程序设计基础

教学团队：徐明星，兴军亮，任炬

renju@tsinghua.edu.cn

2023秋，每周一第2节，三教3200

专题讨论 (2)

Hatred is blind, as well as love.

—— Oscar Wilde

我想请教一下ppt72页的从文件读入gpa数据的参考代码，如果像您在57页讲的只是读到末尾，eof()会返回false，只有"试图跨过文件末尾"才会返回true，那么我感觉72页这里统计的学生数量应该会比实际数量多1。但是从输出的结果来看，学生数量又是正确的，请问这是什么原因呢?

EX01.cpp

20221119 > EX01.cpp > main()

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    char name[5];
    int score, age;

    int num = 0;
    ifstream fin("input-err.txt");
    while (!fin.eof())
    {
        fin >> name >> score >> age;
        cout << num << " -->> "
             << name << ' ' << score << ' ' << age << endl;
        num++;
    }

    return 0;
}
```

input-err.txt

20221119 > input-err.txt

```
ZHANG 87 18
LI 93 17
WANG 63 18
```

终端

```
0 -->> ZHANG 87 18
1 -->> LI 93 17
2 -->> WANG 63 18
```

EX01.cpp

20221119 > EX01.cpp > main()

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    char name[5];
    int score, age;

    int num = 0;
    ifstream fin("input-err.txt");
    while (!fin.eof())
    {
        fin >> name >> score >> age;
        cout << num << " -->> "
             << name << ' ' << score << ' ' << age << endl;
        num++;
    }

    return 0;
}
```

input-err.txt

20221119 > input-err.txt

```
ZHANG 87 18
LI 93 17
WANG 63 18

```

注意：这里多了一个空行

终端

```
0 -->> ZHANG 87 18
1 -->> LI 93 17
2 -->> WANG 63 18
3 -->> WANG 63 18
```

注意：这里重复输出了

【出错原因】输入文件的格式与代码没有配合好

D:\FOP.VS.Projects\20221119\input-err.txt

```
00000000  5A 48 41 4E 47 20 38 37  20 31 38 0D 0A 4C 49 20  ZHANG 87 18..LI
00000010  39 33 20 31 37 0D 0A 57  41 4E 47 20 36 33 20 31  93 17..WANG 63 1
00000020  38 0D 0A                                          8..
```

D:\FOP.VS.Projects\20221119\input.txt

```
00000000  5A 48 41 4E 47 20 38 37  20 31 38 0D 0A 4C 49 20  ZHANG 87 18..LI
00000010  39 33 20 31 37 0D 0A 57  41 4E 47 20 36 33 20 31  93 17..WANG 63 1
00000020  38                                                8
```

```
2022.11.18   21:30                912 BUG01.cpp
2022.11.18   21:24                380 EX01.cpp
2022.11.18   22:54                618 EX02.cpp
2022.11.18   21:26                 35 input-err.txt
2022.11.18   21:10                 33 input.txt
               5 个文件              1.978 字节
               2 个目录 764.009.693.184 可用字节
```

文件大小有差异

说明：在程序测试input.txt时，要将其更名为input-err.txt，否则就需要修改源程序中文件名字符串。

解决问题的办法

EX01.cpp　EX01A.cpp

20221119 > EX01A.cpp > main()

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    char name[5];
    int score, age;

    int num = 0;
    ifstream fin("input-err.txt");
    while (!fin.eof())
    {
        fin >> name >> score >> age;
        if (!fin)
            break;
        cout << num << " -->> "
             << name << ' ' << score << ' ' << age << endl;
        num++;
    }

    return 0;
}
```

终端

```
0 -->> ZHANG 87 18
1 -->> LI 93 17
2 -->> WANG 63 18
PS D:\FOP.VS.Projects>
```

注意 ：若文件处于EOF，这只能说明上一次读操作试图跨越文件结尾（这意味着再进行文件读是没有意义的。参见第12行代码），而对于是否成功地读入了想要的类型数据，是并不知道的。如果得到了，则文件状态是正常的，if (fin)判断为真；如果没有得到，则文件状态是异常的，if (fin)判断为假，这时变量中存储的数据是不对的，不能使用（参见第15-16行代码）。

在使用读入的数据前，
先检查一下文件状态！

【调试任务】 OJ-1759 回文串

题目描述

回文串是一个正读和反读都一样的字符串。比如“level”、“noon”、“aba”、“c”都是回文串，但“abc”、“good”、“abab“都不是回文串。
现在输入一行字符串，现只考虑其中字母和数字，并忽略大小写，判断其是否为回文串。

输入描述

一行字符串，字符个数不超过100。（可能包含大小写字母、数字、标点符号、空格等）

输出描述

如果回文串，输出1，否则输出0。

BUG在哪里？

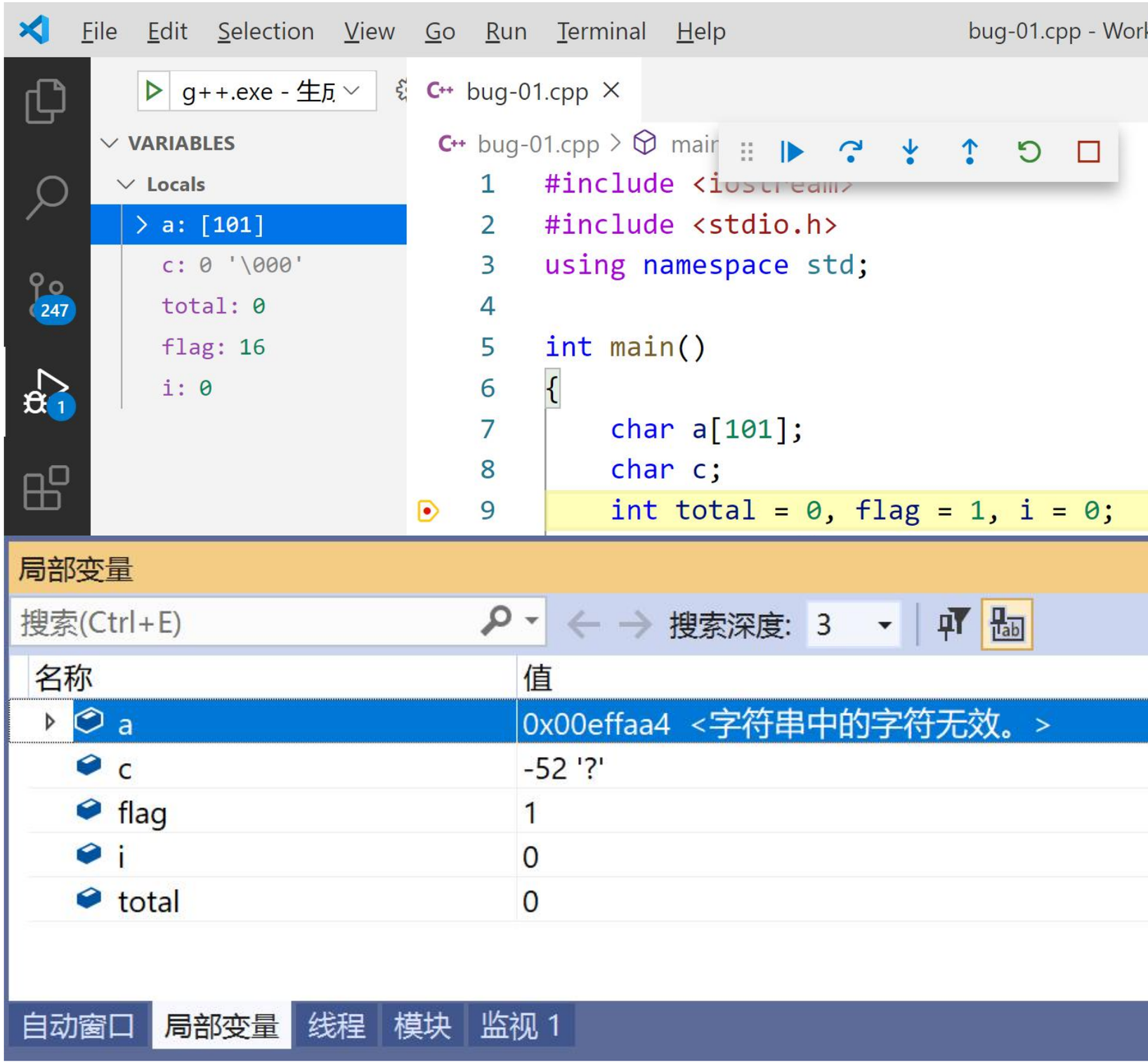
bug-01.cpp

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[101];
    char c;
    int total = 0, flag = 1, i = 0;
    c = getchar();
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
        {
            a[i] = c + 32;
            total += 1;
            i += 1;
        }
        else if ((c <= 'z') && (c >= 'a'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        else if ((c <= '9') && (c >= '0'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        c = getchar();
    }
    total--;
    for (i = 0; i <= (total / 2); i++) {
        if (a[i] != a[total - i])
        {
            flag = 0;
            break;
        }
        else if (total < 0)
        {
            flag = 0;
            break;
        }
    }
    cout << flag;
    return 0;
}
```

