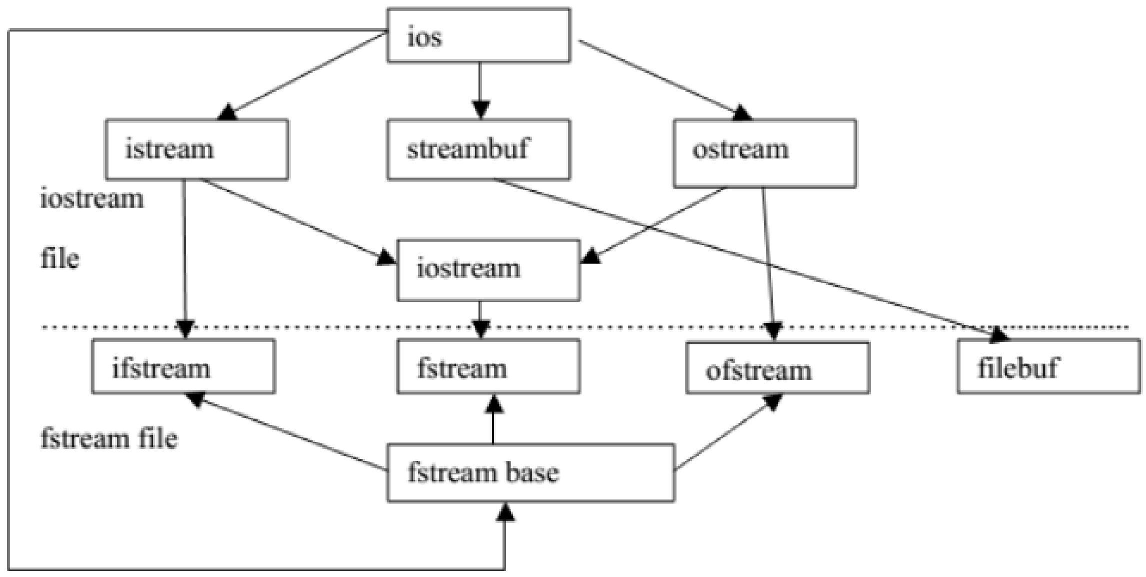
File handling in C++

C++ Stream Classes

What is Stream?

- A stream is an abstraction. It is a sequence of bytes.
- It represents a device on which input and output operations are performed.
- It can be represented as a source or destination of characters of indefinite length.
- It is generally associated to a physical source or destination of characters like a disk file, keyboard or console.
- C++ provides standard **iostream** library to operate with streams.
- The **iostream** is an object-oriented library which provides Input/output functionality using streams.



In the above figure, **ios** is the base class. The **iostream** class is derived from **istream** and **ostream** classes. The **ifstream** and **ofstream** are derived from **istream** and **ostream**, respectively. These classes handle input and output with the disk files.

The **fstream.h** header file contains a declaration of **ifstream, ofstream** and **fstream** classes. The **iostream.h** file contains **istream, ostream** and **iostream** classes and included in the program while doing disk I/O operations.

The **filebuf** class contains input and output operations with files. The **streambuf** class does not organize streams for input and output operations, only derived classes of **streambuf** performs I/O operations. These derived classes arrange a space for keeping input data and for sending output data. The **istream** and **ostream** invokes the **filebuf** functions to perform the insertion or extraction on the streams.

I/O Stream	Meaning	Description
istream	Input Stream	It reads and interprets input.
ostream	Output stream	It can write sequences of characters and represents other kinds of data.
ifstream	Input File Stream	The ifstream class is derived from fstreambase and istream by multiple inheritance. This class accesses the member functions such as get(), getline(), seekg(), tellg() and read(). It provides open() function with the default input mode and allows input operations.
ofstream	Output File Stream	The ofstream class is derived from fstreambase and ostream classes. This class accesses the member functions such as put(), seekp(), write() and tellp(). It provides the member function open() with the default output mode.
fstream	File Stream	The fstream allows input and output operations simultaneous on a filebuf. It invokes the member function istream::getline() to read characters from the file. This class provides the open() function with the default input mode.
fstreambase	File Stream Base	It acts as a base class for fstream, ifstream and ofstream. The open() and close() functions are defined in fstreambase.

Advantages of Stream Classes

- Stream classes have good error handling capabilities.
- These classes work as an abstraction for the user that means the internal operation is encapsulated from the user.
- These classes are buffered.
- These classes have various functions that make reading or writing a sequence of bytes easy for the programmer.

