Chapter 07. Advanced pointer

목차

1. Array used as a parameter of the function
2. Strings and pointers
3. A pointer to a function

학습목표

- You can create a function that array as a parameter.
- You can set relationship between the strings and pointers.
- Learn how to read a string from the command line.
- A pointer can be used in function.

01 Array used as a parameter of the function

A one-dimensional array that uses as a parameter of the function

- When you call a function, the element of the array is stored only once, so it passes the starting address of a one-dimensional array.

  Therefore, the parameter type is to be a one-dimensional pointer variable which declared to store the address.

Two-dimensional array used as a parameter of the function

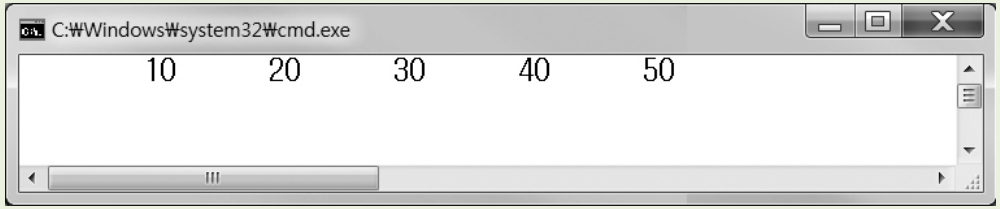
- Two-dimensional array of two-dimensional parameters not hand over the entire array, so as to pass only the starting address of the two-dimensional array.
- Thus, this address is given as the two-dimensional in the pointer.

포인터 변수 기본 형식

```
int (* p) [number of elements of the array;
```

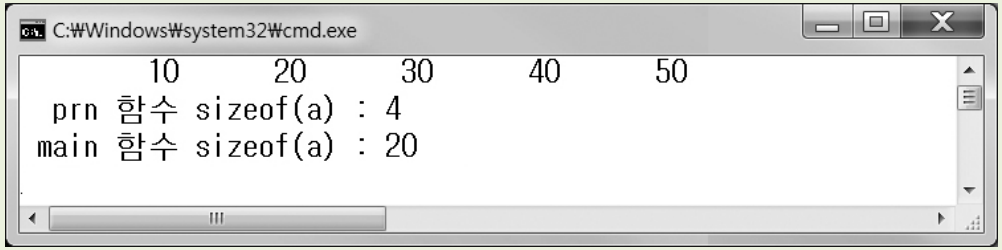
Example 7-1. Using the function to output the elements of the one-dimensional array (07_01.cpp)

```
#include <iostream>
using namespace std;
void prn(int *pa, int size);
void main()
{
    int a[5] = {10,20,30,40,50};
    prn(a, 5);
}
void prn(int *pa, int size)
{
    for(int i=0; i<size; i++) {
        cout<<"₩t"<<*(pa+i);    // pa[i];와 같이 표현할 수 있다.
    }
    cout<<"₩n";
}
```



Example 7-2. To represent the pointer to the array parameter (07_02.cpp)

```
#include <iostream>
using namespace std;
void prn(int a[], int size);
void main()
{
    int a[5] = {10,20,30,40,50};
    prn(a, 5);
    cout << " main 함수 sizeof(a) : "<< sizeof(a) << endl;
}
void prn(int a[], int size)          // void prn(int *a, int size)와 동일함
{
    for(int i = 0;i<size;i++)
        cout<<"₩t"<<a[i];
    cout<< endl;
    cout << " prn 함수 sizeof(a) : "<< sizeof(a) << endl;
}
```