在代码中设置断点

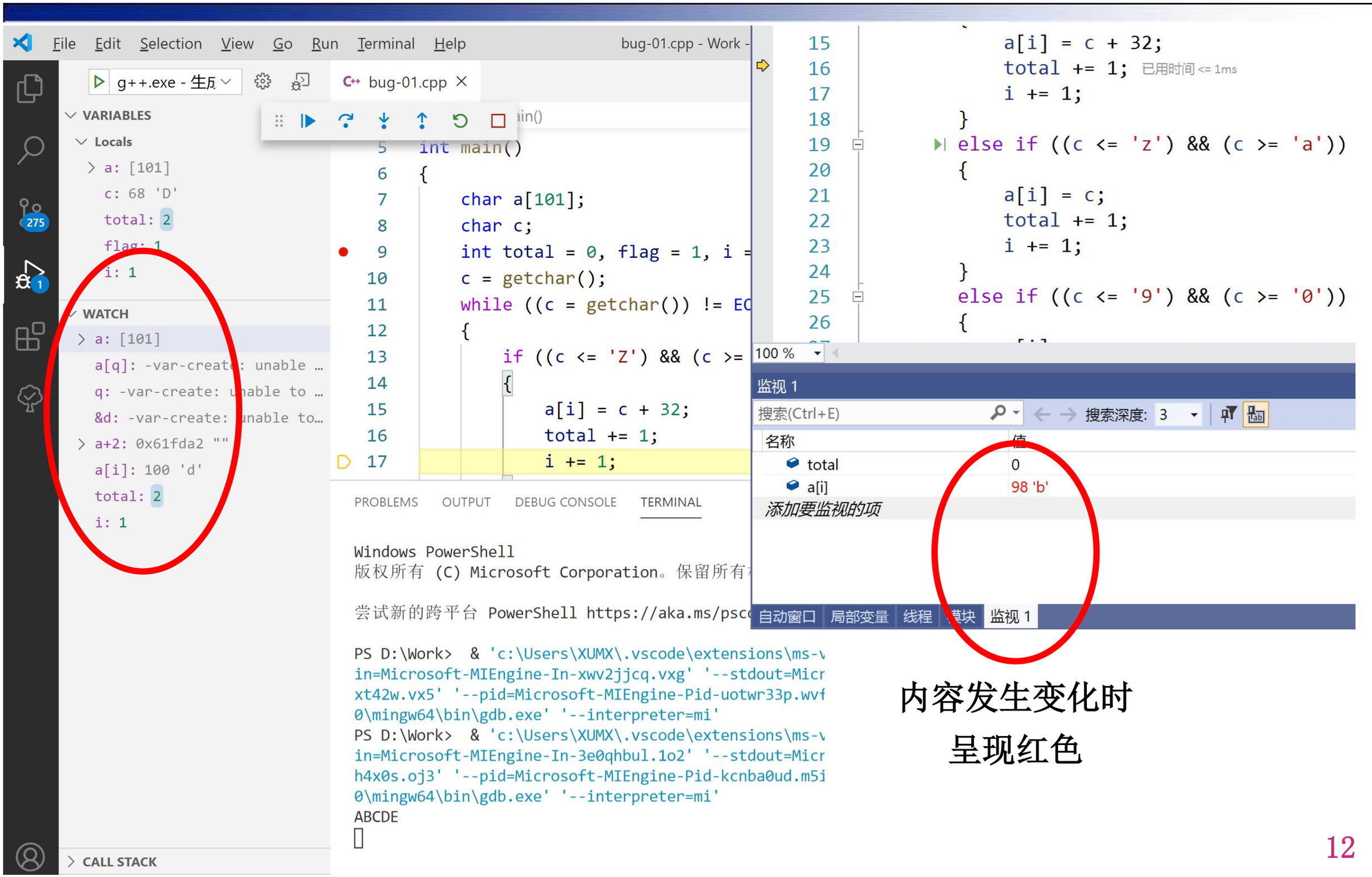
在行边条上单击鼠标左键设置断点

观察变量


File Edit Selection View Go Run Terminal Help
bug-01.cpp - Work
g++.exe - 生成
bug-01.cpp
VARIABLES
Locals
a: [101]
c: 0 '\000'
total: 0
flag: 16
i: 0
bug-01.cpp > main
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
char a[101];
char c;
int total = 0, flag = 1, i = 0;
局部变量
搜索(Ctrl+E)
搜索深度: 3
名称
值
类型
a
0x00effaa4 <字符串中的字符无效。>
char[101]
c
-52 '?'
char
flag
1
int
i
0
int
total
0
int
自动窗口
局部变量
线程
模块
监视 1

调试方法：随时监控变量（表达式）的值

VS Code, Visual Studio



内容发生变化时
呈现红色

观察指定内存单元的内容

```
11  while ((c = getchar()) != EOF)
12  {
13      if ((c <= 'Z') && (c >= 'A'))
14      {
15          a[i] = c + 32;
16          total += 1;  已用时间 <= 1ms
17          i += 1;
18      }
```

100 %

内存 1

地址: 0x00AFF798　　列: 自动

```
0x00AFF798  62 cc cc cc cc cc cc cc cc cc cc cc cc cc  b??????????
                                cc cc cc cc  ??????????
                                cc cc cc cc  ??????????
                                cc cc cc cc  ??????????
                                cc cc cc cc  ??????????
0x00AFF7D9  cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ??????????
0x00AFF7E6  cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ??????????
0x00AFF7F3  cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ??????????
0x00AFF800  cc cc cc cc 24 f8 af 00 83 21 43 00 01     ????$??.?!C..
0x00AFF80D  00 00 00 e8 55 ec 00 88 cc ec 00 01 00     ...?U?.???...
0x00AFF81A  00 00 e8 55 ec 00 88 cc ec 00 80 f8 af     ..?U?.???.€??
0x00AFF827  00 d7 1f 43 00 44 e4 ce 52 39 13 43 00     .?.C.D??R9.C.
0x00AFF834  39 13 43 00 00 e0 94 00 00 00 00 00 00     9.C..??......
0x00AFF841  00 00 00 00 00 00 00 00 00 00 00 00 00     .............
0x00AFF84E  00 00 00 00 00 00 00 00 00 00 00 00 00     .............
0x00AFF85B  00 7c a5 43 00 88 a5 43 00 00 00 00 00     .|?C.??C.....
```

在“地址”栏中输入&a[i]，可观察这个数组元素内存单元的内容变化

自动窗口　局部变量　内存 1　内存 2　线程　模块　监视 1

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[101];
    char c;
    int total = 0, flag = 1, i = 0;
    c = getchar();
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
        {
            a[i] = c + 32;
            total += 1;
            i += 1;
        }
        else if ((c <= 'z') && (c >= 'a'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        else if ((c <= '9') && (c >= '0'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        c = getchar();
    }
    total--;

    for (i = 0; i <= (total / 2); i++)
    {
        if (a[i] != a[total - i])
        {
            flag = 0;
            break;
        }
        else if (total < 0)
        {
            flag = 0;
            break;
        }
    }
    cout << flag;

    return 0;
} // bug-01.cpp
```

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[101];
    char c;
    int total = 0, flag = 1, i = 0;
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
        {
            a[i] = c + 32;
            total += 1;
            i += 1;
        }
        else if ((c <= 'z') && (c >= 'a'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        else if ((c <= '9') && (c >= '0'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
    }
    total--;

    for (i = 0; i <= (total / 2); i++)
    {
        if (a[i] != a[total - i])
        {
            flag = 0;
            break;
        }
        else if (total < 0)
        {
            flag = 0;
            break;
        }
    }
    cout << flag;

    return 0;
} // bug-01-A.cpp
```

