EGR 125 - Introduction to Engineering Methods (C++) Due date: \_\_11/4/2021\_\_\_

File: N125-Ch8LB\_and\_Ch12LB Name: \_\_David Vermaak\_\_

**Chapter 8 and Chapter 12 Homework:**

**Multidimensional Arrays and the Vector Class**

**Reading Assignment:**

Read Chapters 8 and 12 in Introduction to Programming with C++, 3rd Edition by Liang

**Problem Assignment:**

1. (10 pts) Work the following ***Quick Check Exercises*** in the textbook. As usual, include the instructions and the given code with the answers.

* ***Quick Check Exercise*** 8.4: What is the output of the following code?

#include <iostream>

using namespace std;

int main()

{

int matrix[4][4] =

{{1, 2, 3, 4},

{4, 5, 6, 7},

{8, 9, 10, 11},

{12, 13, 14, 15}};

int sum = 0;

for (int i = 0; i < 4; i++)

sum += matrix[i][i];

cout << sum << endl;

return 0;

}

31

* ***Quick Check Exercise*** 8.5: What is the output of the following code?

#include <iostream>

using namespace std;

int main()

{

int matrix[4][4] =

{{1, 2, 3, 4},

{4, 5, 6, 7},

{8, 9, 10, 11},

{12, 13, 14, 15}};

int sum = 0;

for (int i = 0; i < 4; i++)

cout << matrix[i][1] << " ";

return 0;

}

2 5 9 13

2. (22 pts) ***Sum the major diagonal in a matrix***: Programming Exercise 8.2: Write a function that sums all the double values in the major diagonal in an n \* n matrix of double values using the following header:

const int SIZE = 10;

double sumMajorDiagonal(const double m[][SIZE]);

Write a test program that reads a 4-by-4 matrix and displays the sum of all its elements

on the major diagonal. Additional details:

* The size of an array cannot be changed based on user input (dynamic memory allocation), so the matrix should be dimensionsed to the max size ever expected (10 x 10 perhaps).
* Prompt user to enter N (the size of the N x N matrix). The program should work for any N >= 2. N should not be used anywhere for matrix dimensions. It is only used in for loops to control user input and printing. Note that a 3x3 matrix is really just using the upper corner of the 10x10 matrix.
* Prompt the user to enter the elements in the matrix row-by-row.
* Display the NxN matrix.
* Display the sum of the elements in the major diagonal. The sum should be displayed from the main function, not from the function sumMajorDiagonal.
* Include a printout of the main program and the function.
* Include printouts for the test case in the textbook as well as for a 2x2 matrix and a 3x3 matrix.

**Program 8\_2:**

// \_\_\_\_ \_\_ \_\_

// / / \ / \

// / / /

// / \\_\_\_/ \\_\_\_/

//

// Date: 11/4/2021

// Name: David Vermaak

// Project Description: HWK 8 & 12 Problem 2

// Inputs: doubles in a matrix

// Outputs: a single double

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream> //This header containing cout and cin

#include <conio.h> //This header declares getch which pauses intil a key is pushed

#include <math.h> //This header allows for the use of more complex mathmatical operators

#include <string> //This header enables string functions

#include <cctype> //This header contains functions that give boolian answers for given types of char

#include <iomanip> //This header contains functions to better format output

#include <ctype.h> // This header contains functions to manipulate chars

using namespace std;

const int SIZE = 10;

double sumMajorDiagonal( double [][SIZE]); //Function declaration

int main()

{

int n = 10, d;

double m[n][n] = { };

cout << "Enter the size of your square Matrix (2-10)";

cin >> n;

for (int i = 0; i< n; i++)

{

for (int j = 0; j < n; j++)

{

cout << "Enter the number " << i <<" , " << j <<" unit in the matrix" << endl;

cin >> m[i][j];

}

}

d = sumMajorDiagonal(m[][n]);

cout << "Major Diagonal:" << d << endl;

return 0;

}

double sumMajorDiagonal( double m[][SIZE]) //Function definition

{ double diagonal = 0;

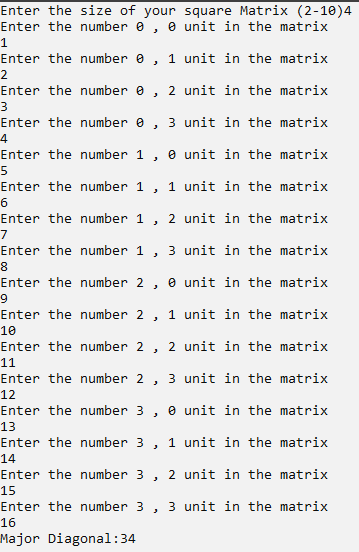
for (int i = 0; i < SIZE; i++)

