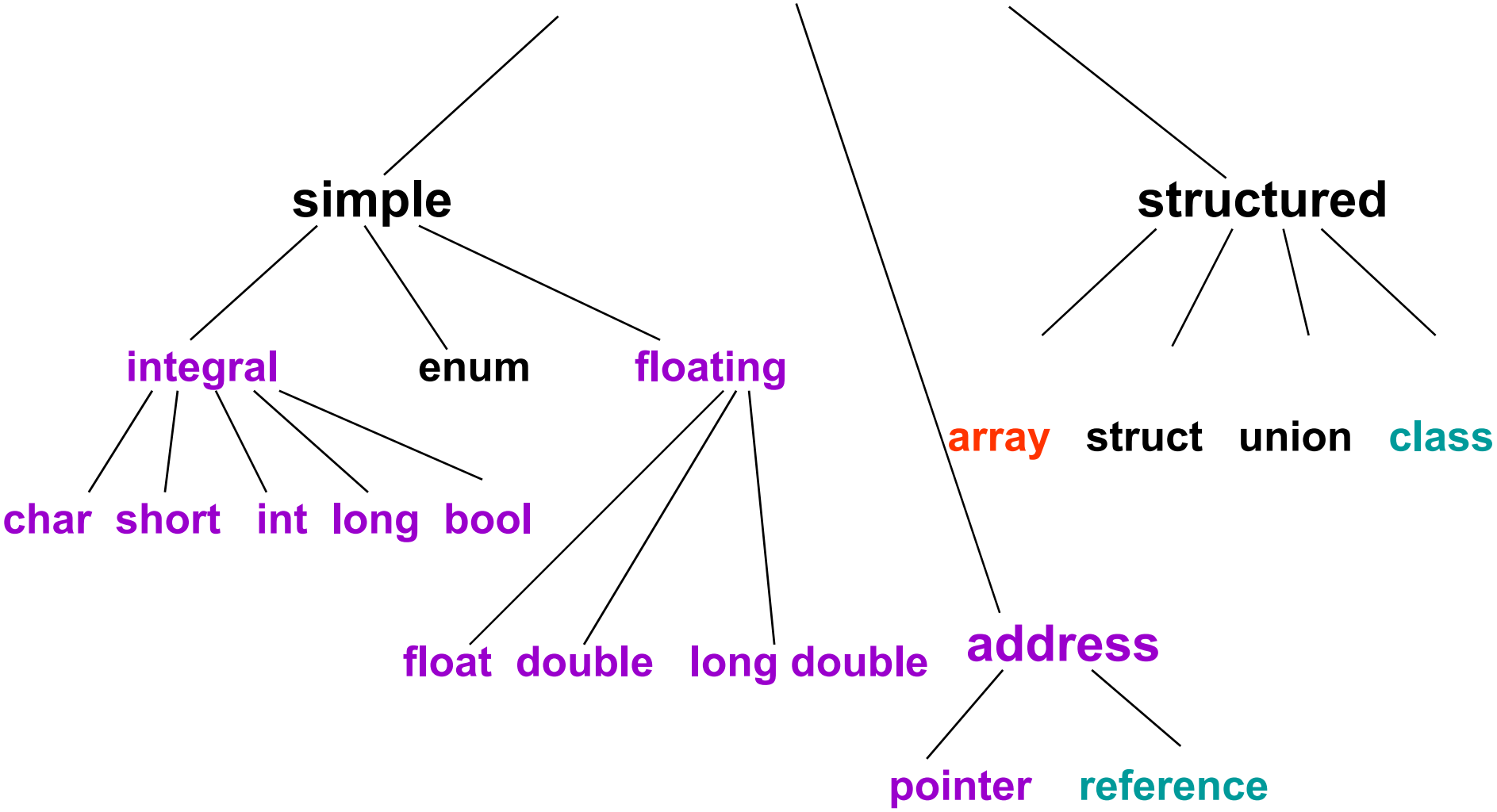
Chapter 6
Numeric Arrays

C++ Data Types

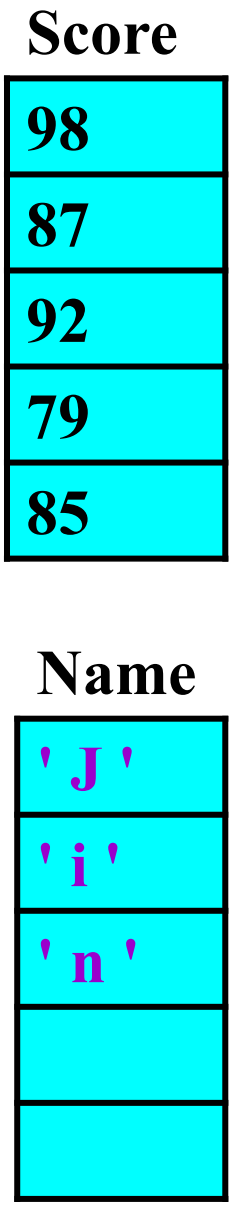

simple
structured
integral
enum
floating
array
struct
union
class
char
short
int
long
bool
float
double
long double
address
pointer
reference

6.1 *Array* (Page 270)

- An array is a data structure grouping of **same type** data continuously

  Score: **98 , 87 , 92 , 79 , 85** **integer**

  Name: **' J ' , ' i ' , ' n '** **char**



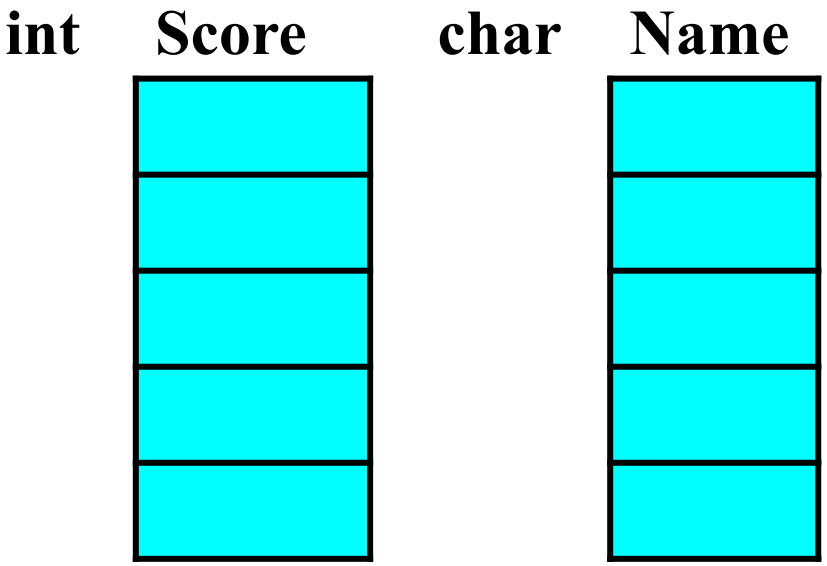
One-Dimensional Array

- To declare a one-dimensional(1D) array

  ElementType ArrayName [NumberOfElements];

  int Scores[5];

  char Name[5];