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[101];
    char c;
    int total = 0, flag = 1, i = 0;
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
        {
            a[i] = c + 32;
            total += 1;
            i += 1;
        }
        else if ((c <= 'z') && (c >= 'a'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        else if ((c <= '9') && (c >= '0'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
    }
    total--;

    for (i = 0; i <= (total / 2); i++)
    {
        if (a[i] != a[total - i])
        {
            flag = 0;
            break;
        }
        else if (total < 0)
        {
            flag = 0;
            break;
        }
    }
    cout << flag;

    return 0;
} // bug-01-A.cpp
```

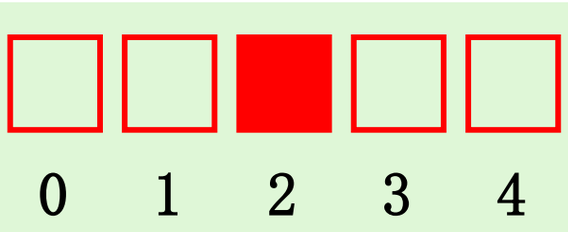
极端情况应提前，不能放在循环中。

循环中放的应该是“通项”！

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char a[101];
    char c;
    int total = 0, flag = 1, i = 0;
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
        {
            a[i] = c + 32;
            total += 1;
            i += 1;
        }
        else if ((c <= 'z') && (c >= 'a'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
        else if ((c <= '9') && (c >= '0'))
        {
            a[i] = c;
            total += 1;
            i += 1;
        }
    }
    total--;

    if (total <= 0)
    {
        cout << 1;
        return 0;
    }

    for (i = 0; i <= (total / 2); i++)
    {
        if (a[i] != a[total - i])
        {
            flag = 0;
            break;
        }
    }
    cout << flag;

    return 0;
} // bug-01-B.cpp
```

用简单实例检查边界情况

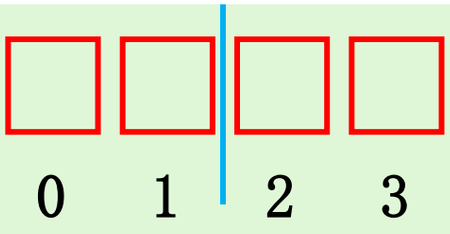
```
for (i = 0; i <= (total / 2); i++)
{
    if (a[i] != a[total - i])
    {
        flag = 0;
        break;
    }
}
```



total = 4

for: [0 .. 2]

下标2处的元素重复检查了

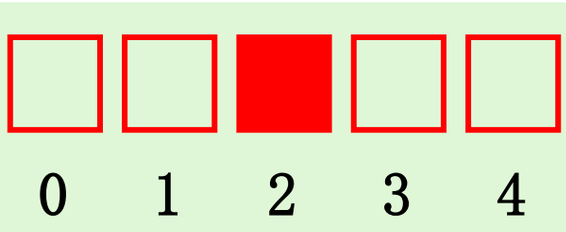


total = 3

for: [0 .. 1]

0-based array !

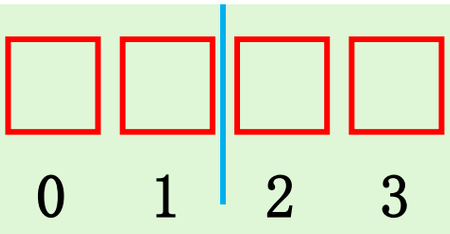
```
/// 令 total 记录元素的总数
for (i = 0; i < total / 2; i++)
{
    if (a[i] != a[total - 1 - i])
    {
        flag = 0;
        break;
    }
}
```



total = 5

for: [0 .. 2)

下标2处的元素不会被检查



total = 4

for: [0 .. 2)

下标2处的元素不会被检查

要避免出现重复代码片段

```
while ((c = getchar()) != EOF)
{
    if ((c <= 'Z') && (c >= 'A'))
    {
        a[i] = c + 32;
        total += 1;
        i += 1;
    }
    else if ((c <= 'z') && (c >= 'a'))
    {
        a[i] = c;
        total += 1;
        i += 1;
    }
    else if ((c <= '9') && (c >= '0'))
    {
        a[i] = c;
        total += 1;
        i += 1;
    }
}
```

虽不算错误，但有隐患！

参考代码V1

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char c, a[101];

    int total = 0;
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
            a[total++] = c + 32;
        else if (((c <= 'z') && (c >= 'a')) ||
                 ((c <= '9') && (c >= '0')))
            a[total++] = c;
    }
    if (total <= 1) {
        cout << 1;
        return 0;
    }
    for (int i = 0; i < total / 2; i++){
        if (a[i] != a[total - 1 - i]) {
            cout << 0;
            return 0;
        }
    }
    cout << 1;
    return 0;
}
```

要避免出现重复代码片段

参考代码V2

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char c, a[101];

    int total = 0;
    while ((c = getchar()) != EOF)
    {
        if ((c <= 'Z') && (c >= 'A'))
            a[total++] = c + 32;
        else if (((c <= 'z') && (c >= 'a')) ||
                 ((c <= '9') && (c >= '0')))
            a[total++] = c;
    }
```

```
    if (total > 1)
    {
        for (int i = 0; i < total / 2; i++)
        {
            if (a[i] != a[total - 1 - i])
            {
                cout << 0;
                return 0;
            }
        }
    }
    cout << 1;
    return 0;
}
```

缜密的逻辑思维极端重要

思考题

1、检查任意字符数组的内容是否满足回文要求

2、能设置任意过滤规则，对字符序列进行处理

```
{
    if ((c <= 'Z') && (c >= 'A'))
        a[total++] = c + 32;
    else if (((c <= 'z') && (c >= 'a')) || ((c <= '9') && (c >= '0')))
        a[total++] = c;
}
```

程序命令行参数

下列程序的功能是什么？

```
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
} // EX1.CPP
```

程序**EX1**的特点：

- 加法的两个操作数在程序运行时输入
- 用户在被程序“问到”时才会输入操作数
- 属于“规定了流程的人机交互”类型

能否有其他的人机交互方式？

使用main函数的参数argc，argv

请看下面的示例代码：

```
// ex2.cpp
#include <iostream>
#include <cstdio>  // atoi()
int main(int argc, char** argv)
{
   int a, b;
//std::cin >> a >> b;
   a = atoi(argv[1]);
   b = atoi(argv[2]);
   std::cout << a + b << std::endl;
   return 0;
}
```

命令提示符

```
D:\>type ex2.cpp
// ex2.cpp @ 20090831
#include <iostream>
#include <cstdio>        // atoi()
int main(int argc, char** argv)
{
        int a, b;
//      std::cin >> a >> b;
        a = atoi(argv[1]);
        b = atoi(argv[2]);
        std::cout << a + b << std::endl;
        return 0;
}

D:\>cl -GX ex2.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

ex2.cpp
Microsoft (R) Incremental Linker Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:ex2.exe
ex2.obj

D:\>ex2 4 5
9

D:\>
```

命令行参数（main函数实参）与形参的对应关系


```
// ex2.cpp
#include <iostream>
#include <cstdio>   // atoi()
int main(int argc, char** argv)
{
    int a, b;
//  std::cin >> a >> b;
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    std::cout << a + b << std::endl;
    return 0;
}
```

不按“常理”出牌 ... 程序运行崩溃

命令提示符 - ex2

```
D:\>ex2 4 5
9

D:\>ex2
```

ex19.cpp > main(int, char **)

```
1   #include <iostream>
2   #include <cstdio> // atoi()
3   using namespace std;
4
5   int main(int argc, char **argv)
6   {
7       int a, b;
8       //  cin >> a >> b;
9       a = atoi(argv[1]);
10      b = atoi(argv[2]);
```