File I/O with stream classes:

- In C++,file handling is done by using C++ streams.
- The classes in c++ for file I/O are ***ifstream*** for input files,***ofstream*** for output files, and ***fstream*** for file used for both input and output operation. These classes are derived classes from ***istream,ostream,***and ***iostream*** respectively and also from fstreambase.

  -The header file for *ifstream,ofstream and fstream* classes is <fstream.h>

  -To create and write disk file we use ofstream class and create object of it.

  e.g. **ofstream outf;**

- The creation and opening file for write operation is done either using its constructor or using ***open()*** member function which had already been defined in ofstream class.

- Creating and opening file for write operation is as:

```
ofstream outf("myfile.txt");       //using constructor of ofstream class.
                    Or
ofstream outf;
outf.open("myfile.txt");        // using open() member function.
```

Writing text into file

- We use the object of *ofstream* to write text to file created as:
  - **outf<<"This is the demonstration of file operation\n";**
  - **outf<<"You can write your text\n";**
  - **outf<<"The text are written to the disk files\n";**

An example for writing to disk file.

```
#include<fstream>
using namespace std;
int main()
{                              //constructor creates file and ready to write
               ofstream outf(“myfile.txt”);

                /* Alternate for above line is
                 ofstream outf;    //using open() member function
               outf.open(“myfile.txt”);
               */

               outf<<”File demonstration program\n”;
                              //writes strings to file myfile.txt
               outf<<”These strings are written to disk\n”;
   }
```

<u>Writing data to file:</u>

```
int x = 20; float f = 2.5;
char ch = 'c'; char* str = " string";
```

<u>Writing to file is done as</u>

```
Outf<<x<<" " <<f<<' '<<ch<<' '<<str;
```

Reading data from file

- To read data from file , we use an object of ***ifstream*** class and file is opened for reading using constructor of ***ifstream*** class or ***open()*** member function as;

```
ifstream fin("test.txt");          //constructor
        or
ifstream fin;
fin.open("test.txt");        // member function open();
```

- Reading data is done as:
- ***fin>>ch>>i>>f>>str;*** which is similar as reading data from keyboard by ***cin*** object.

- **String with embedded blanks:**

  -Require delimiter line \n for each string with embedded blank and read/write operation is easy.

Reading text from file

- To read text from file we use ***ifstream*** class and file is opened for read operation using **constructor or open() member function**.

```
e.g.  :  ifstream  infile("myfile.txt");  //using constructor
                             or
                   ifstream infile;
                   infile.open("myfile.txt");
```

// Reading from file myfile.txt:

```
while(infile)              // or while(!infile.eof())   until end of file
{
  infile.getline(buffer,maxlength);   //buffer to be defined as char
                                      //string of length maxlength
  cout<<buffer;           // for display to screen (optional)
}
```

An example: Reading text from file

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
   const int LEN = 100;
   char  text[LEN];      //for buffer
   ifstream infile("class.cpp");
            while(infile)  //until end of file Alternate is
//while(!infile.eof())
   {
      infile.getline(text,LEN);    // read a line of text
      cout<<endl<<text;            //display line of text
   }
}
```

Character I/O in file[get() and put() function]

- put() and get() functions are members of ostream and istream classes.
- These functions are inherited to ofstream and ifstream objects in class heirarchy.
- put() is used to write a single character in file.
- Similarly get() function is used for reading a character from file.

Example: writhing character

```
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;

int main()
{
        char*str="This is a string written to file one char at a time";
        ofstream fout;
        fout.open("myfile.txt");
        for(int i=0;i<strlen(str);i++)
        {
                fout.put(str[i]);
        }
        cout<<"File write completed";
}
```

Example: getting character

```
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;

int main()
{
		char ch;
		ifstream infile;
		infile.open("myfile.txt");
		while(infile)
		{
				infile.get(ch);
				cout<<ch;
		}
}
```

Writing and reading of user Input

- We can also write user-input (values of variables in a program input from keyboard) by ofstream object.
- Also can read those values in user variable from file by using objects of **ifstream** respectively same as done above .
- Look at the simple program example in next.

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
   //creates and open for writing
   ofstream  fout("test.txt");
  cout<<"Enter the Name:";
    char name[20];
    cin>>name;        //reading from
keyboard
    fout<<name<<endl;  //writing to
the file "test"
    cout<<"Enter telephohe:";
    int tel;
    cin>>tel;    //reading from keyboard
    fout<<tel;   //writing to file "test"
    fout.close();  //Closes the file "test"
```

```
//opens the file test for read

   ifstream fin("test.txt");
   char n[20];
    int t;
    fin>>n;     //reading from file
    fin>>t;      //reading from file
    cout<<endl<<"The name is: "<<n;
    cout<<endl<<"Telephone no: " <<t;
    fin.close();
 }
```

