CSE 2020 Computer Science II

Module 1B – More Review & Warmups

Instructor: Kerstin Voigt, School of CSE at CSUSB

More worthy program: Testing Magic Square Property

A 3x3 Magic Square:

[4] [9] [2] All row sums = 15

[3] [5] [7] All column sums = 15

[8] [1] [6] Main diag sum = 15

Off-diag sum = 15

General Magic Squrare: N x N N >= 3 (odd?)

What is a suitable C++ data structure to represent a Magic Square (“matrix”)?

```
// Representation A

x11 = 4;
x12 = 9;
x13 = 2;
x21 = 3;
x22 = 5;
x23 = 7;
x31 = 8;
x32 = 1;
x33 = 6;
```

```
// Representation B

int row1[] = {4,9,2};
int row2[] = {3,5,7};
int row3[] = {8,1,6};

// Representation C;

int col1[] = {4,3,8};
int col2[] = {9,5,1};
int col3[] = {2,7,6};
```

```
// Representation D;

vector<int> row1{4,9,2};
vector<int> row2{3,5,7};
vector<int> row3{8,1,6};

// Representation E;

vector<int> row1{4,9,2};
vector<int> row2{3,5,7};
vector<int> row3{8,1,6};

vector<vector<int> > matrix{row1,row2,row3};
```

```
// Representation F;

map<pair<int,int>, int> matrix;

matrix[(0,0)] = 4;
matrix[(0,1)] = 9;
matrix[(0,2)] = 2;
matrix[(1,0)] = 3;
matrix[(1,1)] = 5;
matrix[(1,2)] = 7;
matrix[(2,0)] = 8;
matrix[(2,1)] = 1;
matrix[(2,2)] = 6;
```

[4] [9] [2]

[3] [5] [7]

[8] [1] [6]

DISCUSSION …..

More worthy program: Testing Magic Square Property

A 3x3 Magic Square:

[4] [9] [2] All row sums = 15
[3] [5] [7] All column sums = 15
[8] [1] [6] Main diag sum = 15
Off-diag sum = 15

General Magic Squrare: N x N N >= 3 (odd?)

What is a suitable C++ data structure to represent a Magic Square (“matrix”):

```
vector<vector<int> > magicSQ;
```

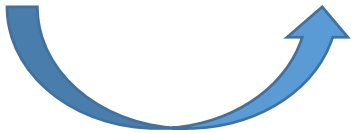
A vector of vectors; inner vectors of type int;

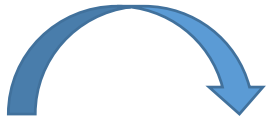
Test with a function

```
bool test_magic(vector<vector<int> > square)
{
    ... code that tests square ...
}
```

File MagicSquare.cpp

```
#include <iostream>
#include <iomanip>
#include <cassert>
#include <vector>
#include <fstream>
#include <math.h>
using namespace std;

void read_square(vector<vector<int> >&);
void print_magic(const vector<vector<int> >&);
bool test_magic(const vector<vector<int> >&, int&);
int row_sum(int i, const vector<vector<int> >&);
int col_sum(int j, const vector<vector<int> >&);
int diag1_sum(const vector<vector<int> >&);
int diag2_sum(const vector<vector<int> >&);
```

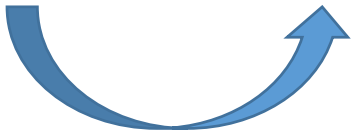


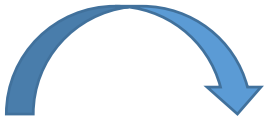


```
int main()
{
    vector<vector<int> > magicSQ;

    read_square(magicSQ);
    print_magic(magicSQ);

    int magic_sum;
    if (test_magic(magicSQ, magic_sum))
    {
        cout << endl;
        cout << "Magic Test passes -- the magic sum is "
             << magic_sum << endl;
    }
    else
    {
        cout << endl;
        cout << "Magic Test fails" << endl;
    }
    return 0;
}
```

File MagicSquare.cpp (to mark up)

```
#include <iostream>
#include <iomanip>
#include <cassert>
#include <vector>
#include <fstream>
#include <math.h>
using namespace std;

void read_square(vector<vector<int> >&);
void print_magic(const vector<vector<int> >&);
bool test_magic(const vector<vector<int> >&, int&);
int row_sum(int i, const vector<vector<int> >&);
int col_sum(int j, const vector<vector<int> >&);
int diag1_sum(const vector<vector<int> >&);
int diag2_sum(const vector<vector<int> >&);
```





```
int main()
{
    vector<vector<int> > magicSQ;

    read_square(magicSQ);
    print_magic(magicSQ);

    int magic_sum;
    if (test_magic(magicSQ, magic_sum))
    {
        cout << endl;
        cout << "Magic Test passes -- the magic sum is "
             << magic_sum << endl;
    }
    else
    {
        cout << endl;
        cout << "Magic Test fails" << endl;
    }
    return 0;
}
```

File MagicSquare.cpp cont.

input file: a_square.txt

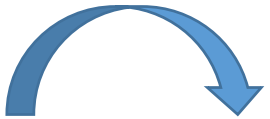
11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

```
// read magic square info from input file a_square.txt
void read_square(vector<vector<int> >& square)
{
    ifstream inp;
    vector<int> nums;
    inp.open("a_square.txt");
    int next;
    inp >> next;
    while (!inp.fail())
    {
        nums.push_back(next);
        inp >> next;
    }
    inp.close();

    assert(sqrt(nums.size()) == floor(sqrt(nums.size())));

    int n = static_cast<int>(sqrt(nums.size()));

    int k = 0;
    for (int i = 1; i <= n; i++)
    {
        vector<int> row;
        for (int j = 1; j <= n; j++)
        {
            row.push_back(nums[k]);
            k++;
        }
        square.push_back(row);
    }
    return;
}
```

```
// print out the magic square
void print_magic(const vector<vector<int> >& mag)
{
    cout << endl;
    for (int i = 0; i < mag.size(); i++)
    {
        for (int j = 0; j < mag.size(); j++)
            if (mag[i][j] == 0)
                cout << left << setw(6) << "~" << "  ";
            else
                cout << left << setw(6) << mag[i][j] << "  ";
        cout << endl;
    }
    cout << endl;
    return;
}
```

File MagicSquare.cpp cont. cont.