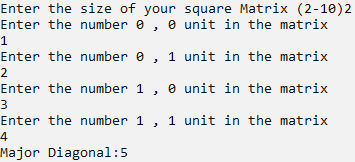
{ diagonal += m[i][i];}

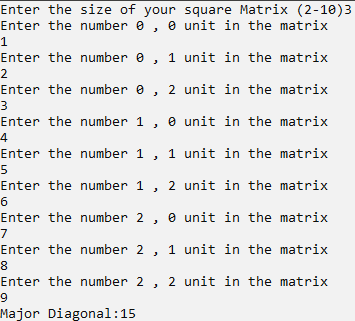
return diagonal;

}

**Program 8\_2 output:**







3. (22 pts) ***Matrix of velocities***: Write a C++ program to convert velocities to alternate units as described below:

* Use a single 2D array with 3 columns and 21 rows for the velocity values as follows:
  + column 0 – velocity in mi/h (mph)
  + column 1 – velocity in km/h
  + column 2 – velocity in ft/s
* The 3D array is illustrated below where velocity varies from 0 – 100 mi/h:

|  |  |  |
| --- | --- | --- |
| 0 |  |  |
| 5 |  |  |
| 10 |  |  |
| 15 |  |  |
| 20 |  |  |
| 25 |  |  |
| 30 |  |  |
| 35 |  |  |
| 40 |  |  |
| 45 |  |  |
| 50 |  |  |
| 55 |  |  |
| 60 |  |  |
| 65 |  |  |
| 70 |  |  |
| 75 |  |  |
| 80 |  |  |
| 85 |  |  |
| 90 |  |  |
| 95 |  |  |
| 100 |  |  |

* Use a loop or a list to assign the velocity values to column 0.
* Use a loop to calculate and assign the velocities in columns 1 and 2.
* Neatly display the velocity in three columns with a table heading. Select a reasonable number of digits for each column.

**Program 8\_3:**

// \_\_\_\_ \_\_ \_\_

// / / \ / \

// / / /

// / \\_\_\_/ \\_\_\_/

//

// Date: //2021

// Name: David Vermaak

// Project Description: Template

// Inputs:

// Outputs:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream> //This header containing cout and cin

#include <conio.h> //This header declares getch which pauses intil a key is pushed

#include <math.h> //This header allows for the use of more complex mathmatical operators

#include <string> //This header enables string functions

#include <cctype> //This header contains functions that give boolian answers for given types of char

#include <iomanip> //This header contains functions to better format output

#include <ctype.h> // This header contains functions to manipulate chars

#include <vector> //This header allows the use of vectors (better arrays)

#include <algorithm> //This header allows the use of algorithm functions

#include <numeric> //This header allows the use of numeric functions

using namespace std;

const int row = 21, col = 3;

int main()

{

double M[col][row];

for (int j = 0; j < 21; j++)

{ int i = 0;

M[0][j] = i;i+=5;

}

for (int j = 0; j < 21; j++)

{ int i = 0;

M[1][j] =i\*1.609344;i+=5;

}

for (int j = 0; j < 21; j++)

{ int i = 0;

M[2][j] = i\*1.466667;i+=5;

}

cout << setw(4) << "mph" << setw(4) << "km/h" << setw(4) << "ft/s" <<endl;

for (int r = 0; r < row; r++)

{

for (int c = 0; c < col; c++)

{

cout << setw(4) << setpresction(3)<< M[col][row];

}

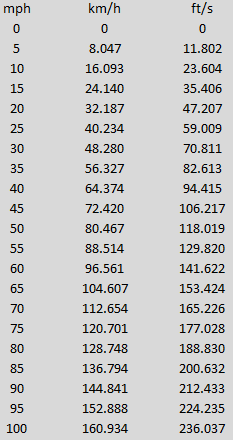
cout << endl;

}

return 0;

}

**Program 8\_3 output:**



4. (22 pts) ***Vector Class***: Write a C++ program that reads an unspecified number of temperatures in Fahrenheit from the keyboard (prompt the user to enter them one at a time). The values should be added to a vector using the ***vector class*** with the ***push\_back*** function. Instruct the user to enter an invalid temperature (<-459.4) to indicate that they are done entering values. Use functions and algorithms in the ***<algorithm>*** and ***<numeric>*** libraries to find the max, min, sum, and count (number of values) and display the results. Also sort the temperatures in increasing order and display the values. Include units with all outputs. Print the program along with an example that includes at least 10 valid temperatures.