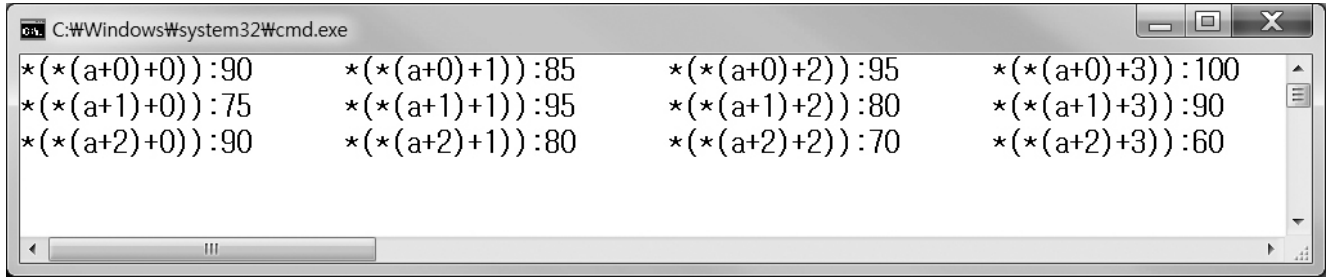
Example 7-3. Creating a function that received the two-dimensional array (07_03.cpp)

```
#include <iostream>
using namespace std;
#define ROW 3
#define COL 4

void prn(int (*p)[COL]);

void main()
{
    int a[ROW][COL] = { { 90, 85, 95, 100 },
                        { 75, 95, 80, 90 },
                        { 90, 80, 70, 60 }
                      };
    prn(a);
}

void prn(int (*p)[COL])          // 2차원 배열의 주소값을 전달할 포인터
{
    int r, c;
    for (r = 0; r < ROW; r++) {
        for (c = 0; c < COL; c++) {
            cout << "*(*(a+" << r << ")+" << c << ")):" << *(*(p + r) + c) << "    ";
        }
        cout << "₩n";
    }
}
```



02 Strings and pointers

String storage method

- Use the array.

  char str[10]="fox"; // An array of characters

- When stored in a character array to store memory is allocated directly to the string.

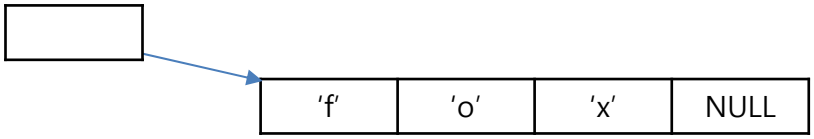
str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]	str[9]
'f'	'o'	'x'	NULL						

- The form of the pointer variable is stored.

  char *ptr="fox"; //Pointer Variables

- Pointer variable is the starting address of the string constants which are stored in memory.

ptr(String storing the starting address of the memory, the string is saved)



(Located somewhere in memory the string is stored)

- Ptr as a pointer variable that stores only the starting address can output the string as follows:
- cout << ptr;

02 Strings and pointers

The string constant to the pointer variable assignment

- Str is declared as a character array and "Apple" as the initial value.

```
char str[256]="Apple";  // When declaring a character array initialization
```

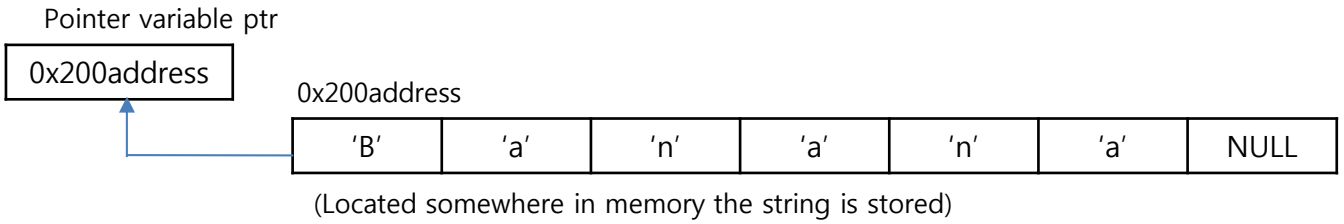
- The following is an example to assign a string constant to a character array that has already been declared.. It is in the character array stored in a string constant so as possible error occurs because the following compile error like.

```
str="Grapes";  //Compile error
```

