C++ Programming

Week 7

Two-Dimensional Arrays & More about Arrays

(الشرح والأفكار الرئيسية)

نوت السي ++

تتكون النوت من عشرة أسابيع.
نوت كل أسبوع يحتوي على الشرح
وتمارين من امتحانات سابقة.

نوت السي ++

يحتوي على شرح المواضيع
وأمثلة للأفكار الرئيسية من
هوموركات وامتحانات سابقة.

(لمن يريد المزيد)

يتوفر على الموقع ملفات الأسابيع
لتمارين وأمثلة إضافية من واقع
امتحانات سابقة.

لماذا لا تقتني الأحدث؟

النوت يتم تنقيحها وتجديدها نهاية
كل أسبوع، راجع eng-hs.net
للتأكد من شرائك الإصدار الأحدث.

هناك دائما أماكن شاغرة
على القمة.

2-D Arrays (*Output*)

Choose the best answer in the following questions:

int b[3][2] = {{1, 2}, {3}, {4, 5}};
the value of the second row second element is
answer:

a. 3
b. 0 (correct)
c. 2
d. none of the answers

1	2
3	0
4	5

int table [3][3] = {{1, 8}, {2, 4, 6}, {5}};
the value of table [1][1] is
answer:

a. 2
b. 8
c. 1
d. 4 (correct)

1	8	0
2	4	6
5	0	0

double response[3][3] = {9, 8, 7, 5, 3, 2, 1, 5, 6};
the value of response [2][2] is
answer:

a. 6 (correct)
b. 5
c. 3
d. 2

9	*8*	*7*
5	*3*	*2*
1	*5*	*6*

int freq[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
the value of freq[3][3] is
answer:

a. 9
b. 6
c. none is correct (correct)
d. 8

1	*5*	*3*
4	*5*	*6*
7	*8*	*9*

الحقيقــة مثــل الشــمس، قــد تغيب للحظات لكنها ستعود.

2-D Arrays (*Sum-evens*)

Write a program that reads an integer 2-D array of 4 × 3 elements, then call a function that returns the sum of even integers.

```
# include <iostream>
  using namespace std;
```

3	4	15
7	8	5
3	4	9
6	11	10

```
Enter 12 integers: 3 4 15 7 8 5 3 4 9 6 11 10
Sum of even numbers: 32
```

```
int  sum_evens (int x [  ][3], int n,  int m)
{
   int  i, j, sum = 0;

   for (i = 0;  i < n;  i++)
         for (j = 0;   j < m;   j++)
               if (x [i][j] % 2 == 0)
                     sum += x [i][j];

   return sum;
}
```

لقـراءة أو طباعـة مصـفوفة ثنائية الأبعاد لابد أن نستخدم (2 nested loops).

```
int main ( )
{
   int  x [4][3], i, j, sum;

   cout << "Enter 12 integers: ";

   for (i =  0;  i < 4;  i++)
      for (j =  0;  j < 3;  j++)
            cin >> x [i][j];

   sum = sum_evens (x,  4,  3);

   cout << "Sum of even numbers = " << sum  << endl;

   return 0;
}
```

سـيكون بإمكاننـا يومـا مـا حسـاب حركـة الأجرام السماوية، ولكن ليس جنون البشر.

2-D Arrays (*Range*)

Write a program that reads a double 2-D array of 3 × 4 elements, then call a function that returns the range of the array elements.
The main function will print the range of elements.
Note: *range = maximum - minimum*

```
# include <iostream>
  using namespace std;
```

Enter 12 elements: ***3 4 15 7 8 5 3 4 9 6 11 10***
Range = ***12***

3	*4*	*15*	*7*
8	*5*	*3*	*4*
9	*6*	*11*	*10*

```
int getRange (int [ ][ 4 ], int, int);
int main ( )
{
   int  x [3][4], i, j, range;

   cout << "Enter 12 elements: ";

   for (i = 0;  i < 3;  i++)
      for (j = 0;  j < 4;  j++)
          cin >> x [i][j];

   range = getRange (x, 3, 4);

   cout << "Range =   " <<   range   << endl;

   return 0;
}
```

range
هو الفرق بين أكبر
وأصغر عنصرين.

```
int  getRange (int   x [ ][ 4 ], int n, int m)
{
    int i, j, max,  min;

    max = min = x [0][0];

    for (i = 0;  i < n;  i++)
      for (j = 0;   j < m;  j++)
      {
          if (x [i][j] > max)
                    max = x [i][j];

          if (x [i][j] < min)
                    min = x  [i][j];
      }

     return (max – min);
}
```

