→ Reading and writing to the same file :-

If we want to write to file via object we have two option. First make object to connect to that file for writing using ofstream class. Then, pass an string by this object in which you have written something, which you want to print/write in that file.

Second, make an object of ofstream class and direct pass what you want to write in that file via that object, like

out<<"what you want to write";

here, out is object of that ofstream class.

We can also print new line to file by :

out<<endl;

If you are passing string to the object to write in that file and you are taking input from user via cin in that string, always we getline() function for that, as if you use

contd

Cin directly string will only store first letter not whole line.

Also note that one file can only have one output object, in you used second one everything written via first object will be erased, also if you have closed first object and then made second one. so, never make two objects of ofstream for same file.

To close an object write

{{object name}}.close();

~~After running this~~ After writing this line of code, you can't use that object to write to that file.

Reading a file is quite easy, you have to make an object of class ifstream and use getline() function to get the one line of that file in string and then print that string using cout.

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

int main()
{
    string st;
    string st2;
    cout<<"write what you want to save in file"
        <<endl;
    getline(cin,st);
    ofstream out("file.txt");
    out<<st;
    out<<endl;
    out<<"what happen bro";
    out<<endl;
    ifstream in("file.txt");
    getline(in,st2);
    cout<<st2<<endl;
    getline(in,st2);
    cout<<st2<<endl;
    out<<"gt is end";
    out<<endl;
    out.close;
    out<<"gt is not indeed thing";
    get(in,st2);
    cout<<st2;
    return 0;
```

Also note that you can't write two programms to write in one same file, by doing so, it will erase what is written by previous programm.

→ Using open() and eof() :-

open() is used to open a file. You have to make in object then by that object invoke open() function to open the file.

like

```
ofstream x;
x.open("file.txt");
x<<st;
```

eof() is used to find end of file it will return 0 untill end of file comes. So, this function is very useful in case we want to print every line using getline() untile end of file comes.

```
ifstream X;
X.open("file.txt");
string st2;
while(X.eof()==0)
{
    getline(X, st2);
    cout<< st2 <<endl;
}
```

→ **Template :-**

Template are paramatrize classes, in which we pass data type as arguments.

Template are used or helpful in case we need to make different classes for different datatypes but we wont to do so as, in this way we will repeat same thing again and again.

To avoid so we define template, by which make one class but as we pass datatype to that it will be differ

Syntax of templat:-

```
#include<iostream>
using namespace std;
Template <class t>
class demo{
    // T num1;
       T num2;
       T sum=0;
    public:
        demo(T a, T b):num1(a),num2(b)
        {
            sum=num1+num2;
            cout<<"sum is"<<sum<<endl;
        }
};
int main{
    demo<int> obj(3,5);
    demo<float> obj2(3.5,7.9);
    return 0;
}
```

Q WAP to Calculate dot product of two vectors.

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

Template <class T>
class vector{

    T *arr;
    int size;
    T* sum;
  public:
    vector(int a):size(a){
      arr = new int[size];
    }
  void getin(void)
  {
    for(int i=0;i<size;i++)
      {
        cout<<"input componet of vector"
                                <<endl;
        cin>> arr[i];
      }
  }

T dotproduct(vector &V)
{
  for(int i=0;i<size;i++)
  {
```

```
        sum+= this->arr[i] * v.arr[i];
    }
    return sum;
  }
};
int main(){

    vector<float> obj1(3);
    obj1.getin();
    vector<float> obj2(3);
    obj2.getin();
   float a= obj1.dotproduct(obj2);
   cout<< a<<endl;
   return 0;
}
```

→ Templates with multiple parameters:-

We can make a class template with more than one prameters.

Syntax:-

```
Template <class T1, class T2>
  class demo
    {
       //load;
    };
```

now, we can pass two different datatypes with class.
We can also pass classes ~~with this~~ instead of datatype.

→ Template with default parameters:

Template with default parameter works same as function with default parameter.

Syntax:-

```
Template <class T1 = int, class T2 = float>
class Demo{
    T1 x;
    T2 y;
public:
void sum( int a, float b)
{    x=a ; y=b
     cout<< x+y <<endl;
}
};

int main
{  Demo<> obj;
   obj.sum(3, 5.7);
   return 0;
}
```

→ function template :-

Function template means function with template.

Syntax :-

```
Template <class T>
void Sum ( T a , T b)
{
    T x = a;
    T y = b;
    cout << x+y << endl;
}
int main()
{
    Sum (5, 5.7);
    return 0;
}
```

For any non-member function to be function template, that function should be paramatize and what value we pass to that function will automatically decided by type of passed

values:

How to define a member function of class template outside the class -

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

Template <class T>
class Demo{

    T data;

    Public:
      Void show(T a);
};
class Template <class T>
Void Demo<T> :: show(T a)
{
        data = a;
    Cout<< " data is: " << data <<endl;
}

int main()
{
   Demo <char> obj;
   obj. show('c');
   return 0;
}
```

NOTE :- What if you have two paramatsize function or two functions with same name. One is normal function and one of is function template.

If you invoke one of those function which will be invoked.

Here that function will be invoked which is exact match, if there is no exact match then function template will be invoked as it will become match due to its ability.

ex -

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

Template <class T>
Class Demo
{ int a;
  T b;
  Public:
    void func1(int num);
    void func2(T num2);
};
```

```
void Demo :: func(int num)
{
        a=num;
    cout<<" Normal function"<<a<<endl;
}

template <class T>
void Demo<T> :: func(T num2)
{
        b = num2;
    cout<<"function template"<<b<<endl;
}

int main()
{
    Demo <int> obj;
    obj.func(5);

    Demo<float> obj2;
    obj2.func(5.7);

    return 0;
}
```

Output:-

Normal functions
function template 5.7