程序设计基础

教学团队：徐明星，兴军亮，任炬

renju@tsinghua.edu.cn

2024秋，每周一第2节，三教2301

疑问解答与作业回顾

【调试技巧】练就“火眼金睛”，让程序错误“现原形”！

代码哪里错了？

debug.cpp

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

int main()
{

    int k;
 int g=0;
 char thisman;
 int sum;
 for ( k=0;k<4;k++);
 {

 thisman ='A'+k;

    sum=(thisman!='A')
     +(thisman=='C')
     +(thisman=='D')
     +(thisman!='D');

     if (sum==3)
   {
  cout << "做好事的人为" << thisman <<endl;
  g=1;
   }


     if(g!=1)
     {
     cout << "找不到做好事的人" <<endl;
     }

  return 0;
    }
}
```

【调试技巧】阅读编译警告信息，发现错误或隐患

```
~$ g++ debug.cpp
debug.cpp:11:20: warning: for loop has empty body [-Wempty-body]
 for ( k=0;k<4;k++);
                   ^
debug.cpp:11:20: note: put the semicolon on a separate line to silence this warning
1 warning generated.
~$
```

```
11    for ( k=0;k<4;k++);
12    {
```

去掉循环语句后面多余的分号

```
~$ g++ debug1.cpp
~$ a.out
找不到做好事的人
~$
```

【调试技巧】调整源程序缩进格式，发现算法逻辑错误

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

int main()
{

    int k;
    int g = 0;
    char thisman;
    int sum;
    for (k = 0; k < 4; k++)
    {
        thisman = 'A' + k;

        sum = (thisman != 'A')
            +(thisman == 'C')
            +(thisman == 'D')
            +(thisman != 'D');

        if (sum == 3)
        {
            cout << "做好事的人为" << thisman <<endl;
            g=1;
        }

        if (g != 1)
        {
            cout << "找不到做好事的人" <<endl;
        }

        return 0;
    }
}
```

源程序格式规范化，问题就变得显然了。

移到循环的后面，程序就运行正确了

“勿以恶小而为之，勿以善小而不为。”

《三国志·蜀书·先主传》

（蜀汉昭烈帝刘备遗诏）

“火眼金睛” —— 代码风格美化工具

各种IDE均提供代码排版风格调整的功能

Artistic Style 3.1

A Free, Fast, and Small Automatic Formatter
for C, C++, C++/CLI, Objective-C, C#, and Java Source Code

http://astyle.sourceforge.net/astyle.html

```
astyle L04-01.cpp --style=bsd --convert-tabs --indent=spaces=4 --attach-closing-while --indent-switches --indent-namespaces --indent-continuation=4 --indent-preproc-block --indent-preproc-define --indent-preproc-cond --indent-col1-comments --pad-oper --unpad-paren --delete-empty-lines --align-pointer=name --align-reference=name --break-elseifs --add-braces --pad-comma --add-one-line-braces --pad-header
```

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

int main()
{

    int k;
  int g = 0;
  char thisman;
  int sum;
  for (k = 0; k < 4; k++)
  {

  thisman = 'A' + k;

     sum = (thisman != 'A')
      +(thisman == 'C')
      +(thisman == 'D')
      +(thisman != 'D');

      if (sum == 3)
    {
   cout << "做好事的人为" << thisman <<endl;
   g=1;
    }


      if (g != 1)
      {
      cout << "找不到做好事的人" <<endl;
      }

    return 0;
      }
} // D04-01.cpp
```

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

int main()
{
    int k;
    int g = 0;
    char thisman;
    int sum;
    for (k = 0; k < 4; k++)
    {
        thisman = 'A' + k;
        sum = (thisman != 'A')
            + (thisman == 'C')
            + (thisman == 'D')
            + (thisman != 'D');
        if (sum == 3)
        {
            cout << "做好事的人为" << thisman << endl;
            g = 1;
        }
        if (g != 1)
        {
            cout << "找不到做好事的人" << endl;
        }
        return 0;
    }
} // D04-01B.cpp
```

