Chapter 06. Arrays and pointers

목차

학습목표

- Learn about the data set which is an array of the same type.
- The one-dimensional array and the relationship between the pointers.
- The study of various types of two-dimensional pointer.

01 Understanding of the array

A one-dimensional array

- The array is called to store multiple data, and the elements of the array a respective value stored in the array (element).

배열 기본 형식

Array data type name [number of elements];

Data type for the value stored in each element

int a [5] ;

The number of array elements

Array name

[Picture 6-1] For the structure of the array declaration

a[0]	a[1]	a[2]	a[3]	a[4]

[Picture 6-2Allocation of memory arrays

배열의 원소 개수 선언 형식

Name array [0] to array name [n-1]

01 Understanding of the array

Initialization of the one-dimensional array,

- The number and the number of initial values of the array is equal.

  int a[5] = {10, 20, 30, 40, 50};

- If the initial value specified in the array may be dispensed with the number of arrangement.

  int a[ ] = {10, 20, 30};

- If the initial value is smaller than the number of the array and fills the remaining area with zero.

  int a[5] = {10, 20, 30};

- If given away a lot of the initial value than the number of arrays, an error occurs.

  int a[5] = {10, 20, 30, 40, 50, 60, 70} // error

- In that not the initial value of the arrangement can be omitted, the number of elements in the array.

  int a[ ]; // error

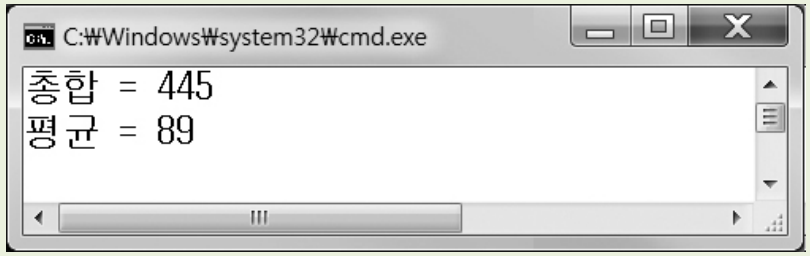
Example 6-2. Using an array to obtain the total and average (06_02.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a[5]={85, 90, 75, 100, 95};
    int tot=0;
    double avg;
    int i;

    for(i=0; i<5; i++)
        tot+=a[i];

    avg = (double)tot/5.0;

    cout << "총합 = " << tot <<"₩n";
    cout << "평균 = " << avg <<"₩n";
}
```



01 Understanding of the array

Two-dimensional array

- If you specify a two-dimensional array is added to only the number of rows and columns in the form of a two-dimensional array data types for declaring a one-dimensional array rather than in separately.

2차원 배열 선언 기본 형식

[Number of rows] data type array name [number of columns;

int a[3][4];

	0 Ten	1열	2열	3열
0열	a[0][0]	a[0][1]	a[0][2]	a[0][3]
1열	a[1][0]	a[1][1]	a[1][2]	a[1][3]
2열	a[2][0]	a[2][1]	a[2][2]	a[2][3]

a[1][2] =10;

[Picture 6-3] A two-dimensional array structure of the

01 Understanding of the array

Two-dimensional array initialization

```
int a[3][4] = { {90, 85, 95, 100}, // Initialization for the 0-th row
                {75, 95, 80, 90},    // 1번째 행에 대한 초기화
                {90, 80, 70, 60}     // 2번째 행에 대한 초기화
               };
```

- To use nested braces to the initial value of the two-dimensional array.

The output of the two-dimensional array using double for loops

```
int a[3][4];
```

=

```
cout <<"₩t" << a[0][0];
cout <<"₩t" << a[0][1];
cout <<"₩t" << a[0][2];
cout <<"₩t" << a[0][3];
cout <<"₩n";
cout <<"₩t" << a[1][0];
cout <<"₩t" << a[1][1];
:
:
cout <<"₩t" << a[2][2];
cout <<"₩t" << a[2][3];
```

=

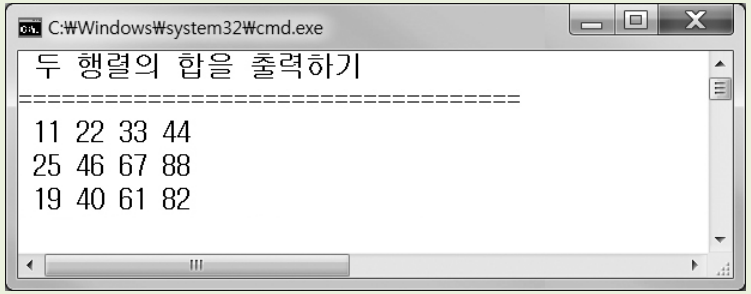
```
for(r= 0; r<3; r++) {
    for(c= 0; c<4; c++)
        cout<<"₩t"<<a[r][c];
    cout<<"₩n";
}
```

Example 6-5. Obtaining the sum of two matrices (06_05.cpp)