مهما أخطأنا بحقهن، مازلن يتحاملن
لأجلنا، لأنهن ببساطة أمهات.

2-D Arrays (*Average-odds*)

Write a program that reads an integer 2-D array of 4 × 3 elements then calls a function that returns the average of odd integers.

```
# include <iostream>
  using namespace std;
```

enter 12 elements: *3 4 15 7 8 5 3 4 9 6 11 10*
average odds: *7.08333*

```
double  average_odds (int  x [ ][ 3 ],  int n,  int m)
{
    int  i, j, c = 0, sum = 0;

    for (i = 0;  i < n;  i++)
      for (j = 0;  j < m;  j++)
          if (x [i][j] % 2  != 0)
          {
                  c++;
                  sum += x [i][j];
          }

   if (c == 0)
      return 0;

   return (sum * 1.0) / c;
}
```

3	*4*	*15*
7	*8*	*5*
3	*4*	*9*
6	*11*	*10*

```
int main ( )
{
   int x [4][3], i, j;

   cout << "enter 12 elements: ";
   for (i = 0;  i < 4;  i++)
      for (j = 0;  j < 3;  j++)
          cin >> x [i][j];

   cout << "average odds: " << average_odds (x , 4 , 3) << endl;

   return 0;
}
```

يحزنني إدراكي أني سأفارق الحياة وفي قلبي شوق لكتاب لم أقرأه بعد.

2-D Arrays (*Lower-triangle-elements*)

Write a function that gets a 2-D array of size n × n, the function should return sum of *lower-triangle* odd elements.

1	2	3	4
5	6	7	8
8	0	3	9
2	5	6	2

In the array shown, it should return 10

```
int  lower  (int  x [ ][ 100 ], int  n)
{
      int  i,   j,  sum = 0;

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

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

            if (j < i && x [i][j] % 2 != 0)

                sum += x [i][j];

       return sum;
}
```

شرط lower-triangle
(j < i)

في حال تساوي عدد الصفوف وعدد الأعمدة يكفي رقم واحد للتعبير عن كليهما (n).

الأمور السارة تأتي جماعات، والمصائب كذلك للأسف.

2-D Arrays (*Upper-triangle-elements*)

Write a function that gets a 2-D array of size n × n, the function should return the average of the *upper-triangle* odd elements.

1	2	3	4
5	6	7	8
9	0	3	9
2	5	6	2

In the array shown, it should return 6.333

```
double upper (int  x [ ][100], int  n)
{
    int  i,  j,  c = 0, sum = 0;

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

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

            if (i < j  && x [i][j] % 2 != 0)
            {
                c++;
                sum += x [i][j];
            }

     if (c == 0)
          return  0;

     return  (sum * 1.0) / c;
}
```

كل الحدود الموجودة في المثلث العلوي تتميز بأن رقم الصف لها أصغر من رقم العمود.
(i < j)

البعض يختارالطريق الصحيح أحيانا لمجرد أن الطريق الخاطئ ليس متاحا.

2-D Arrays (*Diagonal-elements*)

Write a function that gets a 2-D array of size n × n, the function should return the sum of diagonal even elements.
Use only one loop.

1	2	3	4
5	6	7	8
9	0	3	9
2	5	6	2

In the array shown, it should return 8

```
int diagonal  (int  x [ ][100], int  n)
{
    int   i,  sum = 0;

    for (i = 0;  i < n;  i++)
        if (x [i][i] % 2 == 0)
                sum += x [i][i];

     return   sum;
}
```

العناصر الموجودة على القطر يتساوى فيها رقم الصف ورقم العمود لذا يمكن التعبير عنها بـ x [i][i].