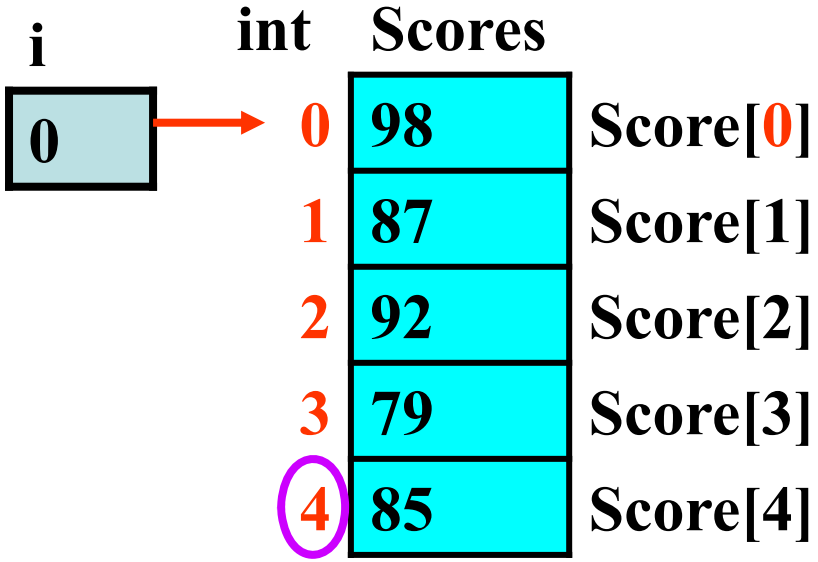


Index (subscript) (Page 274)

- ***Index*** is an integer to designate a particular element in the array
- The first ***Index*** is **0**
- Thus ***size-1*** is the maximum ***Index***.

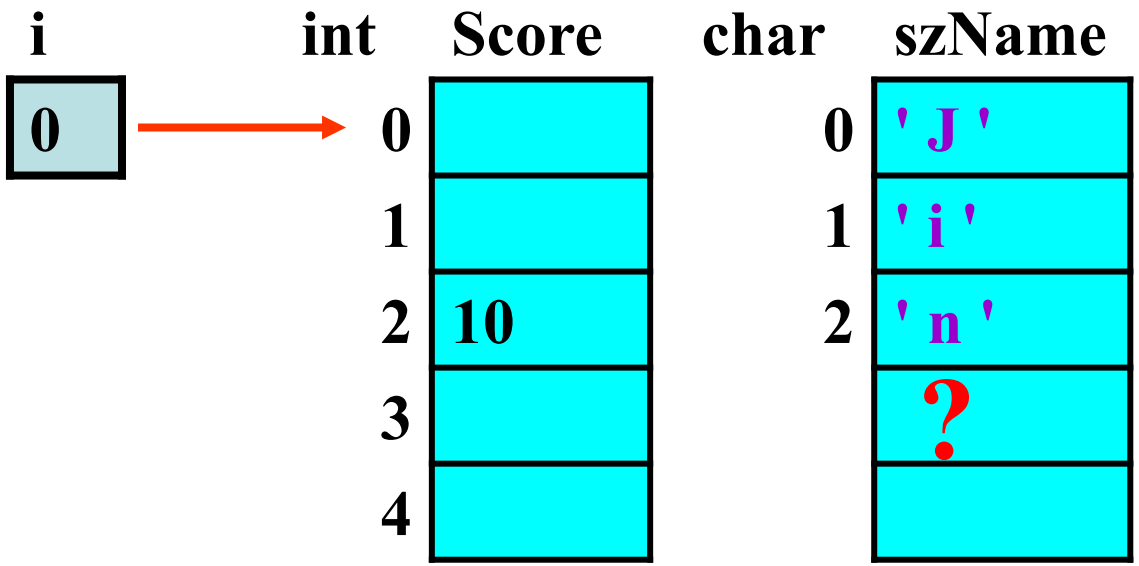
int Scores[5];

Scores[5]=???



1-D Array Data(Page 275)

```
int     Score[5];
char    szName[5];
  Score[2]=10;
  for(i=0;i<=9;i++)
     Score[i] =i;
  szName[0]='J';
  szName[1]='i';
  szName[2]='n';
```

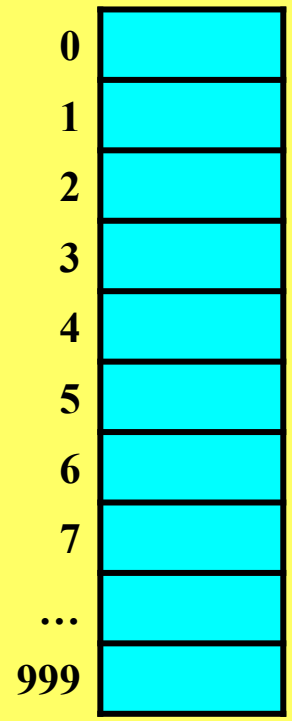


printf("Score[2]=%d,szName[0]=%c\n",Score[2],szName[0]);

printf("My name is %s\n",**szName**);???