Opening file in different mode

- In above example we have used the **ofstream** and **ifstream** constructors or **open()** member function using only one argument i.e. filename e.g. “test.txt” etc.
- However this can be done by using two argument : one is filename and another is filemode.

The Syntax for using file mode is:

Stream-object.open(“filename”,filemode);

The File Mode Parameter

- The second argument **filemode** is the parameter which is used for what purpose the file is opened.
- If we haven’t used any **filemode** argument and only filename with **open()** function, the default mode is as:

- **ios::in** for **ifstream** functions means open for reading only.
- i.e. **fin.open(“test.txt”);** is equivalent to **fin.open(“test”,ios::in);** as default
- **ios::out** for **ofstream** functions means open for writing only.

fout.open(“test”); is same as **fout.open(“test”,ios::out);** as default.

NOTE:

Class fstream inherits all features of ifstream and ofstream so we can use fstream object for both input/output operation in file.

When fstream class is used , we should mention the second parameter <filemode> with open() member function.

The File Mode Parameter

- The file mode parameter can take one or more such predefined
  constants in **ios** class. The following are such file mode parameters.

Parameters	**Meanings**
ios::app	Append to end of file
ios::ate	Go to end-of-file on opening
ios::binary	Binary file
ios::in	open file for reading only
ios::nocreate	Opens fails if the file does not exists
ios::noreplace	Open files if the file already exists
ios::out	Open file for writing only
ios::trunc	Delete the contents of files if it exits

The File Mode Parameter

- Opening file in **ios::out** mode also opens in the **ios::trunc** mode default
- **ios::app** and **ios::ate** takes to the end-of-file when opening but only difference is that **ios::app** allows to add data only end-of-file but **ios::ate** allows us to add or modify data at anywhere in the file. In both case file is created if it does not exists.
- Creating a stream **ofstream** default implies output(write) mode and **ifstream** implies input(read), but **fstream** stream does not provide default parameter so we must provide the mode parameter with **fstream.**
- **The mode can combine two or more parameters using bitwise OR operator (|)**

