知识点 1【字符串操作函数】

字符串操作函数 一般以 str 开头 默认遇到'\0' .

需要的头文件#include <string.h>

1、测试字符串长度 strlen

遇到\0 结束 不包含\0

```c
#include <stdio.h>
#include <string.h>
void test01()
{
    char *str = "hello world";
    printf("%ld\n", strlen(str)); // 11
}
```

2、字符串拷贝 strcpy strncpy

2.1 strcpy

char *strcpy(char *dest, const char *src);

功能:将 src 指向的空间中的字符串 拷贝到 dst 指向的空间中 遇到\0 结束

返回值：返回的是 dest 保存的空间起始地址编号

注意：遇到字符串赋值 必须使用 strcpy

2.2 strncpy 拷贝前 n 个字符

```
char *strncpy(char *dest, const char *src, size_t n);
```

strncpy 拷贝 n 个字节 不足补 0.

3、字符串追加函数 strcat strncat

```
char *strcat(char *dest, const char *src);

char *strncat(char *dest, const char *src, size_t n);
```

```c
void test03()
{
    char dst[128] = "hello world";
    char *src = "xixi haha";

    strcat(dst, src);
    printf("dst=%s\n", dst);
}
int main(int argc, char const *argv[])
{
    test03();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
dst=hello worldxixi haha
edu@edu:~/work/c/day07$
```

```c
void test03()
{
    char dst[128] = "hello\0world";
    char *src = "xixi haha";

    strcat(dst, src);
    printf("dst=%s\n", dst);
}
int main(int argc, char const *argv[])
{
    test03();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
dst=helloxixi haha
edu@edu:~/work/c/day07$
```

4、strcmp 和 strncmp 字符串比较

```
int strcmp(const char *s1, const char *s2);
```

```
int strncmp(const char *s1, const char *s2, size_t n);

相等返回 0

大于返回>0

小于返回<0
```

```
void test04()
{
    char src[] = "hello";
    char dst[] = "haha";


    if(strcmp(src, dst) == 0)
    {
        printf("相等\n");
    }
    else if(strcmp(src, dst) > 0)
    {
        printf("大于\n");
    }
    else if(strcmp(src, dst) < 0)
    {
        printf("小于\n");
    }
```

```
}
```

```
50  void test05()
51  {
52      char buf1[] = "pub:xxxxxxxxxxxxxxx";
53      if (strncmp(buf1, "pub", 3) == 0)
54      {
55          printf("%s是发布命令\n", buf1);
56      }
57  }
58  int main(int argc, char const *argv[])
59  {
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
pub:xxxxxxxxxxxxxxx是发布命令
edu@edu:~/work/c/day07$
```

5、strchr 字符查找

```
char *strchr(const char *s, int c);

功能：从 s 指向的字符串中查找第一次出现字符 c 的位置（地址编号）

char *strrchr(const char *s, int c);
```

```c
void test06()
{
    char buf[] = "hello world";
    char *ret = strchr(buf, 'o');
    printf("%s\n", ret);
    *ret = ':';
    printf("%s\n", buf);
}
int main(int argc, char const *argv[])
{
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
o world
hell: world
edu@edu:~/work/c/day07$
```

6、strstr 查找字符串

char *strstr(const char *haystack, const char *needle);

从 haystack 指向的字符中查找 needle 指向的字符串 返回第一次出现的位置

找不到返回 NULL

```c
void test07()
{
    char buf[] = "http://www.sex.777.sex.999.sex.cn";
    char *ret = strstr(buf, "sex");
    printf("%s\n", ret);
}
int main(int argc, char const *argv[])
{
    test07();
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
sex.777.sex.999.sex.cn
edu@edu:~/work/c/day07$
```

```c
void test07()
{
    char buf[] = "http://www.sex.777.sex.999.sex.cn";
    char *ret = NULL;
    while (ret = strstr(buf, "sex"))
    {
        memset(ret, '*', strlen("sex"));
    }

    printf("%s\n", buf);
}
int main(int argc, char const *argv[])
{
    test07();
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
http://www.***.777.***.999.***.cn
edu@edu:~/work/c/day07$
```

