SSCM 1313
C++ COMPUTER PROGRAMMING

Chapter 4:
Array and File I/O

Authors:
Farhana Johar
Professor Dr. Shaharuddin Salleh

Array

An *array* is a tabular representation of data in the form of rows and columns.

$$x_i \rightarrow x[i]$$
$$x_{ij} \rightarrow x[i][j]$$
$$x_{ijk} \rightarrow x[i][j][k]$$

One Dimensional Array

$$v = [v_1 \ v_2 \ \ldots \ v_N] = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \\ v_N \end{bmatrix} \rightarrow v[1], v[2], \ldots, v[N]$$

```
double v[5];
v[0]=3;  v[1]=2.7;  v[2]=-0.45;  v[3]=3.70;  v[4]=-0.5;
```

OR

```
    double  v[5]={3,2.7,-0.45,3.70,-0.5};
```

OR

```
    double  v[]={3,2.7,-0.45,3.70,-0.5};
```

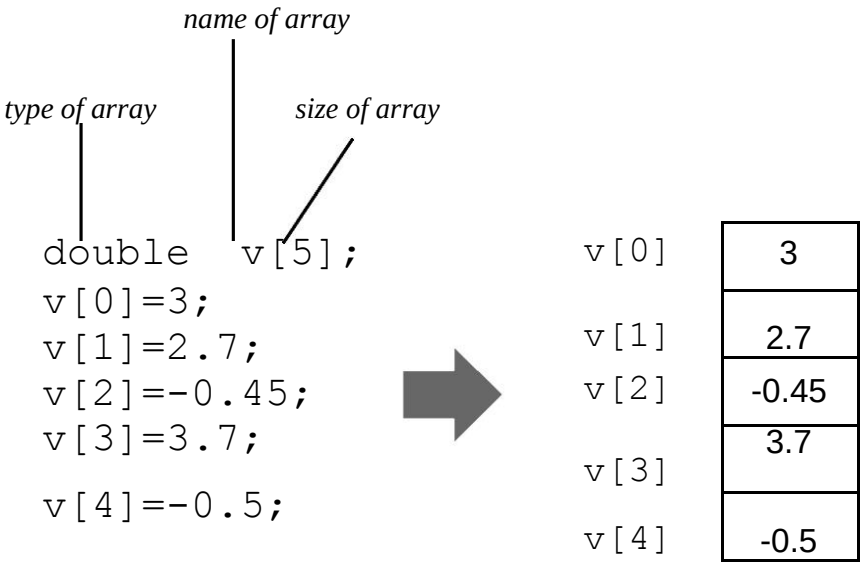
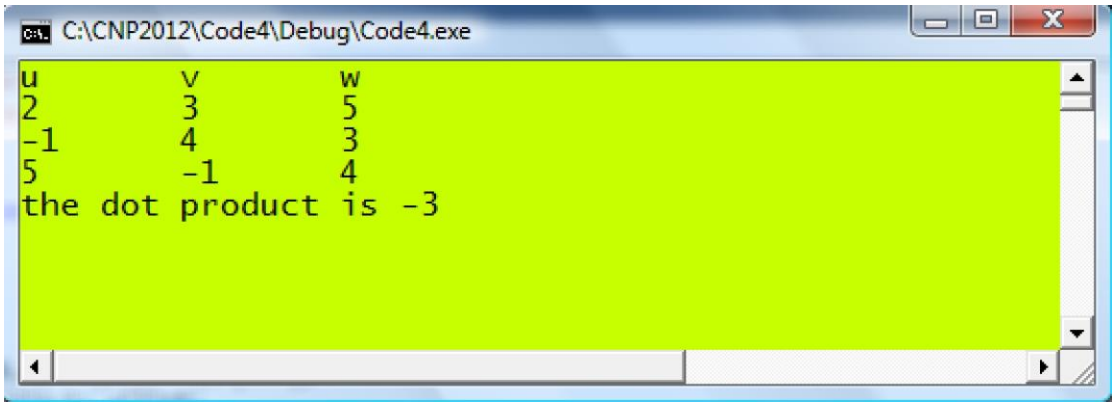


Figure 4.2. One-dimensional array.