Exception has occurred.
Segmentation fault

```
11      cout << a + b << endl;
12      return 0;
13  } //! EX19.cpp
```

ex2.exe - 应用程序错误

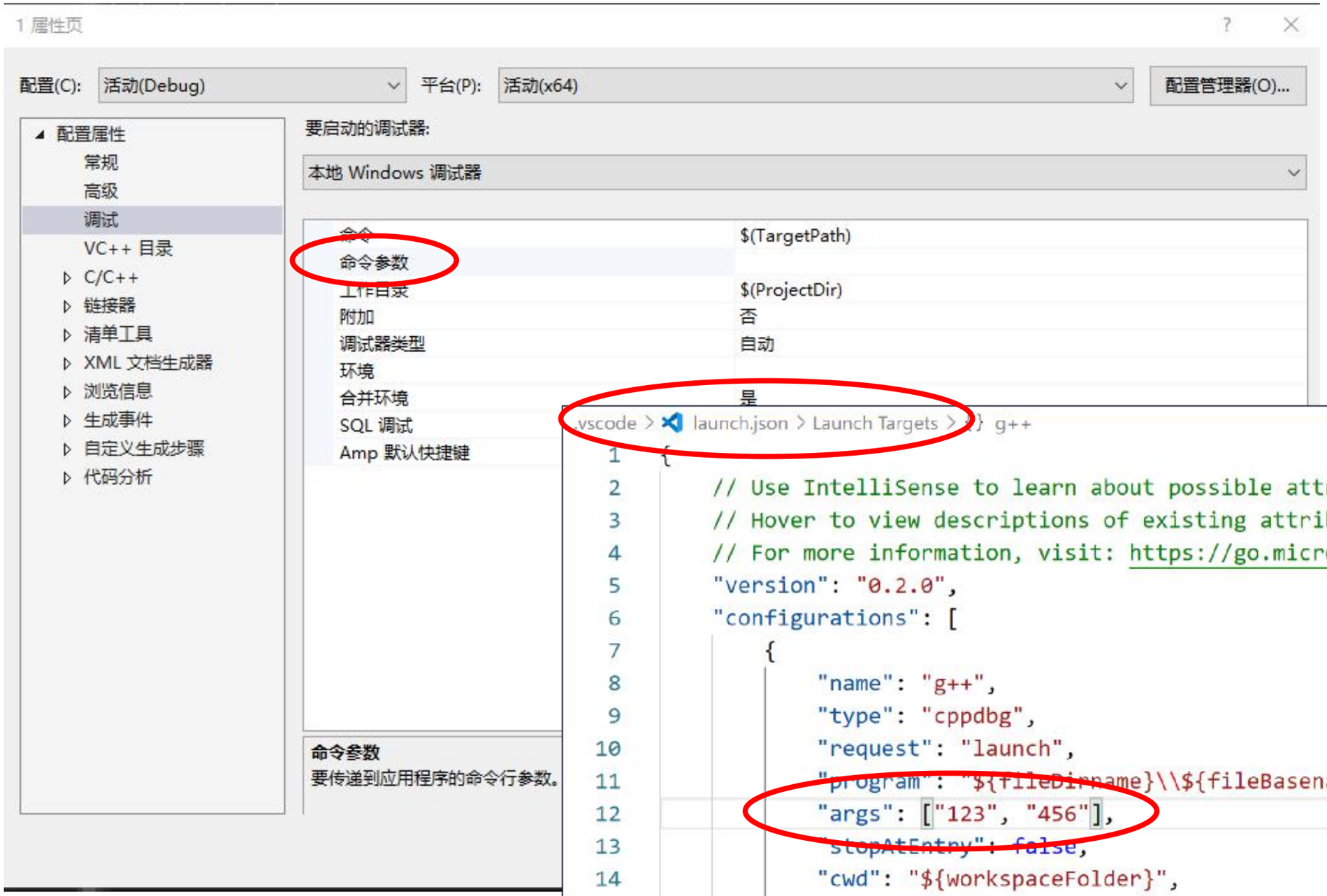
“0x00406987”指令引用的“0x00000000”内存。该内存不能为“read”。

要终止程序，请单击“确定”。
要调试程序，请单击“取消”。

确定　　取消

用IDE调试时，如何设置程序的命令行参数？

运行时没有机会输入！



Visual Studio 2019

VS Code

应对方法：“防御性编程”

代码对可能发生的“意外情况”要有预案

```
2  #include <iostream>
3  #include <cstdio>   // atoi()
4  int main(int argc, char** argv)
5  {
6      if (argc != 3)
7      {
8          std::cout << "Usage: " << argv[0]
9                    << " op1 op2" << std::endl;
10         return 1;
11     }
12
13     int a, b;
14 //  std::cin >> a >> b;
15     a = atoi(argv[1]);
16     b = atoi(argv[2]);
17     std::cout << a + b << std::endl;
18
19     return 0;
20 }
```

提供对用户友好的提示信息

多选题 1分

设置

在C++语言的表达式中，若运算符中出现了字符&，则它的含义是：

A 取操作数的地址

B 按二进制位求与的运算

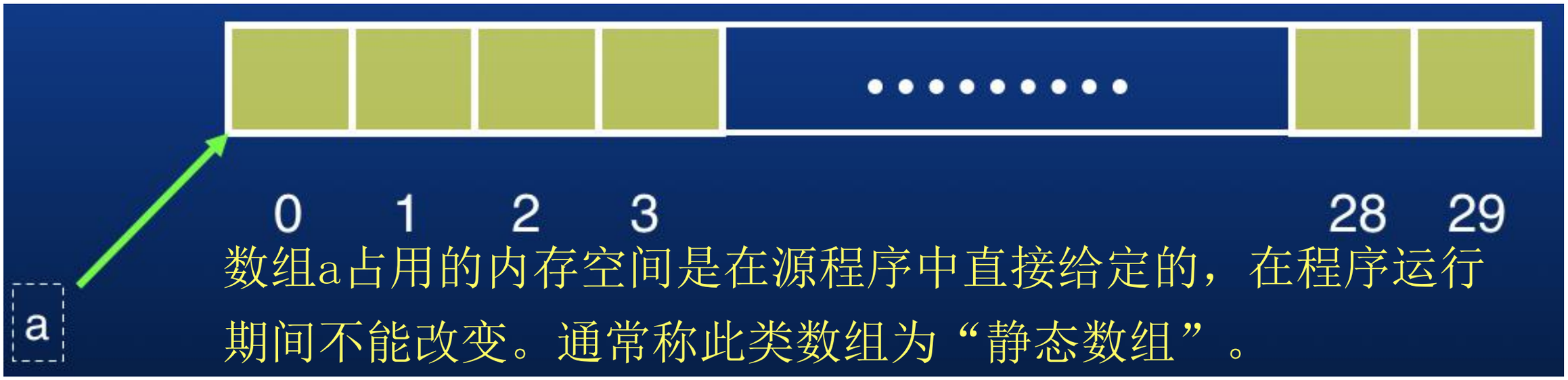
C 逻辑与运算符的一部分

D 听说是一种称为“引用”的东西 ☹

提交

数组名与指针变量的加减运算

int a[30]; // 数组a实际上一个常量指针，因此不能改变它




```
a     == &(a[0])
a+8   == &(a[8]) == &(a[0]) + 8
a+9   == &(a[8]) + 1
...
sizeof(a) == sizeof(int) * 30
```

数组元素在内存中的存放方式

```
#include <iostream>
using namespace std;

int main() {
    int a[30];
    for (int i=0; i<=30; i++)
        cout << i << ": " << &(a[i])
             << " - " << a+i << endl;

    return 0;
}
```

元素内容

元素地址

一维数组中的元素是依次存放的；下标相邻的单元，内存地址也相邻；第一个元素地址值最小。

参考输出

```
0: 0x22fec0 - 0x22fec0
1: 0x22fec4 - 0x22fec4
2: 0x22fec8 - 0x22fec8
3: 0x22fecc - 0x22fecc
4: 0x22fed0 - 0x22fed0
5: 0x22fed4 - 0x22fed4
6: 0x22fed8 - 0x22fed8
7: 0x22fedc - 0x22fedc
8: 0x22fee0 - 0x22fee0
9: 0x22fee4 - 0x22fee4
10: 0x22fee8 - 0x22fee8
11: 0x22feec - 0x22feec
......
26: 0x22ff28 - 0x22ff28
27: 0x22ff2c - 0x22ff2c
28: 0x22ff30 - 0x22ff30
29: 0x22ff34 - 0x22ff34
30: 0x22ff38 - 0x22ff38
```

数组变量和指针变量可以混用

```
#include <iostream>
using namespace std;
int main() {
    int *p, data[] = {1, 2, 3, 4, 5};
    for (int i=0; i<sizeof(data)/sizeof(data[0]); i++)
        cout << data[i] << ' ' << *(data+i) << endl;

    p = data;
    for (int i=0; i<sizeof(data)/sizeof(data[0]); i++)
        cout << p[i] << ' ' << *(p+i) << endl;

    return 0;
}
```

程序运行结果:

```
D:\>a
1 1
2 2
3 3
4 4
5 5
1 1
2 2
3 3
4 4
5 5
```

注意程序代码中求数组长度的编程技巧: sizeof(数组名)/sizeof(数组元素)

把二维数组当作一维数组使用

2D array ➔ 1D array

```
#include <iostream>
#include <iomanip>
using namespace std

void func0(int *data, int num)
{
    for (int i = 0; i < num; i++)
        cout << setw(2) << data[i] << ' ';
    cout << endl;
}

int main()
{
    // 可以用一维数组来初始化二维数组
    int data1[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
    func0((int *)data1, 12);  // 将二维数组当作一维数组来使用

    return 0;
}
```