**Program 8\_4:**

// \_\_\_\_ \_\_ \_\_

// / / \ / \

// / / /

// / \\_\_\_/ \\_\_\_/

//

// Date: //2021

// Name: David Vermaak

// Project Description: Template

// Inputs:

// Outputs:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream> //This header containing cout and cin

#include <conio.h> //This header declares getch which pauses intil a key is pushed

#include <math.h> //This header allows for the use of more complex mathmatical operators

#include <string> //This header enables string functions

#include <cctype> //This header contains functions that give boolian answers for given types of char

#include <iomanip> //This header contains functions to better format output

#include <ctype.h> // This header contains functions to manipulate chars

#include <vector> //This header allows the use of vectors (better arrays)

#include <algorithm> //This header allows the use of algorithm functions

#include <numeric> //This header allows the use of numeric functions

using namespace std; //introduces namespace std

int main ( )

{

vector<double> temps;

vector<double>::iterator iterate;

double degrees = 0;

double sum = 0;

double max = 0;

double min = 0;

double count = 0;

while(degrees > -459.4 && degrees < 459.4)

{

cout << "Enter the temperature values: ";

cin >> degrees;

if(degrees > -459.4 && degrees < 459.4)

{

temps.push\_back(degrees);

}

}

iterate = max\_element(temps.begin(),temps.end());

max = \*iterate;

iterate = min\_element(temps.begin(),temps.end());

min = \*iterate;

sum = accumulate(temps.begin(),temps.end(),0);

count = temps.size();

sort(temps.begin(), temps.end());

cout << "\n\n\n\n\n\nMax temp = " << max << endl

<< "Min temp = " << min << endl

<< "The sum of the temperatures is: " << sum << endl

<< "The count of the temperatures is: " << count << endl << endl;

for (int i = 0; i<temps.size(); i++)

{

cout <<temps[i] <<" , ";

}

return 0;

}

**Program 8\_4 output:**

Enter the temperature values: 100

Enter the temperature values: 59

Enter the temperature values: 57

Enter the temperature values: 90

Enter the temperature values: 38

Enter the temperature values: -375

Enter the temperature values: 0.96

Enter the temperature values: 58

Enter the temperature values: -100

Enter the temperature values: 93

Enter the temperature values: 287

Enter the temperature values: 900

Max temp = 287

Min temp = -375

The sum of the temperatures is: 308

The count of the temperatures is: 11

-375 , -100 , 0.96 , 38 , 57 , 58 , 59 , 90 , 93 , 100 , 287

5. (24 points) Determine the output for each part below by hand (do not compile the programs).

|  |  |  |
| --- | --- | --- |
| **Part A** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = 0; i < R; i++)**  **{ for (int j = 0; j < C; j++)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **1 2 3**  **4 5 6**  **7 8 9**  **10 11 12** |
| **Part B** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = 1; i < R; i++)**  **{ for (int j = 1; j < C; j++)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **5 6**  **8 9**  **11 12** |
| **Part C** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int j = 0; j < C; j++)**  **{ for (int i = 0; i < R; i++)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **1 4 7 10**  **2 5 8 11**  **3 6 9 12** |
| **Part D** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = R-1; i >= 0; i--)**  **{ for (int j = C-1; j >= 0; j--)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **12 11 10**  **9 8 7**  **6 5 4**  **3 2 1** |
| **Part E** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = 0; i < R; i++)**  **{ for (int j = 0; j < i+1; j++)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **1**  **4 5**  **7 8 9**  **10 11 12** |
| **Part F** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = 0; i < R; i++)**  **{ for (int j = 0; j < C; j++)**  **if(i>=j)cout<<setw(3)<<A[i][j];**  **cout << endl; }** | **1**  **4 5**  **7 8 9**  **10 11 12** |
| **Part G** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = R-1; i >= 0; i--)**  **{ for (int j = 0; j < C; j++)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **10 11 12**  **7 8 9**  **4 5 6**  **1 2 3** |
| **Part H** | **const int R = 4, C = 3;**  **double A[R][C]={1,2,3,4,5,6,7,8,9,10,11,12};**  **for (int i = 0; i < R; i++)**  **{ for (int j = C-1; j >=0; j--)**  **cout << setw(3) << A[i][j];**  **cout << endl; }** | **3 2 1**  **6 5 4**  **9 8 7**  **12 11 10** |