```
#include <iostream>
using namespace std;
void main()
{

    int a[3][4]={ {10, 20, 30, 40}, {20, 40 ,60 ,80}, {10, 30, 50, 70} };
    int b[3][4]={ { 1, 2, 3, 4}, { 5, 6, 7, 8}, { 9, 10 ,11 ,12} };
    int c[3][4];
    int row, col;
    for(row = 0; row < 3; row ++)
        for(col = 0; col < 4; col ++)
            c[row][col] = a[row][col]+ b[row][col];

    cout<<" 두 행렬의 합을 출력하기";
    cout<<"₩n==================================₩n";
    for(row = 0; row < 3; row ++){
        for(col = 0; col < 4 ; col ++)
            cout<<" "<<c[row][col];
        cout<<'₩n';
    }
}
```



02 The one-dimensional arrays and pointers

Address of the array element

- Address of the array element is a stick and a subscript in the array to the specified element.

```
int a[10];
== & Address of the array element [2]
```

An array pointer name

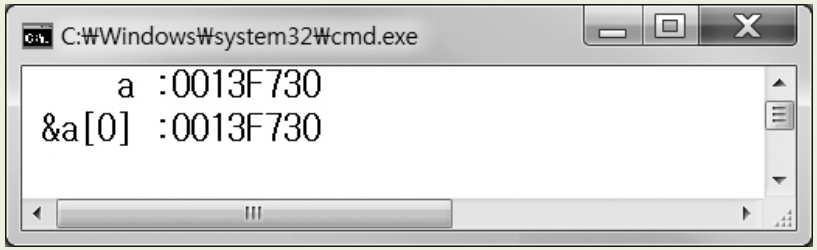
- Simply array technology, people interpreting it as the start address value, that is a pointer to an array.

```
int a[10];
배열의 시작 주소값 == a == &a[0]
```

Example 6-8. Who arranged to output values (06_08.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a[5] = {10, 20, 30, 40, 50};

    cout << "      a :" << a << "₩n";
    cout << " &a[0] :" << &a[0] << "₩n";
    cout << " &a[1] :" << &a[1] << "₩n";
}
```



02 The one-dimensional arrays and pointers

Array and pointer arithmetic

- Pointer is array names indicating the start address of the array

```
Start address of the array == a == &a[0]
```

- * A stick in front of the array, as people visit the * operator that address indicates the value stored in its memory space. That is, attaching the array name * operator informs the first element value in the starting address of the array.

```
*a == *(&a[0]) == a[0]
```

- Subscript i is the address of the array element a [i] might put a & operator in front but i may be obtained in addition to the array of people, such as a + i.

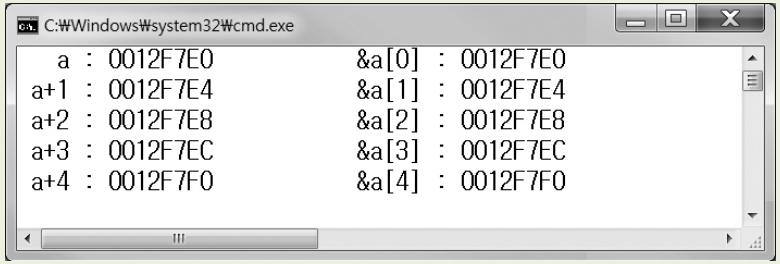
```
a+i == &a[i];
```

[Table 6-1] Arithmetic operators that can be used for pointer

Operator	Meaning	result
+	It indicates the address of the next element to come out.	Address (pointer)
-	It indicates the address of the previous element.	Address (pointer)