Code4A.cpp: Sum and dot-product of vectors.

```
#include <iostream>
#define N 3

using namespace std;

void main()
{
      int i;
      double z=0,u[N+1],v[N+1],w[N+1];
      u[1]=2; u[2]=-1; u[3]=5;
      v[1]=3; v[2]=4; v[3]=-1;
      cout << "u\t" << "v\t" << "w" << endl;
      for (i=1; i<=N; i++)
      {
            w[i]=u[i]+v[i];
            cout << u[i] << "\t" << v[i] << "\t" << w[i] << endl;
            z += u[i]*v[i];
      }
      cout << "the dot product is " << z << endl;
      cin.get();
}
```



Sum of numbers

$$z = \sum_{i=1}^{8} x_i$$

The sum is evaluated easily using the `for` loop, given by

```
int z=0;
for (int i=1;i<=8;i++)
      z += x[i];
```

Note that an initial value of `z=0` is required in the loop.

Product of numbers

Similarly, the product $w = \prod_{i=1}^{8} x_i$ is evaluated by assigning `w=1` as its initial value, as follows:

$$\text{int } w = 1;$$
$$for\ (\text{int } i = 1; i <= 8; i++)$$
$$w* = x[i];$$

Summation of product numbers

$$s = \sum_{i=1}^{8} x_i \sin y_i$$

has its solution written as

```
double s=0;
for (int  i=1;i<=8;i++)
      s+= pow(x[i],2)*sin(y[i]);
```

$$z = \frac{1+3\sum_{i=1}^{5} x_i \cos y_i}{\sqrt{7\sum_{i=1}^{3} x_i^{\,2} - 5\prod_{i=1}^{4} y_i \sin x_i}}$$

Let $p = \sum_{i=1}^{5} x_i \cos y_i$, $q = \sum_{i=1}^{3} x_i^{\,2}$ and $r = \prod_{i=1}^{4} y_i \sin x_i$

This simplifies the above expression into $z = \frac{1+3p}{\sqrt{7q-5r}}$

Code4B.cpp: Summation using loop.

```
#include <iostream>
using namespace std;
void  main()
{
      int i;
      double x[6],y[6],p=0,q=0,r=1,z;
      x[1]=3; x[2]=7; x[3]=-1; x[4]=2; x[5]=9;
      y[1]=-5; y[2]=2; y[3]=8; y[4]=-4; y[5]=1;
      cout << "x\t" << "y\t" << endl;
      for (i=1;  i<=5;  i++)
      {
            cout << x[i] << "\t" << y[i] << endl;
            p += x[i]*cos(y[i]);
      }
      for (i=1;i<=3;i++)
            q += pow(x[i],2);
      for (i=1;i<=4;i++)
            r *= y[i]*sin(x[i]);
      z=(1+3*p)/sqrt(7*q-5*r);
      cout << "p=" << p << endl;
      cout << "q=" << q << endl;
      cout << "r=" << r << endl;
      cout << "z=" << z << endl;
      cin.get();
}
```

Random Number Generation

time()	Initializes the clock.
return type arguments prototype	void void time.h

srand()	Provides a seed value for the start.

return type
arguments
example
prototype
	`TRUE or FALSE.`
	`(int)`
	`srand(3154)` sets the initial seed to 3154.
	`iostream.h`

rand()	Generates random number.
return type arguments prototype	int void iostream.h

expression	numbers generated
`rand()%10`	0, 1, ..., 9
`rand()%100`	0, 1, ..., 99
`1+rand()%10`	1, 2, ..., 10
`1/(1+rand()%10)`	0.01, 0.02, ..., 1.0

We illustrate the above concept through an example on finding the maximum and minimum of a set of numbers. Suppose there are six numbers, as follows:

```
49        64        95        83        27
```

First, assign an array to hold these values:

```
a[1]=49    a[2]=64    a[3]=95    a[4]=83    a[5]=27
```

Next, assign the first value into a variable, call it `max`:

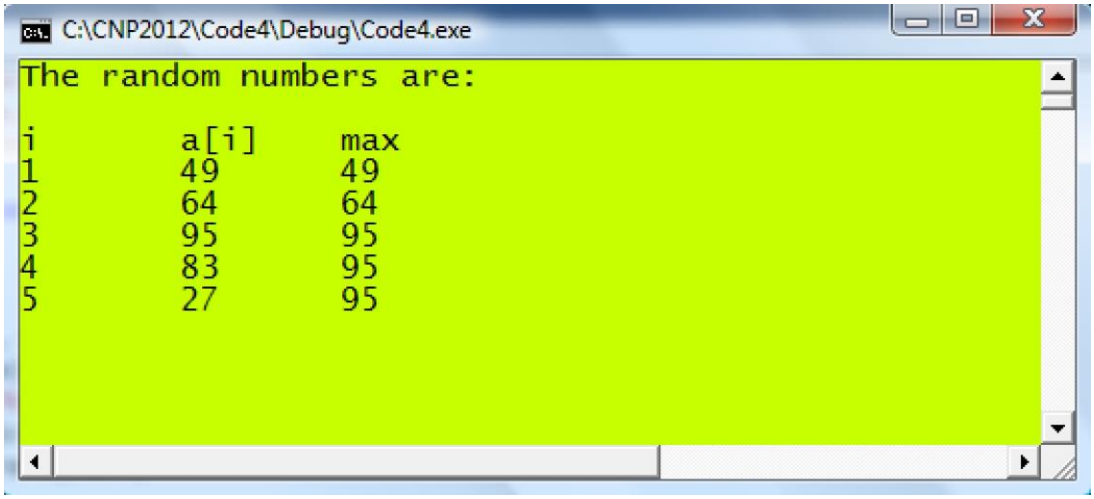
```
max=a[1]
```

Run a loop with five repeats, and at each time compare the values to get the following results:

	Initial	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
Comparison	`a[i]`	`max`	`max`	`Max`	`max`	`max`
`max<a[1]?`	49	49				
`max<a[2]?`	64		64			
`max<a[3]?`	95			95		
`max<a[4]?`	83				95	
`max<a[5]?`	27					95

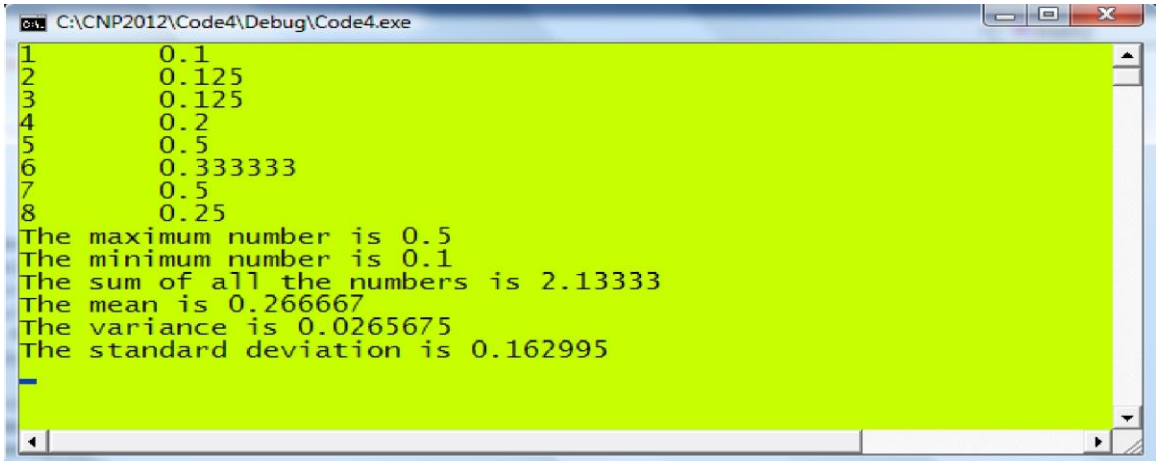
Code4C.cpp: Computing the maximum of numbers.

```
#include <iostream>
#include <time.h>

using namespace  std;

void main()
{
      int a[10], max, i; srand(time(0));
      cout << "The random numbers are:" << endl << endl;
      for (i=1;i<=5;i++)
            a[i]=rand()%100; max=a[1];
      cout << "i\t" << "a[i]\t" << "max" << endl; for
      (i=1;i<=5;i++)
      {
            if (a[i]>max) max=a[i];
                  cout << i << "\t" << a[i] <<"\t"<<max<<endl;
      }
      cin.get();
}
```



