2. 포인터

[학습목표]

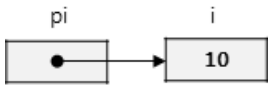
- 포인터의 선언 및 초기화
- 배열, 주소상수, 포인터
- 배열 포인터와 포인터배열
- 함수인자로서의 배열 포인터
- 함수포인터와 함수포인터배열

2.1. Summary

포인터의 선언과 초기화 방법

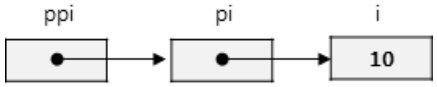
단일 포인터 선언과 역참조

```
1   int i = 10;
2   int* pi = &i;       // 단일 포인터 pi의 선언과 초기화
3                       // pi는 int형 변수 i의 주소를 저장
    *pi = *pi + 10;     // *pi는 변수 i를 의미함, 이 때의 *는 역참조 연산자
```



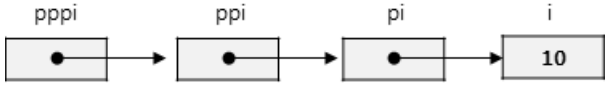
이중 포인터 선언과 역참조

```
1   int i = 10;
2   int* pi = &i;
3   int** ppi = &pi;            // 이중 포인터 ppi의 선언과 초기화
4                               // ppi는 단일 포인터 pi의 주소를 저장
5
6   **ppi = **ppi + 10;         // **ppi는 변수 i를 의미함
```



삼중 포인터 선언과 역참조

```
1   int i = 10;
2   int* pi = &i;
3   int** ppi = &pi;
4   int*** pppi = &ppi;         // 삼중 포인터 pppi의 선언과 초기화
5                               // pppi는 이중 포인터 ppi의 주소를 저장
6
7   ***pppi = ***pppi + 10;     // ***pppi는 변수 i를 의미함
```