  e.g. **fout.open("test",ios::app|ios::out);**

File Pointers

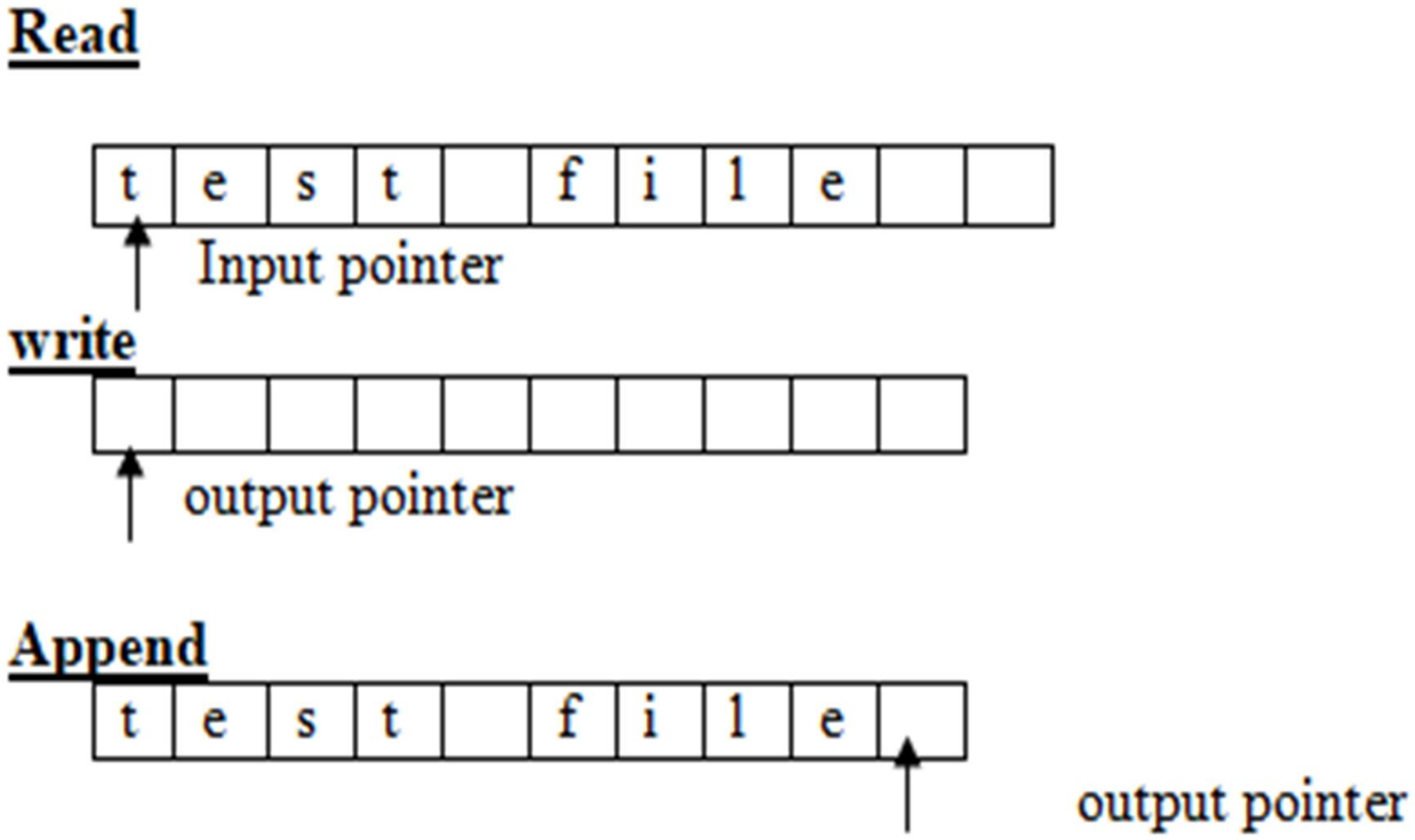
- The file management system associates two types of pointers with each file.
- get pointer (input pointer)
- put pointer (output pointer)
- These pointers facilitate the movement across the file while reading and writing.
- The get pointer specifies a location from where current read operation initiated.
- The put pointer specifies a location from where current write operation initiated.
- The file pointer is set to a suitable location initially depending upon the mode which it is opened.

• **Read-only Mode**: When a file is opened in read-only mode, the input (get) pointer is initialized to the beginning of the file.

• **Write-only: mode**: In this mode, existing contents are deleted if file exists and put pointer is set to beginning of the file.

• **Append mode**: In this mode, existing contents are unchanged and put pointer is set to the end of file so writing can be done from end of file.

I/O Pointer in different Mode



Functions manipulating file pointers

- C++ I/O system supports 4 functions for setting a file to any desired position inside the file. The functions are

Function	member of class	Action
seekg()	ifstream	moves get file pointer to a specific location
seekp()	ofstream	moves put file pointer to a specific location
tellg()	ifstream	Return the current position of the get ptr
tellp()	ofstream	Return the current position of the put ptr

- These all four functions are available in f**stream** class by inheritance. The two seek() functions have following prototypes.

istream & seekg (long offset, seek_dir origin =ios::beg);

ostream & seekp (long offset, seek_dir origin=ios::beg);

- Both functions set file ptr to a certain offset relative to specified origin. The origin is relative point for offset measurement. The default value for origin is ios::beg.

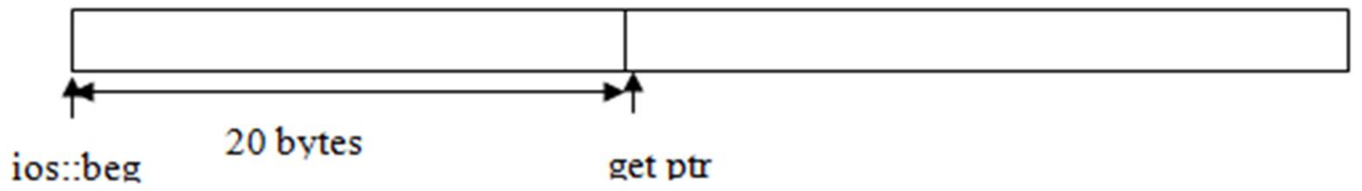
(seek_dir) an enumeration declaration given in ios class as

orgin value	seek from
ios::beg	seek from beginning of file
ios::cur	seek from current location
ios::end	seek from end of file

