例：程序const\_pointer1

int a = 58;

int b = 68;

int c = 78;

int d = 88;

int e = 98;

int f = 108;

//指向常量的指针，在指针类型前加const，

// 表示指针指向的对象是常量，指针本身可以修改，

// 但指针指向的对象是常量，是不能修改的，

// 即使指针本身修改了，但它指向的对象仍然是常量，还是不能修改。

const int\* p1 = &a;

//与指向常量的指针用法相同

int const\* p2 = &b;

//指针常量，在指针名前加const，表示指针本身是常量，

// 指针本身不能修改，但指针指向的对象可以修改。

int\* const p3 = &c;

//指向常量的指针常量

const int\* const p4 = &d;

// \*p1 = 108; // error，指定的为const对象

p1 = &f;

p2 = &e;

// p3 = &f; // error，不能修改指针本身

\*p3 = 118;

printf("%d\n", \*p1); // 输出为108

printf("%d\n", \*p2); // 输出为98

printf("%d, %d\n", \*p3, c); // 输出为118,118

const指针做函数的形参：

const指针可以接受const和非const地址，但是非const指针只能接受非const地址。所以const指针的能力更强一些，所以尽量多用const指针，这是一种习惯。

// const指针做函数的形参

例：程序const\_pointer2

#include <stdio.h>

void func(const int\* p)

{

// \*p = 4; // error，不能修改p指向的对象

// p = NULL; // 可以修改p指针本身

printf("%d\n", \*p); // 输出为2

}

int main(int argc, char\* argv[])

{

int a = 2;

int\* p = &a;

func(p);

return 0;

}