Example 6-9. Learn array name and relationship of the + operator (06_09.cpp)

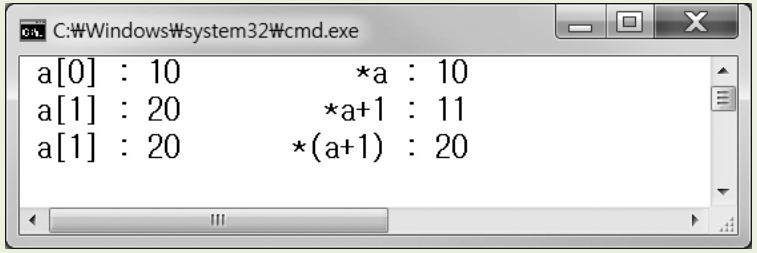
```
#include <iostream>
using namespace std;
void main()
{
    int a[5] = {10, 20, 30, 40, 50};
    cout<<" a : "<< a <<" ₩t &a[0] : "<<&a[0]<<"₩n" ;
    cout<<" a+1 : "<<a+1 <<" ₩t &a[1] : "<<&a[1]<<"₩n" ;
    cout<<" a+2 : "<<a+2 <<" ₩t &a[2] : "<<&a[2]<<"₩n" ;
    cout<<" a+3 : "<<a+3 <<" ₩t &a[3] : "<<&a[3]<<"₩n" ;
    cout<<" a+4 : "<<a+4 <<" ₩t &a[4] : "<<&a[4]<<"₩n" ;
}
```



Example 6-10. Outputting each value in the array elements using pointers (06_10.cpp)

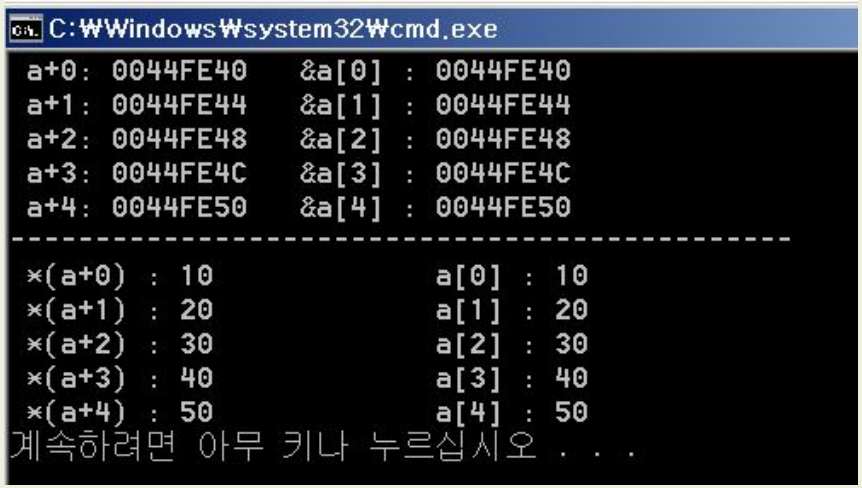
```
#include <iostream>
using namespace std;
void main()
{
    int a[5] = {10, 20, 30, 40, 50};

    cout<<" a[0] : "<<a[0]<<" ₩t *a : "<< *a<<"₩n";
    cout<<" a[1] : "<<a[1]<<" ₩t *a+1 : "<< *a+1<<"₩n";
    cout<<" a[1] : "<<a[1]<<" ₩t *(a+1) : "<<*(a+1)<<"₩n";
}
```



