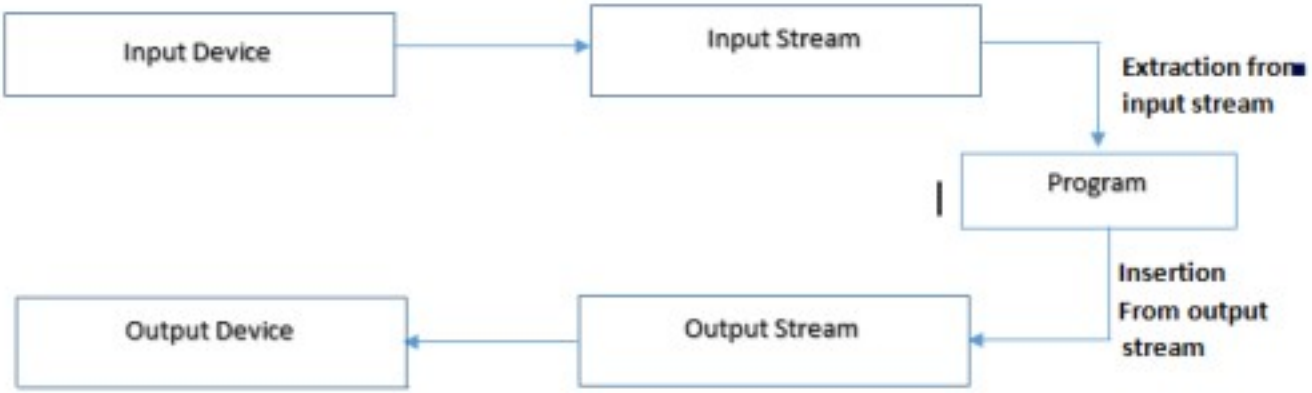
1 Explain Input stream and Output stream in brief.

- I/O system પ્રોગ્રામરને એક ઇન્ટરફેસ આપે છે જે actual device ને access કરવા માટે independent હોય છે. આ ઇન્ટરફેસને ***stream*** તરીકે ઓળખવામાં આવે છે.
- stream એ bytes નું sequence છે. તે કા તો જ્યાંથી ઇનપુટ ડેટા મેળવી શકાય છે એવા source તરીકે કામ કરે છે અથવા એક destination તરીકે કે જ્યાં આઉટપુટ ડેટા મોકલી શકાય છે.
- પ્રોગ્રામને ડેટા પૂરા પાડતા source stream ને *input stream* અને પ્રોગ્રામમાંથી આઉટપુટ મેળવનાર destination stream ને *output stream* કહેવામાં આવે છે.
- બીજા શબ્દોમાં કહીએ તો, પ્રોગ્રામ input stream માંથી bytes ને મેળવે છે અને તે બાઇટ્સને output stream માં inserts કરે છે. જે નીચે આપેલા આકૃતિમાં બતાવેલ છે:



- આપણે જાણી છીએ કે cin એ input stream ને દર્શાવે છે કે જે standard input device સાથે જોડાયેલુ હોય છે અને cout એ output stream ને દર્શાવે છે કે જે standard output device સાથે જોડાયેલુ હોય છે

Header files available in C++ for Input – Output operation are:

- **iostream**: iostream એ standard input output stream માટે વપરાય છે આ header file માં cin, cout, cerr જેવા objects ની definitions હોય છે.
- **iomanip**: iomanip એ input output manipulators માટે વપરાય છે. આ header file માં setw, setprecision જેવા manipulator ની definitions હોય છે
- **fstream**: આ header file main file stream ને describes કરે છે. આ header file નો ઉપયોગ file માથી data કે જે input તરીકે read કરવામાં આવે છે તેને handle કરવા માટે વપરાય છે. અને તે data file માં output તરીકે લખવામાં આવે છે.