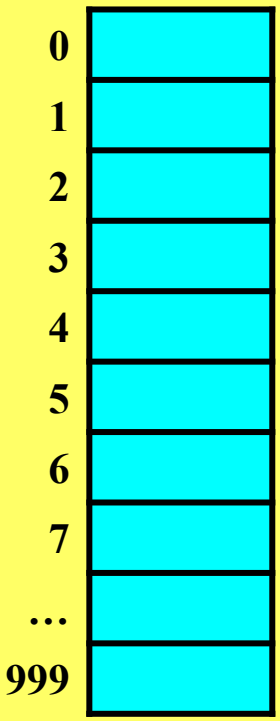
Try:Generate 1000 Random Numbers

```
#include <stdlib.h>
#include <time.h>
int _tmain(int argc, _TCHAR* argv[])
{
int  Data???, nMin,nMax,i;
  srand((unsigned)time(NULL));
  nMin=1; nMax=10;
  for(i=0;i<1000;i++)
     Data???=(double)rand()/(RAND_MAX+1)*(nMax-nMin+1)+nMin;
  return 0;
}
```



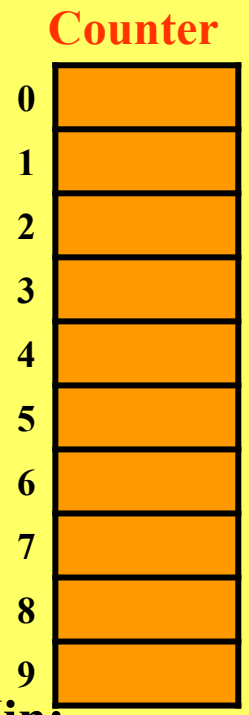
Try:Generate 1000 Random Numbers

```
#include <stdlib.h>
#include <time.h>
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[1000], nMin,nMax,i;
  srand((unsigned)time(NULL));
  nMin=1; nMax=10;
  for(i=0;i<1000;i++)
     Data[i]=(double)rand()/(RAND_MAX+1)*(nMax-nMin+1)+nMin;
  return 0;
}
```



Try:Counting Random Numbers

```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[1000], nMin,nMax,i,Counter[10]={?};
  srand((unsigned)time(NULL));
  nMin=1; nMax=10;
  for(i=0;i<1000;i++)
     Data[i]=(double)rand()/(RAND_MAX+1)*(nMax-nMin+1)+nMin;
     Counter[???]++;//???
  for(i=0;i<10;printf("Counter[%d]=%d\n",i,Counter[?]),i++);
  return 0;
}
```



```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[1000], nMin,nMax,i,Counter[10]={0,0,0,0,0,0,0,0,0,0};
  srand((unsigned)time(NULL));
  nMin=1; nMax=10;
  for(i=0;i<1000;i++)
  {
    Data[i]=(double)rand()/(RAND_MAX+1)*(nMax-nMin+1)+nMin;
     Counter[Data[i]-1]++;
  }
  for(i=0;i<10;printf("Counter[%d]=%d\n",i,Counter[i]),i++);
  return 0;
}
```

6.2 *Array Initialization* (Page 277)

```
int      Scores[5]={10,20,30,40,50};
double   y[10]={1.2,3.6      ?      };
int      z[?]={10,20,30};
char     MyName[5]={'J', 'i', 'n', '1'};
char     szName[5]={"Li"};
```

Scores

0	10	x[0]
1	20	x[1]
2	30	x[2]
3	40	x[3]
4	50	x[4]

y

0	1.2	y[0]
1	3.6	y[1]
2		y[2]
3		y[3]
4		y[4]

z

0	10	z[0]
1	20	z[1]
2	30	z[2]

MyName

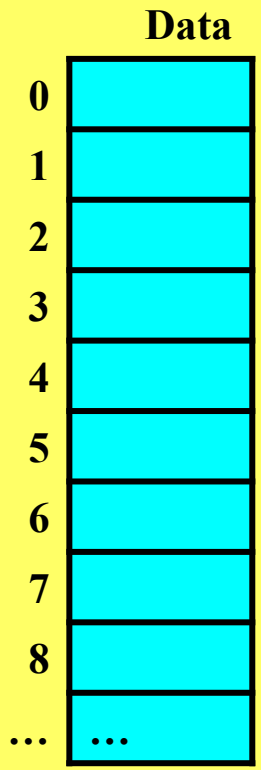
0	'J'	MyName[0]
1	'i'	MyName[1]
2	'n'	MyName[2]
3	'1'	MyName[3]
4	?	MyName[4]

szName

0	'L'	szName[0]
1	'i'	szName[1]
2	?	szName[2]
3		szName[3]
4		szName[4]

6.3 *Reading from File* (Page 282)

```
#include "stdio.h"
int _tmain(int argc, _TCHAR* argv[])
{
FILE   *fpReadFile;
int      Data[1000],i,nRet;
  fpReadFile=fopen("d:\\DataFile2.txt","r");
  i=0;
  do{
     nRet=fscanf(fpReadFile,"%d,",&Data[i]);
     i++;
  }while(nRet!=EOF);
  fclose(fpReadFile);
  return 0;
}
```



Home Work

(1) Generate 1000 random alphabetic characters(mixd of 'a'-'z' or 'A'- 'Z') in an array ***Data***.

(2) Generate 100 lines of random addition calculations with 2 digitals(such as 38+24=), and write these calculations to a file.

(3) Page 287-3

6.4 *Two-Dimensional Array* (Page 288)

- A two-dimensional array consists of both rows and columns of elements.

int **Scores1[3][4];**

Scores1

8	16	5	92
3	15	27	6
14	15	2	10

Scores1

		0	1	2	3
Scores1[0]	0	Score[0][0]	Score[0][1]	Score[0][2]	Score[0][3]
Scores1[1]	1	Score[1][0]	Score[1][1]	Score[1][2]	Score[1][3]
Scores1[2]	2	Score[2][0]	Score[2][1]	Score[2][2]	Score[2][3]

2-D Array Initialization (Page 293)

```
int   Scores1[3][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
int   Scores2[3][4]={{8,16,5,92},
                     {3,15,27 ?},
                     {14,15  ?  }};
int   Scores3[][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
int   Scores4[][4]={8,16,5,92,3,15,27,6,1      ?          };
   Scores1[0][0]=20;      Scores1[2][3]=Scores1[0][0];
   Scores1[3][4]=40;//???
```