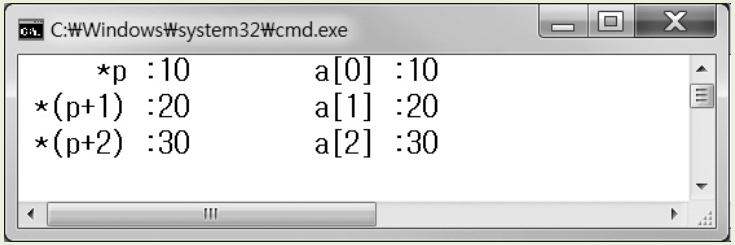
Example 6-11. Form elements of the array as a pointer value output (06_11.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a[5] = {10, 20, 30, 40, 50};
    int i;
    for(i=0;i<5;i++)
        cout << " a+" << i << ": " << a + i << "₩t &a[" << i << "] : " << &a[i] << "₩n";
    cout<<"--------------------------------------------₩n";
    for(i=0;i<5;i++)
        cout << " *(a+" << i << ") : " << *(a + i) << "₩t₩t a[" << i << "] : " << a[i] << "₩n";
}
```



Example 6-12. People know the relationship array pointer variable view (06_12.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a[5] = {10,20,30,40,50};
06     int *p;                          // 포인터 변수 선언
07     p = a;                           // 포인터 변수 초기화
08
09     cout << "₩n *p :" << *p;
10     cout << "₩t a[0] :" << a[0];
11
12     cout << "₩n *(p+1) :" << *(p+1);
13     cout << "₩t a[1] :" << a[1];
14
15     cout << "₩n *(p+2) :" << *(p+2);
16     cout << "₩t a[2] :" << a[2] << "₩n";
17 }
```



03 Two-dimensional pointer

① A pointer to the pointer (two-dimensional pointer)

- **<u>The one-dimensional pointer's pointer to a pointer to store the address of the declared twice to describe the * symbol.</u>**

포인터의 포인터 기본 형식

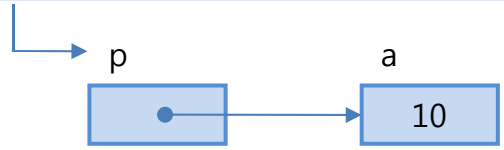
** Pointer data type variable name;

EX) int **pp; // Two-dimensional pointer

- Operators & related pointer *

int a = 10;
int *p = &a;

Pointer variable p is the address of where the value of a variable stored.

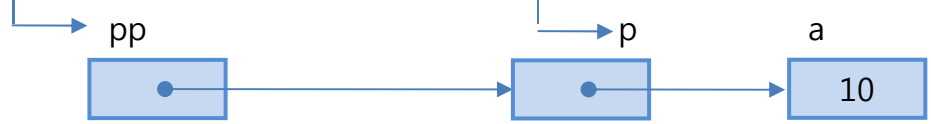


- Pointer's Pointer

int **pp = &p;

Pointer variable phpsms
The value in the pointer variable p
The address where the stored value.

P is a pointer variable
The value of the variable a
The address where the stored value.

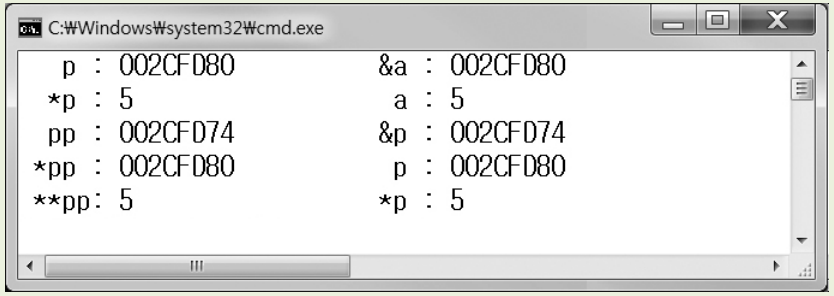


Example 6-13. Using two-dimensional pointer (06_13.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a=5;
    int *p;
    int **pp;

    p=&a;
    pp=&p;

    cout<<" p : "<< p <<" ₩t &a : "<< &a << endl;
    cout<<" *p : "<< *p <<" ₩t ₩t a : "<< a << endl;
    cout<<" pp : "<< pp <<" ₩t &p : "<< &p << endl;
    cout<<" *pp : "<< *pp <<" ₩t p : "<< p << endl;
    cout<<" **pp: "<< **pp <<" ₩t ₩t *p : "<< *p << endl;
}
```



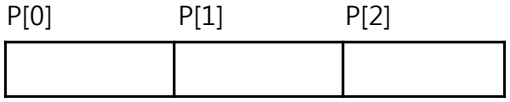
03 Two-dimensional pointer

② A pointer for storing the one-dimensional array of pointers

❶ int *p1, *p2, *p3;

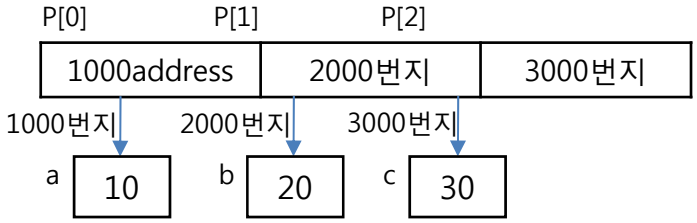
❷ int *p[3];

- Creating ❷ like and the array elements that can store three generated, each element may store a one-dimensional pointer.



- Let the stored one-dimensional pointer in the respective elements.