任务：输出日历

上机实验三

实验任务

如下是某一年 1 月到 4 月的日历。

```
01                      1
   2  3  4  5  6  7  8
   9 10 11 12 13 14 15
  16 17 18 19 20 21 22
  23 24 25 26 27 28 29
02 30 31  1  2  3  4  5
   6  7  8  9 10 11 12
  13 14 15 16 17 18 19
  20 21 22 23 24 25 26
03 27 28 29  1  2  3  4
   5  6  7  8  9 10 11
  12 13 14 15 16 17 18
  19 20 21 22 23 24 25
04 26 27 28 29 30 31  1
   2  3  4  5  6  7  8
   9 10 11 12 13 14 15
  16 17 18 19 20 21 22
  23 24 25 26 27 28 29
```

请参照上述排版格式（最左列绿色数字表示月份，需要输出；每行从左向右对应的是周一至周日；示例中的底色与字体颜色是为了方便观察，实验只要求输出纯文本，不需要设置字体颜色），编写程序，在运行时根据用户输入的指定年份（要求年份范围为 2023~2050）输出全年 12 个月的日历（从 1 月 1 日到 12 月 31 日）。

除“2023.01.01 是周日”可以作为已知信息使用外，程序中不得直接使用其他“某年某月某日是星期几”的信息。

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

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }
```

然后，怎么写？

蛮力输出！

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

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }
```

```
    if (year == 2023)
    {
        cout << "01                    1" << endl;
        cout << "    2  3  4  5  6  7  8" << endl;
        cout << "    9 10 11 12 13 14 15" << endl;
        cout << "   16 17 18 19 20 21 22" << endl;
        cout << "   23 24 25 26 27 28 29" << endl;
        cout << "02 30 31  1  2  3  4  5" << endl;
        cout << "    6  7  8  9 10 11 12" << endl;
        cout << "   13 14 15 16 17 18 19" << endl;
        cout << "   20 21 22 23 24 25 26" << endl;
        cout << "03 27 28  1  2  3  4  5" << endl;
        cout << "    6  7  8  9 10 11 12" << endl;
        cout << "   13 14 15 16 17 18 19" << endl;
        cout << "   20 21 22 23 24 25 26" << endl;
        cout << "04 27 28 29 30 31  1  2" << endl;
        cout << "    3  4  5  6  7  8  9" << endl;
        cout << "   10 11 12 13 14 15 16" << endl;
        cout << "   17 18 19 20 21 22 23" << endl;
        cout << "   24 25 26 27 28 29 30" << endl;
        cout << "05  1  2  3  4  5  6  7" << endl;
        cout << "    8  9 10 11 12 13 14" << endl;
        cout << "   15 16 17 18 19 20 21" << endl;
        cout << "   22 23 24 25 26 27 28" << endl;
        cout << "06 29 30 31  1  2  3  4" << endl;
        cout << "    5  6  7  8  9 10 11" << endl;
```

