INDEX :

File IO :

```cpp
#include<iostream>

#include<fstream>

using namespace std;

/*

The useful classes for working with files in C++ are :-

1. fstreambase

2. ifstream --> Derived from fstreambase

3. ofstream --> Derived from fstreambase


In order to work with files in C++, you will have to open it.

There are 2 way to open a file :

1. Using the Constructor

2. Using the member function open() of the class

*/


int main(){

    string st = "Aman Verma";

    string st1;

    // Opening files using constructor and writing it.

    ofstream out("ZPA~SampleFiles.txt"); // Write operation

    out<<st;

    out.close(); // This will disconnect the link btw file and this program.


    // Opening files using constructor and reading it.

    ifstream in("ZPA~SampleFiles.txt"); // Read operation

    // in>>st1; --> This will take only first word of the line.

    getline(in, st1); // --> This will take only first line from the file.

    getline(in, st1); // This will take next line from the file.

    cout<<st1;

    in.close();

    // We can use any word instead of 'out' and 'in' that use in this program.


return 0;

}
```

Read And Write Operation :

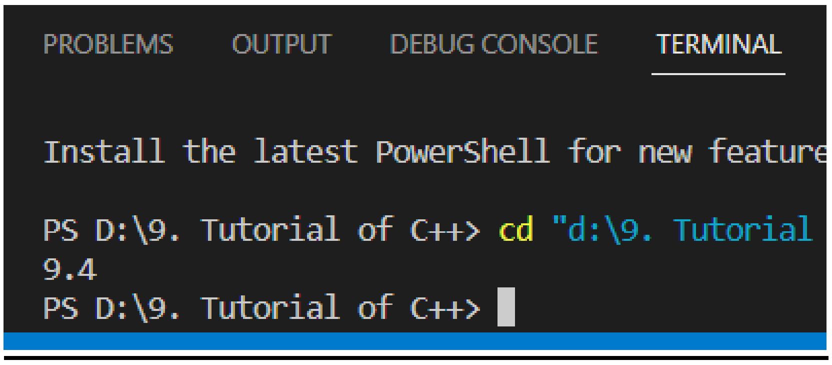
```cpp
#include<iostream>
#include<fstream>
using namespace std;

int main(){
    ofstream outIt;
    outIt.open("ZPA~SampleFiles.txt");
    outIt<<"Hello World..!\n";
    outIt<<"I'm here\n";
    outIt<<"Let me be at there with you.\n";
    outIt.close();

    ifstream inIt;
    string st;
    inIt.open("ZPA~SampleFiles.txt");
    while(inIt.eof() == 0){
        getline(inIt, st);
        cout<<st<<endl;
    }
    inIt.close();

return 0;
}
```

Templates :

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

class Vector{
    public:
    int *arr;
    int size;
    Vector(int m){
        size = m;
        arr = new int[size];
    }
    int dotProduct(Vector &v){
        int d = 0;
        for(int i=0; i<size; i++){
        d += this->arr[i] * v.arr[i];
        }
        return d;
    }
};

int main(){
    Vector v1(3);
    v1.arr[0] = 4;
    v1.arr[1] = 3;
    v1.arr[2] = 1;

    Vector v2(3);
    v2.arr[0] = 0;
    v2.arr[1] = 1;
    v2.arr[2] = 0;

    int a = v1.dotProduct(v2);
    cout<<a<<endl;
```

```
return 0;
}
*/


#include<iostream>
using namespace std;

template <class T>
class Vector{
    public:
    T *arr;
    int size;
    Vector(int m){
        size = m;
        arr = new T[size];
    }
    T dotProduct(Vector &v){
        T d = 0;
        for(int i=0; i<size; i++){
        d += this->arr[i] * v.arr[i];
        }
        return d;
    }
};

int main(){
    // Vector <int>v1(3);
    // v1.arr[0] = 4;
    // v1.arr[1] = 3;
    // v1.arr[2] = 1;

    // Vector <int>v2(3);
    // v2.arr[0] = 0;
    // v2.arr[1] = 1;
    // v2.arr[2] = 0;


    // int a = v1.dotProduct(v2);

    // cout<<a<<endl;