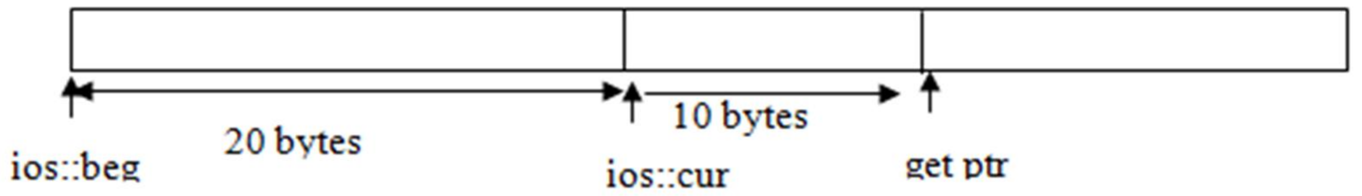
Seek function offset position

e.g. `ifstream infile;`

`infile.seekg(20,ios::beg);` or `infile.seekg(20);` // default ios::beg move file ptr to 20th byte in the file. The reading start from 21st item [byte start from 0] with file.



Then after, `infile.seekg(10,ios::cur);` moves get pointer 10 bytes further from current position.



Similarly:

ofstream outfile;

outfile.seekp(20,ios::beg); // out file. seek p (20);

moves file put pointer to 20th byte and if write operation is initiated, start writing from 21st item.

Example

<u>Consider below example:</u>

```
ofsteam outfile("student",ios::app);
int size=outfile.tellp();
```

- Returns the size of file in byte to variable size since ios::app takes file put ptr at end of file. The function tellp() returns the takes file put ptr at end of file. The function tellp() returns the current position of put ptr.

<u>Equivalently</u>:

```
ifstream infile("student");
infile.seekg(0,ios::end);
int size=infile.tellg() ;
```

- This returns the current file pointer position which is at end of file so we get he size of fife "student".

Pointer Call with seek()

Some of pointer offset calls and their actions:

Assume ofstream object: **ofstream fout;**

Seek	**Action**
fout.seekg(0,ios::beg)	Go to beginning of the file
fout.seekg(0,ios::cur)	Stay at current location
fout.seekg(0,ios::end)	Go to the end of file
fout.seekg(n,ios::beg)	move to (n+1) byte from beginning of file.
fout.seekg (n,ios::cur)	move forword by n bytes from currrent position
fout.seekg(-n,ios:: cur)	move backward by n bytes from currnt position
fout.seekp(n,ios:: beg)	move write pointer (n+1) byte location
fout.seekp(-n,ios:: cur)	move write ptr n bytes backwards.

File I/O with fstream class

- fstream class supports simultaneous input/output operations using same object.
- It inherits function from istream and ostream class through iostream.

```
#include<iostream>      // Example for writing data to file
#include<fstream>
#include<process.h>
using namespace std;
int main()
{
                fstream fout;
                int i, count, percentage;
                char name[20];
                cout<<"Enter no of student:";
                cin>>count;
                fout.open("student.in",ios::out);
                if(fout.fail())                 // if operation failed.
                {
                                cout<<"Error: student.in Create/open fail";
                                exit(1);
                }
                fout<<count<<' ';
                for (i=0;i<count;i++)
                {
                                cout<<"Name:";
                                cin>>name;
                                cout<<"Percentage:";
                                cin>>percentage;
                                fout<<name<<' '<<percentage<<' ';
                }
                fout.close();
}
```

```
#include<iostream>
#include<fstream>                          //Assume file student.in
#include<conio.h>                          //is created with
#include<process.h>
using namespace std;        //1. no of student (count)
int main()                                                //2. for n
students
{
            fstream infile;            // input file name
            fstream outfile;           // output file percentage sane
            int i, count, percentage;
            char name[20];
                                       //open for read mode
            infile.open("student.in",ios::in);
            if(infile.fail())// if operation failed.
            {
                        cout<<"Error: student.in open fail";
                        exit(1);
            }
                        //open next file for write
            outfile.open("student.out",ios::out);
            if(outfile.fail())
            {           cout<<"Error:......"; exit(1);
            }

            infile>>count;             // no of student
            outfile<<"      student Information processing" <<endl;
            outfile<<"      -----------------------------" <<endl;
```