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

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }
```

```cpp
    if (year == 2023)
    {
        cout << "01                    1" << endl;
        cout << "    2  3  4  5  6  7  8" << endl;
        cout << "    9 10 11 12 13 14 15" << endl;
        cout << "   16 17 18 19 20 21 22" << endl;
        cout << "   23 24 25 26 27 28 29" << endl;
        cout << "02 30 31  1  2  3  4  5" << endl;
        cout << "    6  7  8  9 10 11 12" << endl;
        cout << "   13 14 15 16 17 18 19" << endl;
        cout << "   20 21 22 23 24 25 26" << endl;
        cout << "03 27 28  1  2  3  4  5" << endl;
        cout << "    6  7  8  9 10 11 12" << endl;
        cout << "   13 14 15 16 17 18 19" << endl;
        cout << "   20 21 22 23 24 25 26" << endl;
        cout << "04 27 28 29 30 31  1  2" << endl;
        cout << "    3  4  5  6  7  8  9" << endl;
        cout << "   10 11 12 13 14 15 16" << endl;
        cout << "   17 18 19 20 21 22 23" << endl;
        cout << "   24 25 26 27 28 29 30" << endl;
        cout << "05  1  2  3  4  5  6  7" << endl;
        cout << "    8  9 10 11 12 13 14" << endl;
        cout << "   15 16 17 18 19 20 21" << endl;
        cout << "   22 23 24 25 26 27 28" << endl;
        cout << "06 29 30 31  1  2  3  4" << endl;
        cout << "    5  6  7  8  9 10 11" << endl;
```

然后，怎么办？

愚公移山！

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

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }
```

```
    cout << "    23 24 25 26 27 28 29 30 31" << endl;
    cout << "08  1  2  3  4  5  6  7" << endl;
    cout << "    8  9 10 11 12 13 14" << endl;
    cout << "   15 16 17 18 19 20 21" << endl;
    cout << "   22 23 24 25 26 27 28" << endl;
    cout << "09 29 30 31  1  2  3  4" << endl;
    cout << "    5  6  7  8  9 10 11" << endl;
    cout << "   12 13 14 15 16 17 18" << endl;
    cout << "   19 20 21 22 23 24 25" << endl;
    cout << "10 26 27 28 29 30  1  2" << endl;
    cout << "    3  4  5  6  7  8  9" << endl;
    cout << "   10 11 12 13 14 15 16" << endl;
    cout << "   17 18 19 20 21 22 23" << endl;
    cout << "   24 25 26 27 28 29 30" << endl;
    cout << "11 31  1  2  3  4  5  6" << endl;
    cout << "    7  8  9 10 11 12 13" << endl;
    cout << "   14 15 16 17 18 19 20" << endl;
    cout << "   21 22 23 24 25 26 27" << endl;
    cout << "12 28 29 30  1  2  3  4" << endl;
    cout << "    5  6  7  8  9 10 11" << endl;
    cout << "   12 13 14 15 16 17 18" << endl;
    cout << "   19 20 21 22 23 24 25" << endl;
    cout << "   26 27 28 29 30 31" << endl;
    }

    return 0;
}
```