    Vector <float>v1(3);

    v1.arr[0] = 4.6;

    v1.arr[1] = 3.2;

    v1.arr[2] = 1;


    Vector <float>v2(3);

    v2.arr[0] = 1;

    v2.arr[1] = 1.5;

    v2.arr[2] = 0;


    float a = v1.dotProduct(v2);

    cout<<a<<endl;



return 0;

}
```

<u>Output :-</u>

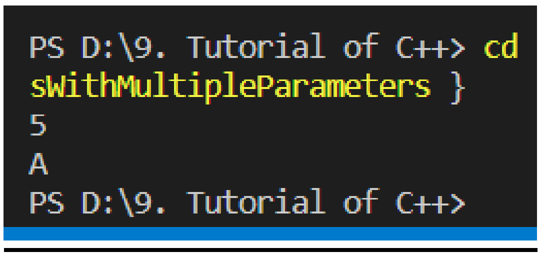


```
Install the latest PowerShell for new feature

PS D:\9. Tutorial of C++> cd "d:\9. Tutorial
9.4
PS D:\9. Tutorial of C++> 
```

Templates With Multiple Parameters :

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

template<class T1, class T2> // class templates with multiple parameters.
class Myclass{
    public:
    T1 data1;
    T2 data2;
    Myclass(T1 A, T2 a){
        data1 = A;
        data2 = a;
    }
    void display(){
        cout<<this->data1<<endl<<this->data2;
    }
};

int main(){
    Myclass<int, char>obj(5,'A');
    obj.display();

return 0;
}
```

Output :-

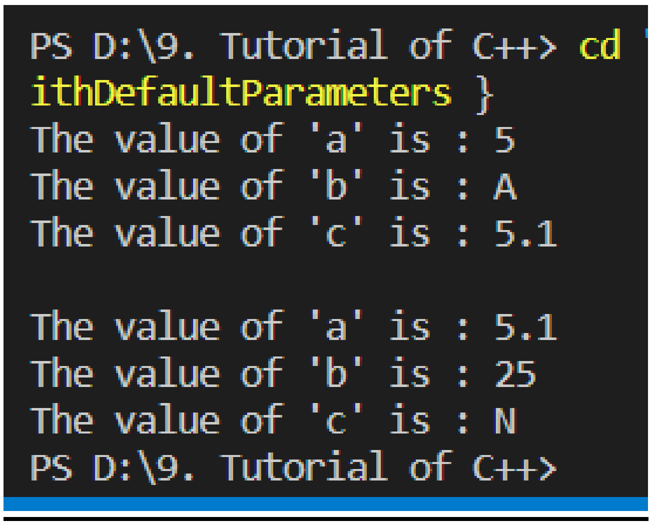


```
PS D:\9. Tutorial of C++> cd
sWithMultipleParameters }
5
A
PS D:\9. Tutorial of C++>
```

Templates With Default Parameters :

```
#include<iostream>

using namespace std;


template <class T1 = int, class T2 = char, class T3 = float> // Default setting of data.

class ItsMe{

    T1 a;

    T2 b;

    T3 c;

    public:

    ItsMe(T1 a, T2 b, T3 c){

        this->a = a;

        this->b = b;

        this->c = c;

    }

    void display(){

        cout<<"The value of 'a' is : "<<a<<endl;

        cout<<"The value of 'b' is : "<<b<<endl;

        cout<<"The value of 'c' is : "<<c<<endl;

    }

};


int main(){

    ItsMe <>me1(5, 'A', 5.1);

    me1.display();

    cout<<endl;

    ItsMe <float, int , char>me2(5.1, 25, 'N'); // We can make changes in defalut data also.

    me2.display();


return 0;

}
```

