READ()
WRITE()

```
#include<Windows.h>
#include<iostream>
using namespace std;
int main()
{
//Local Variable
HANDLE hFile;
BOOL bFile;
char chBuffer[] = "Sample for Readfile and WriteFile";
DWORD dwNoBytesToWrite = strlen(chBuffer);//size of written data
LPDWORD lpNoByteWritten = 0;//Set Write pointer to 0
DWORD dwNoBytesToRead = strlen(chBuffer);//size of Buffer
LPDWORD lpNoByteRead = 0;//Set Read Pointer to 0

   //STEP-1 CreateFile Function
hFile = CreateFile(
L"C:\\SampleFolder\\SampleFile.txt",//File Path and Name
GENERIC_READ | GENERIC_WRITE,      //File Permission
FILE_SHARE_READ | FILE_SHARE_WRITE,  //File Sharing Mode
NULL,                              //Security Attribute
CREATE_NEW | OPEN_EXISTING,   //if  File exist then open existing File.
FILE_ATTRIBUTE_NORMAL,          //Flag for File
NULL);                         //Default File Template
```

```
◦ //STEP-2- Check file has successfully created or not
◦ if (INVALID_HANDLE_VALUE == hFile)
◦ {
◦ cout << "Create File Failed" << endl;
◦ cout << "Error No - " << GetLastError() << endl;
◦ }
◦ cout << "Create File Success " << endl;

◦ //STEP-3 Write File Function
◦ bFile = WriteFile(
◦ hFile,
◦ chBuffer,
◦ dwNoBytesToWrite,
◦ lpNoByteWritten,
◦ NULL);
◦ if (FALSE == bFile)
◦ {
◦ cout << "WriteFile Failed & Error No- " << endl;
◦ cout << "Error No- " << GetLastError() << endl;
◦ }
◦ cout << "WriteFile Success" << endl;
◦ //STEP-4 Read File Function
◦ bFile = ReadFile(
◦ hFile,
◦ chBuffer,
◦ dwNoBytesToRead,
◦ lpNoByteRead,
◦ NULL);
◦ if (FALSE == bFile)
◦ {
◦ cout << "ReadFile Failed "<< endl;
◦ cout << "Error No- " << GetLastError() << endl;
◦ }
◦ cout << "ReadFile Success" << endl;

◦ //STEP-5 Read data from Buffer
◦ cout << "Data Reading from Buffer - " << chBuffer << endl;

◦ //STEP-6 CloseHandle
◦ CloseHandle(hFile);

◦ system("PAUSE");
◦ return 0;
◦ }
```

API’s Used

CreateFile()

WriteFile()

ReadFile()