形参data看起来像是一维数组变量

实参data1实际上是二维数组变量

把一维数组当作二维数组使用

1D array ➔ 2D array

```
#include <iostream>
#include <iomanip>
using namespace std;

void fun1(int data[][4], int rows) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < 4; j++)
            cout << setw(2) << data[i][j] << ' ';
        cout << endl;
    }
}

void fun2(int *data, int rows) {
    const int cols = 4; // 数组大小需用常量定义
    int(*ptr)[cols];    // 定义“指向数组的指针”
    ptr = (int(*)[cols])data;

    fun1(ptr, rows);
}

int main() {
    int data[12] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
    fun1((int(*)[4])(data), 3); // 将一维数组强制转换为“指向数组的指针”
    fun2(data, 3);              // 直接将一维数组作为函数参数

    return 0;
}
```

语法要求：作为函数形式参数的二维数组，必须**在声明中给出列的大小！**

借助于“**指向数组的指针**”将一维数组当作二维数组来使用

语法：int(*VAR)[SIZE]

```cpp
int main() {
    int a1[4] = {1, 2, 3}, a2[5] = {1, 2, 1, 2, 9};
    int *p = a1, *q; // 注意指针q的定义方式

    for (int i = 0; i < 4; i++) cout << p[i] << '=' << *(p + i) << ' ';
    cout << endl;

    for (p = a2 + 4; p >= a2; p--) cout << *p << ' '; // 指针用作循环变量
    cout << endl;

    q = a2;
    for (; q < a2 + 5; q++) cout << *q << ' ';
    cout << endl;

    p = a1; // 数组变量可以直接赋值给指针变量
    cout << "p = " << p << " a1 = " << a1 << endl;

    p = a2;
    cout << "      p = " << p          << ", a2    = " << a2     << endl;
    cout << "    p+1 = " << p + 1      << ", a2+1  = " << a2 + 1 << endl; // 注意 p+1 的实际值
    cout << " *(p+2) = " << *(p + 2)   << ", p[2]  = " << p[2]   << endl;
    cout << "*(a2+2) = " << *(a2 + 2) << ", a2[2] = " << a2[2]  << endl;
    return 0;
} // pointer-array-1.cpp
```

```
1=1 2=2 3=3 0=0
9 2 1 2 1
1 2 1 2 9
p = 0x22ff30 a1  = 0x22ff30
      p = 0x22ff10, a2    = 0x22ff10
    p+1 = 0x22ff14, a2+1  = 0x22ff14
 *(p+2) = 1, p[2]  = 1
*(a2+2) = 1, a2[2] = 1
```

填空题 4分

设置

下列代码输出4个变量的大小，这些变量的大小依次为：

[填空1] [填空2] [填空3] [填空4]

```
#include <iostream>
using namespace std;
int main() {
    int a1[4] = {1,2,3}, a2[5] = {1,2,1,2,9};
    int *p = a1, *q;
    cout << "sizeof(a1) = " << sizeof(a1) << endl;
    cout << "sizeof(a2) = " << sizeof(a2) << endl;
    cout << "sizeof(p)  = " << sizeof(p) << endl;
    cout << "sizeof(q)  = " << sizeof(q) << endl;
    return 0;
} // pointer-array-2.cpp
```

正常使用填空题需3.0以上版本雨课堂

作答

```
D:\Work>g++ pointer-array-2.cpp

D:\Work>a
sizeof(a1) = 16
sizeof(a2) = 20
sizeof(p)  = 8
sizeof(q)  = 8
```

编译器 G++

```
d:\Work>cl /EHsc pointer-array-2.cpp
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.27.29111 版
版权所有(C) Microsoft Corporation。保留所有权利。

pointer-array-2.cpp
Microsoft (R) Incremental Linker Version 14.27.29111.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:pointer-array-2.exe
pointer-array-2.obj

d:\Work>pointer-array-2
sizeof(a1) = 16
sizeof(a2) = 20
sizeof(p)  = 4
sizeof(q)  = 4
```

编译器 CL

```
#include <iostream>
using namespace std;

void test_1(int a[], int len) // 数组变量作函数参数
{
    cout << "test_1():  a = " << a << ", len = "
         << len << endl << "\t\t";
    for (int i=0; i<len; i++)
        cout << a[i] << '=' << *(a+i) << ' ';
    cout << endl;
}

void test_2(int* a, int len) // 指针变量作函数参数
{
    cout << "test_2():  a = " << a << ", len = "
         << len << endl << "\t\t";
    for (int i=0; i<len; i++)
        cout << a[i] << '=' << *(a+i) << ' ';
    cout << endl;
}
```

```
int main()
{
    int a1[4] = {1, 2, 3};
    int a2[5] = {1, 2, 1, 2, 9};

    cout << "main():    a1 = " << a1
         << ", a2 = " << a2 << endl;

    test_1(a1, 4);
    test_1(a2, 5);

    test_2(a1, 4);
    test_2(a2, 5);

    int* p = a2;
    test_1(p, 2);
    test_2(p, 3);

    return 0;
} // pointer-array-3.cpp
```

```
main():    a1 = 0x61fe00, a2 = 0x61fde0
test_1():  a = 0x61fe00, len = 4
                1=1 2=2 3=3 0=0
test_1():  a = 0x61fde0, len = 5
                1=1 2=2 1=1 2=2 9=9
test_2():  a = 0x61fe00, len = 4
                1=1 2=2 3=3 0=0
test_2():  a = 0x61fde0, len = 5
                1=1 2=2 1=1 2=2 9=9
test_1():  a = 0x61fde0, len = 2
                1=1 2=2
test_2():  a = 0x61fde0, len = 3
                1=1 2=2 1=1
```

投票 最多可选1项

设置

猜猜下面程序编译和运行结果

```
#include <iostream>
using namespace std;
void show(int array[5], int len) {
    for (int i=0; i<len; i++)
        cout << array[i] << ' ';
    cout << endl;
}
int main() {
    int data[6] = {1, 2, 3, 4, 5};
    show(data, 6);
    return 0;
}
```

A 编译不通过

B 编译通过，运行崩溃

C 编译运行一切正常

D 实在猜不出来（我需要上机试试）

提交

结论：数组作函数参数的两种形式等效

```
#include <iostream>
using namespace std;
void show(int array[], int len) {
    for (int i=0; i<len; i++)
        cout << array[i] << ' ';
    cout << endl;
}

int main() {
    int data[] = {1, 2, 3, 4, 5};
    show(data,
         sizeof(data)/sizeof(data[0]));
    return 0;
} // version 1.0
```

```
#include <iostream>
using namespace std;
void show(int* array, int len) {
    for (int i=0; i<len; i++)
        cout << array[i] << ' ';
    cout << endl;
}

int main() {
    int data[] = {1, 2, 3, 4, 5};
    show(data, 5);

    return 0;
} // version 2.0
```

单选题 1分

设置

函数show能输出array数组中的所有元素吗？ A 可以 B 不能

```
#include <iostream>
using namespace std;

void show(int array[]) {
    int len = sizeof(array)/sizeof(array[0]);
    for (int i=0; i<len; i++)
        cout << array[i] << ' ';
    cout << endl;
}

int main() {
    int data[] = {1, 2, 3, 4, 5};
    show(data);
    return 0;
}
```

提交

20221119 > C++ EX03.cpp > ...

```cpp
#include <iostream>
using namespace std;

void show(int array[])
{
    int len = sizeof(array) / sizeof(array[0]);
    for (int i = 0; i < len; i++)
        cout << array[i] << ' ';
    cout << endl;
}

int main()
{
    int data[] = {1, 2, 3, 4, 5};
    show(data);
    return 0;
}

```

终端

```
1 2
PS D:\FOP.VS.Projects>
```

```
#include <iostream>
using namespace std;

void show(int a[], int len) // 数组变量作函数参数
{
    cout << "show():  a = " << a << ", len = "
         << len << endl << "\t\t";
    for (int i=0; i<len; i++)
        cout << a[i] << '=' << *(a+i) << ' ';
    cout << endl;
}
```

```
void test_3(int a[], int len) // 数组参数a中的元素内容被更新了!
{
    cout << "test_3():  a = " << a << ", len = " << len << endl;
    for (int i=0; i<len; i++)
    {
        cout << "\t\ta[" << i << "] = " << a[i] << " ==> ";
        a[i] = a[i] + i * 2;   // 在这里把数组元素更新了
        cout << a[i] << endl;
    }
    cout << endl;
}
```

