Two-Dimensional Arrays

**Problem:**
Assume that each student in a class has four grades representing marks on four exams. The instructor wishes to find various statistics:
- The average mark on each exam.
- The highest and lowest mark on each exam.
- Each student's average over all four exams.

**Solution:**
To solve this problem efficiently, we need a two-dimensional array.

**Two-Dimensional Array Declaration Syntax:**
`data_type identifier[num_rows][num_columns];`

**Example of a 3 x 6 array:**
```
int number[3][6];  //This is the declaration
```

```
<table>
<thead>
<tr>
<th>r</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>0</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td>-27</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>2</td>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Example Usage:**
```
number[0][0] = 95;
number[0][5] = -27;
number[1][3] = 17;
number[2][1] = 68;
```
Processing a Two-Dimensional Array in a Main Program

/* program to read data into a two-dimensional array */
#include <iostream>
using namespace std;

const int MAXSIZE = 50;
const int NUMEXAMS = 4;

int main()
{
    int grade[MAXSIZE][NUMEXAMS];
    int class_size;

    cout << "How many students in the class? ";
    cin >> class_size;

    for (int stnum = 0; stnum < class_size; stnum++)
    {
        cout << "Type in four grades for student " << stnum
             << endl;
        for (int exam = 0, exam < NUMEXAMS; exam++)
            cin >> grade[stnum][exam];
        cout << "The grades for student " << stnum << " were:"
             << endl;
        for (int exam = 0, exam < NUMEXAMS; exam++)
            cout << " " << grade[stnum][exam];
    }
    return 0;
}
Passing a Two-Dimensional Array as a Parameter

● The Function findstudentavg():

/* Function findstudentavg() 
* Input: 
* grade - a 2-dimensional array of grades 
* NUMEXAMS - numbers of exams for each student 
* class_size - number of students in the class 
* Process: 
* finds each student's average 
* Output: 
* prints each student's average 
*/

void findstudentavg(int grade[][NUMEXAMS], int class_size)
{
    int sum;
    double avg;

    for (int stnum = 0; stnum < class_size; stnum++)
    {
        sum = 0;
        for (int exam = 0; exam < NUMEXAMS; exam++)
            sum += grade[stnum][exam];
        avg = (double)sum/NUMEXAMS;
        cout << "Student " << stnum << " had an average of " << avg << endl;
    }
    return;
}

● Function Prototype:
void findstudentavg(int [[NUMEXAMS], int);

● Function Usage:
findstudentavg(grade, class_size);
Processing Down a Column of a Two-Dimensional Array

- The Function *findexamavg()*:

```c
/* Function findexamavg()
   * Input:
   *   grade - a 2-dimensional array of grades
   *   NUMEXAMS - numbers of exams for each student
   *   class_size - number of students in the class
   * Process:
   *   finds the class average on each exam
   * Output:
   *   prints the class average on each exam
   */
void findexamavg(int grade[][NUMEXAMS], int class_size) {
    int sum;
    double avg;