Scores1

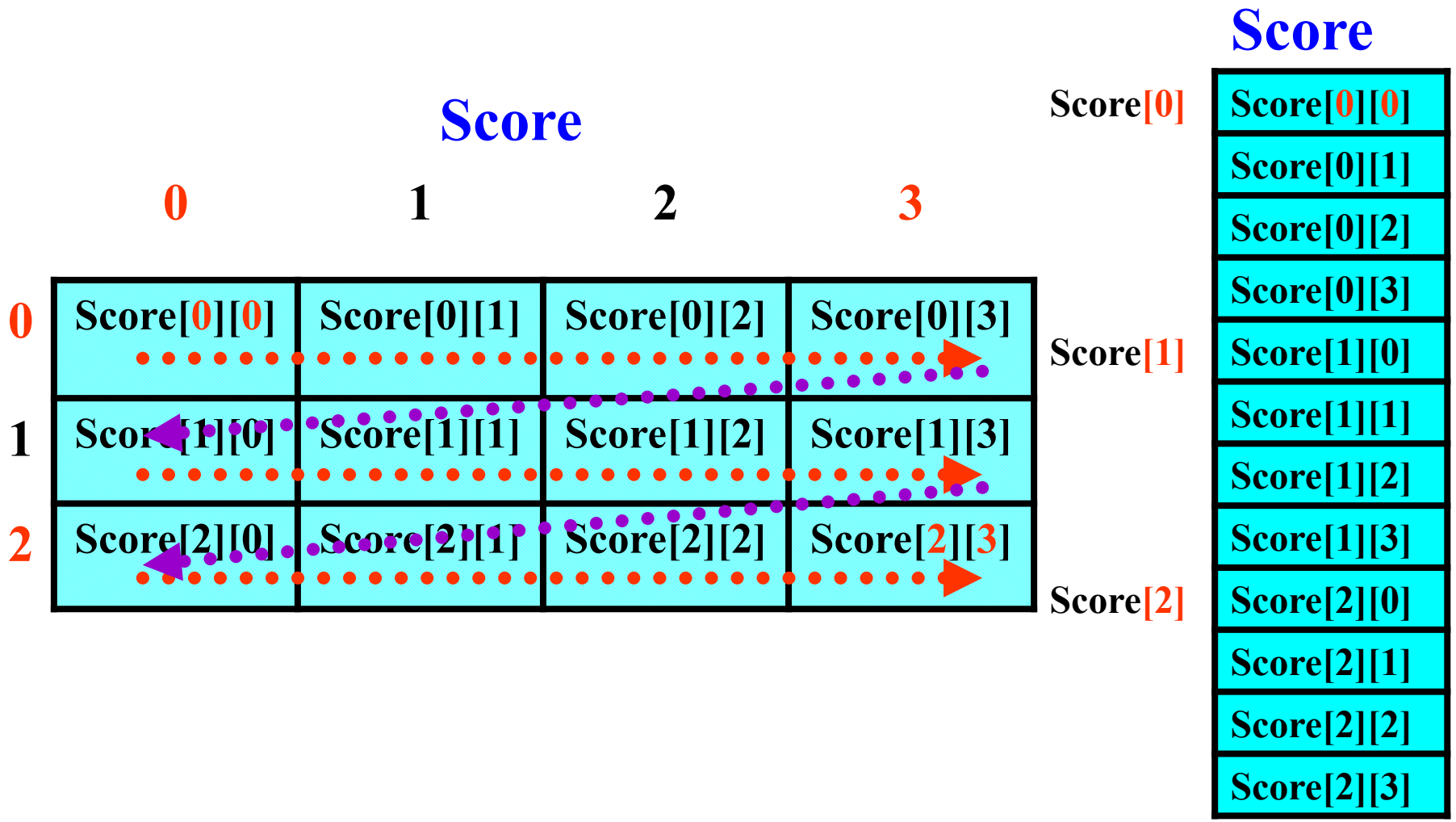
	0	1	2	3
0	8	16	5	92
1	3	15	27	6
2	14	15	2	10

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[])
{
int   Scores1[3][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
int   Scores2[3][4]={{8,16,5,92},
                     {3,15,27   },
                     {14,15     }};
int   Scores3[][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
  Scores1[0][0]=20;   Scores1[2][3]=30;
  Scores1[3][4]=40;//???
  for(int i=0;i<3;i++)
  {
     for(int j=0;j<4;j++)
        printf("%4d",Scores1[i][j]);
     printf("\n");
  }
  return 0;
}
```

Scores1

	0	1	2	3
0	8	16	5	92
1	3	15	27	6
2	14	15	2	10

Stored Contiguously in Increasing Memory Locations(Page 294)



Home Work

Write Programs:

1. Using 2-D array initialization to assign a two dimensional array as follows.

Num

1	0	0	0	0
1	3	0	0	0
1	3	5	0	0
1	3	5	7	0
1	3	5	7	9

3. Sum the rows.
4. Sum the columns.
5. Sum all elements
6. Change low triangle into up triangle.
7. Print the array.

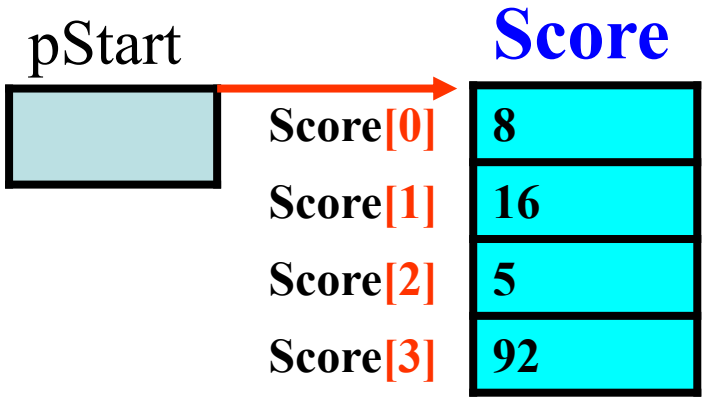
Home Work

1. Page 297-3
2. Page 297-5

1-D Array Name as Pointer

- Array name is an address of the array