Example :

```
#include<iostream>
using namespace std;
int main()
{
   int age;
   cout << "Enter your age:";
   cin >> age;
   cout << "\nYour age is: "<<age;
```

```
    return 0;
}
```

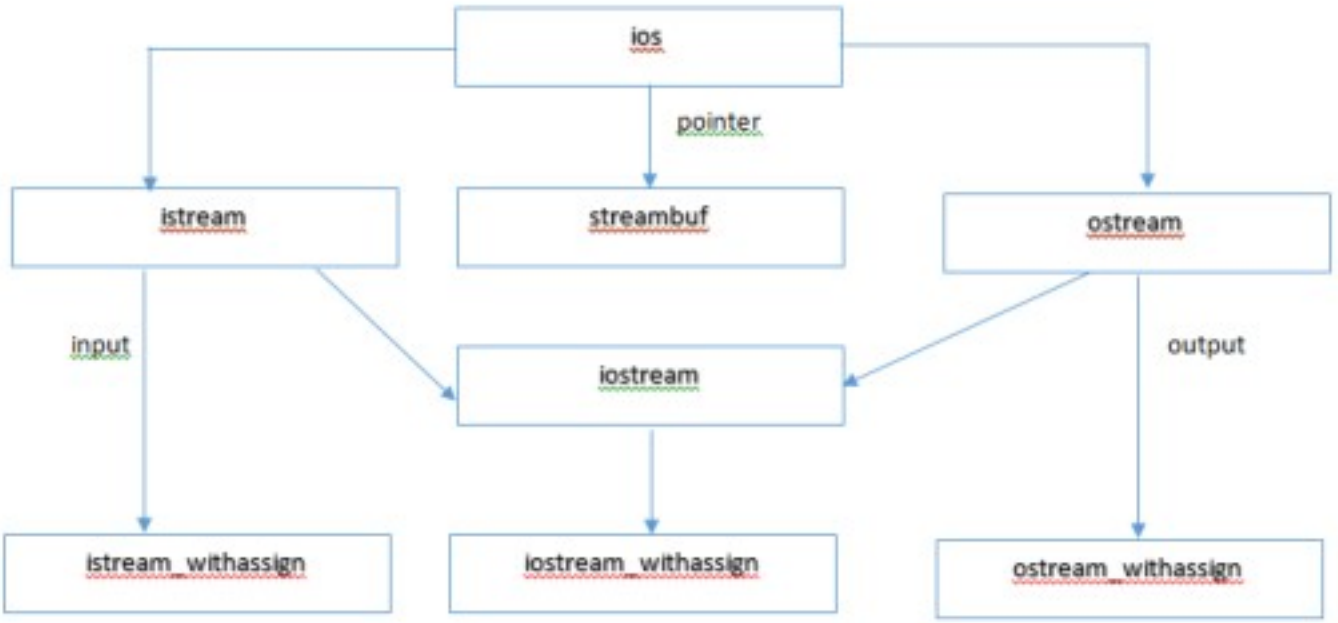
Output :

Enter your age: 18

Your age is: 18

2 Explain C++ Stream class hierarchy.

- C++ I/O system માં classes નું hierarchy આવેલું છે જેનો ઉપયોગ console અને ડિસ્ક ફાઇલો બંને સાથે deal કરવા માટે અલગ અલગ streams ને define કરવા માટે થાય છે. આ classes ને *stream classes* કહેવામાં આવે છે.
- Figure માં console unit સાથે input અને output operations માટે stream class's ના user ની hierarchy બતાવે છે. આ classes header file iostream માં declare કરવામાં આવ્યા છે. આ file તે બધા programs માં આવેલી હોવી જોઈએ કે જે console unit સાથે communicate કરે છે



- Figure માં બતાવ્યા પ્રમાણે, *ios* એ *istream (input stream)* અને *ostream (output stream)* નો બેઝ ક્લાસ છે, જેમાં પછી, *istream* અને *ostream* એ *iostream(input/output stream)* માટે બેઝ ક્લાસ છે. class *ios* ને virtual base class તરીકે declare કરવામાં આવે છે જેથી તેના members ની માત્ર એક copy iostream દ્વારા મળે છે.
- Class *ios* એ formatted અને unformatted I/O operations માટે basic support પૂરો પાડે છે. class *istream* એ formatted & unformatted input માટેની facilities આપે છે જ્યારે class *ostream* એ formatted output માટેની facilities આપે છે.
- class *iostream* બંને ઇનપુટ અને આઉટપુટ streams ને handling કરવાની facilities આપે છે. classes, જેવા કે istream_withassign, ostream_withassign, iostream_withassign.