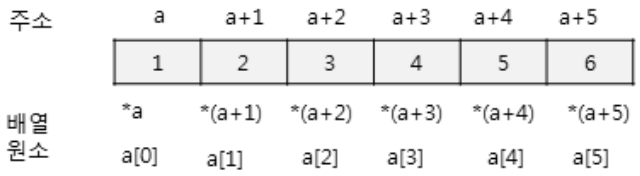


배열과 주소상수

1차원 배열과 주소상수를 사용한 역참조

```
int a[ ] = { 1, 2, 3, 4, 5, 6 };
```

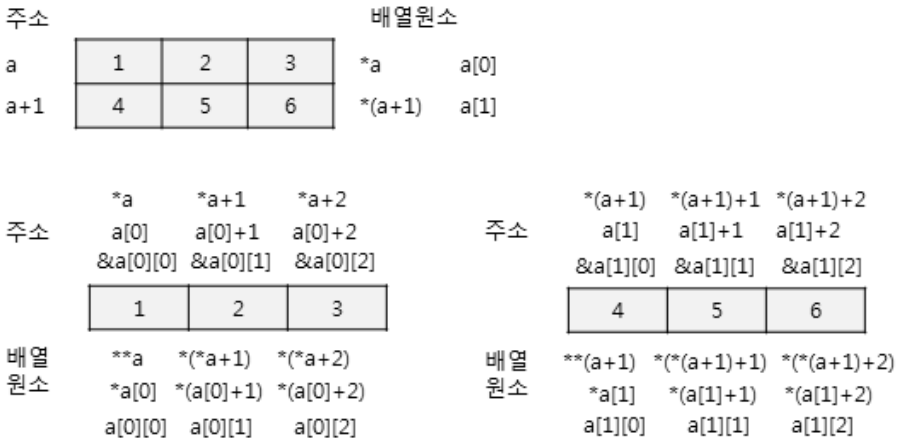
a는 &a[0], 즉, 첫 번째 배열원소의 주소를 의미하는 주소상수



2차원 배열과 주소상수를 사용한 역참조

```
int a[ ][3] = { {1, 2, 3}, {4, 5, 6} };
```

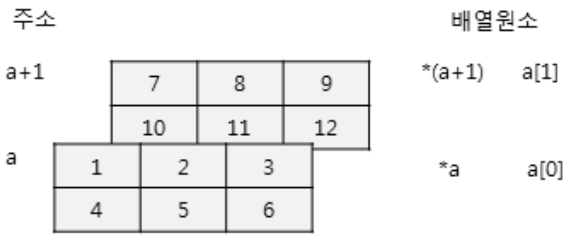
a는 &a[0], 즉, 첫 번째 행의 주소를 의미



3차원 배열과 주소상수를 사용한 역참조

```
int a[ ][2][3] = {{{1, 2, 3}, {4, 5, 6}},
                  {{7, 8, 9}, {10, 11, 12}}};
```

a는 &a[0], 즉, 첫 번째 2차원 배열의 주소를 의미

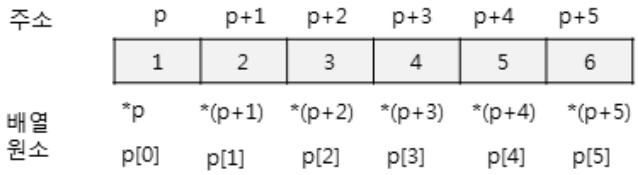


배열과 포인터

1차원 배열과 포인터를 사용한 역참조

```
int a[ ] = { 1, 2, 3, 4, 5, 6 };
int *p = a;         // int *p = &a[0];
```

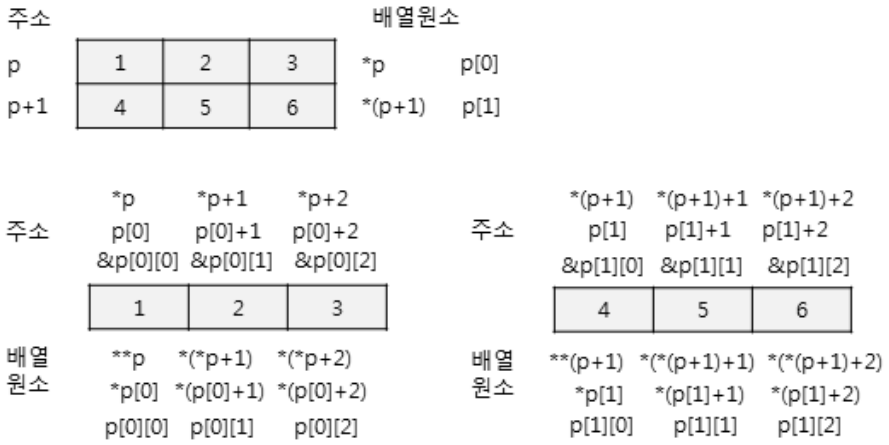
p는 &a[0], 즉, 첫 번째 배열원소의 주소를 가지는 1차원 배열 포인터



2차원 배열과 포인터를 사용한 역참조

```
int a[ ][3] = { {1, 2, 3}, {4, 5, 6} };
int (*p)[3] = a;   // int (*p)[3] = &a[0];
```

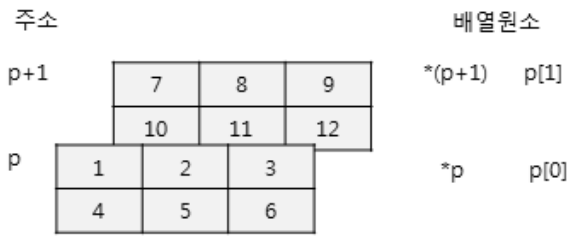
p는 &a[0], 즉, 첫 번째 행의 주소를 가지는 2차원 배열 포인터



3차원 배열과 포인터를 사용한 역참조

```
int a[ ][2][3] = {{{1, 2, 3}, {4, 5, 6}},
                  {{7, 8, 9}, {10, 11, 12}}};
int (*p)[2][3] = a;       // int (*p)[2][3] = &a[0];
```

