<"Value of z in above
    class : "<<obj2.z<<std::endl;

return 0;
}
```

Default Constructor and Parameterized Constructor can be implemented for the same class using different objects

Problem 41. Function overloading

```
#include<iostream>
class overloading
{
  public :

  int add()
  {
   int a, b;
   std::cout<<"Enter the value of
      a : ";
   std::cin>>a;
   std::cout<<"Enter the value of
      b : ";
   std::cin>>b;
   int c = a+b;
   return c;
  }

  void add(int a, int b)
  {
  std::cout<<"Addition : "<<a+b<<
     std::endl;
  }

};

int main()
{

 overloading f;
 int x = f.add();
 std::cout<<"Addition of a and b
    is : "<<x<<std::endl;
 f.add(5, 6);

return 0;
}
```

16.2 Run-Time Polymorphism

```
#include<iostream>
class Base
{
  public :
  void add()
  {
  int a, b;
  std::cout<<"Enter the value of a
     : ";
  std::cin>>a;
  std::cout<<"Enter the value of b
     : ";
  std::cin>>b;
  std::cout<<"Summation of a & b
     is : "<<a+b<<std::endl;
  }

};

class Derived : public Base
{
 public :
 void add()
 {
  float p = 8.2, q = 4.5;
  std::cout<<"Addition of p and q
     is : "<<p+q<<std::endl;
 }
};

int main()
{
 Derived d;
 d.add();

 Base b;
 b.add();

return 0;
}
```

Note :

1) Run-Time Polymorphism is also know as OVERRIDING
2) Run-Time Polymorphism is implemented using inheritance concept.

SDR Driver code using SoapySDR architecture is designed with the help of Run-Time polymorphism of *Device.cpp*

17 Namespace

Problem 42. Introduction to namespace

```
/*
Description : Introduction to
   namespace concept
Programmer : K. Prasanna Kumar
Last Modified Date : 02/07/2021
*/

#include<iostream>
using namespace std;

// User Defined Namespace
namespace first
{
 int a = 500;
}

// Globel Variable
int a = 200;

int main()
{
 // Variable inside the main
    function
 int a = 100;

 // Output
 cout<<"Value of variable a  from
    the namespace : ";
 cout<<first::a<<endl;
 cout<<"Value of variable a  from
    the Globel Variable : ";
 {
 extern int a;
 cout<<a<<endl;
 }
 cout<<"Value of vairable a  from
    the main function : ";
 cout<<a<<endl;

return 0;
}
```

Problem 43. Display variable with same name from various namespace

```
/*
```

```
Description : Printing the value
   of the variable with same name
   from vairous namespaces
Programmer : K. Prasanna Kumar
Last Modified Date : 02-July-2021
*/

#include<iostream>
using namespace std;

// User Defined namespaces
namespace one
{
 int X = 10;
}
namespace two
{
 int X = 20;
 int Y = 50;
}
namespace three
{
 int X = two::Y + one::X;
}

int main()
{

 cout<<"Value of X from namespace
    one is : "<<one::X<<endl;
 cout<<"Value of X from namespace
    two is : "<<two::X<<endl;
 cout<<"Value of X from namespace
    three is : "<<three::X<<endl;

return 0;
}
```

Problem 44. Defining a function in namespace

```
/*
 Description : Defining a function
     which gives sin value in the
    interval [0, 2pi]
 in the namespace and calling it
    in main function
 Programmer : K. Prasanna Kumar
 Last Modified Date : 7th July
    2021
 */

#include<iostream>
#include<math.h>
using namespace std;

// Defining a name space
namespace SoapySignal
{       void sin_fun(int ar_size,
   float ar[], float sin_ar[])
        { for (int i = 0; i<
           ar_size; i++)
          {sin_ar[i] = sin(ar[i])
             ;}
        }
}

int main()
{       float T = 2*22/7;
        float Ts = 0.1;
        int t_size = T/Ts;
        float t[t_size];
        for (int i = 0; i<t_size;
           i++)
        { if (i == 0)
                t[0] = 0;
          else
                t[i] = t[i-1] + Ts
                   ;
        }

        float sint[t_size];

        //calling function defined
            in namespace
           SoapySignal
        SoapySignal::sin_fun(
           t_size, t, sint);

        // Displaying t
        cout<<"Time Scale Values :
           t ="<<endl;
        for (int i = 0; i<t_size;
           i++)
        {cout<<t[i]<<" ";}
        cout<<endl; cout<<endl;

        // Displaying Sin of t
        cout<<"Sin function Values
           for corrosponding t
           values : sin(t) = "<<
```

```
        endl;
    for (int i = 0; i<t_size;
        i++)
    {cout<<sint[i]<<" ";}
    cout<<endl;

    return 0;
}
```

Problem 45. Declaring a function in namespace

```
/*
 Description : Defining a function
     which gives sin value in the
    interval [0, 2pi],
 declaring it in the namespace and
     calling it in main function
 Programmer : K. Prasanna Kumar
 Last Modified Date : 7th July
    2021
 */

#include<iostream>
#include<math.h>
using namespace std;

// Defining a name space
namespace SoapySignal
{
    void sin_fun(int , float
        ar[], float sin_ar[]);
}

int main()
{   float T = 2*22/7;
    float Ts = 0.1;
    int t_size = T/Ts;
    float t[t_size];
    for (int i = 0; i<t_size;
        i++)
    { if (i == 0)
            t[0] = 0;
      else
            t[i] = t[i-1] + Ts
                ;
    }

    float sint[t_size];

    //calling function defined
        in namespace
        SoapySignal
    SoapySignal::sin_fun(
        t_size, t, sint);

    // Displaying t
    cout<<"Time Scale Values :
        t =  "<<endl;
    for (int i = 0; i<t_size;
        i++)
    {cout<<t[i]<<" ";}
    cout<<endl<<endl;

    // Displaying Sin of t
    cout<<"Sin function Values
        for corresponding t
        values : sin(t) = "<<
        endl;
    for (int i = 0; i<t_size;
        i++)
    {cout<<sint[i]<<" ";}
    cout<<endl;

    return 0;
}

// Defining a function for
    SoapySignal namespace

void SoapySignal::sin_fun(int
    ar_size, float ar[], float
    sin_ar[])
{ for (int i = 0; i<ar_size; i++)
    {sin_ar[i] = sin(ar[i]);}
}
```

Problem 46. Defining a class in namespace

Problem 47. Declaring a class in namespace

18 File Handling

References

[1] Sundeep Saradhi Kanthety, "C++ PROGRAMMING", youtube video playlist https://www.youtube.com/playlist?list=PLLOxZwkBK52Ad3zogUDbxIVGo-rgx4Pum