7、字符串转换数值

atoi/atol/atof //字符串转换功能

```c
#include <stdlib.h>
void test08()
{
    printf("%d\n", atoi("1234"));
    printf("%ld\n", atol("1234"));
    printf("%f\n", atof("12.34"));
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 00_code.c
edu@edu:~/work/c/day07$ ./a.out
1234
1234
12.340000
edu@edu:~/work/c/day07$
```

晚上案例：分装函数 实现将字符串 转数值 类似 atoi

8、字符串切割 strtok

```
char *strtok(char *str, const char *delim)

参数 str:指向被切割的字符串首元素地址（第一次必须赋字符串首元素地址 以后的切割赋值为 NULL)

参数 delim：按 delim 指向的字符串作为切割符号集合

返回值:

    成功 返回切割到子串首元素地址

    失败 返回 NULL
```

方式一:

```
void test09()
{
    char str[] = "hehehe:xixixi:lalala:heiheihei:wuwuwu:henhenhenhen";
    char *buf[128] = {NULL};


    int i = 0;
    // 第一次切割
    buf[i] = strtok(str, ":");
    while (buf[i] != NULL)
    {
        i++;
        buf[i] = strtok(NULL, ":");
    }
```

```
    i = 0;
    while (buf[i] != NULL)
    {
        printf("%s\n", buf[i]);
        i++;
    }
}
```

方法二:

```
void test09()
{
    char str[] = "hehehe:xixixi:lalala:heiheihei:wuwuwu:henhenhenhen";
    char *buf[128] = {str, NULL};

    int i = 0;
    while (1)
    {
        buf[i] = strtok(buf[i], ":");
        if (buf[i] == NULL)
            break;
        i++;
    }
```

```
    i = 0;
    while (buf[i] != NULL)
    {
        printf("%s\n", buf[i]);
        i++;
    }
}
```

方法三：

```
87  void test09()
88  {
89      char str[] = "hehehe:xixixi:lalala:heiheihei:wuwuwu:henhenhenhen";
90      char *buf[128] = {str, NULL};
91
92      int i = 0;
93      while ((buf[i] = strtok(buf[i], ":")) && ++i);
94
95      i = 0;
96      while (buf[i] != NULL)
97      {
98          printf("%s\n", buf[i]);
99          i++;
100     }
101 }
```

例 10：作业

以下为我们的手机收到的短信的格式，请利用指针数组与 strtok 函数对其解析

char msg_src[]="+CMGR:REC

UNREAD,+8613466630259,98/10/01,18:22:11+00,ABCdefGHI";

参考以下的函数名字以及参数，完成相应的要求

int msg_deal(char *msg_src, char *msg_done[],char *str)

参数 1：待切割字符串的首地址

参数 2：指针数组：存放切割完字符串的首地址

参数 3：切割字符

返回值：切割的字符串总数量

手机号:13466630259

日期：98/10/01

时间：18:22:11

内容：ABCdefGHI

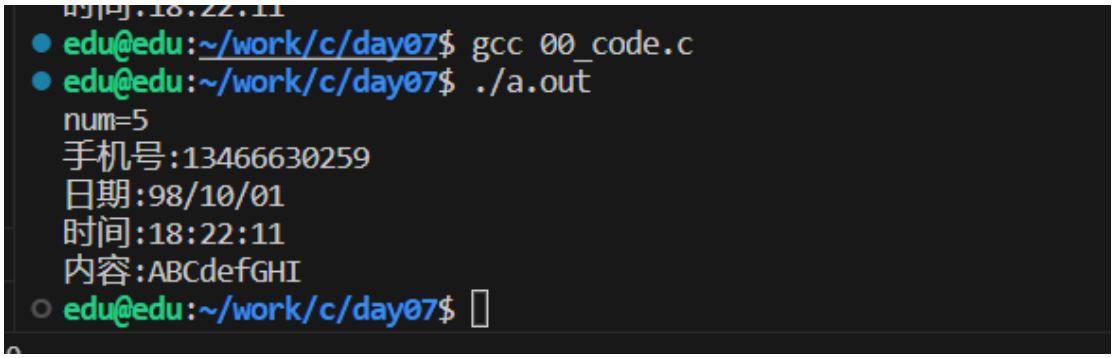
```
int msg_deal(char *msg_src, char *msg_done[], char *str)
{
    msg_done[0] = msg_src;

    int i = 0;
    while ((msg_done[i] = strtok(msg_done[i], str)) && ++i)
        ;

    return i;
}
void test10()
{
    char msg_src[] = "+CMGR:REC
UNREAD,+8613466630259,98/10/01,18:22:11+00,ABCdefGHI";
    char *msg_done[128] = {NULL};
```

```
    int num = msg_deal(msg_src, msg_done, ",");
    printf("num=%d\n", num);

    printf("手机号:%s\n", msg_done[1] + 3);
    printf("日期:%s\n", msg_done[2]);
    char *ret = strchr(msg_done[3], '+');
    if (ret != NULL)
        *ret = '\0';
    printf("时间:%s\n", msg_done[3]);
    printf("内容:%s\n", msg_done[4]);
}
```