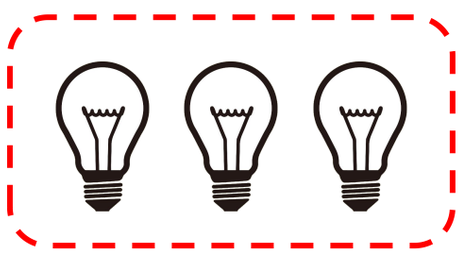
```
void test_4(int a[], int len, int b[])
{
    cout << "test_4():  a = " << a << ", len = " << len
         << " b = " << b << endl << "\t\t";
    for (int i=0; i<len; i++)
        b[i] = a[i] + i * 2;  // 运算结果赋值给了参数数组 b
    cout << endl;
}

int main()
{
    int a1[4] = {1, 2, 3}; // 第4个元素自动为0
    int a2[5] = {1, 2, 1, 2, 9};
    test_3(a1, 4);
    show(a1, 4);

    test_4(a1, 4, a2); // 数组a2内容发生改变
    show(a1, 4);
    show(a2, 5);

    return 0;
} // pointer-array-4.cpp
```

```
test_3():  a = 0x61fe10, len = 4
                a[0] = 1 ==> 1
                a[1] = 2 ==> 4
                a[2] = 3 ==> 7
                a[3] = 0 ==> 6

show():  a = 0x61fe10, len = 4
                1=1 4=4 7=7 6=6
test_4():  a = 0x61fe10, len = 4 b = 0x61fdf0

show():  a = 0x61fe10, len = 4
                1=1 4=4 7=7 6=6
show():  a = 0x61fdf0, len = 5
                1=1 6=6 11=11 12=12 9=9
```

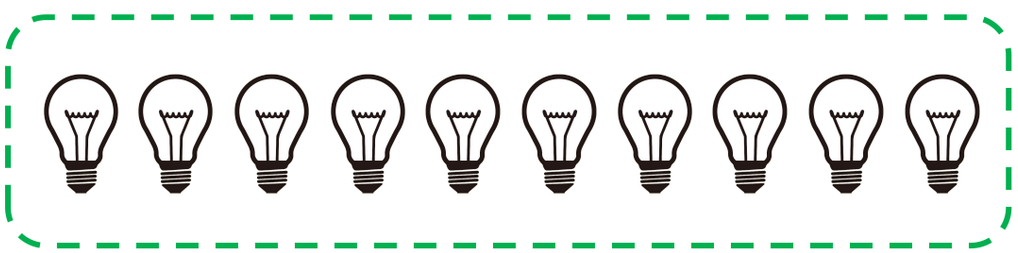
【思考题】用灯泡的点亮与熄灭来表达数值

如果想表达0~999之间的数，需要多少灯泡？

3

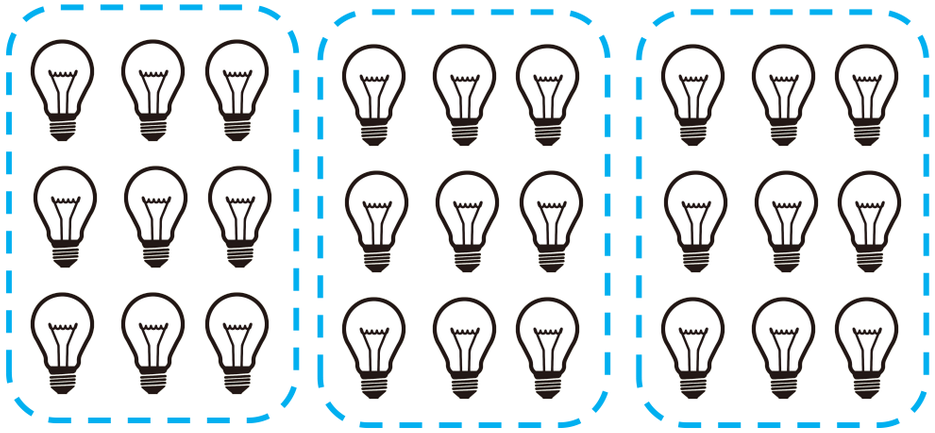


⇨ 999盏灯



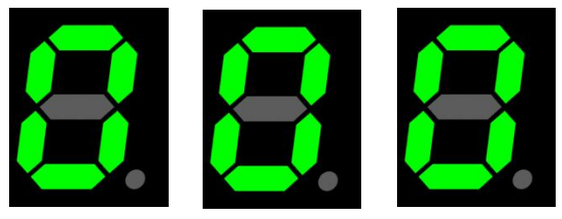
⇨ 10盏灯

999

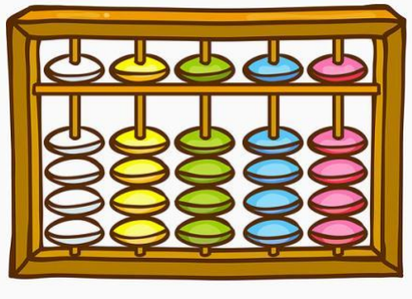


百　　十　　个

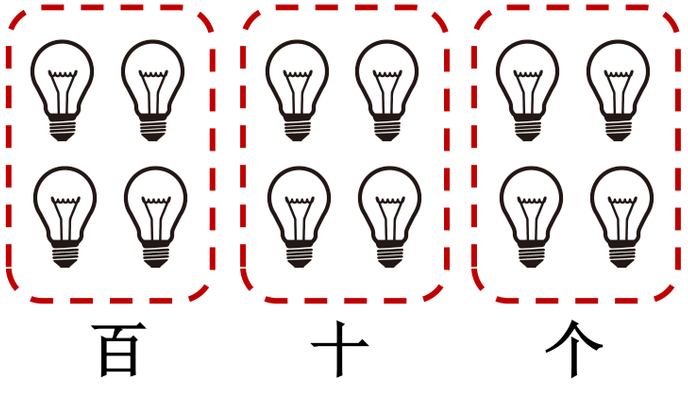
⇨ 27盏灯



⇨ 21盏灯



⇨ 15盏灯



百　　十　　个

⇨ 12盏灯

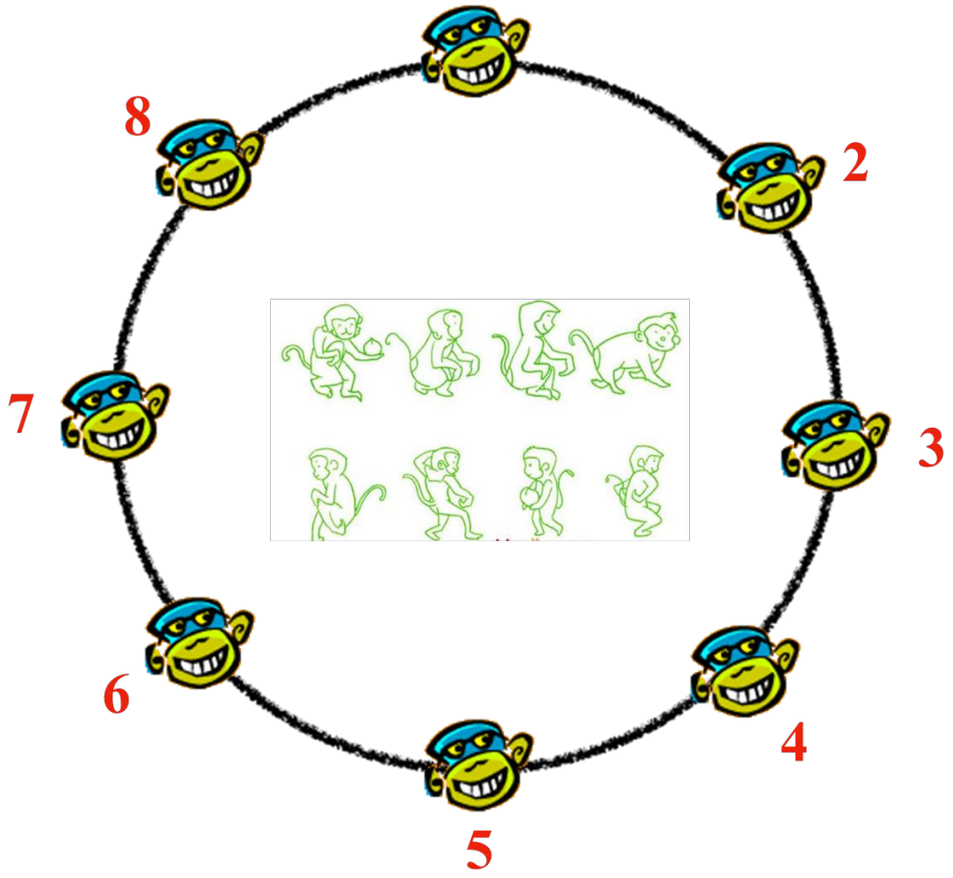
【重构练习】猴子选大王 —— 循环链表版

有n 只猴子围成一圈，顺时针方向从 1 到 n 编号。之后从 1 号开始沿顺时针方向让猴子从 1，2，…，m 依次报数。凡报到 m 的猴子，都让其出圈，取消候选资格。然后不停地按顺时针方向逐一让报出 m 者出圈，最后剩下一个就是猴王。