```
int   Score[4]={8,16,5,92};
int   *pStart,x,y;
      pStart=Score;
      x=*pStart;
      pStart++;
      y=*pStart;
```



- Array element can be expressed by pointer

```
      pStart=&Score[2];
      x=*pStart;
      y=*(pStart+1);
```

6.5 Functions and Arrays(Page 298)

- Passing individual array elements to functions
- Passing entire arrays to functions
- Access an array in a function
- Ex:

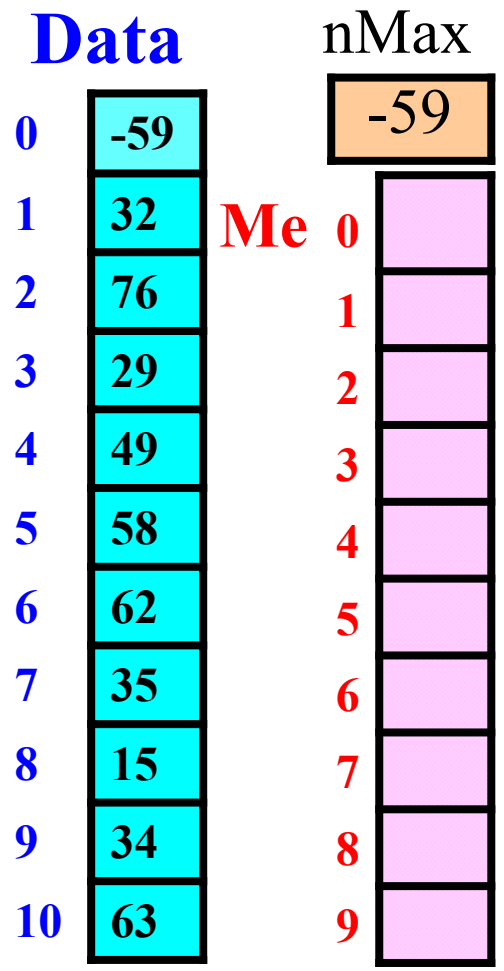
Generate10 random numbers in Data[1] to Data[10] and put the maximum in Data[0]

```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[11];
  srand((unsigned)time(NULL));
  for(int i=1;i<11;i++)
  {
     Data[i]=rand();
     printf("%d\n",Data[i]);
  }
  return 0;
}
```

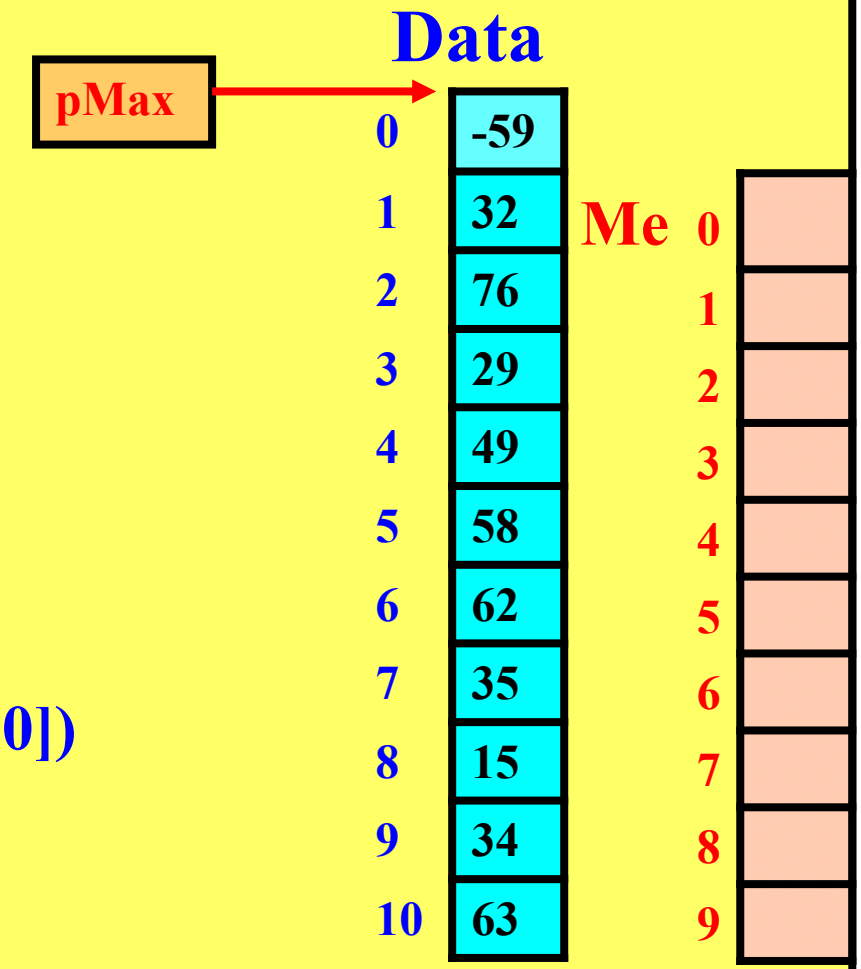
Data

0	-59
1	32
2	76
3	29
4	49
5	58
6	62
7	35
8	15
9	34
10	63

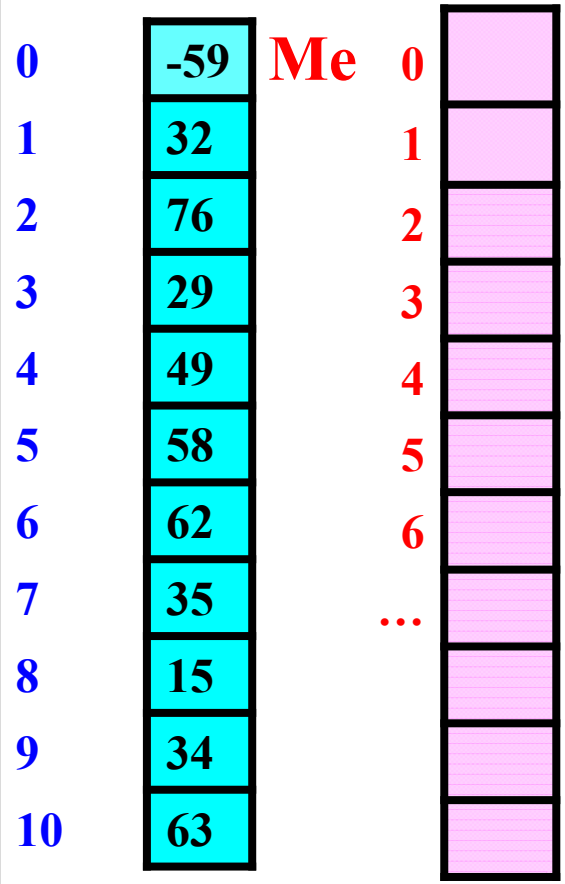
```
void FindMax1(int nMax,int Data[10]);
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[11];
…
  FindMax1(Data[0],&Data[1]);
  printf("Max1=%d\n",Data[0]);
  return 0;
}
void FindMax1(int nMax,int Me[10])
{
  nMax=-1;
  for(int i=0;i<10;i++)
     nMax=(nMax>=Me[i])?nMax:Me[i];
}
```



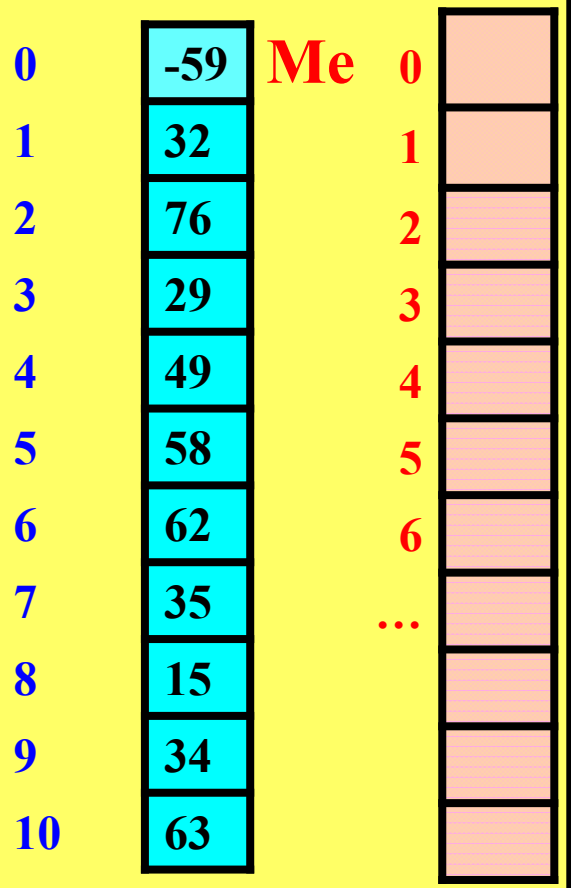
```
void FindMax2(int *pMax,int Data[10]);
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[11];
…
  FindMax1(Data[0],&Data[1]);
  FindMax2(&Data[0],&Data[1]);
  printf("Max2=%d\n",Data[0]);
  return 0;
}
void FindMax1(int nMax,int Data[10])
{
…
}
void FindMax2(int *pMax,int Me[10])
{
  *pMax=-1;
  for(int i=0;i<10;i++)
     *pMax=(*pMax>=Me[i])?*pMax:Me[i];
}
```