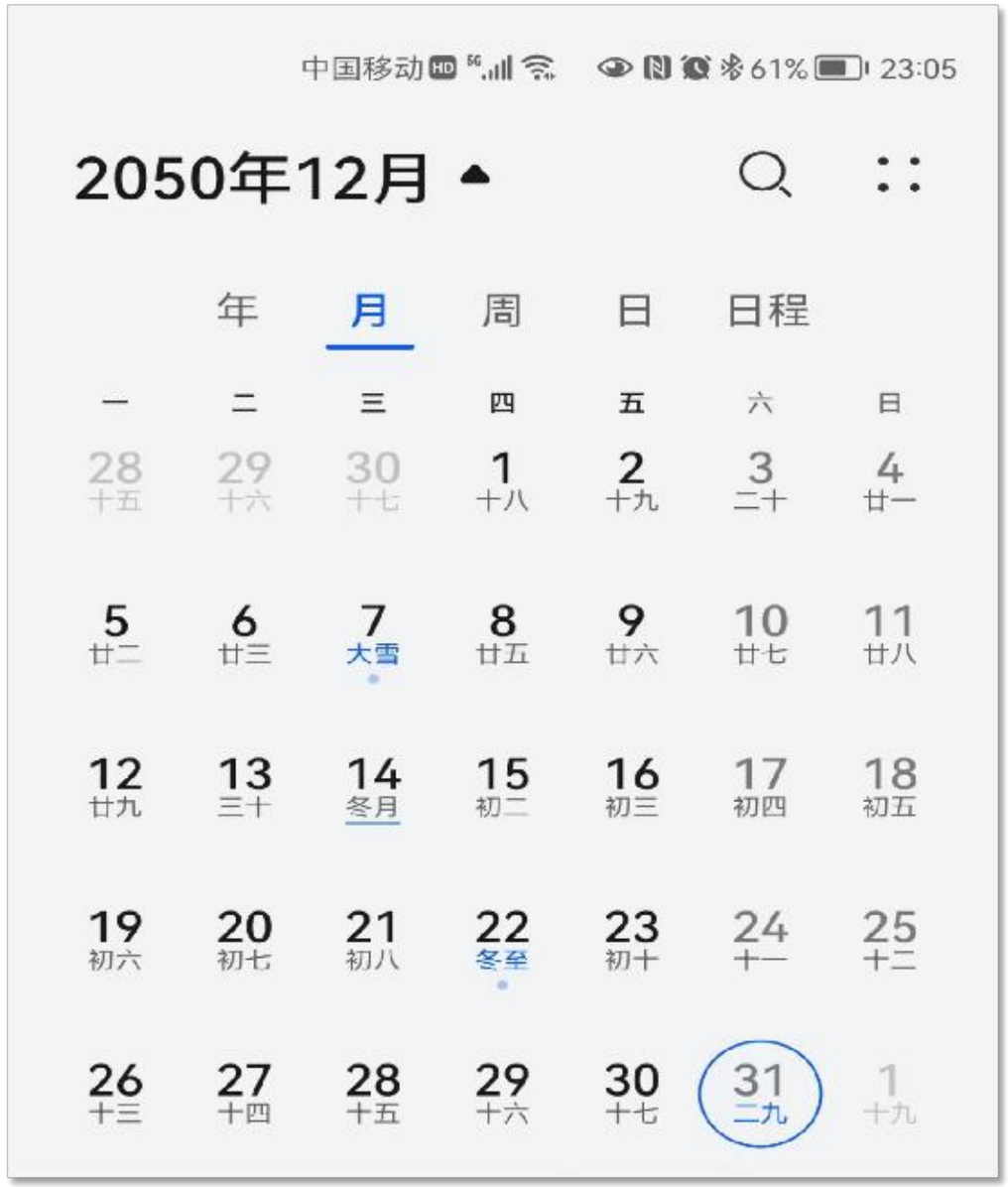
从2023到2050，一共才28年，不长 ☺ ☺ ☺

calender-v0.cpp > main()

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

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }

    if (year == 2023)
    {
        cout << "01                    1" << endl;
        cout << "    2  3  4  5  6  7  8" << endl;
        cout << "    9 10 11 12 13 14 15" << endl;
        cout << "   16 17 18 19 20 21 22" << endl;
        cout << "   23 24 25 26 27 28 29" << endl;
        cout << "02 30 31  1  2  3  4  5" << endl;
        cout << "    6  7  8  9 10 11 12" << endl;
        cout << "   13 14 15 16 17 18 19" << endl;
        cout << "   20 21 22 23 24 25 26" << endl;
        cout << "03 27 28  1  2  3  4  5" << endl;
        cout << "   25 26 27 28 29 30 31" << endl;
        cout << "08  1  2  3  4  5  6  7" << endl;
        cout << "    8  9 10 11 12 13 14" << endl;
        cout << "   15 16 17 18 19 20 21" << endl;
        cout << "   22 23 24 25 26 27 28" << endl;
        cout << "09 29 30 31  1  2  3  4" << endl;
        cout << "    5  6  7  8  9 10 11" << endl;
        cout << "   12 13 14 15 16 17 18" << endl;
        cout << "   19 20 21 22 23 24 25" << endl;
        cout << "10 26 27 28 29 30  1  2" << endl;
        cout << "    3  4  5  6  7  8  9" << endl;
        cout << "   10 11 12 13 14 15 16" << endl;
        cout << "   17 18 19 20 21 22 23" << endl;
        cout << "   24 25 26 27 28 29 30" << endl;
        cout << "11 31  1  2  3  4  5  6" << endl;
        cout << "    7  8  9 10 11 12 13" << endl;
        cout << "   14 15 16 17 18 19 20" << endl;
        cout << "   21 22 23 24 25 26 27" << endl;
        cout << "12 28 29 30  1  2  3  4" << endl;
        cout << "    5  6  7  8  9 10 11" << endl;
        cout << "   12 13 14 15 16 17 18" << endl;
        cout << "   19 20 21 22 23 24 25" << endl;
        cout << "   26 27 28 29 30 31" << endl;
    }