请你编写程序模拟这个过程，即按取消候选资格的次序，依次输出被淘汰的猴子编号，以及最后剩下的猴王的编号。

猴子被淘汰的顺序

3 6 1 5 2 8 4



【解题技巧】观察实例

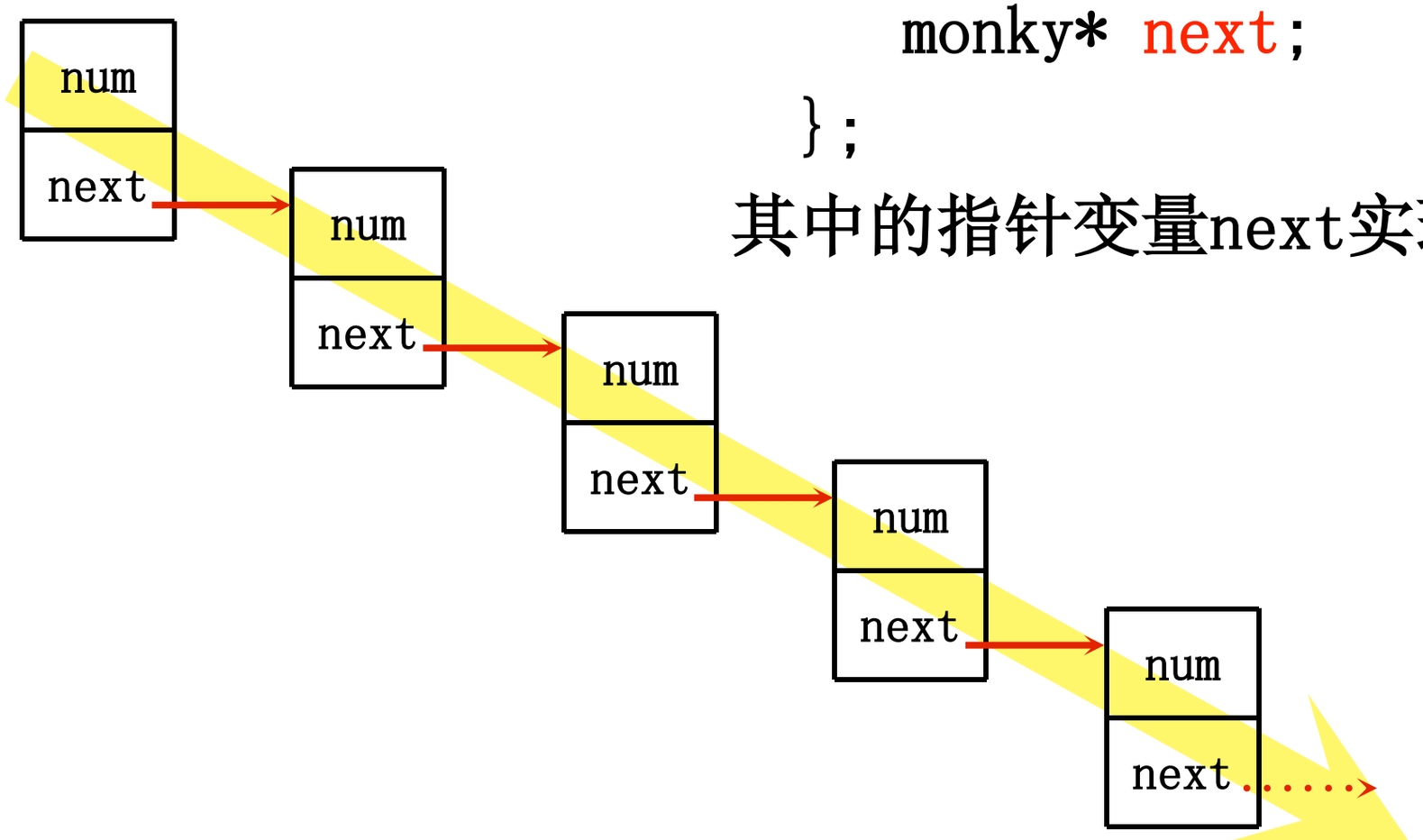
（以n=8，m=3为例）

使用结构定义链表节点的类型

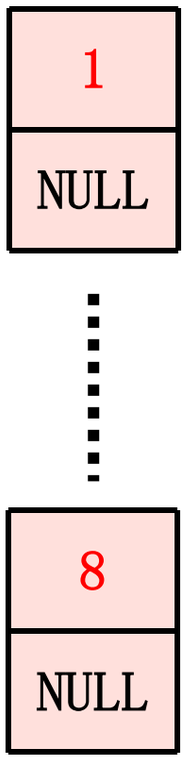
下面的结构定义了链表中每个节点的类型：

```
struct monkey
{
    int num;
    monky* next;
};
```

其中的指针变量next实现了节点之间的衔接。



循环链表

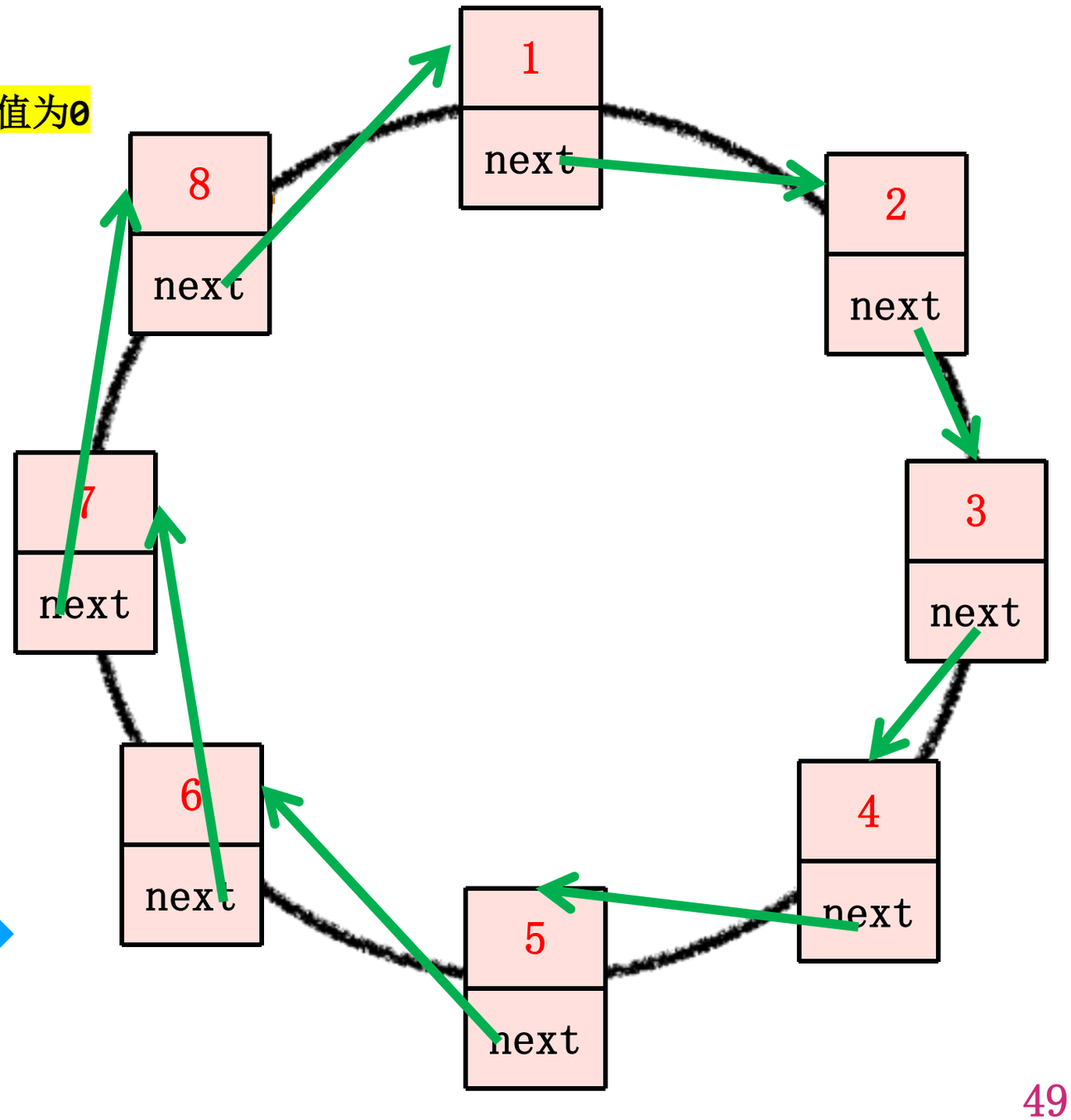



```
monkey *p;
p = new monkey;
p->num = 1;
p->next = NULL; // NULL为空指针，值为0
```

```
p = new monkey;
p->num = 8;
p->next = NULL;
```