التاريخ هو صيغة أحداث الماضي التي قرر الناس الاتفاق عليها.

2-D Arrays (*output*)

What is the output of the following codes:

```
void readmat  (int a[][3], int n, int m);
void printmat (int a[][3], int n, int m);

int main()

{
    int a[4][3]={0};
    readmat (a,4,3);
    printmat(a,4,3);
    return 0;
}

void readmat(int a[][3],int n,int m)
{
    cout << "enter a matrix:\n";
    for (int i = n - 1; i >= 0; i--)
        for (int j = 0; j < m; j++)
            cin >> a[i][j];

}

void printmat (int a[][3],int n,int m)
{
    cout << "the new matrix is:\n";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
            cout << a[i][j]<< " ";
        cout << endl;
    }
}
```

10	*11*	*12*
7	*8*	*9*
4	*5*	*6*
1	*2*	*3*

```
enter a matrix:
1 2 3 4 5 6 7 8 9 10 11 12
the new matrix  is:
10  11  12
 7   8   9
 4   5   6
 1   2   3
```

الجميع يتمنى أن يتغير الناس،
القليل يتمنى تغيير نفسه.

Use a double-subscripted array to solve the following problem. A company has four sales-people (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of products sold. Each slip contains the following:

a) The salesperson number.
b) The product number.
c) The total dollar value of that product sold that day.

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all the slips for last month is available, write a program that will read all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in the double-subscripted array *sales*. After processing all the information for last month, print the results in tabular format with each of the columns representing a particular salesperson and each of the rows representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross totals to the right of totaled rows and the bottom of the totaled columns.

```
# include <iostream>
# include <iomanip>
  using namespace std;

int main ( )
{
      int sales[6][5] = { 0 };
      int slips, prod, person, total;
      double value;


      cout << "How many slips do you have for last month? ";
      cin >> slips;
```

الحاجة أرخص ما تكون
حين يستغنى عنها.

```
        for ( int i = 1; i <= slips; i++ )
        {
                cout << "Enter product number, salesperson's number and its
                        total value: ";
                cin >> person >> prod >> value;
                sales[prod][person] += value;
        }

        cout << "\nperson" << setw(7) << "1" << setw(8) << "2" << setw(8)
             << "3" << setw(8) << "4" << setw(10) << "total\n";

        for ( int i = 1; i < 6; i++ )
        {
                total = 0;
                cout << "prod" << i;

                for ( int j = 1; j < 5; j++ )
                {
                        cout << setw(8) << sales[i][j];
                        total += sales[i][j];
                }

                cout << setw(8) << total << endl;
        }

        cout << "total";

        for ( int j = 1; j < 5; j++ )
        {
                total = 0;

                for ( int i = 1; i < 6; i++ )
                        total += sales[i][j];

                cout << setw(8) << total;
        }
        return 0;
}
```

الحرب مجزرة تدور بين أناس لا يعرفون بعضهم لحساب آخرين يعرفون بعضهم، لكنهم لا يتقاتلون.

How many slips do you have for last month? 15

Enter product number, salesperson's number and its total value: 1 1 12
Enter product number, salesperson's number and its total value: 1 1 10
Enter product number, salesperson's number and its total value: 1 2 8
Enter product number, salesperson's number and its total value: 1 3 5
Enter product number, salesperson's number and its total value: 2 4 4
Enter product number, salesperson's number and its total value: 3 2 100
Enter product number, salesperson's number and its total value: 3 3 7
Enter product number, salesperson's number and its total value: 3 4 15
Enter product number, salesperson's number and its total value: 4 1 13
Enter product number, salesperson's number and its total value: 4 1 4
Enter product number, salesperson's number and its total value: 4 3 5
Enter product number, salesperson's number and its total value: 4 4 8
Enter product number, salesperson's number and its total value: 5 2 1
Enter product number, salesperson's number and its total value: 5 3 15
Enter product number, salesperson's number and its total value: 5 3 3

person	1	2	3	4	total
prod1	22	8	5	0	35
prod2	0	0	0	4	4
prod3	0	100	7	15	122
prod4	17	0	5	8	30
prod5	0	1	18	0	19
total	39	109	35	27	

الخبير هو من ارتكب كل الأخطاء التي يمكن ارتكابها في مجال محدود.