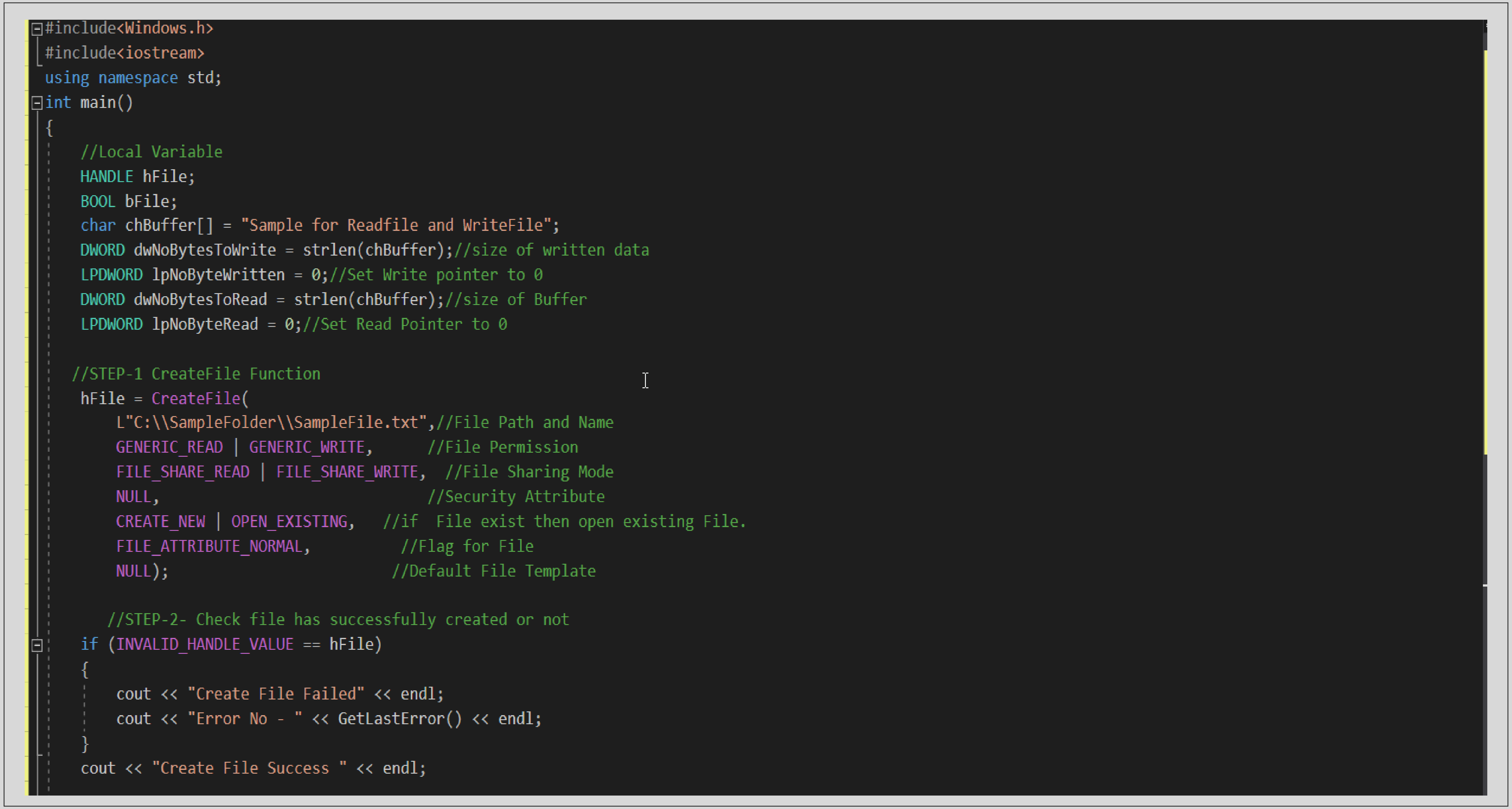
The Handle is first created in CreateFIle()

The Handle is passed to WriteFile() for writing into the file

Finally the handle is then passed to ReadFile()

