# “const“的使用

1. **保护函数的参数内容**

```
/**
 * 数组作为函数参数传进函数中去。如下函数，在函数的执行过程中array[i]++会将数组的每一项都加上1。
 * （这样子，数组的值就意外的被修改了）
 */
int sum_per_add (char array[], int n)
{
    int i = 0;
    int sum = 0;

    for (i = 0; i < n; i++) {
        sum += array[i]++;
    }

    return sum;
}

/**
 * 那么，如果想在函数的执行过程中，数组的值不被修改。
 * 可以在函数的参量定义的过程中加入const
 * （在程序的编译过程中，如果你本来不希望被修改的数组，如果有改变数组值的操作，array[i]++，则会报错）
 */
int sum_per_add (const char array[], int n)
{
    int i = 0;
    int sum = 0;

    for (i = 0; i < n; i++) {
        sum += array[i]++;      /* 此时，编译会报错，应该改成，sum += array[i]; */
    }

    return sum;
}
```

1. **定义常量（与#define作用相同，但是可以创建数组常量，指针常量，以及指向常量的指针）**

```
/* 创建常量 */
const double PI = 3.14159;

/* 常量尝试变修改的时候，编译会报错 */
PI = 4.1413;

/**
 * 创建一个指向常量的指针
 * 然而只有非常量数据的地址才可以赋值给普通的指针
```

```
 */
const char *pt ;
char const *pt ;

double data[2] = {0, 1};
pt = data;
*pt = 89;   /* 不允许 */
*data = 98; /* 但是原来的数据是可以操作的 */

/**
 * 创建一个常量指针
 * 防止该指针指向别的地方
 */
char * const pt = data;

pt = &data[1];  /* 不合法的 */

/**
 * 函数定义的形参，非常量形参不可以传递常量参数
 */
const char cc[] = {1, 2};
char c[] = {1, 3};
void add (const char ary[], int n);

void add (cc, sizeof(cc));    /* 合法 */
void add (c, sizeof(c));      /* 合法 */

void mult (char ary[], int n);

void mult (cc, sizeof(cc));    /* 不合法 */
void mult (c, sizeof(c));      /* 合法 */
```

1.
2.
3.