一道题目玩转指针数组和二级指针

请大家先看下面的代码：

```
#include <stdio.h>

int main(){
    char *lines[5] = {
        "COSC1283/1284",
        "Programming",
        "Techniques",
        "is",
        "great fun"
    };

    char *str1 = lines[1];
    char *str2 = *(lines + 3);
    char c1 = *(*(lines + 4) + 6);
    char c2 = (*lines + 5)[5];
    char c3 = *lines[0] + 2;

    printf("str1 = %s\n", str1);
    printf("str2 = %s\n", str2);
    printf("  c1 = %c\n", c1);
    printf("  c2 = %c\n", c2);
    printf("  c3 = %c\n", c3);

    return 0;
}
```

运行结果：

```
str1 = Programming
str2 = is
  c1 = f
  c2 = 2
  c3 = E
```

为了方便说明问题，我们将上面的字符串数组改成下面的形式，它们都是等价的：

```c
#include <stdio.h>

int main(){
    char *string0 = "COSC1283/1284";
    char *string1 = "Programming";
    char *string2 = "Techniques";
    char *string3 = "is";
    char *string4 = "great fun";

    char *lines[5];
    lines[0] = string0;
    lines[1] = string1;
    lines[2] = string2;
    lines[3] = string3;
    lines[4] = string4;

    char *str1 = lines[1];
    char *str2 = *(lines + 3);
    char c1 = *(*(lines + 4) + 6);
    char c2 = (*lines + 5)[5];
    char c3 = *lines[0] + 2;

    printf("str1 = %s\n", str1);
    printf("str2 = %s\n", str2);
    printf("  c1 = %c\n", c1);
    printf("  c2 = %c\n", c2);
    printf("  c3 = %c\n", c3);

    return 0;
}
```

`char *lines[5]`定义了一个指针数组，它的每个元素的类型都是 `char *`。在表达式中使用 lines 时，它会转换为一个类型为 `char **`的指针，这样`*lines` 就表示一个指向字符的指针，而`**lines` 表示一个具体的字符，这一点很重要，读者一定要明白。

指针是可以进行运算的，lines 表示数组的首地址（第 0 个元素的地址），lines+0、lines+1、lines+2 … 分别表示第 0、1、2 …个元素的地址，*(lines+0)或 lines[0]、

(lines+1)或 lines[1]、(lines+2)或 lines[2] ... 分别是字符串 string0, string1, string2 ... 的首地址。所以：

```
*lines == *(lines+0) == lines[0] == string0

*(lines+1) == lines[1] == string1

*(lines+2) == lines[2] == string2

...
```

注意：lines 是二级指针，*(lines+i) 是一级指针，**(lines+i) 才是具体的字符。

上面的题目中：

- lines[1]：它是一个指针，指向字符串 string1，即 string1 的首地址。
- *(lines + 3)：lines + 3 为数组中第 3 个元素的地址，*(lines + 3) 为第 3 个元素的值，它是一个指针，指向字符串 string3。
- *(*(lines + 4) + 6)：*(lines + 4) + 6 == lines[4] + 6 == string4 + 6，表示字符串 string4 中第 6 个字符的地址，即 f 的地址，所以 *(*(lines + 4) + 6) 就表示字符 f。
- (*lines + 5)[5]：*lines + 5 为字符串 string0 中第 5 个字符的地址，即 2 的地址，(*lines + 5)[5]等价于*(*lines + 5 + 5)，表示第 10 个字符，即 2。
- *lines[0] + 2：lines[0] 为字符串 string0 中第 0 个字符的地址，即 C 的地址；*lines[0] 也就表示第 0 个字符，即字符 C。字符与整数运算，首先转换为该字符对应的 ASCII 码，然后再运算，所以 *lines[0] + 2 = 67 + 2 = 69，69 对应的字符为 E。