7. Array, Pointer and Strings

7.1 One Dimensional Array (일차원 배열)

- 많은 수의 자료를 가진 집합을 다룰 때 사용

ex) int grade1, grade2, grade3 ;

→ int grade[3] ;

(1) declaration (선언)

- declaration (선언 방법)

```
type arrayname[size]   ;
```

- size and bound

```
int a[size] ;
lower bound = 0
upper bound = size  - 1
size  = upper bound  +  1
```

- #define 문의 활용

ex)

```
#define N 100
int i, a[N] ;
:
for(i=0;i<N;i++)
{
          :
}
```

(2) Initialization (초기화)

- storage class of array : automatic, external, static
- 초기화 시키지 않을 경우

  external,static → 0 으로 자동 초기화

  automatic → garbage
- 초기화 방법 :

  float f[5] = { 0.0, 1.0, 2.0, 3.0, 4.0} ;
- 초기화 시에 괄호안의 값이 모자라면, 그 나머지는 0으로 초기화 시키고, 너무 많은 값이 존재한다면 에러로 처리된다.

  ex) int a[100] = {1} ;
- 배열의 초기화 시에 빈괄호 ([]) 를 사용하면 컴파일러는 중괄호 내의 항목의 수를 세어 그 크기의 배열을 만들어 준다.

  ex) int a[] = { 2,3,5, -7}

  int a[4] = { 2,3,5, -7}

(3) Subscripting

- a[expr]

  expr : subscript or index
- 프로그램의 실행부에 배열의 첨자가 범위를 벗어나는 경우 C컴파일러는 에러를 내지 않는다. 에러의 결과는 예측이 불가능함.
- 배열의 첨자를 주어진 범위 안에서 사용하도록 주의를 기울여야 함.
- 함수기호 괄호 () 와 배열첨자 대괄호 [] 를 혼동하지 않도록 주의

7.2 Pointers

* C언어에서의 함수 호출 규칙 (Calling Convention in C -language) :

- Call by Value : 함수를 호출하면 인자들의 새로운 변수영역에 복사해 놓고 작업함
- Returns only single value : 함수는 한 개의 값 만을 리턴함

* 한 개가 아닌 여러 개의 값 을 리턴하는 함수를 만들려면 다른 메커니즘이 필요함.

→ Pointer

* variable value and variable's address : 주소연산자 (address operator)

```
int v ;
v → variable value
&v → variable's address
```

ex)

```
#include <stdio.h>
void main()
{
        int a ;
        a = 1 ;
        printf("%d %p \ n", a, &a) ;
}
```

(결과)

```
1 FFDC
```

* Pointer Variable:

- 어떤 값이 아니라 주소를 저장하기 위한 변수

ex)

```
int a ;
int *p ;
p = &a ;
```

(포인터 변수 p에 변수 a의 주소가 들어감)

* Dereferencing operator (간접연산자) : *

- 포인터 변수 앞에 * 연산자가 오면 그 포인터가 가리키는 값을 나타낸다.

```
ex)  int a,b *p ;   (1)
      a = 1 ;       (2)
     p = &a;        (3)
     b = *p ;       (4)
```

→ b 라는 변수에 포인터 p 가 가리키는 값 (a의 값) 이 들어감

(결과 분석)

	a	b	p
주소	11111	11112	11113
값 (1)	garbage	garbage	garbage
값 (2)	1	garbage	garbage
값 (3)	1	garbage	11111
값 (4)	1	1	11111

(see figures on page 249)

* 포인터 변수의 선언 방법

type *pointername ;

여기서 형명을 명기하는 것은 매우 중요함.

향후 배열과 연관되어 포인터 연산 시 매우 중요

* 포인터 변수의 초기화

250 page 예제 참조) BOX

* ANSI C convention

- assignment of different type of pointer is not allowed
- only void type is allowed for any type of pointer

→ 251 page Box

* Illegal expresion

- do not point at constant

ex) &3

- do not point at ordinary expression

ex) &(k+33)

- do not point at register variables

```
ex) register v ;
        p = &v ;
```

7.3 Call -by -Reference

* C언어에서의 함수 호출 규칙 (Calling Convention in C -language) :

- Call by Value : 함수를 호출하면 인자들의 새로운 변수영역에 복사해 놓고 작업함
- Returns only single value : 함수는 한 개의 값 만을 리턴 함

* 한 개가 아닌 여러 개의 값 을 리턴 하는 함수를 만들려면 다른 메커니즘이 필요함.

→ Pointer

* Pointer를 이용하여 함수에 변수들의 주소 값을 넘겨준 후 주소에 해당하는 값에 어떤 operation을 가한다면 함수를 호출한 원래 변수의 값이 변경이 될 수 있다.

ex)

```
#include <stdio.h>
void swap(int *, int *) ;
int main(void)
{
        int i=3, j=5 ;
        swap(&i,&j) ;
        printf(“%d %d \  n”, i,j) ;
        return 0 ;
}

void swap (int *p, int *q)
{
        int tmp ;
        tmp = *p ;
        *q = *p ;
        *p = tmp ;
}
```