    return 0;
} // 2022.10
```

1616行

```
1590        cout << "   25 26 27 28 29 30 31" << endl;
1591        cout << "08  1  2  3  4  5  6  7" << endl;
1592        cout << "    8  9 10 11 12 13 14" << endl;
1593        cout << "   15 16 17 18 19 20 21" << endl;
1594        cout << "   22 23 24 25 26 27 28" << endl;
1595        cout << "09 29 30 31  1  2  3  4" << endl;
1596        cout << "    5  6  7  8  9 10 11" << endl;
1597        cout << "   12 13 14 15 16 17 18" << endl;
1598        cout << "   19 20 21 22 23 24 25" << endl;
1599        cout << "10 26 27 28 29 30  1  2" << endl;
1600        cout << "    3  4  5  6  7  8  9" << endl;
1601        cout << "   10 11 12 13 14 15 16" << endl;
1602        cout << "   17 18 19 20 21 22 23" << endl;
1603        cout << "   24 25 26 27 28 29 30" << endl;
1604        cout << "11 31  1  2  3  4  5  6" << endl;
1605        cout << "    7  8  9 10 11 12 13" << endl;
1606        cout << "   14 15 16 17 18 19 20" << endl;
1607        cout << "   21 22 23 24 25 26 27" << endl;
1608        cout << "12 28 29 30  1  2  3  4" << endl;
1609        cout << "    5  6  7  8  9 10 11" << endl;
1610        cout << "   12 13 14 15 16 17 18" << endl;
1611        cout << "   19 20 21 22 23 24 25" << endl;
1612        cout << "   26 27 28 29 30 31" << endl;
1613    }
1614
1615    return 0;
1616} // 2022.10
```



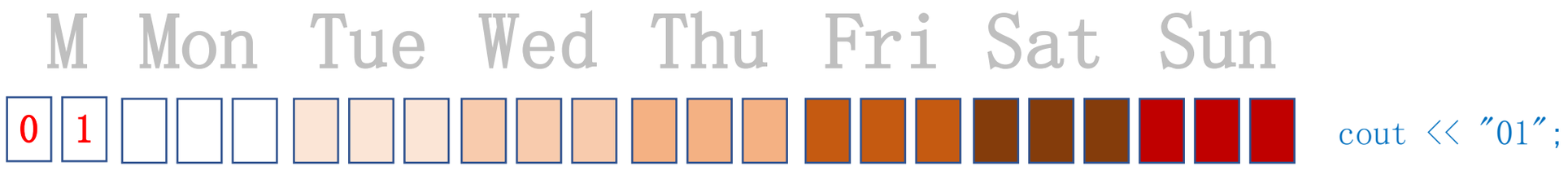
Version 1.0

C++ calender-v0.cpp > ...

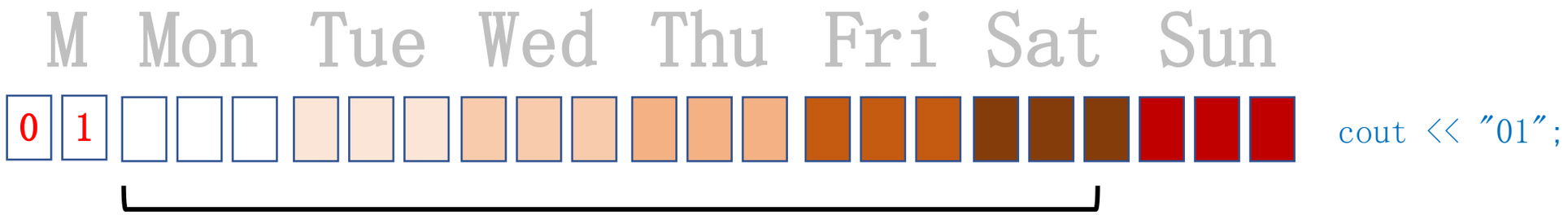
```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "请输入年份(2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "输入年份超出范围!\n";
        return 1;
    }