```
int a=10, b= 20, c=30;
int *p[3]={&a, &b, &c};
```



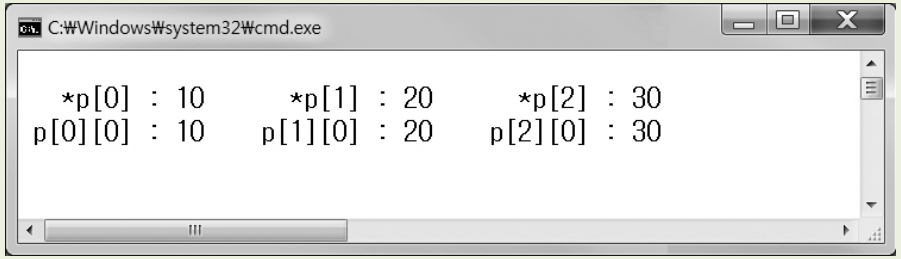
- Since the address information of each element is 10, 20, 30 to output should describe the * operator in front of each element.
- That is, the p [0] == & because a * (p [0]) == * p [0] == a.

Example 6-14. An array of pointers to store one-dimensional pointer (06_14.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a=10, b= 20, c=30; // 정수형 변수
    // 포인터 배열에 변수의 주소를 저장해 둔다.
    int *p[3]={&a, &b, &c};

    // 배열 원소에 * 연산자로 정수값을 얻어온다.
    cout<<"₩n *p[0] : "<< *p[0];
    cout<<"₩t *p[1] : "<< *p[1];
    cout<<"₩t *p[2] : "<< *p[2];

    // * 연산자 대신 [ ]로 정수값을 얻어온다.
    cout<<"₩n p[0][0] : "<< p[0][0];
    cout<<"₩t p[1][0] : "<< p[1][0];
    cout<<"₩t p[2][0] : "<< p[2][0];
    cout<<"₩n";
}
```



03 Two-dimensional pointer

- Let's start pointer array to store the address of the one-dimensional array expert.

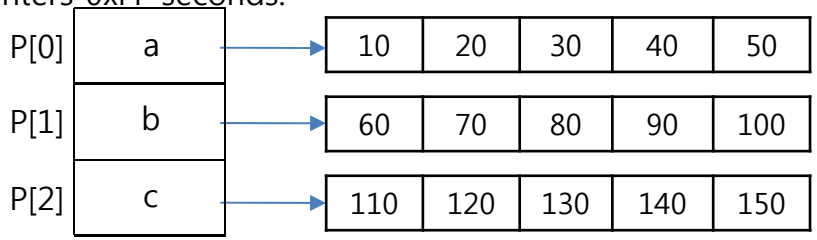
  Int a[5]={10, 20, 30, 40, 50};

  Int b[5]={60, 70, 80, 90, 100};

  Int c[5]={110, 120, 130, 140, 150};

  Int *p[3]={a, b, c};

- The array name itself gives the array name without the & operator because the pointer to an array of pointers 0xFF seconds.



- Because the content of each element are described when the address * operator, the first element of information, each one-dimensional array is output.

  p[0]==a==&a[0]이므로 *p[0]==*a==*(&a[0])==a[0]이 되고

  p[1]==b==&b[0]이므로 *p[1]==*b==*(&b[0])==b[0]이 되고

  p[2]==c==&c[0]이므로 *p[2]==*c==*(&c[0])==c[0]이 된다.