Continue…

```
        for(i=0; i<count; i++)
        {                               // Read data percentage from input file
                infile>>name;
                infile>>percentage;
                                // write in output file.
                outfile<<"Name:"<<name<<endl;
                outfile<<"precentage:"<<percentage<<endl;
                outfile<<"passed in:";
                if(percentage>=75)
                                outfile<<"first Division/distinction";
                else if(percentage>=45)
                                outfile<<" Second Div";
                else if(percentage>=35)
                                outfile<<"Passed";
                else
                outfile<<"Failed";
                outfile<<endl;
                outfile<<"....................."<<endl;
        }
                                                // close files;
                infile.close();
                outfile.close();
}
```

Character I/O in file

<u>The put () and get () function</u>:

- The function get() is a member function of the file stream class fstream, and used to read a single character from file.

- The function put() is member function of fstream class and used to write a single character into file.

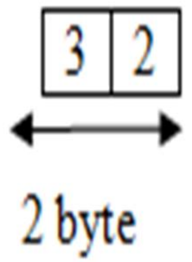
```
// An example of character I/O in file
#include<fstream>
#include<iostream>
#include<stdlib.h>
using namespace std;
int main()
{
            char c, string[100];
            fstream file("student.txt",ios::in|ios::out);
            if(file.fail())
            {
                        cout<<"File Open Error:";
                        exit(0);
            }
            cout<<"Enter string:";
            cin.getline(string,100);
    for (int i=0; string[i]!='\0'; i++)
    file.put(string[i]);
    file.seekg(0);          // seek to the beging
    cout<<"output string:";
    while(file)
    {
                        file.get(c);
                        cout<<c;
    }
}
```

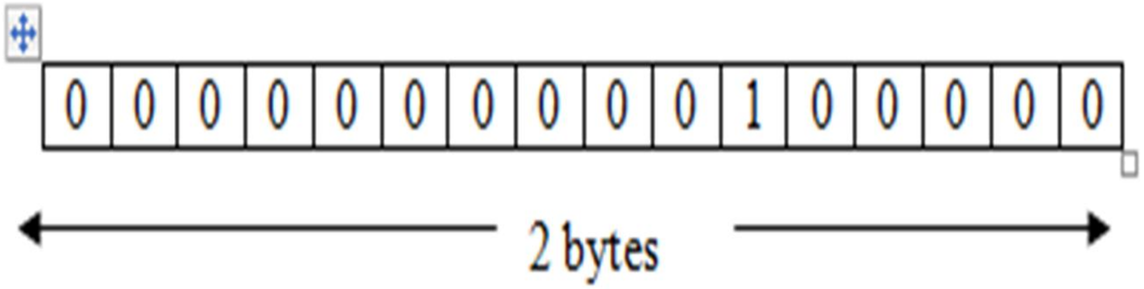
The write () and read () function

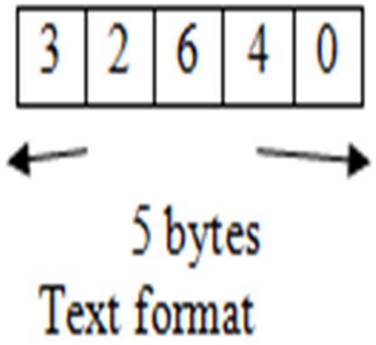
- The **write()** function is a member of stream class fstream and used to write data in file as binary format.
- The **read()** function is used to read data (binary form) from a file.
- The data representation in binary format in file is same as in system. The no of byte required to store data in text form is proportional to its magnitude but in binary form, the size is fixed.

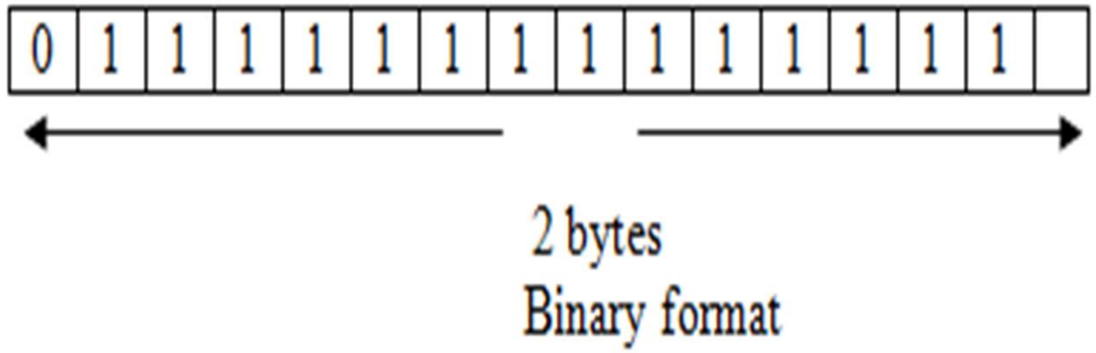
Size of data in file

e.g.