p는 &a[0], 즉, 첫 번째 2차원 배열의 주소를 가지는 3차원 배열 포인터



2.2. Code Patterns

배열을 함수로 전달하기

1차원 배열의 전달

```
 1    int main(void)
 2    {
 3        int a[3] = { 1, 2, 3 };
 4        int sum = sum1DArr(a, 3);     // 배열이름을 실인자로 사용
 5        ...
 6        return 0;
 7    }
 8
 9    // 1차원 배열 포인터를 사용한 전달
10    int sum1DArr( int *a, int size )
11    {
12        int i, sum = 0;
13
14        for( i = 0; i < size; i++)
15                sum += a[i];   //배열 포인터 이름과 첨자를 이용해서 참조
16
17        return sum;
18    }
```

2차원 배열의 전달

```
      int main(void)
 1    {
 2        int a[2][3] = { {1, 2, 3}, {4, 5, 6} };
 3        int sum = sum2DArr(a, 2, 3);
 4        ...
 5        return 0;
 6    }
 7
 8    // 2차원 배열 포인터를 사용한 전달
 9    int sum2DArr( int (*a)[3], int rows, int cols)
10    {
11        int r, c, sum = 0;
12
13        for( r = 0; r < rows; r++)
14            for( c = 0; c < cols; c++)
15                sum += a[r][c];
16
17        return sum;
      }
```

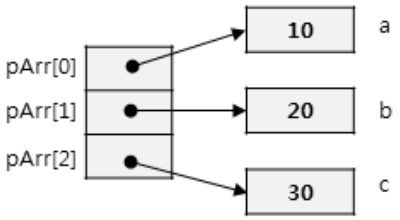
3차원 배열의 전달

```
    int main(void)
    {
 1      int a[2][2][3] = {{{1, 2, 3}, {4, 5, 6}},
 2                        {{7, 8, 9}, {10, 11, 12}}};
 3      int sum = sum3DArr(a, 2, 2, 3);
 4      ...
 5      return 0;
 6  }
 7
 8  // 3차원 배열 포인터를 사용한 전달
 9  int sum3DArr( int (*a)[2][3], int sec, int rows, int cols)
10  {
11      int s, r, c, sum = 0;
12
13      for( s = 0; s < sec; s++)
14          for( r = 0; r < rows; r++)
15              for( c = 0; c < cols; c++)
16                  sum += a[s][r][c];
17
        return sum;
    }
```

포인터 배열, 함수 포인터, 함수 포인터 배열

포인터 배열

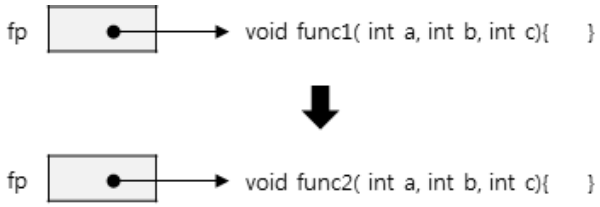
```
int a = 10, b = 20, c = 30;
int* pArr[3] = { &a, &b, &c };    // 포인터 배열 선언과 초기화
```



함수 포인터

```
void func1(int, int, int);
void func2(int, int, int);

…
void (*fp)(int, int, int) = func1;        // 함수포인터 선언과 초기화
fp(3, 4, 5);                              // func1(3, 4, 5);
fp = func2;
fp(7, 8, 9);                              // func2(7, 8, 9);
```



함수 포인터 배열

```
void func1(int, int, int);
void func2(int, int, int);
void func3(int, int, int);
…
// 함수포인터 배열 선언과 초기화
void (*fpArr[3])(int, int, int) = {func1, func2, func3};

fpArr[0](3, 4, 5);                        // func1(3, 4, 5);
fpArr[1](6, 7, 8);                        // func2(6, 7, 8);
fpArr[2](3, 5, 5);                        // func3(3, 5, 5);
```