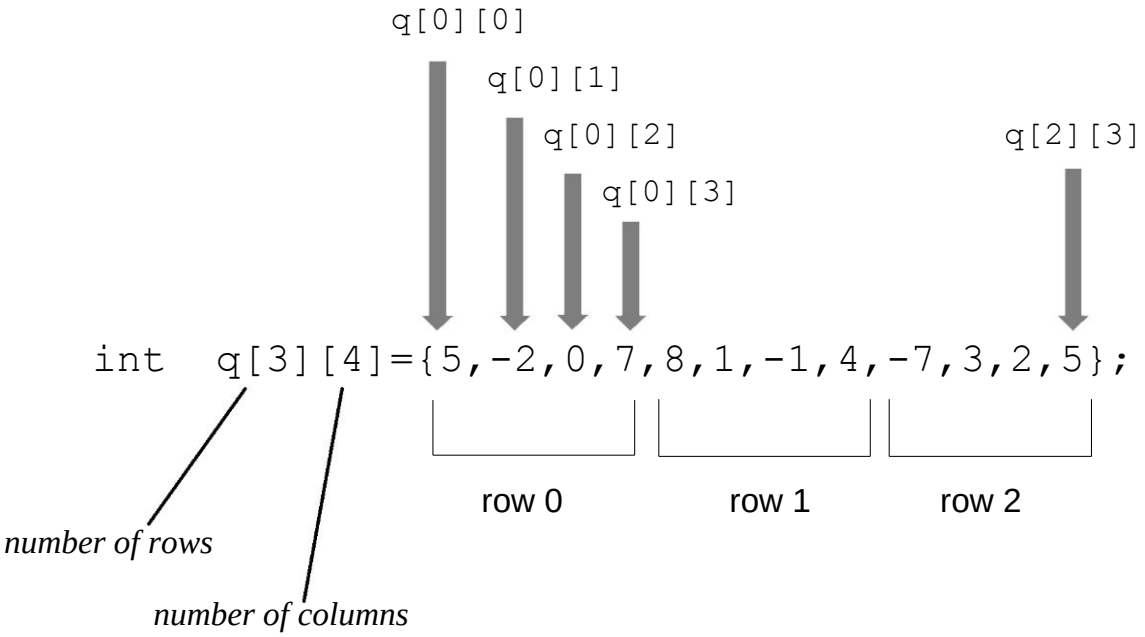
Code4D.cpp: Statistical analysis of numbers.

```
#include
<iostream> #define
N 8

using namespace  std;

void main()
{
       int  i;
       double a[N+1], Sum=0, Max, Min, Mean;
       srand(time(0));
       for  (i=1;i<=N;i++)
         a[i]=1/((double)1+rand()%10); // random numbers from 0.1 to 1.00
       Max=a[1];
        Min=a[1];
       for (i=1;i<=N;i++)
       {
              cout << i << "\t" << a[i] <<
              endl; if (Max<=a[i])
                     Max=a[i]
              ; if
              (Min>=a[i])
                     Min=a[i]
              ; Sum += a[i];
       }
       Mean=Sum/N;
       cout << "The maximum number is " << Max <<
       endl; cout << "The minimum number is " << Min
       << endl;
       cout << "The sum of all the numbers is " << Sum <<
       endl; cout << "The mean is " << Mean << endl;

       double
       SumSquare=0,Variance,StdDev;
       SumSquare=0;
       for (i=1;i<=N;i++)
              SumSquare+=pow(a[i]-
              Mean,2);
       Variance=SumSquare/(N-
       1);
       StdDev=sqrt(Variance);
       cout  <<  "The  variance  is  "  <<  Variance  <<  endl;
       cout << "The standard deviation is " << StdDev <<
       endl;
       cin.get();
}
```



Two-dimensional Array

```
int  q[3][4]={5,-2,0,7,8,1,-1,4,-7,3,2,5};
```



	column 0	*column* 1	*column* 2	*column* 3
row 0	5	-2	0	7
row 1	8	1	-1	4
row 2	-7	3	2	5

Matrix Multiplication.

$$AB = C$$

For example, $A=[a_{ik}]$ and $B=[b_{kj}]$ for $i=1,2,3.$ $k=1,2,3,4$ and $j=1,2$ are multiplied to produce $C=[c_{ij}]$ according to:

$$C = AB \rightarrow [c_{ij}] = \sum_{k=1}^{p} a_{ik} b_{kj}$$

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{4} a_{1k} b_{k1} & \sum_{i=1}^{4} a_{1k} b_{k2} \\ \sum_{i=1}^{4} a_{2k} b_{k1} & \sum_{i=1}^{4} a_{2k} b_{k2} \\ \sum_{i=1}^{4} a_{3k} b_{k1} & \sum_{i=1}^{4} a_{3k} b_{k2} \end{bmatrix}$$

Simplifies as:

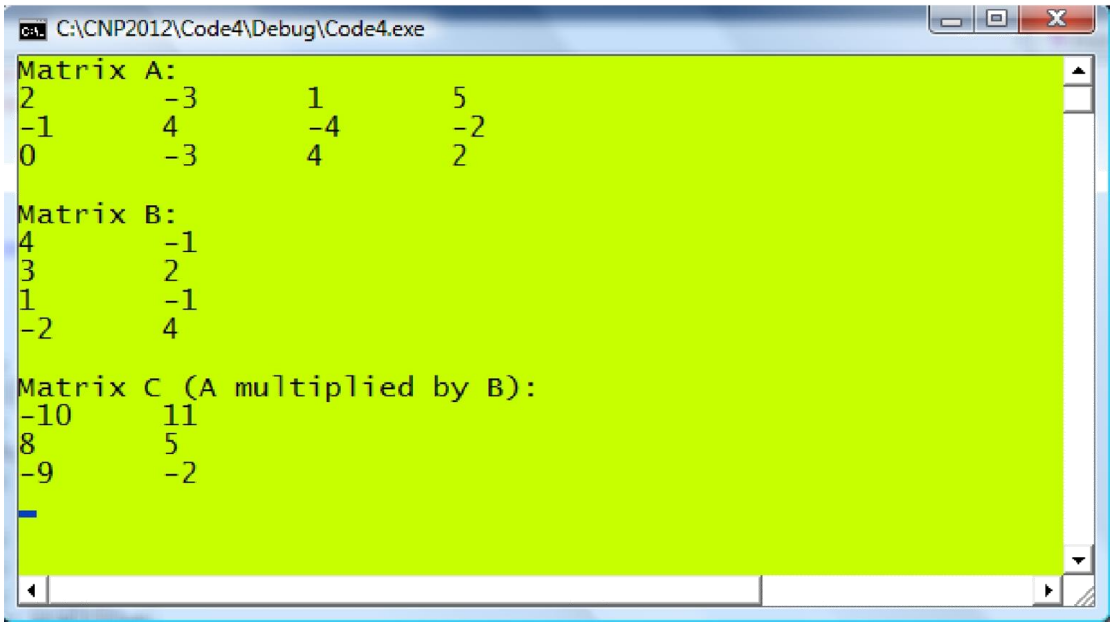
$$[c_{ij}] = \sum_{k=1}^{p} a_{ik} b_{kj}$$

for $i=1,2, \ldots, m$ and $j=1,2,\ldots, n,$ where $m=3$, $n=2$ and $p=4$.

```
int m=3, n=2, p=4;
for (int i=1; i<=m; i++)
      for(int j=1; j<=n; j++)
            for(int k=1; k<=p; k++)
                  c[i][j]+=a[i][k]*b[k][j];
```

Code4E.cpp: Matrix Multiplication.

```
#include <iostream>
#define m 3
#define n 2
#define p 4

using namespace std;

void main()
{
      int i,j,k;
      int a[m+1][p+1], b[p+1][n+1], c[m+1][n+1];
      a[1][1]=2;  a[1][2]=-3; a[1][3]=1;  a[1][4]=5;
      a[2][1]=-1; a[2][2]=4;  a[2][3]=-4; a[2][4]=-2;
      a[3][1]=0;  a[3][2]=-3; a[3][3]=4;  a[3][4]=2;

      b[1][1]=4; b[1][2]=-1;
      b[2][1]=3; b[2][2]=2;
      b[3][1]=1; b[3][2]=-1;
      b[4][1]=-2; b[4][2]=4;
      cout << "Matrix A:" << endl;
      for (i=1; i<=m; i++)
      {
            for (j=1;j<=n;j++)
                  cout << a[i][j] << "\t";
            cout << endl;
      }
      cout << endl << "Matrix B:" << endl;
      for (i=1; i<=p; i++)
      {
            for (j=1; j<=n; j++)
                  cout << b[i][j] << "\t";
            cout << endl;
      }
      cout << endl << "Matrix C (A multiplied by B):" << endl;
      for (i=1; i<=m; i++)
      {
            for (j=1;j<=n;j++)
            {
                  c[i][j]=0;
                  for (k=1;k<=p;k++)
                        c[i][j] += a[i][k]*b[k][j];
                  cout << c[i][j] << "\t";
            }
            cout << endl;
    }
    cin.get ();
}
```


C:\CNP2012\Code4\Debug\Code4.exe
Matrix A:
2 -3 1 5
-1 4 -4 -2
0 -3 4 2
Matrix B:
4 -1
3 2
1 -1
-2 4
Matrix C (A multiplied by B):
-10 11
8 5
-9 -2

String

- an array of characters

```
char str[20];
strcpy(str,”Kedah”);
```

OR

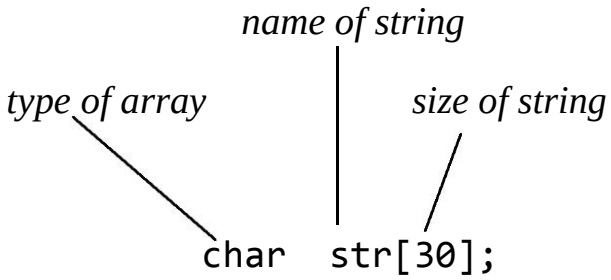
```
char  str[20]=”Kedah”;
```

OR

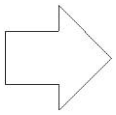
```
char  str[]=”Kedah”;
```

OR

```
char  *str=”Kedah”;
```




```
str[0]='K';
str[1]='e';
str[2]='d';
str[3]='a';
str[4]='h';
str[5]='\0';
```



str[0]	'K'
str[1]	'e'
str[2]	'd'
str[3]	'a'
str[4]	'h'
str[5]	'\0'