read() and write() function

- The prototype for read () & write () functions are as:.

```
infile.read((char*)&variable, sizeof(variable));
outfile.write((char*)&variable, sizeof(variable));
```

- The first parameter is a pointer to a memory location at which the data is to be retrieved [read()] or to be written [write()] function.

- The second parameter indicates the number of bytes to be transferred.

<u>Example</u> :writing variable in to files

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
		int number1=530;
		float number2=100.50;
				//  open file in read binary mode, read integer and class
		ofstream ofile("number.bin",ios::binary);
		ofile.write((char*)&number1, sizeof(number1) );
		ofile.write((char*)&number2, sizeof(float) );
		ofile.close();
		//  open file in read binary mode, read integer & close
		ifstream ifile ("number.bin",ios::binary);
		ifile.read((char*)&number1,sizeof(number1));
		ifile.read((char*)&number2,sizeof(number2));
		cout<<number1<<" "<<number2<<endl;
		ifile.close();
}
```

<u>**Object I/O in file**</u>

- C++ is Object-oriented language so we need objects to be written in file and read from file .
- Following examples show the I/O operations .

- <u>**Writing object to disk file:**</u>
  - Generally binary mode is used which writes object in disk in bit configurations
  - Follow the example in next.

```
//example file10_1
#include<fstream>
#include<iostream>
using namespace std;
class emp
{
        char empname[20];
        int eno;
        float sal;
        public:
        void getdata()
        {
                cout<<"Enter Name:"; cin>>empname;
                cout<<"Enter Emp No:"; cin>>eno;
                cout<<"Enter salary:"; cin>>sal;
        }
};

int main()
{
        emp em;
        cout<<"Enter the detail of employee"<<endl;
        em.getdata();
        ofstream fout("emp.dat");
        fout.write((char*)&em,sizeof(em));
        cout<<"Object written to file";
}
```

```
//READING FROM FILE file10_2
#include<fstream>
#include<iostream>
using namespace std;
class emp
{
        char empname[20];
        int eno;
        float sal;
        public:

        void showdata()
        {
                cout<<"\nName:"<<empname<<endl;
                cout<<"Emp NO:"<<eno<<endl;
                cout<<"Salary:"<<sal<<endl;
        }
};

int main()
{
        emp em;
        ifstream fin("emp.dat");
        fin.read((char*)&em,sizeof(em));
        cout<<"Object detail from file";
        em.showdata();
}
```

```
//Writing and reading objects:
//file10_3.cpp
#include<iostream>
#include<fstream>
#include<iomanip>
using namespace std;
class student
{
		char name[20];
		int roll;
		char add[20];
		public:
		void readdata()
		{
				cout<<"Enter name:";cin>>name;
				cout<<"Enter Roll. no.:";cin>>roll;
				cout<<"Enter address:";cin>>add;
		}
		void showdata()
		{
				cout<<setw(10)<<roll<<setiosflags(ios::left)<<setw(10)
				<<name<<setiosflags(ios::left)<<setw(10)<<add<<endl;
		}
};
```

```
int main()
{

        student s[5],r[5];
        fstream file;
        file.open("record.dat", ios::in|ios::out);
        cout<<"enter detail for 5 students:";
        for(int i=0;i<5;i++)
        {
                s[i].readdata();
                file.write((char*)&s[i],sizeof(s[i]));
        }
        file.seekg(0);//move pointer begining.
        cout<<"Output from file"<<endl;
        cout<<setiosflags(ios::left)<<setw(10)<<"RollNo"
        <<setiosflags(ios::left)<<setw(10)<<"Name"
        <<setiosflags(ios::left)<<setw(10)<<"Address"<<endl;
        for(int i=0;i<5;i++)
        {
                file.read((char*)&s[i],sizeof(s[i]));
                s[i].showdata();
        }
        file.close();
}
```