```
void FindMax3(int Data[]);
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[11];
…
  FindMax2(&Data[0],&Data[1]);
  FindMax3(&Data[0]);
  printf("Max3=%d\n",Data[0]);
  return 0;
}
…
void FindMax2(int *pMax,int MyData[10])
{
…
}
void FindMax3(int Me[])
{
  Me[0]=-1;
  for(int i=1;i<11;i++)
     Me[0]=(Me[0]>=Me[i])?<Me[0]:Me[i];
}
```



```
void FindMax3(int Data[]);
int _tmain(int argc, _TCHAR* argv[])
{
int  Data[11];
…
  FindMax3(&Data[0]);
  printf("Max3=%d\n",Data[0]);
  FindMax3(Data);
  printf("Max4=%d\n",Data[0]);
  return 0;
}
…
void FindMax3(int Me[])
{
  Me[0]=-1;
  for(int i=1;i<11;i++)
     Me[0]=(Me[0]>=Me[i])?Me[0]:Me[i];
}
```



Passing Array as Parameter

- When we passed simple variables between functions, we had only two choices
  - ***Passing by Value***
    - copy the value of the simple variable
  - ***Passing by Address***
    - pass the address of the variable

Passing by Value(Page 300)

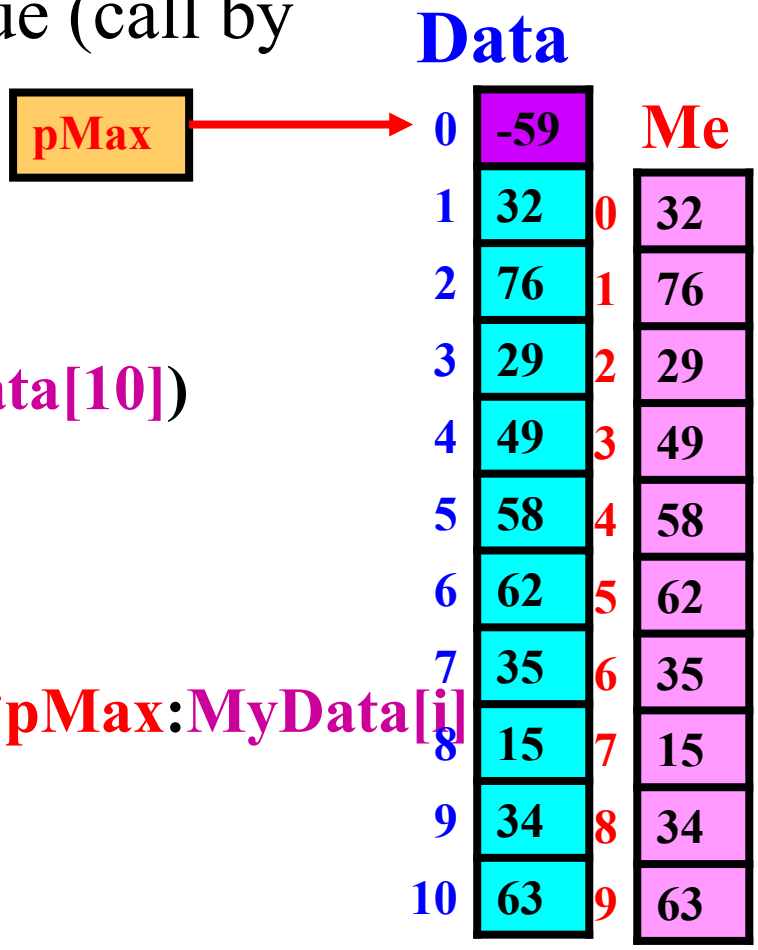
- To pass single array element to a function, we treat it just like that of simple variable
- You **can't** modify the original variable's value