1-D arrays with functions (*Bubble Sort*)

Write a function that gets an integer array and its number of elements. The function should sort the array in an ascending order.

```
# include <iostream>
  using namespace std;
void bubble_sort (int [ ], int);

int  main ( )
{
      int  x [10], i, sum;
      cout << "Enter 10 integers: ";
      for (i = 0; i <= 9; i++)
            cin >> x[i];

      bubble_sort (x, 10);

      cout << "The sorted array is: ";
      for (i = 0; i <= 9; i++)
            cout << x[i] << " " ;

      cout << endl;
      return 0;
}

void bubble_sort (int   x[ ], int   n)
{
      int temp;
      for ( int i = 1; i < n; i++ )
            for ( int j = 0; j < n - i; j++ )
                  if ( x[j] > x[j + 1] )
                  {
                        temp = x[j];
                        x[j] = x[j + 1];
                        x[j + 1] = temp;
                  }
}
```

```
Enter 10 integers: 2  1  5  7  8  10  3  4  9  6
The sorted array is: 1  2  3  4  5  6  7  8  9  10
```

خلاصة الحكمة: عش وأنت على قيد الحياة ولا تمت قبل موتك.

1-D arrays with functions (*recursive-sum-evens*)

Write a recursive function that gets an integer array and its number of elements. The function should return the sum of even elements in the array.

```
Enter 10 integers: 2  1  5  7  8  10  3  4  9  6
Sum of even numbers = 30
```

```
# include <iostream>
  using namespace std;

int sum_evens (int [ ], int);

int  main ( )
{
      int  x [10], i, sum;

      cout << "Enter 10 integers: ";

      for (i = 0; i <= 9; i++)
            cin >> x [i];

      sum = sum_evens (x, 10);

      cout << "Sum of even numbers = " << sum << endl;

      return 0;
}


int sum_evens (int  x[ ], int   n)
{
      if (n == 0)
            return 0;

      if (x [n – 1] % 2  ==  0)

            return  x [n – 1] + sum_evens (x, n - 1);

      return   0 + sum_evens (x, n - 1);
}
```

حياة أغلب البشر لا تتعدى إخماد حرائق حولهم والتغلب على بعض المشاكل الطارئة، هذا كل شيء.

يمكن عدم كتابة 0

1-D Arrays with functions (*Recursive-Search-Key*)

Write a recursive function that gets an array, its number of elements and a key. The function should return true if key exists in the array, otherwise it will return false.

```
# include <iostream>
  using namespace std;
```

Enter 7 numbers: ***2.5 1.7 7.2 15.5 17.8 7.2 1.1***
Enter the search key: ***17.8***
Exists

```
bool  Search (double x [ ], int n, double key)
{
      if (n == 0)
            return  false;
      if (x [n – 1] == key)
            return true;
      return Search (x, n – 1, key);
}

int  main ( )
{
      double  x [7], key;
       bool  y;

      cout << "Enter 7 numbers: ";
      for (i = 0; i < 7 ; i++)
            cin >> x [i];

      cout << "Enter the search key: ";
      cin >> key;

      y = Search (x, 7, key);

      if  (y == false)
            cout << "Does not exist\n";
      else
            cout << "Exists\n";

      return 0;
}
```

Enter 7 numbers: ***2.5 1.7 7.2 15.5 17.8 7.2 1.1***
Enter the search key: ***3.6***
Does not exist

لا يوجد اختلاف في طريقة كتابة main عندما تكون الدالة recursive.

تتكون الحياة من لحظات رحيل كثيرة واحدة تلو أخرى.