- આ Table માં classes ની detail આપેલી છે

Class Name	Contents
ios (General I/O Stream Class)	બધા input અને output classes દ્વારા use કરવામાં આવતી basic facilities આવેલી હોય છે.
istream (Input Stream)	Ios ની properties ને Inherit કરીને input functions આપે છે જેવા કે get(),getline() and read(),અને extraction operator >>
ostream (Output Stream)	Ios ની properties ને Inherit કરીને output functions આપે છે જેવા કે write(),અને insertion operator <<
iostream (I/O Stream)	istream અને ostream ની properties ને multiple inheritance દ્વારા Inherits કરે છે અને તેમાં input અને output functions આવેલા હોય છે.
streambuf	physical devices ને buffers દ્વારા interface Provides કરે છે.

3 **Explain Unformatted I/O Operations**

Overloaded Operators >> and <<:

- આપણે અલગ અલગ પ્રકારનાં ડેટાના ઇનપુટ અને આઉટપુટ માટે cin અને cout objects નો ઉપયોગ કરીએ છીએ. operators >> અને << એ basic C++ types ને ઓળખવા માટે overloading કરીને આ possible બન્યું છે.
- >> operator એ istream class માં ઓવરલોડ થયેલ છે અને << ostream class માં ઓવરલોડ થયેલ છે.
- operators નું General form નીચે બતાવ્યા પ્રમાણે છે.
- cin >> variable1 >> variable2>>.....>>variableN;
- variable1,variable2..... .. valid C++ ના variable names છે જે પહેલેથી declare કરવામાં આવ્યા છે. આ statement કમ્પ્યુટર ના execution ને stop કરે છે અને કીબોર્ડમાંથી ઇનપુટ ડેટા શોધશે.

Example નીચે મુજબ છે;

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"This is the example of COUT"<<endl;
    int a;
    cout<<"Enter Value of a:";
    cin>>a;
    cout<<"Entered Value of a is: "<<a;
    return 0;
}
```

put() and get() Functions:

- એક character ના input/output operations ને handle કરવા માટે classes istream અને ostream બે member functions ને get() અને put() ને define કરે છે.
- get() functions, આપણે blank space, tab અને નવી line ના character સહિતના character ને મેળવવા માટે get(char *) prototype નો ઉપયોગ કરી શકીએ છીએ.

- get() functions તેની argument માટે input character ને assigns કરવામાં આવે છે.
- ostream class ના member, Put() ફંક્શનનો ઉપયોગ ટેક્સ્ટની લાઇનમાં character by character આઉટપુટ કરવા માટે કરી શકાય છે

Example નીચે મુજબ છે;

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
        char c1;
        cout<<"First Method: Using get(* char)"<<endl;
        cin.get(c1);
        cout.put(c1);
        return 0;
}
```

getline() and write() Function:

- આપણે line-oriented input/output functions getline() and write() નો ઉપયોગ કરીને ટેક્સ્ટની લાઇનને વધુ સારી રીતે વાંચી અને display કરી શકીએ છીએ.
- જે નવી લાઇન character સાથે end થાય છે તે text ની આખી લાઇન ને getline() function વાંચે છે Function માં cin નો ઉપયોગ કરીને આ ફંક્શનનો ઉપયોગ કરી શકાય છે.
- આ Function call કરવાથી function getline() call થાય છે જે variable ના character input ને લાઇનમાં read કરે છે.
- ગેટલાઇન() ફંક્શન માં cin ફંક્શન કરતાં એક advantage છે કારણ કે cin તે strings ને read કરી શકે છે જેમાં white spaces નથી.પરંતુ getline થી જો white spaces હોય તો પણ તે string ને આખી read કરી શકે છે.
- write function નું format નીચે મુજબ છે;

  cout.write(line,size);
- પહેલી Argument line એ string ના નામને દર્શાવવા માટે use થાય છે અને બીજી argument size, દર્શાવવા માટે character ના number ને display કરે છે.
- જ્યારે null character આવે છે ત્યારે write function એ characters ને display કરવાનું બંધ કરતું નથી.

