**12-b6.cpp**

/\*2152041 计科 王浩\*/

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：

功 能：

输入参数：

返 回 值：

说 明：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void f\_sub(int x)

{

cout << "x = " << x << endl;

}

void \*f\_str(const char\* str)

{

cout << "str = " << str << endl;

void(\*fun)(int);

fun = f\_sub;

return fun;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：

功 能：

输入参数：

返 回 值：

说 明：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

void(\*(\*f[3])(const char\* p))(int x);

f[0] = (void(\*(\*)(const char\* p))(int x))f\_str;

f[0]("Hello")(2);

f[0]("Tongji")(7);

return 0;

}

/////////////////////////////////////////代码部分结束

首先对定义部分进行分析

void(\*(\*f[3])(const char\* p))(int x);

从f开始向外看，\*f[3]是一个大小为3的一维指针数组，每个指针都指向一个以const char\*p为参数的函数，函数的返回值又是另一个以int x为参数的函数的地址，而void则是声明返回的函数返回类型是void

**添加部分**

void \*f\_str(const char\* str)

{

cout << "str = " << str << endl;

void(\*fun)(int);

fun = f\_sub;

return fun;

}

//函数f\_str

f[0] = (void(\*(\*)(const char\* p))(int x))f\_str;

一开始的时候，察觉到函数声明中拥有两个参数，且在不同的()中，意识到以const char\*为参数的函数应该返回一个函数指针，指向f\_sub，所以就定义了一个f\_str，先输出str(因为f\_sub中没这个内容)，然后再返回一个指向f\_sub函数的指针。之后便是将f[0]进行赋值，可是此时vs却报错了，提示表达式左右两侧类型不同，但是f[0]=f\_str后

//一开始报错如上图;

我认为肯定能得到正确结果，所以根据vs的类型提示进行了强制类型转换，将f\_str转换后赋值给f[0]，也确实得到了预期结果。