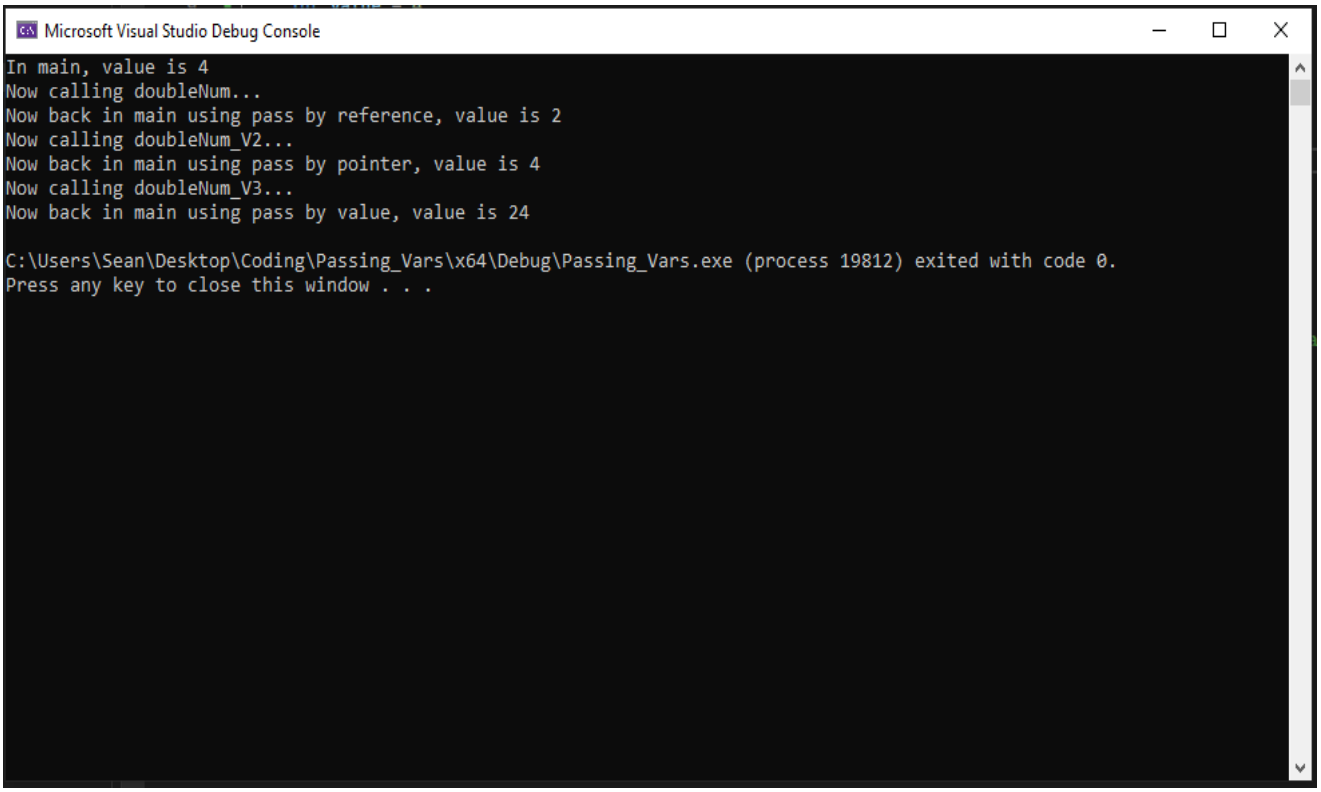

Header File: Passing.h

```
#pragma once


class Passing

{

private:


public:

void doubleNum(int& refVar);

void doubleNum_V2(int* refVar2);

int doubleNum_V3(int refVar0);

};
```

Implementation File: Passing.cpp

```
#include "Passing.h"
// This program uses a reference variable as a function parameter.
#include <iostream>
#include <cmath>
using namespace std;

/**************************************************************
*                    doubleNum                    *
* This function's parameter is a reference variable. The &   *
* tells us that. This means it receives a reference to the   *
* original variable passed to it, rather than a copy of that *
* variable's data. The statement refVar *= 2 is doubling the *
* data stored in the value variable defined in main.         *
**************************************************************/
void Passing::doubleNum(int& refVar) // Passing by reference because & is used
{
refVar = 2;
}

void Passing::doubleNum_V2(int* refVar2)
{
*refVar2 = pow(*refVar2, 2); // don't forget to dereference
}

int Passing::doubleNum_V3(int refVar0)
{
refVar0 = refVar0 * 6;
return refVar0;
}
```

EndUserDriver: EndUserDriver.cpp

```
#include "Passing.h"

#include <iostream>

using namespace std;


int main()

{

Passing p;


int value = 4;

int* value1 = &value;


cout << "In main, value is " << value << endl;

cout << "Now calling doubleNum..." << endl;

p.doubleNum(value);

cout << "Now back in main using pass by reference, value is " << value << endl;


cout << "Now calling doubleNum_V2..." << endl;

p.doubleNum_V2(value1);

cout << "Now back in main using pass by pointer, value is " << value << endl;


cout << "Now calling doubleNum_V3..." << endl;

value = p.doubleNum_V3(value); // must set value to equal the method called so that when it returns a value to main the value is changed

cout << "Now back in main using pass by value, value is " << value << endl;


return 0;

}
```