```cpp
/*
    Marshall Lindsay
    Max Houck
    Formula SAE Tire Temperature Visualization
    ECE 3220 Final Project
    serial.cpp
*/

#include "SerialClass.h"

Serial::Serial(const char *portName)
{
    //We're not yet connected
    this->connected = false;

    //Try to connect to the given port throuh CreateFile
    this->hSerial = CreateFile(portName,
            GENERIC_READ | GENERIC_WRITE,
            0,
            NULL,
            OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL,
            NULL);

    //Check if the connection was successfull
    if(this->hSerial==INVALID_HANDLE_VALUE)
    {
        //If not success full display an Error
        if(GetLastError()==ERROR_FILE_NOT_FOUND){

            //Print Error if neccessary
            printf("ERROR: Handle was not attached. Reason: %s not available.\n",
            portName);
            throw(1);

        }
        else
        {
            printf("ERROR!!!");
            throw(2);
        }
    }
    else
    {
        //If connected we try to set the comm parameters
        DCB dcbSerialParams = {0};

        //Try to get the current
        if (!GetCommState(this->hSerial, &dcbSerialParams))
        {
            //If impossible, show an error
            printf("failed to get current serial parameters!");
        }
        else
        {
            //Define serial connection parameters for the arduino board
            dcbSerialParams.BaudRate=CBR_9600;
            dcbSerialParams.ByteSize=8;
            dcbSerialParams.StopBits=ONESTOPBIT;
            dcbSerialParams.Parity=NOPARITY;
            //Setting the DTR to Control_Enable ensures that the Arduino is properly
            //reset upon establishing a connection
            dcbSerialParams.fDtrControl = DTR_CONTROL_ENABLE;

             //Set the parameters and check for their proper application
             if(!SetCommState(hSerial, &dcbSerialParams))
             {
                printf("ALERT: Could not set Serial Port parameters");
             }
            else
            {
                //If everything went fine we're connected
                this->connected = true;
                //Flush any remaining characters in the buffers
                PurgeComm(this->hSerial, PURGE_RXCLEAR | PURGE_TXCLEAR);
                //We wait 2s as the arduino board will be reseting
                Sleep(ARDUINO_WAIT_TIME);
            }
        }
    }

}

Serial::~Serial()
{
    //Check if we are connected before trying to disconnect
    if(this->connected)
    {
        //We're no longer connected
        this->connected = false;
        //Close the serial handler
        CloseHandle(this->hSerial);
    }
}

int Serial::ReadData(char *buffer, unsigned int nbChar)
{
    //Number of bytes we'll have read
    DWORD bytesRead;
    //Number of bytes we'll really ask to read
    unsigned int toRead;

    //Use the ClearCommError function to get status info on the Serial port
    ClearCommError(this->hSerial, &this->errors, &this->status);

    //Check if there is something to read
    if(this->status.cbInQue>0)
    {
        //If there is we check if there is enough data to read the required number
        //of characters, if not we'll read only the available characters to prevent
        //locking of the application.
        if(this->status.cbInQue>nbChar)
        {
            toRead = nbChar;
        }
        else
        {
            toRead = this->status.cbInQue;
        }

        //Try to read the require number of chars, and return the number of read bytes on success
        if(ReadFile(this->hSerial, buffer, toRead, &bytesRead, NULL) )
        {
            return bytesRead;
        }

    }

    //If nothing has been read, or that an error was detected return 0
    return 0;

}


bool Serial::WriteData(const char *buffer, unsigned int nbChar)
{
    DWORD bytesSend;
```

```cpp

    //Try to write the buffer on the Serial port
    if(!WriteFile(this->hSerial, (void *)buffer, nbChar, &bytesSend, 0))
    {
        //In case it don't work get comm error and return false
        ClearCommError(this->hSerial, &this->errors, &this->status);

        return false;
    }
    else
        return true;
}

bool Serial::IsConnected()
{
    //Simply return the connection status
    return this->connected;
}

```