Array of strings

maximum number of strings

maximum size of each string

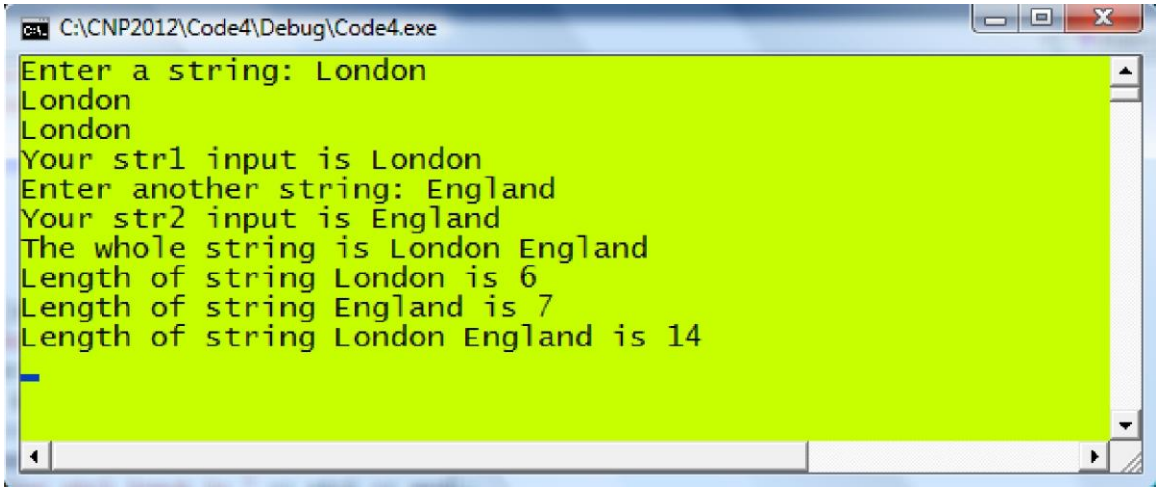
```
char  str[5][15];
```

```
strcpy(str[0],"Tokyo");
strcpy(str[1],"Istanbul");
strcpy(str[2],"Sydney");
strcpy(str[3],"Auckland");

strcpy(str[4],"Washington");
```

str[0]	"Tokyo"
str[1]	"Istanbul"
str[2]	"Sydney"
str[3]	"Auckland"
str[4]	"Washington"

VC2010 **Code4F.cpp:** String manipulation.

```
#include  <iostream>

using  namespace  std;

void  main()
{
       int  i;
       char  str1[10],  str2[10],  str3[20];
       cout  <<  "Enter  a  string:  ";
       cin  >>  str1;
       cout  <<  str1  <<  endl;  //  display  the  whole  string  in  str1
       for  (i=0;  i<=strlen(str1)-1;  i++)
              cout  <<  str1[i];  //  display  str1  character  by  character
       cout  <<  endl;
       cout  <<  "Your  str1  input  is  "  <<  str1  <<  endl;
       cout  <<  "Enter  another  string:  ";
       cin  >>  str2;
       cout  <<  "Your  str2  input  is  "  <<  str2  <<  endl;
       strcpy(str3,str1);          //  copy  and  overwrite  str1  into  str3
       strcat(str3,"  ");          //  add  "  "  into  str3
       strcat(str3,str2);          //  add  str2  into  str3
       cout  <<  "The  whole  string  is  "  <<  str3  <<  endl;
       cout  <<  "Length  of  string  "  <<  str1  <<  "  is  "  <<  strlen(str1)  <<  endl;
       cout  <<  "Length  of  string  "  <<  str2  <<  "  is  "  <<  strlen(str2)  <<  endl;
       cout  <<  "Length  of  string  "  <<  str3  <<  "  is  "  <<  strlen(str3)  <<  endl;
       cin.get();
}
```



File Input/Output (I/O)

To read data from a text file:

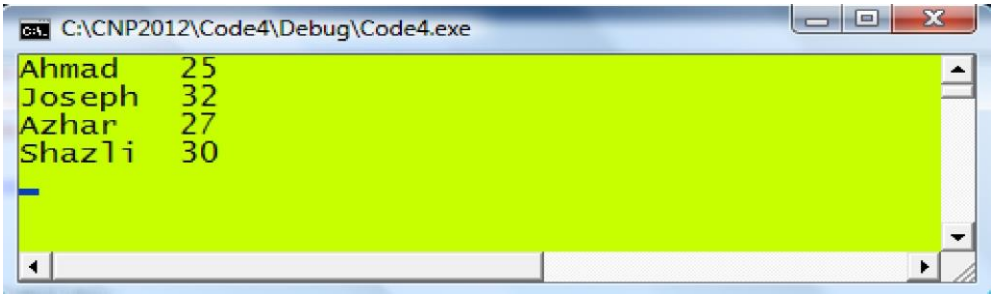
Step	***How to do it?***
Declare an input pointer	`ifstream` *ifp*
Open the file by referring it as the input pointer	*ifp*`.open()`
Read the data through the right indirection operator (>>)	*ifp* `>>` *array*
Close the file	*ifp*`.close()`

To save data into a text file:

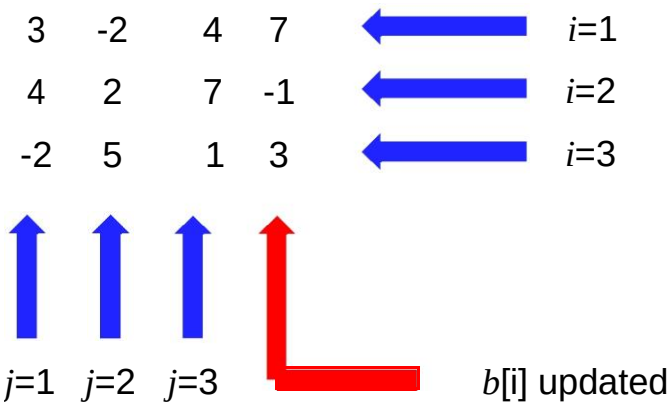
Step	***How to do it?***
Declare an output pointer	`ofstream` *ofp*
Open the file by referring it as the output pointer	*ofp*`.open()`
Store the data through the left indirection operator (<<)	*ofp* `<<` *array*
Close the file	*ofp*`.close()`

Code4G.cpp: File saving and reading.

```
#include <iostream>
#include <fstream>

using  namespace  std;

void  main()
{
      char name[10][15];
      int i,age[10];

      // write the array data into a file
      ofstream ofp;
      ofp.open("code4G.out",ios::out);
      strcpy(name[1],"Ahmad"); age[1]=25;
      strcpy(name[2],"Joseph"); age[2]=32;
      strcpy(name[3],"Azhar"); age[3]=27;
      strcpy(name[4],"Shazli"); age[4]=30;
      for (i=1;i<=4;i++)
            ofp << name[i] << "\t" << age[i] << endl;
      ofp.close();

      // read the saved data to
      verify char Name[10][15];
      int Age[10];
      ifstream ifp;
      ifp.open("code4G.out",ios::in);
      for (i=1;i<=4;i++)
      {
            ifp  >>  Name[i]  >>  Age[i];
            cout  <<  Name[i]  <<  "\t"  <<  Age[i]  <<  endl;
      }
      ifp.close();
      cin.get();
}
```



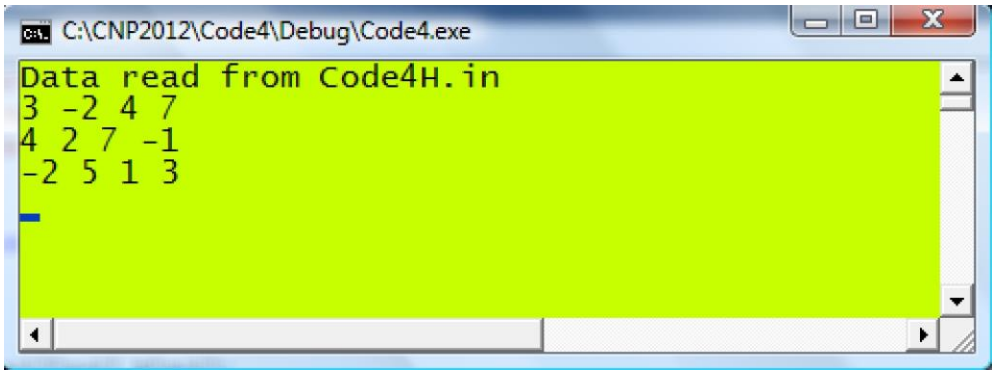
Reading/Writing a Table of Data




```
int  i,j;
for  (int  i=1;i<=n;i++)          // the rows
{
    for  (int  j=1;j<=n;j++)      // the columns
        ifp  >>  a[i][j];         // read columns 1-3, assign to A
    ifp  >>  b[i];                // read column 4, assign to b
 }
```