    if (year == 2023)
    {
        cout << "01                      1" << endl;
        cout << "    2  3  4  5  6  7  8" << endl;
        cout << "    9 10 11 12 13 14 15" << endl;
        cout << "   16 17 18 19 20 21 22" << endl;
        cout << "   23 24 25 26 27 28 29" << endl;
```

这样写是不是有点太坑了？

我们人类是如何制作日历的呢？



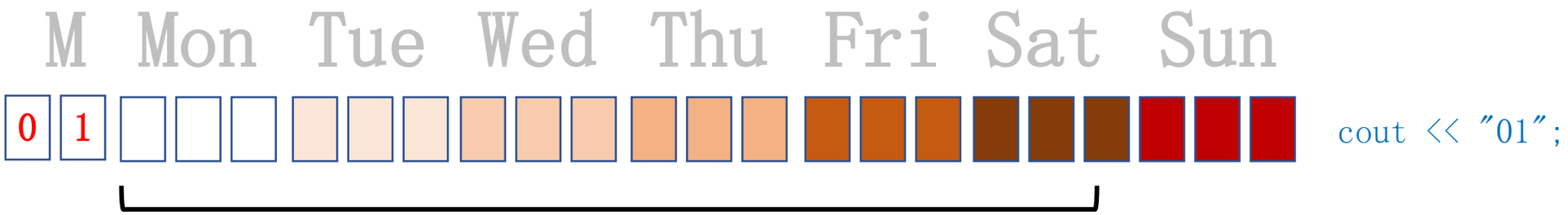
输出1月的序号



上一年所占第一周的天数？

（本年1月1日星期几？）

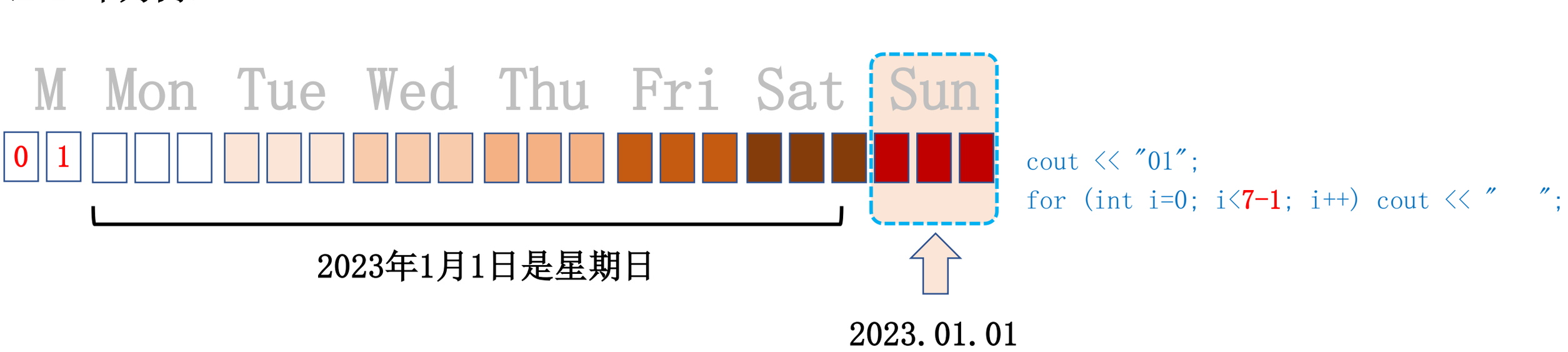
输出1月1日前的空白（如果有的话）



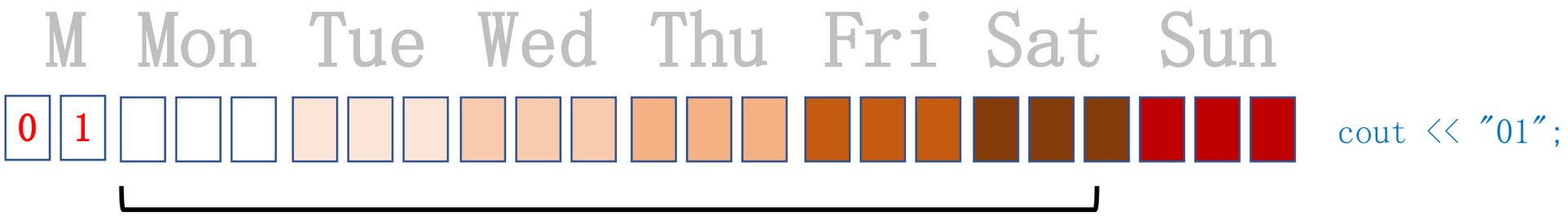
上一年所占第一周的天数？

（本年1月1日星期几？）

以2023年为例：



输出1月1日前的空白（如果有的话）