```
FindMax1(Data[0],&Data[1]);

…
void FindMax1(int nMax,int Me[10])
{
   nMax=-1;
   for(int i=0;i<10;i++)
      nMax=(nMax>=Me[i])?nMax:Me[i];
}
```

Passing by Address(1) (Page 301)

- In passing the address, we allowed the function to modify the original variable's value (call by reference)

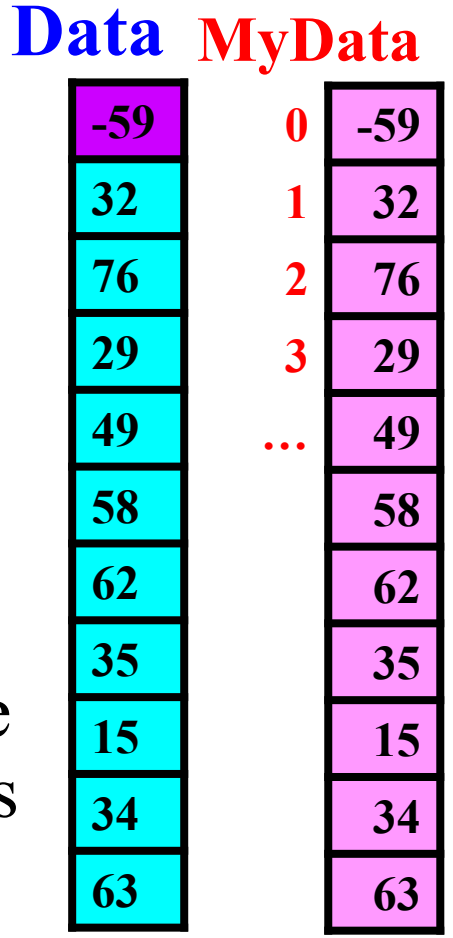
```
FindMax2(&Data[0],&Data[1]);
…
void FindMax2(int *pMax,int MyData[10])
{
   *pMax=-1;
   for(int i=0;i<10;i++)
     *pMax=(*pMax>=MyData[i])?*pMax:MyData[i];

}
```



Passing by Address(2) (Page 301)

- Array name is the same as the address of the first element of an array itself

```
FindMax3(Data);
…
void FindMax3(int MyData[])
{
  MyData[0]=-1;
…
}
```



- To pass an entire 1D array to a function, we need only to pass the first element's address to a function

Passing by Address(3) (Page 303)

- In some cases, we just want the function to read the array, we can use ***const***

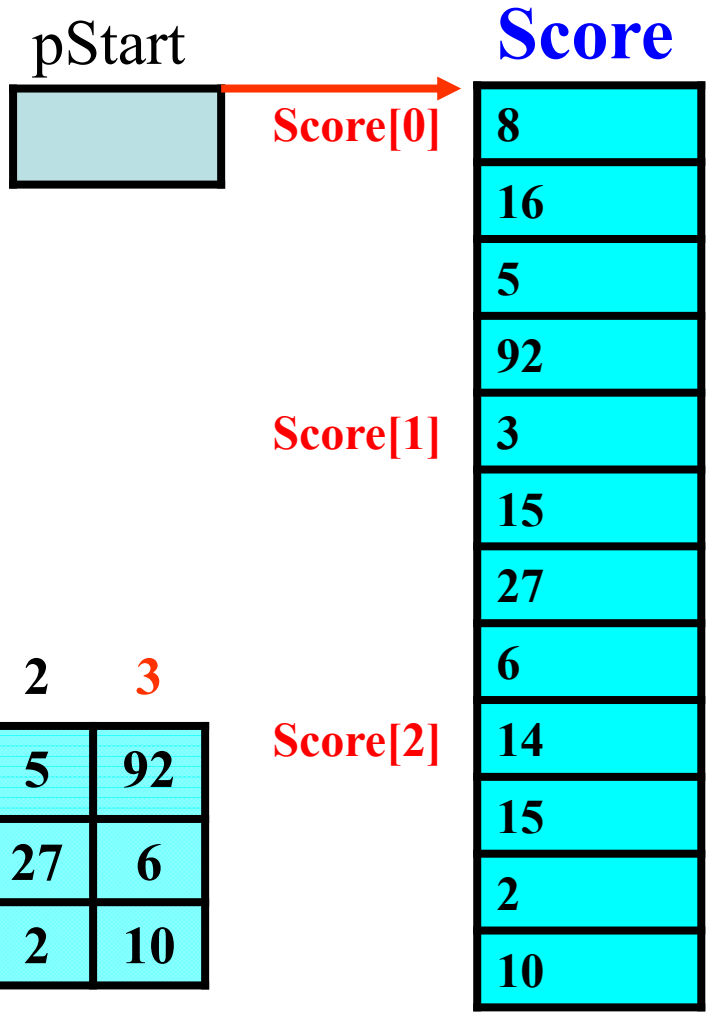
```
void FindMax4(const int Data[])
{
  Data[0]=-1;//???
  for(int i=1;i<11;i++)
    Data[0]=(Data[0]>=Data[i])?Data[0]:Data[i];
}
```

- Using ***const*** qualifier, C will indicate an error if the function attempt to modify the array

2-D Array Name as Pointer

- Array name is an address of the array