知识点 2【格式化字符串操作函数】

1、sprintf 用于组包（将零散的数据组成一个字符串）


```c
#include <stdio.h>
void test01()
{
    int year = 2023;
    int month = 7;
    int day = 28;

    char buf[128] = "";
    int len = sprintf(buf, "%d年%d月%d日", year, month, day);
    printf("buf=%s\n", buf);
    printf("len=%d\n", len);
}
int main(int argc, char const *argv[])
{
    test01();
    return 0;
}

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
buf=2023年7月28日
len=16
edu@edu:~/work/c/day07$
```

```c
void test02()
{
    char buf[128] = "";
    sprintf(buf, "%d", 1234);
    printf("%s\n", buf);
}
int main(int argc, char const *argv[])
{
    test02();
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
1234
edu@edu:~/work/c/day07$
```

2、sscanf 解包函数

2.1 %s 和 sscanf 遇到空格回车结束提取

注意：%s 提取到的内容是字符串

```c
void test03()
{
    char buf[128] = "hello world";
    char buf2[128] = "";

    sscanf(buf, "%s", buf2);
    printf("buf2=%s\n", buf2);
}
int main(int argc, char const *argv[])
{
    test03();
    return 0;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
buf2=hello
edu@edu:~/work/c/day07$
```

2.2 %d %ld %hd %u %hu %lu 和 sscanf 提取的是数值

遇到非数值字符结束。

```c
void test04()
{
    char buf[128] = "123abc456";
    int data = 0;

    sscanf(buf, "%d", &data);
    printf("data=%d\n", data);
}
int main(int argc, char const *argv[])
{
    test04();
    return 0;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
data=123
edu@edu:~/work/c/day07$
```

2.3 %c 和 sscanf 提取一个字符

```c
void test05()
{
    char buf[128] = "123abc456";
    char ch;
    sscanf(buf, "%c", &ch);
    printf("ch=%c\n", ch);
}
int main(int argc, char const *argv[])
{
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
ch=1
edu@edu:~/work/c/day07$
```