Code4H.cpp: Reading a table of data.

```
#include <iostream>
#include <fstream>
#define n 3

using  namespace  std;

void  main()
{
      double a[n+1][n+1],b[n+1];
      ifstream ifp;
      ifp.open("Code4H.in",ios::in);
      cout << "Data read from Code4H.in" << endl;
      for (int i=1;i<=n;i++)
      {
            for  (int  j=1;j<=n;j++)
            {
                  ifp  >>  a[i][j];
                  cout  <<  a[i][j]  <<  "  ";
            }
            ifp  >>  b[i];
            cout  <<  b[i]  <<  endl;
      }
      ifp.close();
      cin.get();
}
```



Dynamic Memory Allocation

Static memory allocation refers to a fixed allocation of memory chunks to arrays no matter the memories are actively used or not.

Dynamic memory allocation is a dynamic way of allocating memory to arrays where only active arrays are allocated.

1D-Array

Static Memory Allocation

```
int  v[N+1];
// place more codes here
```

Dynamic Memory Allocation

```
int *v;                    // declares a pointer to the array
v=new int  [N+1];          // allocates memory of size N+1
// place more codes here
delete v;                  // destroys the array and return the memory
```

Code4I.cpp: Sum and dot product problem revisited.

```
#include <iostream>
#define N 3

using  namespace  std;

void  main()
{
       int  i;
       double z=0, *u, *v, *w;
       u=new double [N+1]; v=new
       double [N+1]; w=new double
       [N+1]; u[1]=2; u[2]=-1;
       u[3]=5; v[1]=3; v[2]=4;
       v[3]=-1;
       cout << "u\t" << "v\t" << "w" << endl; for
       (i=1; i<=N; i++)
       {
              w[i]=u[i]+v[i];
              cout << u[i] << "\t" << v[i] << "\t" << w[i] << endl; z
              += u[i]*v[i];
       }
       cout << "the dot product is " << z << endl;
       delete u,v,w;
       cin.get();
}
```

2D-Array

Static Memory Allocation

```
int  q[M+1][N+1];
// place more codes here
```

Dynamic Memory Allocation

```
int  **q;                          // declares a pointer to the array
v=new  int  *[M+1];                // allocates memory to M+1 rows
for  (int  i=0;i<=M;i++)
   q[i]=new  int  [N+1];           // allocates N+1 columns
// place more codes here
delete  q;                         // destroys the array and return the memory
```

Code 4J.cpp:Matrix multiplication problem revisisted

```
#include  <iostream>
#include  <fstream>
#define  m  3
#define  n  2
#define  p  4
using  namespace  std;

void  main()
{
       int  i,j,k;
       int  **a,**b,**c;    a=new  int  *[p+1];
       b=new  int  *[p+1];
       c=new  int  *[p+1];
       for  (i=0;i<=p;i++)
       {
              a[i]=new  int  [p+1];
              b[i]=new  int  [p+1];
              c[i]=new  int  [p+1];
       }      ifstream  ifp("Code4J.in");
       cout << "Matrix A" << endl;
       for (i=1;i<=m;i++)
       {
              for  (j=1;j<=p;j++)
              {
                     ifp  >>  a[i][j];
                     cout  <<  a[i][j]  <<  "  ";
              }
              cout  <<  endl;
       }
       cout << endl << "Matrix B" <<
       endl; for (i=1;i<=p;i++)
       {
              for  (j=1;j<=n;j++)
              {
                     ifp  >>  b[i][j];
                     cout  <<  b[i][j]  <<  "  ";
              }
              cout  <<  endl;
       }
       ifp.close();
       cout << endl << "Matrix C (A multiplied by B):" << endl;
        for (i=1; i<=m; i++)
       {
              for  (j=1;j<=n;j++)
              {
                     c[i][j]=0;
                     for  (k=1;k<=p;k++)
                     c[i][j] += a[i][k]*b[k][j];
                     cout << c[i][j] << " ";
              }
              cout  <<  endl;
       }
       delete a,b,c;
       cin.get();
}
```

Code4J.in

```
2    -3    1    5
-1   4     -4   -2
0    -3    4    2
4    -1
3    2
1    -1
-2   4
```

MAIN REFERENCE:

Shaharuddin Salleh (2012), C++ Numerical Programming.