```
int   Score[3][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
int   *pStart,x[10];
      pStart=(int *)Score;
      x[0]=*pStart;
      pStart++;
      x[1]=*pStart;
```



Score

	0	1	2	3
0	8	16	5	92
1	3	15	27	6
2	14	15	2	10

To Pass Multidimensional Array to a Function

- During function call, only array name is required
- In function prototype/definition, except for the first pair of brackets, all of the brackets should be filled with their declared sizes.

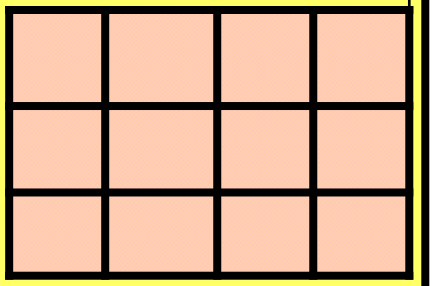
```
#include "stdafx.h"
int Sum1(int Data[3][4]);
int _tmain(int argc, _TCHAR* argv[])
{
int   Scores[3][4]={{8,16,5,92},
                    {3,15,27,6},
                    {14,15,2,10}};
  for(int i=0;i<3;i++)
  {
    for(int j=0;j<4;j++)
      printf("%4d",Scores[i][j]);
    printf("\n");
  }
int   nSum;
  nSum=Sum1(Scores);
  printf("nSum1=%d\n",nSum);
  return 0;
}
```

Scores

	0	1	2	3
0	8	16	5	10
1	3	15	27	6
2	14	15	2	92

```
int Sum1(int Data[3][4])
{
int  nSum=0;
  for(int i=0;i<3;i++)
    for(int j=0;j<4;j++)
      nSum+=Data[i][j];
  return nSum;
}
```

Data



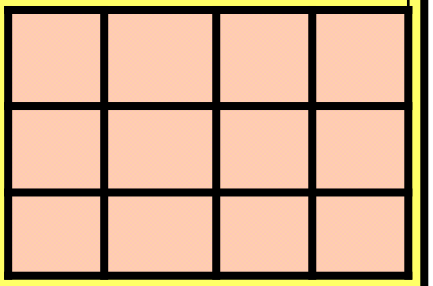
```
#include "stdafx.h"
int Sum1(int Data[3][4]);
int Sum2(int Data[][4]);
int _tmain(int argc, _TCHAR* argv[])
{
int   Scores[3][4]={{8,16,5,92},
                    {3,15,27,6},
                    {14,15,2,10}};
…
int   nSum;
  nSum=Sum1(Scores);
  printf("nSum1=%d\n",nSum);
  nSum=Sum2(Scores);
  printf("nSum2=%d\n",nSum);
  return 0;
}
int Sum1(int Data[3][4])
{
…
}
```

Scores

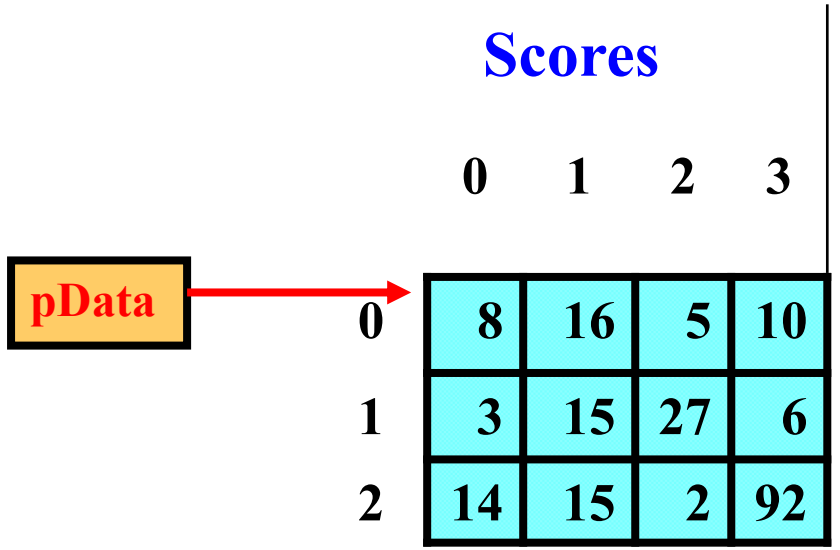
	0	1	2	3
0	8	16	5	10
1	3	15	27	6
2	14	15	2	92

```
int Sum2(int Data[][4])
{
int  nSum=0;
  for(int i=0;i<3;i++)
     for(int j=0;j<4;j++)
        nSum+=Data[i][j];
  return nSum;
}
```

Data



```
#include "stdafx.h"
int Sum1(int Data[3][4]);
int Sum2(int Data[][4]);
int Sum3(int *pData);
int _tmain(int argc, _TCHAR* argv[])
{
…
int   nSum;
…
  nSum=Sum3(&Scores[0][0]);
  printf("nSum3=%d\n",nSum);
  return 0;
}
int Sum1(int Data[3][4])
{
…
}
int Sum2(int Data[][4])
{
…
}
```




```
int Sum3(int *pData)
{
int  a,nSum=0;
  for(int i=0;i<3*4;i++)
  {
     a=*pData++;
     nSum+=a;
  }
  return nSum;
}
```

```
#include "stdafx.h"
…
int SumOneLine(int Line[4]);
int _tmain(int argc, _TCHAR* argv[])
{
…
int   nSum;
…
  nSum=SumOneLine(Scores[2]);
  printf("SumOneLine=%d\n",nSum);
  return 0;
}
int Sum1(int Data[3][4])
{…
}
int Sum2(int Data[][4])
{…
}
int Sum3(int *pData)
{…
}
```

Scores

		0	1	2	3
Scores[0]	0	8	16	5	10
Scores[1]	1	3	15	27	6
Scores[2]	2	14	15	2	92

```
int SumOneLine(int Line[4])
{
int a,nSum=0;
  for(int i=0;i<4;i++)
  {
     a=Line[i];
     nSum+=a;
  }
  return nSum;
}
```

Line

0	14
1	15
2	2
3	92

Home Work

1. Page 305-2

Note:

The program will include a function ***GetWeight1()*** which has **3** input parameters of 1-D array(Length, Width and Thickness) and second function ***GetWeight2()*** which has only **1** input parameter of 2-D array. The functions will return weight.