- * P [0], * p1], * pointer is equivalent to p2] is p [0] [0], p [1] [0], p [2] [0] it can be expressed as an array, as shown, if When outputting the second element of each one-dimensional array can be expressed as follows.
- p[0][1], p[1][1], p[2][1]

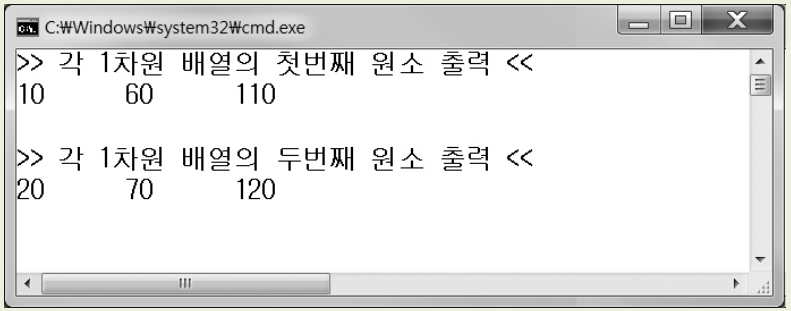
Example 6-15. Storing address values of a one-dimensional array in a pointer array (06_15.cpp)

```
#include <iostream>
using namespace std;
void main()
{
    int a[5]={ 10, 20, 30, 40, 50};
    int b[5]={ 60, 70, 80, 90, 100};
    int c[5]={110, 120, 130, 140, 150};

    int *p[3]={a, b, c};

    cout<<">> 각 1차원 배열의 첫번째 원소 출력 << ₩n";
    cout<<p[0][0]<<"₩t"<<p[1][0]<<"₩t"<<p[2][0]<<"₩n₩n";

    cout<<">> 각 1차원 배열의 두번째 원소 출력 << ₩n";
    cout<<p[0][1]<<"₩t"<<p[1][1]<<"₩t"<<p[2][1]<<"₩n";
}
```



03 Two-dimensional pointer

③ Two-dimensional arrays and pointer variables

- To print the address of the two-dimensional array element?
  - To print the address of array element is the subscript of the array elements and attaching a given.

  In other words, a [0] address value of [0] == & a [0] [0]

Name of array in a two-dimensional array

- * Append the operator twice, the first element of the array is displayed.

  **a==a[0][0];

- ex) 2-dimensional array "int a [3] [4]" Let's use an array pointer name operator +.
  - a + 1 is 16, the start address of the two-dimensional array (4x4) is larger address byte is found, a + 2 is obtained by the 32 (4x4x2) address bytes large. It should be increased by 16 bytes is the address line by line calculated. **Since the two-dimensional array of pointer is also a two-dimensional pointer is a pointer to a result of the add operation, the pointer is a two-dimensional pointer to calculate the addresses of the rows.**

03 Two-dimensional pointer

Two-dimensional array of pointer arithmetic

- Two-dimensional array "int a [b] [c]" is an array of pointers associated operator name *, it can be obtained using only the elements of the array +.

- a[r][c] == * (* (a + r) +c) **// After the line-by-line addition of a pointer value plus thermal units**

- 'a' is name of two-dimensional array , So 'a' is a two-dimensional pointer. *** (A + b) + c is calculated by the address of the location 4 bytes apart times (int) by c, based on the start address of the r-th row. * Operator to the calculated address again paste (* (* (a + r) + c)) to obtain the value of that location.**

Example 6-19. Outputting the elements of the array by using the pointer operator (06_19.cpp)

```
#include <iostream>
using namespace std;
#define ROW 3
#define COL 4
void main()
{
    int a[ROW][COL] = { {90, 85, 95, 100},
                        {75, 95, 80, 90},
                        {90, 80, 70, 60}
                       };
    int r, c;
    for(r=0; r<ROW; r++){
        for(c=0; c<COL; c++) {
            cout<<"*(*(a+"<<r<< ")+"<<c<<")):"<<*(*(a+r)+c)<<" ₩t";
        }
        cout<<"₩n";
    }
}
```

C:₩Windows₩system32₩cmd.exe

```
*(*(a+0)+0)):90        *(*(a+0)+1)):85        *(*(a+0)+2)):95        *(*(a+0)+3)):100
*(*(a+1)+0)):75        *(*(a+1)+1)):95        *(*(a+1)+2)):80        *(*(a+1)+3)):90
*(*(a+2)+0)):90        *(*(a+2)+1)):80        *(*(a+2)+2)):70        *(*(a+2)+3)):60
```

Homework

- **Chapter 6 Exercise: 15, 17, 18, 19, 20, 22, 24, 26**