

@TOC

# 前言

函数指针到底是什么？

先看如下代码：

```
#include <stdio.h>
void test()
{
    printf("hehe\n");
}
int main()
{
    printf("%p\n", test);
    printf("%p\n", &test);
    return 0;
}
```



结果显然相同，两者都达到了得到函数地址的目的，函数名也是函数地址。

提示：那么想一想函数指针的类型是什么样的？

## 一、函数指针类型

其实指针的表达方法大同小异

```
void (*pf) (int x, int y, ... ) = &test
```

括号\*pf指明pf是指针，只不过将函数名替换为一个指针变量。

那么pf就是函数指针变量。

对于函数指针类型，其形参列表可以不写形参名。 `void (*pf) (int , int , ... )`

## 二、函数指针调用

熟悉的解引用 \*pf

之后顺理成章进行函数传参 \*pf ( 参数1, 参数2, .... )

简单的代码例子如下：

```
#include <stdio.h>

int Add(int x,int y){
    return x+y;
}

int main() {
    //函数指针传参
    // int (*Pf) (int x,int y)
    int (*Pf) (int x,int y) = &Add; //在这里取地址函数名和函数名都能得到函数的地址
    int ret = (*Pf)(10,5);
    printf("%d\n",ret);
    return 0;
}
```

```
D:\CLion\C\untitled\cmake-build-debug\untitled.exe
15
```

进程已结束,退出代码0

CSDN @期邈云汉

显然这成功的。那么考虑既然函数名在常规使用时Add (参数1, .....) , Add函数名也是函数的地址,

那么对于指针Pf和函数名Add两者的效果应该是相同。代码测试如下：

```
#include <stdio.h>

int Add(int x,int y){
    return x+y;
}

int main() {
    //函数指针传参
    // int (*Pf) (int x,int y)
    int (*Pf) (int x,int y) = &Add; //在这里取地址函数名和函数名都能得到函数的地址
    //int ret = (*Pf)(10,5);
    int ret = Pf(100,10);
    printf("%d\n",ret);
    return 0;
}
```

结果显然

```
D:\CLion\C\untitled\cmake-build-debug\untitled.exe
110
```

进程已结束,退出代码0

CSDN @期邈云汉

## 三、函数指针数组

那要把函数的地址存到一个数组中，那这个数组就叫函数指针数组，那函数指针的数组如何定义呢？

`int (*parr[10])();`

parr1 先和 `[]` 结合，说明 parr1 是数组，数组的内容是什么呢？是 `int (*)()` 类型的函数指针。

那么函数指针数组要求函数的 参数相同，返回类型相同。**也就是说，函数指针数组可以存放多个【参数相同、返回类型相同】的函数的地址。**

那么考虑将加、减、乘、除运算的四个函数使用函数指针数组来模拟实现简单计算器的功能。

代码如下：

```
#include <stdio.h>

void menu() {
    printf("-----");
    printf("-----1. Add      2. Sub-----");
    printf("-----3. Mul      4. Div-----");
    printf("-----      0.Exit      -----");
    printf("-----");
}

int Add(int x, int y) {
    return x + y;
}

int Sub(int x, int y) {
    return x - y;
}

int Mul(int x, int y) {
    return x * y;
}

int Div(int x, int y) {
    return x / y;
}

int main() {
    int (*pfArr[4])(int, int) = {Add, Sub, Mul, Div}; //函数指针数组
    int choose = 0;
    do {
        int a, b = 0;
        printf_s("请输入你的选择: ");
```

```

scanf("%d", &choose);
switch (choose) {
    case 1:
        printf("请输入两个整形操作数: ");
        scanf("%d %d", &a, &b);
        printf("%d\n", pfArr[0](a, b));
        break;
    case 2:
        printf("请输入两个整形操作数: ");
        scanf("%d %d", &a, &b);
        printf("%d\n", pfArr[1](a, b));
        break;
    case 3:
        printf("请输入两个整形操作数: ");
        scanf("%d %d", &a, &b);
        printf("%d\n", pfArr[2](a, b));
        break;
    case 4:
        printf("请输入两个整形操作数: ");
        scanf("%d %d", &a, &b);
        printf("%d\n", pfArr[3](a, b));
        break;
    case 0:
        break;
    default:
        printf("非法输入! \n");
        break;
}
} while (choose);
return 0;
}