Output :-

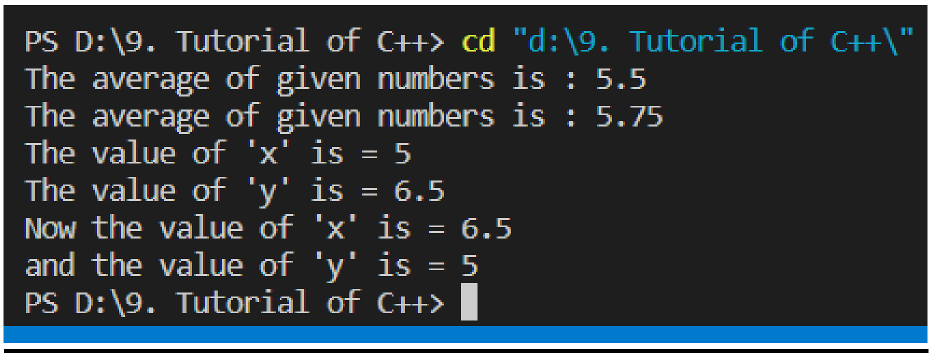


```
PS D:\9. Tutorial of C++> cd
ithDefaultParameters }
The value of 'a' is : 5
The value of 'b' is : A
The value of 'c' is : 5.1

The value of 'a' is : 5.1
The value of 'b' is : 25
The value of 'c' is : N
PS D:\9. Tutorial of C++>
```

Function Templates :

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

template<class T1, class T2>
float funcAverage(T1 a, T2 b){
    float avg = (a + b)/2.0;
    return avg;
}

template <class T>
void swapNumber(T &a, T &b){
    a = a + b;
    b = a - b;
    a = a - b;
}

int main(){
    float a, b;
    a = funcAverage(5, 6);
    cout<<"The average of given numbers is : "<<a<<endl;
    b = funcAverage(5, 6.5);
    cout<<"The average of given numbers is : "<<b<<endl;

    float x = 5, y = 6.5;
    cout<<"The value of 'x' is = "<<x<<endl;
    cout<<"The value of 'y' is = "<<y<<endl;
    swapNumber(x, y);
    cout<<"Now the value of 'x' is = "<<x<<endl;
    cout<<"and the value of 'y' is = "<<y<<endl;

return 0;
}
```

Output :-

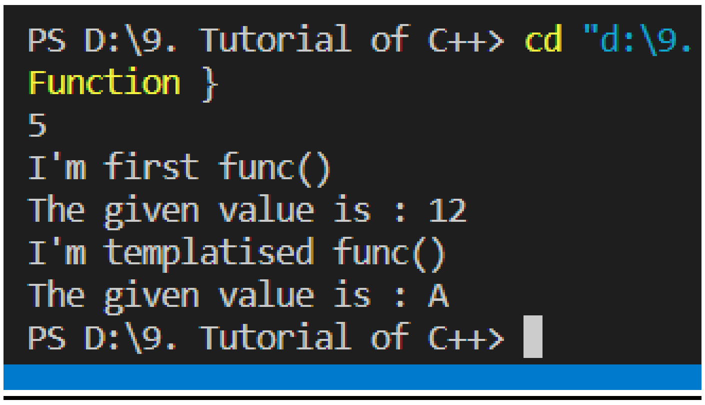


```
PS D:\9. Tutorial of C++> cd "d:\9. Tutorial of C++\"
The average of given numbers is : 5.5
The average of given numbers is : 5.75
The value of 'x' is = 5
The value of 'y' is = 6.5
Now the value of 'x' is = 6.5
and the value of 'y' is = 5
PS D:\9. Tutorial of C++>
```

Overloading Template Function :

```
#include<iostream>

using namespace std;

template <class T>

class ItsMe{

    T data;

    public:

    ItsMe(T a) : data(a){}

    void display();

};


template <class T> // --> This is how we write function out of class.

void ItsMe<T> :: display(){

    cout<<data<<endl;

}


void func(int a){

    cout<<"I'm first func()"<<endl;

    cout<<"The given value is : "<<a<<endl;

}


template <class T>

void func(T a){  // --> Overloding template function.

    cout<<"I'm templatised func()"<<endl;

    cout<<"The given value is : "<<a<<endl;

}

int main(){

    ItsMe <int>me1(5);

    me1.display();


    func(12); // Exact match takes the highest priority.

    func('A');


return 0;

}
```

Output :-



```
PS D:\9. Tutorial of C++> cd "d:\9.
Function }
5
I'm first func()
The given value is : 12
I'm templatised func()
The given value is : A
PS D:\9. Tutorial of C++>
```