1-D Arrays with functions (*Recursive-Search-Key-Count*)

Write a recursive function that gets an array, its number of elements and a key. The function should return how many times key presents in the array.

```
# include <iostream>
  using namespace std;
```

Enter 7 numbers: ***2.5 1.7 17.8 15.5 17.8 7.2 1.1***
Enter the search key: ***17.8***
Exists 2 times

Enter 7 numbers: ***2.5 1.7 7.2 15.5 17.8 7.2 1.1***
Enter the search key: ***3.6***
Does not exist

```
int Search (double [], int, double);
int  main ( )
{
        double  x [7], key;
         int  c;

        cout << "Enter 7 numbers: ";
        for (int i = 0; i < 7; i = i + 1)
              cin >> x [i];

        cout << "Enter the search key: ";
        cin >> key;
        c = Search (x, 7, key);
        if ( c > 0)
              cout << "Exists " << c << "  times" << endl;
        else
              cout << "Does not exist" << endl;
        return 0;
}

int Search (double x [ ], int n, double key)
{
          if (n == 0)
                return 0;

          if (x [n – 1] == key)
                return  1 + Search (x, n – 1, key);

          return  0 + Search (x, n – 1, key);
}
```

تم إضافة (0) للتوضيح فقط، يمكن حذفه لكن لابد نستدعي الدالة حتى لو يتحقق شرط التساوي.

تعد الحياة بلا أهداف جريئة أحد صور الموت لأناس على قيد الحياة.

1-D arrays (*output*)

7.18 What does the following program do?

```
1  // Ex. 7.18: Ex07_18.cpp
2  // What does this program do?
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  int whatIsThis( int [], int );
8
9  int main()
10 {
11    const int arraySize = 10;
12    int a[ arraySize ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
13
14    int result = whatIsThis( a, arraySize );
15
16    cout << "Result is " << result << endl;
17    return 0;
18 }
19
20
21 int whatIsThis( int b[], int size )
22 {
23    if ( size == 1 )
24       return b[ 0 ];
25    else
26       return b[ size - 1 ] + whatIsThis( b, size - 1 );
27 }
```

output

```
Result is: 55
```

طالما أن الموت ليس هدفا في حد ذاته،
فلماذا نعيش حياة الأموات، وكأنه
الهدف الأوحد في حياتنا؟

```
1  // Ex. 7.21: Ex07_21.cpp
2  // What does this program do?
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  void someFunction( int [], int, int ); // function prototype
8
9  int main()
10 {
11    const int arraySize = 10;
12    int a[ arraySize ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
13
14    cout << "The values in the array are:" << endl;
15    someFunction( a, 0, arraySize );
16    cout << endl;
17    return 0; // indicates successful termination
18 } // end main
19
20 // What does this function do?
21 void someFunction( int b[], int current, int size )
22 {
23    if ( current < size )
24    {
25       someFunction( b, current + 1, size );
26       cout << b[ current ] << "  ";
27    } // end if
28 } // end function someFunction
```

output

```
the values  in the array are:
10  9  8  7  6  5  4  3  2  1
```

فطر الناس على الفوز، لكنهم يجاهدون أنفسهم كي يؤلفوها على الهزيمة.

Recursive with functions (*Printing 1-D Array*)

Write a recursive function printArray that takes an array and the size of the array as arguments and returns nothing. The function should stop processing and return when it receives an array of size zero.

```
void printArray ( int a[ ], int n )
{
        if ( n == 0 )
                return ;
        printArray ( a, n-1 );
        cout << a[n-1] << " ";
}
```

الإحسان يَحُطُ من قدر من يتَلّقونه.

Recursive with functions (*Minimum Element*)

Write a recursive function recursive Minimum that takes an integer array and the size as arguments and returns the smallest element of the array. The function should stop processing and return when it receives an array of 1 element.

```
int recursiveMinimum ( int a[ ], int n )
{
        if ( n == 1 )
                return a[0];
        if ( a[n-1] < recursiveMinimum ( a, n-1 ) )
                return a[n-1] ;
        return recursiveMinimum ( a, n-1 );
}
```

يقال إن أقصر قصة كتبها إنسان:
رجل ولد وعاش ومات. وأنا أعتقد أن
سيرة آخرين بشيء من التطويل:
رجل ولد ولم يعش ومع ذلك سيموت.