```

D:\CLion\C\calc\cmake-build-debug\calc.exe

请输入你的选择: 1

请输入两个整形操作数: 12 2

14

请输入你的选择: 3

请输入两个整形操作数: 6 3

18

请输入你的选择: 4

请输入两个整形操作数: 9 2

4

请输入你的选择: 0

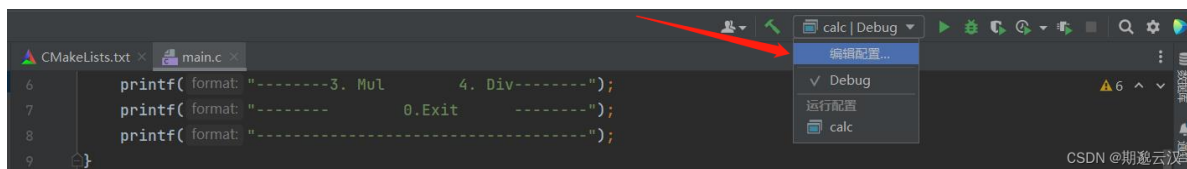
进程已结束,退出代码0

CSDN @期邈云汉

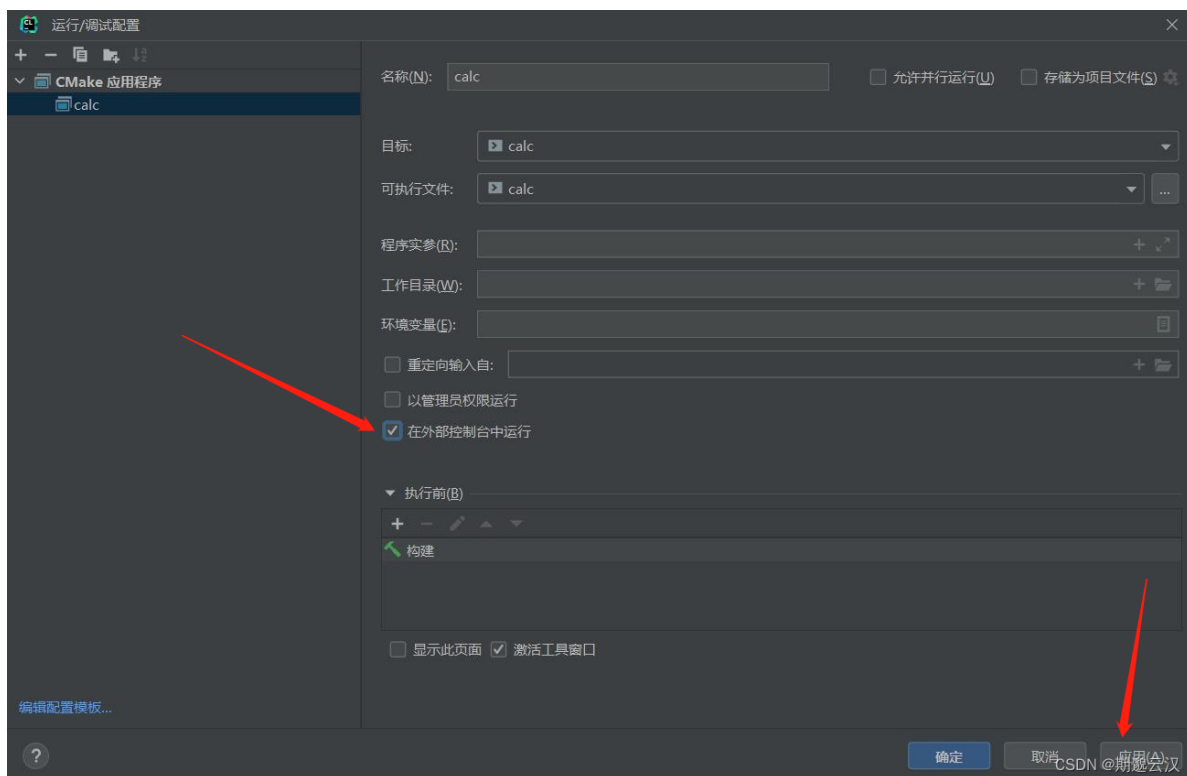
提示: 这里使用CLion环境来写C程序, 那么如果需要使用控制台来展示运行, 像Visual Studio那样

可以参考如下设置：

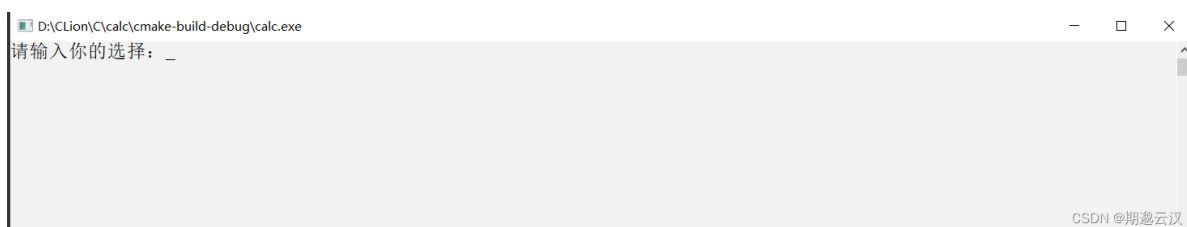
1. 在当前c程序右上角找到编辑配置



2. 选中“在外部控制台中运行”



确定修改后重新运行程序就会在命令行中运行。



## 四、指向函数指针数组的指针

函数指针：

```
int (*pf)(int, int);
```

函数指针数组：

```
int (*pfArr[4])(int, int);
```

//指向函数指针数组的指针

```
int (*(*ptr)[4])(int, int) = &pfArr;
```

既然是数组，那么就会考虑指向这个数组的指针，那么指向函数指针数组的指针也就同理

可以定义。

