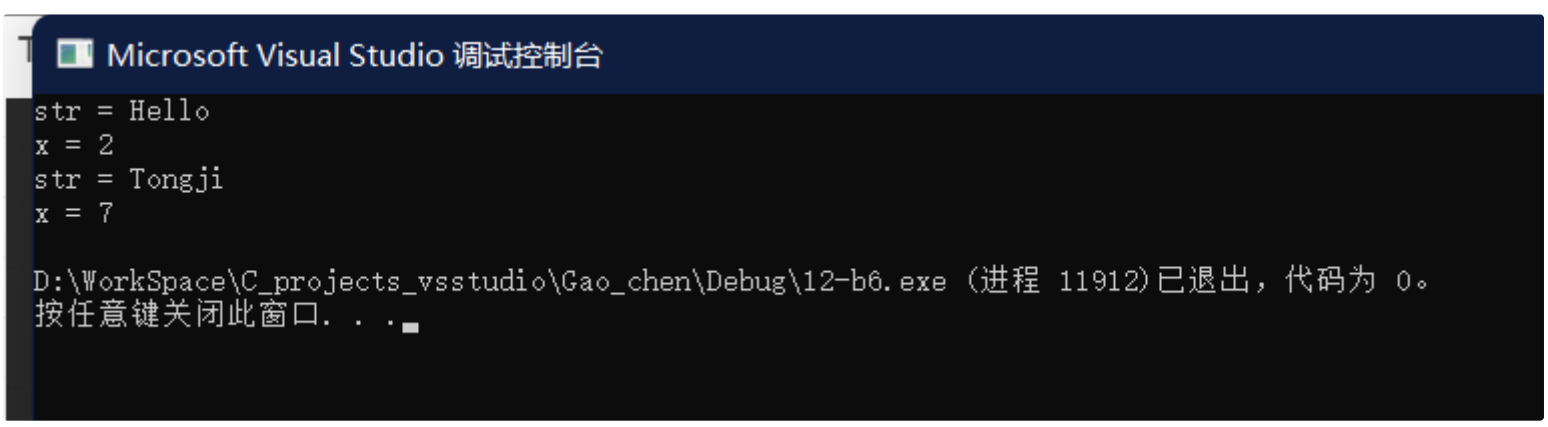
如何理解表达式 $void(*(*f[])(t1))(t2)$

```cpp
#include <iostream>
using namespace std;
using T = void(*)(int);


void f_sub(int x)
{
    cout << "x = " << x << endl;
}

T f_str(const char* p)
{
    cout << "str = " << p << endl;
    return f_sub;//返回指针
}

int main()
{
    void(*(*f[3])(const char* p))(int x);
    //f[3]是一个数组,*f[3]是一个指针数组
    //(*f[3])()是指向函数的指针数组,函数的形参为const char* p
    //(*f[3])(const char* p)代表这个函数返回
    //(*(*f[3])(const char* p))是指向函数的指针,函数返回值为void,形参为int x
    f[0] = f_str;
    f[0]("Hello")(2);
    f[0]("Tongji")(7);

    return 0;
}

```

输出结果



解释

$$void(*(*f[])(t1))(t2) \tag{1}$$

表达式可以拆分为：

`void (* (* f[])(t1) )(t2)`

另 `T = (* f[])(t1)`

则表达式可以表示为：

`void(* T)(t2)`

所以相当于定义了一种 指向函数指针类型的返回值 ， 就是上述中的 `using T = void(*) (int)` ;

（这个没查出来，让 `gpt` 生成的好像就是这样可以跑）

所以整个表达式 (1) 则为定义了一种指向某种函数的指针，其返回类型是 `T = void (*f)(int)`

这个函数的形参表是 `(const char*)`

相当于定义 `T (*fun)(const char *p)` 其中 T 为 `void (*fun)(int)`,

至于其中的 `f[3]` 则是定义的数组，长度为 3.

对于下面的函数调用

则是

$f[0]("hello")$ 先返回 `T = void (*f) (int )`

然后 $f[0]("hello")(2)$ 则是 `void (*f) ( int x = 2 )`的调用

所以就会有输出：

```
str = hello
x = 2
```

同理对于 $f[0]("Tongji")(7)$ 则有：

```
str = Tongji
x = 7
```