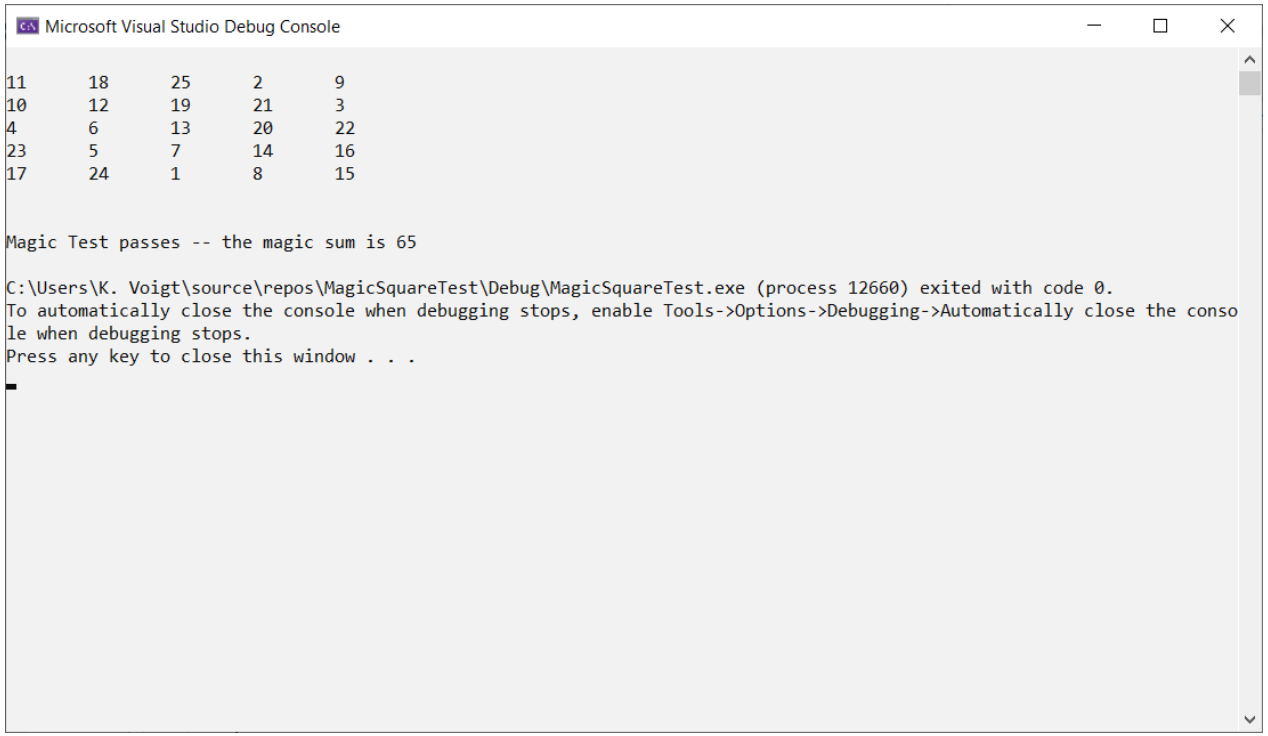
```
// true if vector of vector is magic square: row sums,
// col sums and sums of both diagonals are the same;
// output argument int& msum takes on value of that sum;

bool test_magic(const vector<vector<int> >& mag, int& msum)
{
    int dim = mag[0].size();
    int rsum = row_sum(0, mag);

    for (int i = 1; i < dim; i++)
        if (row_sum(i, mag) != rsum)
            return false;
    for (int i = 0; i < dim; i++)
        if (col_sum(i, mag) != rsum)
            return false;
    if (diag1_sum(mag) != rsum || diag2_sum(mag) != rsum)
        return false;

    msum = rsum;
    return true;
}
```



```
// sum of ith row of mag
int row_sum(int i, const vector<vector<int> >& mag)
{
    int dim = mag[0].size();
    int sum = 0;
    for (int j = 0; j < dim; j++)
        sum += mag[i][j];
    return sum;
}

// sum of jth column
int col_sum(int j, const vector<vector<int> >& mag)
{
    int dim = mag[0].size();
    int sum = 0;
    for (int i = 0; i < dim; i++)
        sum += mag[i][j];
    return sum;
}
```

File MagicSquare.cpp cont. cont. cont.

```
// sum of main diagonal
int diag1_sum(const vector<vector<int> >& mag)
{
    int dim = mag[0].size();
    int sum = 0;
    for (int i = 0; i < dim; i++)
        sum += mag[i][i];
    return sum;

}

// sum of off-diagonal
int diag2_sum(const vector<vector<int> >& mag)
{
    int dim = mag[0].size();
    int sum = 0;
    for (int i = dim - 1, j = 0; j < dim; i--, j++)
        sum += mag[i][j];
    return sum;
}
```



*** End of Module 1.1B ***

Keep Handy for Lab in Week 1