3、sscanf 的高级方法

3.1 提取指定个数的字符或数值 %3s %3d

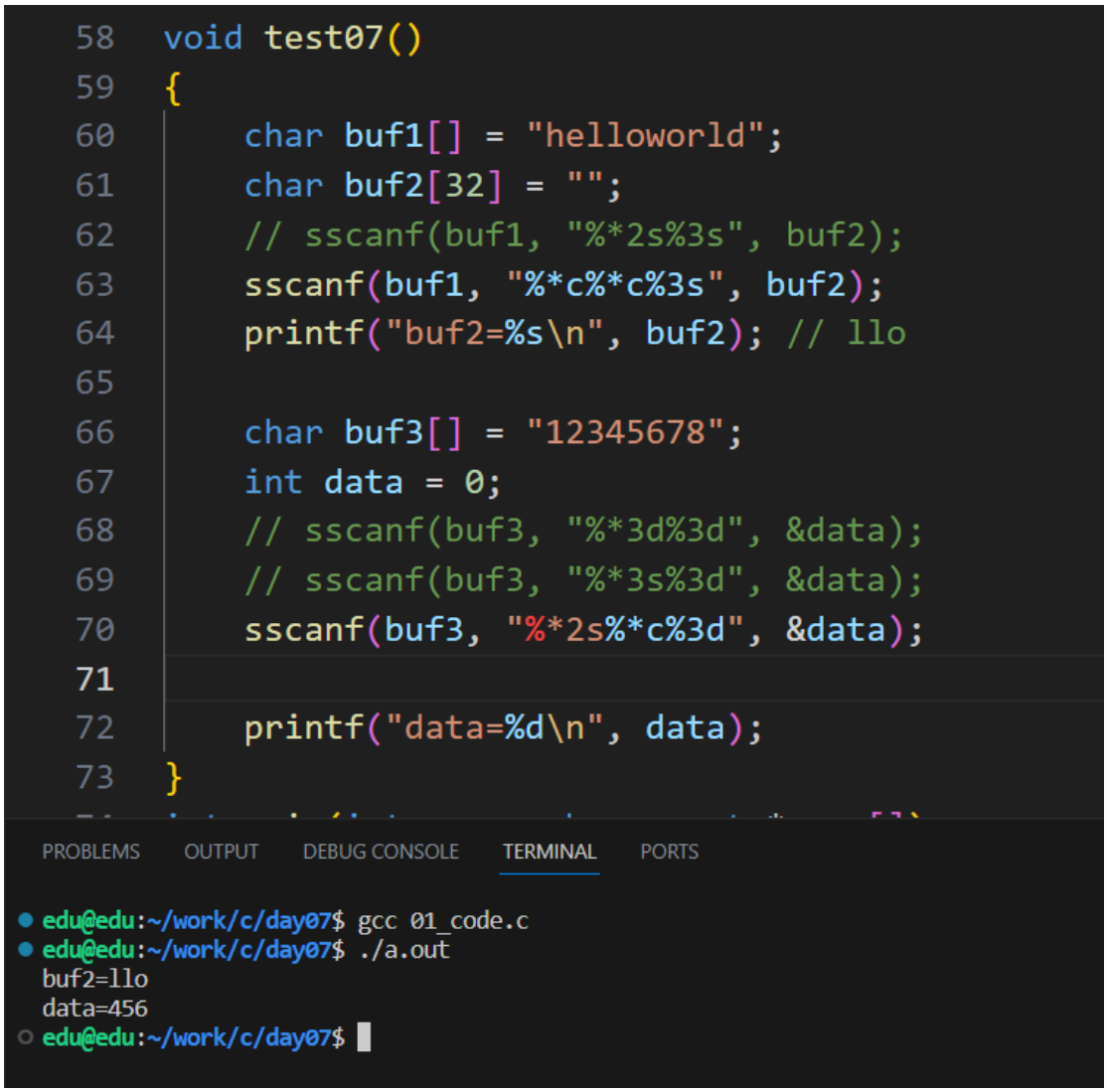
```
45  void test06()
46  {
47      char buf1[] = "helloworld";
48      char buf2[32] = "";
49      sscanf(buf1, "%5s", buf2);
50      printf("buf2=%s\n", buf2); // hello
51
52      char buf3[] = "12345678";
53      int data = 0;
54      sscanf(buf3, "%3d", &data);
55      printf("data=%d\n", data);
56  }
57  int main(int argc, char const *argv[])
58  {
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
buf2=hello
data=123
edu@edu:~/work/c/day07$
```

3.2 %*s 跳过提取到的字符串 %*d 跳过提取的数值

%*3s 跳过 3 个字符　%*3d 跳过 3 个数值



3.2 %[a-z] 表示匹配 a 到 z 中任意字符(尽可能多的匹配)