Example નીચે મુજબ છે;

```
#include<iostream>
#include<conio.h>
 using namespace std;
 int main()
{
 cout<<"Example of write() and getline() Function"<<endl;
 char city[40];
 cout<<"Enter City and State here:"<<endl;
 cin.getline(city,40);
 cout.write(city,18);
```

```
return 0;
}
```

4 Explain Formatted I/O Operations

- ios class માં ઘણા member functions હોય છે જે output ને અલગ અલગ રીતે ફોર્મેટ કરવામાં મદદ કરશે.
- નીચેનું table તેની information બતાવે છે;

Sr. No.	Function	Function Syntax	Syntax Explanation	Function Description
1.	width()	cout.width(w);	અહી w = filed ની width	output ની value ને specify કરવા માટે જરૂરી file size ને display કરવા માટે.
2.	precision()	cout.precision(d);	અહીં d એ decimal point ની જમણી બાજુના digit ની સંખ્યા છે.	floating number ને decimal point 6 digits પછી print કરવા હોય તો આપણે digit ના number દર્શાવીને print કરાવી શકીએ છીએ
3.	fill()	cout.fill(ch);	અહીં ch એ એવા character ને દર્શાવે છે જેનો ઉપયોગ ન વપરાયેલી positions ને ભરવા માટે થાય છે.	field ની ન વપરાયેલી positions માં by default white spaces આવેલી હોય છે, જો કે, આપણે કોઈપણ character દ્વારા ન વપરાયેલી position ને ભરવા માટે fill() ફંક્શનનો ઉપયોગ કરી શકીએ છીએ.
4.	setf()	cout.setf(arg1,arg2);	arg1 એ formatting flags છે જે ios માં define કરવામાં આવ્યા છે. arg2 એ formatting flags છે જે ios માં define કરવામાં આવ્યા છે.	setf() function માં format flags ને specify કરી શકાય છે જે આઉટપુટ display ના ફોર્મને control કરી શકે છે (જેમ કે left-justified, right-justified)
5.	unsetf()	--	--	flags ને clear કરવા માટે વપરાય છે

Example નીચે મુજબ છે;

```
#include<iostream>
```

```
#include<conio.h>
using namespace std;
int main()
{
        cout.width(5);
        cout<< 123 <<endl;
        cout.precision(5);
        cout<< 123.4567890 << "\n";
        cout.fill('*');
        cout.width(10);
        cout<< 31190 << "\n";
        cout.setf(ios::left,ios::adjustfield);
        cout.fill('*');
        cout.width(20);
        cout<<"Programming in C++";
        getch();
        return 0;
}
```

Output :

```
  123
123.453
*****31190
```

5 Explain Formatting with Manipulators

- header file *iomanip* એ મેનિપ્યુલેટર તરીકે ઓળખાતા functions નો set provide કરે છે જેનો ઉપયોગ આઉટપુટ ના ફોર્મેટ્સમાં ફેરફાર કરવા માટે થઈ શકે છે.
- manipulators ને access કરવા માટે, આપણે ફાઇલ પ્રોગ્રામમાં *iomanip* લખવું જરૂરી છે.
- નીચેનું table manipulator અને તેનો અર્થ બતાવે છે;

Manipulator	Meaning
setw(int w)	filed ની width ને w માં Set કરો.
setprecision(int d)	floating point એ precision d પર Set કરો
setfill(int c)	fill એ character c પર Set કરો.
setiosflags(long f)	format flag f Set કરો.
resetiosflags(long f)	F દ્વારા specify કરેલા flag ને Clear કરો
Endl	નવી line ને Insert કરો અને stream ને flush કરો.

Example નીચે મુજબ છે;

```
#include<iostream>
#include<conio.h>
#include<iomanip>
using namespace std;
int main()
  {
        cout<<setw(5)<<setprecision(2)<<123.456<<"\n";
```

```
        cout<<setiosflags(ios::left)<<345.678<<"\n";
        cout<<setw(10)<<setfill('*')<< 123.45<<"\n";
        return 0;
}
```

Output :

```
123.46
345.68
123.45****
```

આપણે આપના manipulator ને પણ design કરી શકીએ છીએ. Manipulator કોઈપણ argument વગર create કરવાનું general form નીચે મુજબ છે.

```
ostream & manipulator (ostream &  output)
{
      .....
      .....
      .....
      return output;
}
```