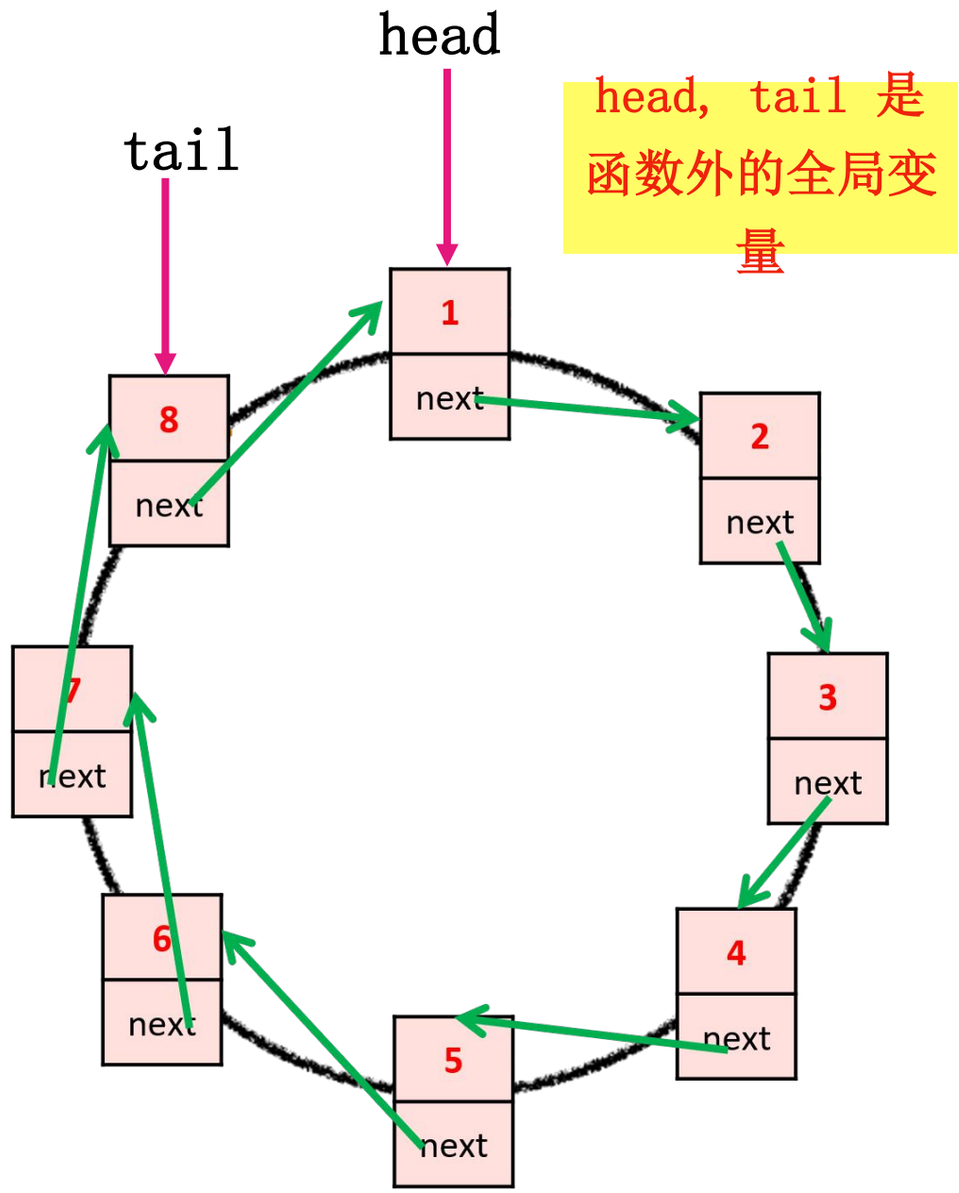
```
monkey *p;
for (int i=1; i<=8; i++) {
    p = new monkey;
    p->num = i;
    p->next = NULL; /// ????
}
```

???



创建循环链表

```
//【创建有nn个结点的循环链表】
void create(int nn) {
    monkey *p, *q;

    p = new monkey;
    p->num = 1;
    p->next = NULL;

    head = p;  // 全局变量head为链表头，赋值为p
    q = p;     // 局部变量q为当前链表尾，赋值为p
    for (int i=2; i<=nn; i++) {
        p = new monkey;
        p->num = i;
        p->next = NULL;// 链表尾部指向空
        q->next = p;   // 将p结点加到链表尾部
        q = p;         // 让q指向链表尾部结点
    }
    tail = q; // 全局变量tail为链表尾
    tail->next = head; // 链表尾部指向链表头，形成了“循环链表”！
}
```



按推选规则，逐步删除链表节点

```
// mm表示结点删除间隔
void select(int mm) {
    int x = 0;
    monkey *p, *q;

    q = tail;       // q 指向循环链表尾部
    do {
        p = q->next;          // p赋值为q相邻的下一个结点
        x++;
        if (x % mm == 0) {  // 表示是否跳过了指定的间隔
            cout << "被删掉的猴子号为" << p->num << "号\n";
            q->next = p->next; // 删除此结点
            delete p;
            p = NULL;
        }
        else
            q = p;                  // q指向相邻的下一个结点p
    } while (q != q->next);       // 剩余结点数不为1，则继续循环
    head = q;   // head指向结点q，q为链表中剩余的一个结点
}
```

链表节点定义、程序输入输出

```
#include <iostream>
using namespace std;

struct monkey {
    int num;              // 猴子号
    monkey *next;         // monkey结构指针
} *head, *tail; // 这是结构类型，结构指针变量同时定义的情形
void create(int nn);
void select(int mm);

int main() {
    int n, m;
    head = NULL;          // 初始化head为NULL，空指针，值为0
    cout << "请输入猴子数\n";
    cin >> n;  // 输入待插入结点数据
    cout << "请输入间隔m\n";
    cin >> m;
    create(n);  // 调用函数create，建立循环链表
    select(m);  // 调用函数select，找出剩下的猴子
    cout << "猴王是" << head->num << "号\n";
    return 0;
}
```

```
请输入猴子数
8
请输入间隔m
3
被删掉的猴子号为3号
被删掉的猴子号为6号
被删掉的猴子号为1号
被删掉的猴子号为5号
被删掉的猴子号为2号
被删掉的猴子号为8号
被删掉的猴子号为4号
猴王是7号
```

课后思考题

吕洞宾不能坐首位。传说八仙到天宫拜会王母娘娘，观音菩萨也在。王母娘娘安排八仙在八仙桌旁就座。为了表示公平，王母娘娘让八仙先排成一圈，然后用两粒骰子掷点数，共掷出几点，就从第一人开始数起，依次数到这个点数，这个人就坐末位，依次周而复始，直到**最后一人成为八仙之首，坐首位**。

由于观音菩萨很看不惯吕洞宾平时的一些行为，不想让他坐在首位，那么，应该如何安排吕洞宾在圆圈中的位置，使得**无论骰子投出的点数如何**，都不会让吕洞宾坐到首位呢?

请你编写程序，帮助观音菩萨把吕洞宾在排成圆圈时的位置想出来。

```
#include <iostream>
using namespace std;
struct Message
{
    char words[50];
    Message* next;
} msg = {"END. See you later!"};

int main()
{
    Message *ptr = &msg;
    cout << ptr->words << endl;
    return 0;
}
```