注意提取的结果是字符串。

```
sscanf("abcABC","%[a-z]")提取的结果是"abc"
```

3.3 %[aBc] 匹配 a、B、c 中一员，贪婪性

```
sscanf("abcABC","%[aBc]")提取的结果是"a"
```

3.4 %[^aFc] 匹配非 a Fc 的任意字符，贪婪性

```
sscanf("abcABC","%[^A]")提取的结果是"abc"

sscanf("[简单爱:啦啦啦]","%[^]]")提取的结果是"[简单爱:啦啦啦"

sscanf("[简单爱:啦啦啦]","%[^:]")提取的结果是"[简单爱"
```

sscanf("[简单爱:啦啦啦]","%*c%[^:]")提取的结果是"简单爱"、

sscanf("[简单爱:啦啦啦]","%*[^:]%*c%[^]]")提取的结果是"啦啦啦"

sscanf("[简单爱:啦啦啦]","%*[^:]:%[^]]")提取的结果是"啦啦啦"

```
75  void test08()
76  {
77      char buf[128] = "";
78      // sscanf("[简单爱:啦啦啦啦]", "%*[^:]%*c%[^]]", buf);
79      sscanf("[简单爱:啦啦啦啦]", "%*[^:]:%[^]]", buf);
80      printf("%s\n", buf);
81  }
82  int main(int argc, char const *argv[])
83  {
84      test08();
85      return 0;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 01_code.c
edu@edu:~/work/c/day07$ ./a.out
啦啦啦啦
edu@edu:~/work/c/day07$
```

```
void test08()

{

    // char buf[128] = "[02:16.33]我想大声宣布对你依依不舍";

    char buf[128] = "[02:16.33][04:11.44][05:11.44]我想大声宣布对你依依不舍";

    char *p_lrc = buf; // 让 p_lrc 定位歌词的位置



    // 定位歌词的位置

    while (*p_lrc == '[')

        p_lrc += 10;
```

```
    // 分析时间
    char *p_time = buf;
    while (*p_time == '[')
    {
        int m = 0, s = 0;
        sscanf(p_time, "[%d:%d", &m, &s);
        printf("%d 秒打印歌词:%s\n", m * 60 + s, p_lrc);
        p_time += 10;
    }
}
```

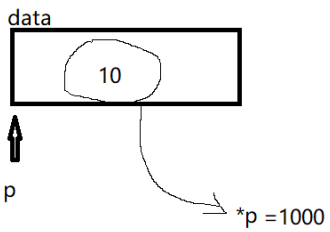
知识点 3【const 和指针的关系】

1、const 修饰普通变量

```
// const 修饰data为只读 data本质还是变量
const int data = 10;
//data = 1000;//error
```

const修饰data为只读 用户不能通过data给空间赋值

用户可以通过其他方式 对data对应的空间内容赋值




```
void test01()
{
    // const 修饰 data 为只读 data 本质还是变量
    const int data = 10;
```

```
    // data = 1000;//error

    int *p = (int *)&data;

    *p = 1000;

    printf("data=%d\n", data);//1000

}
```

2、const 在*的左边

const int *p; int const *p;

```
12  void test02()
13  {
14      int data = 10;
15      // const 修饰的*, 不能对*p赋值   但是p可读可写
16      const int *p = &data;
17      //*p = 1000;//error
18      printf("*p=%d\n", *p); // 10
19
20      int num = 100;
21      p = &num;
22      printf("*p=%d\n", *p); // 100
23  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 02_code.c
edu@edu:~/work/c/day07$ ./a.out
*p=10
*p=100
edu@edu:~/work/c/day07$
```

3、const 在*的右边

int * const p;

```
void test03()

{

    int data = 10;

    // const 修饰的 p   p 只读  *p 可读可写
```

```
    int *const p = &data;

    *p = 1000;

    printf("data=%d\n", data);



    int num = 100;

    //p = &num;//error

}
```