02 Strings and pointers

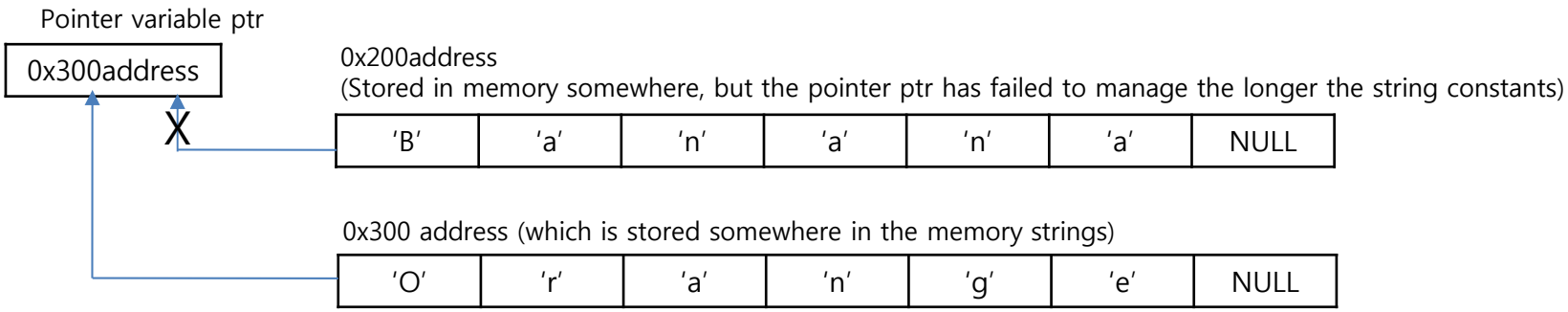
- **<u>To save a string constant as the variable operator must use a pointer.</u>** The following is declared as a pointer variable ptr it gave the "Banana" as the initial value.

```
char *ptr="Banana";  // When declaring a pointer variable initialization
```



- Substituting a string constant "Orange" in ptr after that ptr is not longer remember the start address of the string constant, "Banana" will point to the string constant "Orange".

```
ptr="Orange";
```



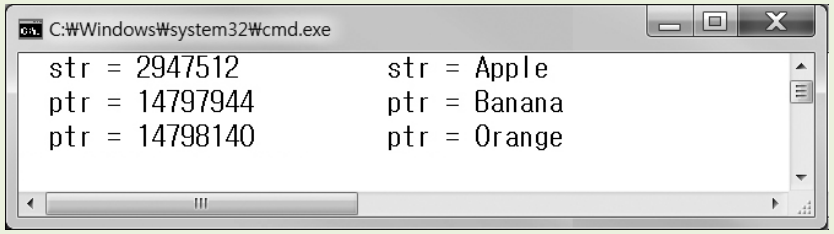
Example 7-6. Assignment to the pointer variable string (07_06.cpp)

```
#include <iostream>
using namespace std;
void main()
{
char str[256] = "Apple";
char *ptr = "Banana";

cout<<" str = "<<(int)str<<" ₩t str = "<<str<<"₩n";
cout<<" ptr = "<<(int)ptr<<"₩t ptr = "<<ptr<<"₩n";

// str="Grapes"; // 문자 배열은 다른 문자열 상수를 대입하지 못한다.

// 포인터 변수에는 다른 문자열 상수를 대입할 수 있다.
ptr = "Orange";
// 포인터 변수 ptr에 다른 주소가 저장되어 있다.
cout<<" ptr = "<<(int)ptr<<"₩t ptr = "<<ptr<<"₩n";
}
```



02 Strings and pointers

Passing the string with several parameters of the function

- Array of pointers to store multiple strings

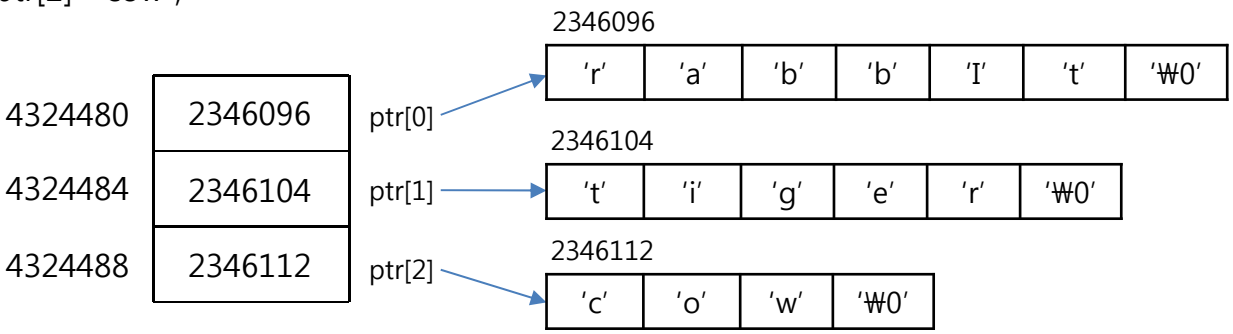
  char *ptr[3];