    for (int exam = 0; exam < NUMEXAMS; exam++) {
        sum = 0;
        for (int stnum = 0; stnum < class_size; stnum++)
            sum += grade[stnum][exam];
        avg = (double)sum/class_size;
        cout << "Exam " << exam << " had a class average of " << avg << endl;
    }
    return;
}
```

- Function Prototype:
  ```c
  void findexamavg(int grade[][NUMEXAMS], int);
  ```

- Function Usage:
  ```c
  findexamavg(grade,class_size);
  ```

- Multi-Dimensional Arrays:
  ```c
  data_type identifier[num_dim_1][num_dim_2]...[num_dim_n];
  ```
Array of Structures

// This program uses an array of structures to hold payroll data.
#include <iostream>
#include <iomanip>
using namespace std;

struct PayInfo {
    int hours;           // Hours worked
    double payRate;           // Hourly pay rate
};

int main()
{
    const int NUM_EMPS = 3;      // Number of employees
    int index;
    PayInfo workers[NUM_EMPS];  // Define an array of structures
    double grossPay;

    // Get payroll data
    cout << "Enter the hours worked and hourly pay rates of " << NUM_EMPS << " employees."
    for (index = 0; index < NUM_EMPS; index++)
    {
        cout << "Hours worked by employee #" << (index + 1) << "": ";
        cin  >> workers[index].hours;
        cout << "Hourly pay rate for employee #" << (index + 1) << ": ";
        cin  >> workers[index].payRate;
    }

    // Display each employee's gross pay
    cout << "Here is the gross pay for each employee:
    for (index = 0; index < NUM_EMPS; index++)
    {
        grossPay = workers[index].hours * workers[index].payRate;
        cout << "Employee #" << (index + 1) << ": ";
        cout << "$" << setw(7) << grossPay << endl;
    }
    return 0;
}
Array of Structures with a Constructor

// This program uses an array of structures to hold payroll data.
#include <iostream>
#include <iomanip>
using namespace std;

struct PayInfo
{
    int hours;       // Hours worked
    double payRate;  // Hourly pay rate

    PayInfo(int h=0, double p=0.0) // Constructor
    {
        hours = h;
        payRate = p;
    }
};

int main()
{
    const int NUM_EMPS = 3;   // Number of employees
    int index;
    // Define and initialize array of structures
    PayInfo workers[NUM_EMPS] = {
        PayInfo(10, 9.75),
        PayInfo(20, 10.00),
        PayInfo(30, 20.00)
    };

    double grossPay;

    // Display each employee's gross pay
    cout << "Here is the gross pay for each employee:\n";
    cout << fixed << showpoint << setprecision(2);
    for (index = 0; index < NUM_EMPS; index++)
    {
        grossPay = workers[index].hours * workers[index].payRate;
        cout << "Employee #" << (index + 1) << " : ": " << setw(7) << grossPay << endl;
    }
    return 0;
}
Array of Class Objects

// This header file contains the Circle class declaration.
#ifndef CIRCLE_H
#define CIRCLE_H
#include <cmath>

class Circle
{
    private:
        double radius;                   // Circle radius
        int centerX, centerY;            // Center coordinates
    public:
        Circle()                          // Default constructor
            {  radius = 1.0;                  // accepts no arguments
                centerX = centerY = 0;
            }

        Circle(double r)                  // Constructor 2
            {  radius = r;                    // accepts 1 argument
                centerX = centerY = 0;
            }

        Circle(double r, int x, int y)    // Constructor 3
            {  radius = r;                    // accepts 3 arguments
                centerX = x;
                centerY = y;
            }

        void setRadius(double r)
            {  radius = r;  }

        int getXcoord()
            {  return centerX;  }

        int getYcoord()
            {  return centerY;  }

        double findArea()
            {  return 3.14 * pow(radius, 2);  }
}; // End Circle class declaration
#endif
// This program uses an array of objects.
// The objects are instances of the Circle class.
#include <iostream>
#include <iomanip>
#include "Circle.h"    // Needed to create Circle objects
using namespace std;

const int NUM_CIRCLES = 4;

int main()
{
    Circle circle[NUM_CIRCLES];  // Define an array of Circle objects

    // Use a loop to initialize the radius of each object
    for (int index = 0; index < NUM_CIRCLES; index++)
    {
        double r;
        cout << "Enter the radius for circle " << (index+1) << " : ";
        cin >> r;
        circle[index].setRadius(r);
    }

    // Use a loop to get and print out the area of each object
    cout << fixed << showpoint << setprecision(2);
    cout << "Here are the areas of the " << NUM_CIRCLES << " circles."
    for (int index = 0; index < NUM_CIRCLES; index++)
    {
        cout << "circle " << (index+1) << " : " << circle[index].findArea() << endl;
    }
    return 0;
}
Array of Class Objects using Overloaded Constructors

// This program demonstrates how an overloaded constructor
// that accepts an argument can be invoked for multiple objects
// when an array of objects is created.
#include <iostream>
#include <iomanip>
#include "Circle.h"     // Needed to create Circle objects
using namespace std;

const int NUM_CIRCLES = 4;

int main()
{
    // Define an array of 4 Circle objects. Use an initialization list
    // to call the 1-argument constructor for the first 3 objects.
    // The default constructor will be called for the final object.
    Circle circle[NUM_CIRCLES] = {0.0, 2.0, 2.5};

    // Display the area of each object
    cout << fixed << showpoint << setprecision(2);
    cout << "\nHere are the areas of the " << NUM_CIRCLES << " circles."

    for (int index = 0; index < NUM_CIRCLES; index++)
    {
        cout << "circle " << (index + 1) << setw(8) << circle[index].findArea() << endl;
    }

    return 0;
}