After that the handle is close using Closehandle function

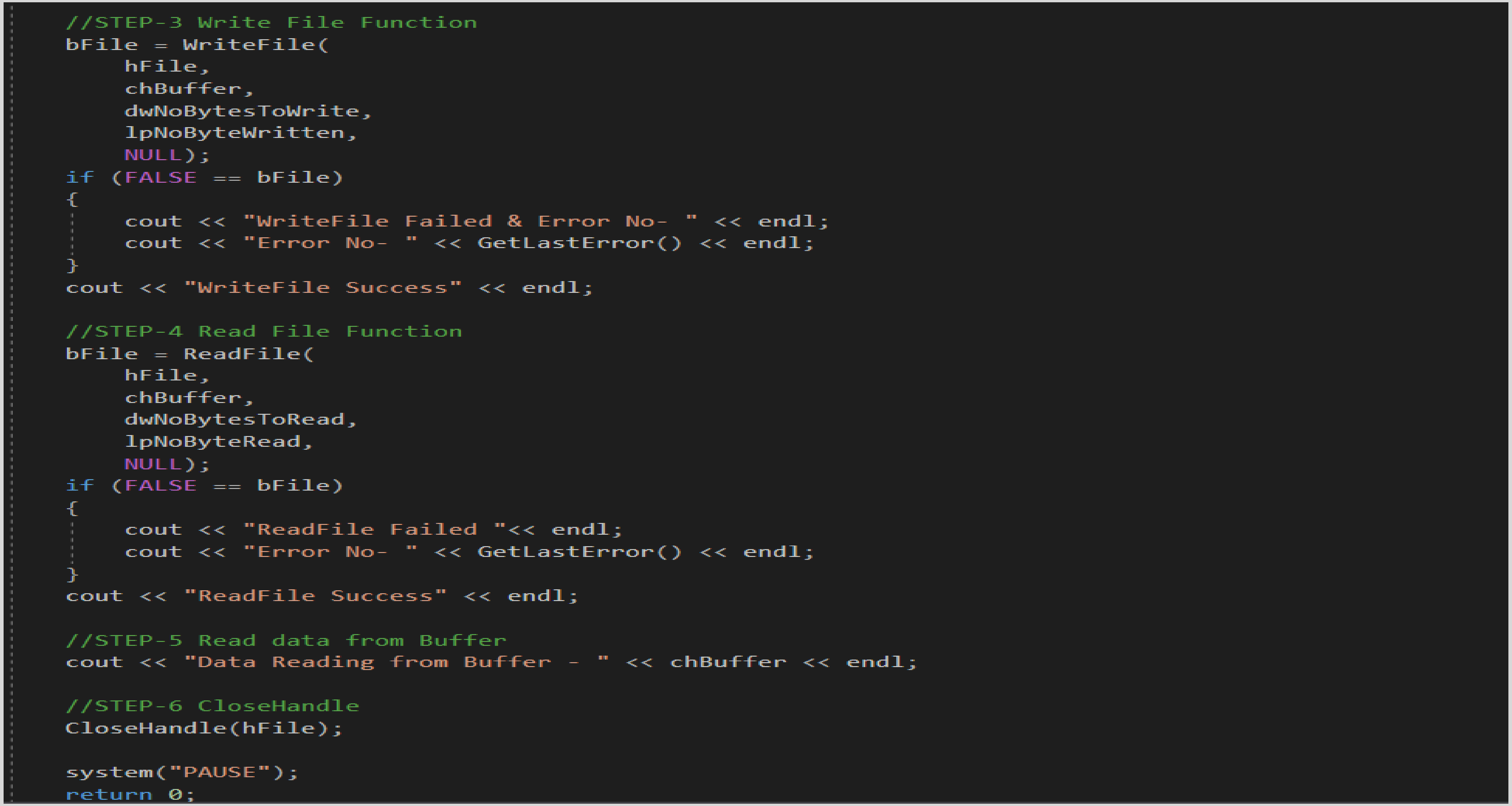
Lpcvoid : can point to any data type such as char,int and float



```
#include<Windows.h>
#include<iostream>
using namespace std;
int main()
{
    //Local Variable
    HANDLE hFile;
    BOOL bFile;
    char chBuffer[] = "Sample for Readfile and WriteFile";
    DWORD dwNoBytesToWrite = strlen(chBuffer);//size of written data
    LPDWORD lpNoByteWritten = 0;//Set Write pointer to 0
    DWORD dwNoBytesToRead = strlen(chBuffer);//size of Buffer
    LPDWORD lpNoByteRead = 0;//Set Read Pointer to 0

   //STEP-1 CreateFile Function
    hFile = CreateFile(
        L"C:\\SampleFolder\\SampleFile.txt",//File Path and Name
        GENERIC_READ | GENERIC_WRITE,      //File Permission
        FILE_SHARE_READ | FILE_SHARE_WRITE,  //File Sharing Mode
        NULL,                              //Security Attribute
        CREATE_NEW | OPEN_EXISTING,   //if  File exist then open existing File.
        FILE_ATTRIBUTE_NORMAL,          //Flag for File
        NULL);                         //Default File Template

       //STEP-2- Check file has successfully created or not
    if (INVALID_HANDLE_VALUE == hFile)
    {
        cout << "Create File Failed" << endl;
        cout << "Error No - " << GetLastError() << endl;
    }
    cout << "Create File Success " << endl;
```



```
    //STEP-3 Write File Function
    bFile = WriteFile(
        hFile,
        chBuffer,
        dwNoBytesToWrite,
        lpNoByteWritten,
        NULL);
    if (FALSE == bFile)
    {
        cout << "WriteFile Failed & Error No- " << endl;
        cout << "Error No- " << GetLastError() << endl;
    }
    cout << "WriteFile Success" << endl;

    //STEP-4 Read File Function
    bFile = ReadFile(
        hFile,
        chBuffer,
        dwNoBytesToRead,
        lpNoByteRead,
        NULL);
    if (FALSE == bFile)
    {
        cout << "ReadFile Failed "<< endl;
        cout << "Error No- " << GetLastError() << endl;
    }
    cout << "ReadFile Success" << endl;

    //STEP-5 Read data from Buffer
    cout << "Data Reading from Buffer - " << chBuffer << endl;

    //STEP-6 CloseHandle
    CloseHandle(hFile);

    system("PAUSE");
    return 0;
```