- Substituting a string constant to each element of the array elements of the array, and stores the starting address of the string constant.

  ptr[0]="rabbit";

  ptr[1]="tiger";

  ptr[2]="cow";

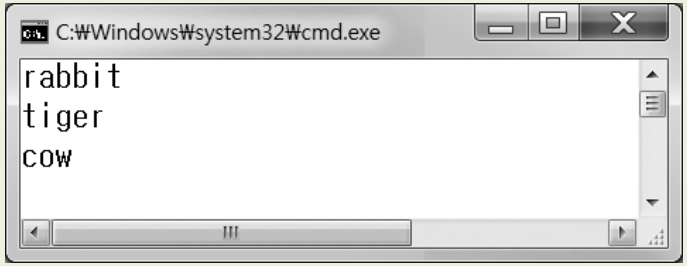


- Function that received the pointer array
  - Since the beginning of the address pointer is a two-dimensional array pointer type, parameter type of a function to the array as a parameter must be a pointer to a two-dimensional form. And also it should be to pass number for the element of the pointer array.

```
void print_string(char **pptr, int n)
```

Example 7-8. Function (07_08.cpp) using the pointer array as a parameter

```
#include <iostream>
using namespace std;
void print_string(char **pptr, int n)
{
    for(int i=0; i<n ; i++)
        cout<< pptr[i]<<" ₩n" ;     // *(pptr+i)
}
void main()
{
    char *ptr[3]={"rabbit", "tiger", "cow"};
    print_string(ptr, 3);
}
```



03 A pointer to a function

- The memory area is present in the code block for the data block for the variable source. When the program execution code for each function is stored in a specific address of the code block. When the function is called, it has a branch to the address returned by the function call statement. **Pointer variable for the function has a start address of the function stored in the memory area.**

```
int (*pf)(int);      // Function pointer declared
```

- pf is a pointer variable that holds the address of the function, the function pointed to by pf has a single integer parameter and returns the result as an int value.

- **technical function name to find out the address of the function. Let's say that abs () function to obtain the absolute value is defined as follows.**

```
int abs(int num)
{
  if(num<0)
    num=-num;
}
```

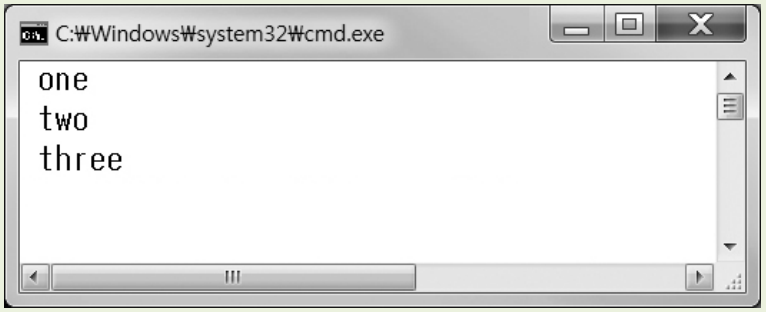
- After storing the address of a particular function to a function pointer, using the pointer to the function call is:
- `pf = abs;  // Function pointer`
- `int y = pf(-5);`

Example 7-9. Using a pointer to a function by function calls (07_09.cpp)

```
#include <iostream>
using namespace std;
/* 함수를 가리키는 포인터 변수 선언 */
void (*pf)(void);
void one()
{
    cout<<" one ₩n";
}
void two()
{
    cout<<" two ₩n";
}
void three()
{
    cout<<" three ₩n";
}
void main()
{
    pf = one;
    pf();

    pf = two;
    pf();

    pf = three;
    pf();
}
```



Homework

- **Chapter 7 Exercise: 2, 3, 4, 6**