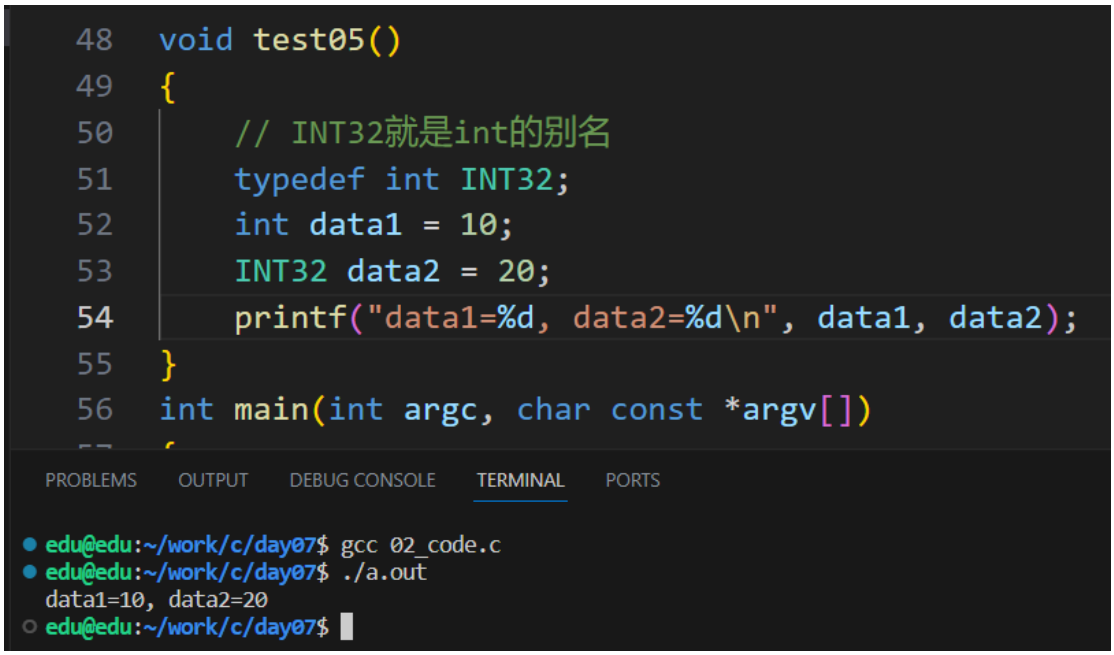
4、const 在*左右两边

const int * const p;

```
36  void test04()
37  {
38      int data = 10;
39      // p只读  *p只读
40      const int *const p = &data;
41      *p = 1000;//error
42      printf("data=%d\n", data);
43
44      int num = 100;
45      p = &num; // error
46  }
47  int main(int argc, char const *argv[])
48  {
49      test04();
50      return 0;
51  }
```

知识点 4【typedef】

typedef 给已有的类型 重新取个别名。别名也是类型，旧类型任然可用



1、typedef 步骤

1、给哪个类型取别名 就用该类型定义变量

2、用别名替换 变量名。

3、在表达式前面加 typedef 关键字

2、常见的 typedef 定义形式

2.1 给指针类型取别名

```c
void test06()
{
    // 需求：给int *取个别名P_TYPE
    typedef int *P_TYPE; // P_TYPE==int *
    P_TYPE p;            // int *p
    int data = 10;
    p = &data;
    printf("*p=%d\n", *p);
}
int main(int argc, char const *argv[])
{
    test06();
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

edu@edu:~/work/c/day07$ gcc 02_code.c
edu@edu:~/work/c/day07$ ./a.out
*p=10
edu@edu:~/work/c/day07$
```

注意该案例的分析：

```c
#include <stdio.h>
typedef int * P_TYPE1;
#define P_TYPE2 int*
void test01()
{
    P_TYPE1 p1,p2;//p1 p2都是int *类型的指针变量

    //int* p3,p4;
    P_TYPE2 p3,p4;//p3是int *类型的指针变量  p4是int类型变量
}
```

```c
void test02()
{
    // ARR_TRPE就是数组类型 该数组必须5个元素 每个元素为int
    typedef int ARR_TRPE[5];
    ARR_TRPE arr;//arr为数组
    printf("%ld\n", sizeof(arr));//20
}
```

const char *argv[]

```c
typedef int (*P_FUN)(int, int);
int my_add(int x, int y)
{
    return x + y;
}
int my_calc(int x, int y, P_FUN func)
{
    return func(x, y);
}
void test03()
{
    P_FUN p = my_add;
    printf("%d\n", my_calc(100, 200, my_add));
}
```