http://ysserve.int-univ.com/Lecture/c2/e_04-04-05.html
http://www.nurs.or.jp/~sug/soft/tora/tora10.htm
いつも混乱するのでメモ。

■わかりやすく一言で

・多次元配列はＣ特有のもの。Java などには無い。
・多次元配列は、メモリ上に並んだ領域
・配列の配列やポインタの配列は、配列やポインタを参照するためのポインタを配列にしたもの。Java の多次元配列はこれに相当する。

■原則

・ポインタはアドレスを格納する
・配列の先頭をポインタで表す
・一番右（？）の次元の配列のみポインタに代入できる
なので

```
int a[3];
int* b;
b = a;
```

とか

```
int a[2][3];
int* b;
b = a[1];
```

は可能。

■二次元配列をポインタで表す

int a[2][3];

```
a[0][0] = 0;
a[0][1] = 1;
a[0][2] = 2;
a[1][0] = 10;
a[1][1] = 11;
a[1][2] = 12;

int (*b)[3]; //int[3]型（？） を指すポインタ
b = a;

int *c[2];  // ポインタの配列
c[0] = a[0]; // この場合は、各次元をそれぞれ代入する
c[1] = a[1];
```

ポインタの配列と配列のポインタ

http://homepage2.nifty.com/c-labo/Pointer.html

```
char *foo[100];
char (*bar)[100];
```

C/C++ 言語において、演算子 [] は演算子 * より高いので、コンパイラは [] を先に解釈します。

つまり、"char *foo[100]" の場合、この宣言の解釈順は以下の通りです。

> **宣言のうち、最も優先順位が高いのは [] である。つまり、変数 foo は「何か」が 100 個ある配列である。**
> **宣言から [100] を除外する。**
> **宣言の残りは "char* foo" すなわち、char へのポインタなので、「何か」は char へのポインタである。**
> **よって、変数 foo は char 型へのポインタが 100 個ある配列である。**

一方、"char (*bar)[100]" の場合は、

> **宣言のうち丸かっこの内側が優先順位が高い。**
> **丸かっこの内側は "*bar" なので、この変数 bar は「何か」へのポインタである。**
> **宣言から "(*bar)" を除外する。**
> **宣言の残りは "char[100]" なので、「何か」は char 型が 100 個ある配列である。**
> **よって、変数 bar は char 型 100 個の配列へのポインタである。**

かなり強引な解釈ですが、演算子の優先順位に従って慎重に解釈していけば、特に難しいことはないと思います。