```
cout << "01";
```

上一年所占第一周的天数？

（本年1月1日星期几？）

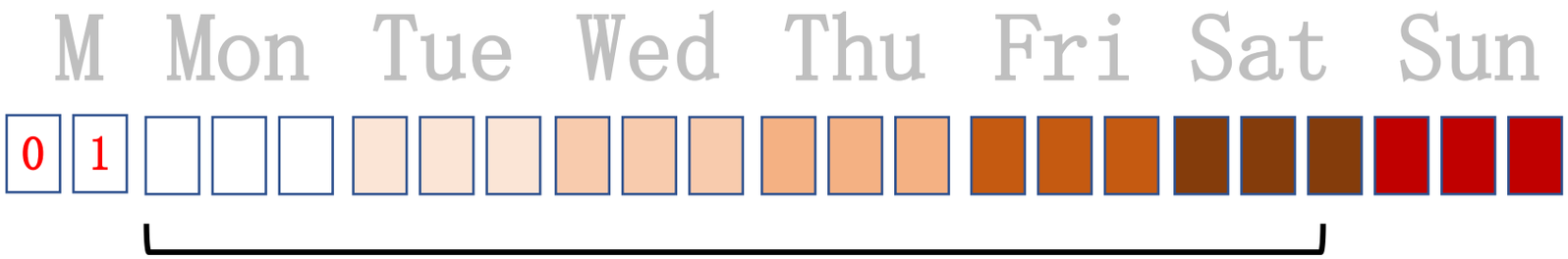
以2023年为例：

2023年1月1日是星期日

2023.01.01

```
cout << "01";
for (int i=0; i<7-1; i++) cout << "    ";
printf("%3d", 1);
```

输出1月1日

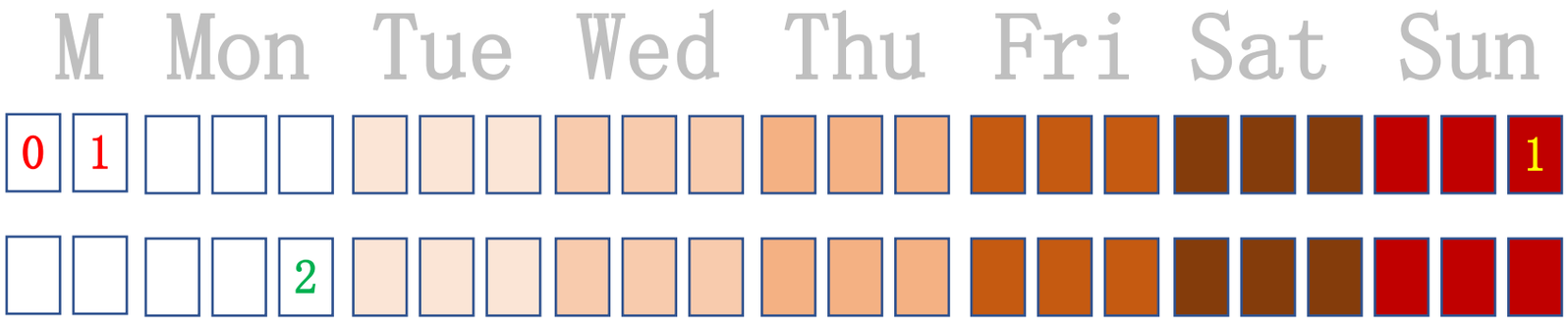



```
cout << "01";
```

上一年所占第一周的天数？

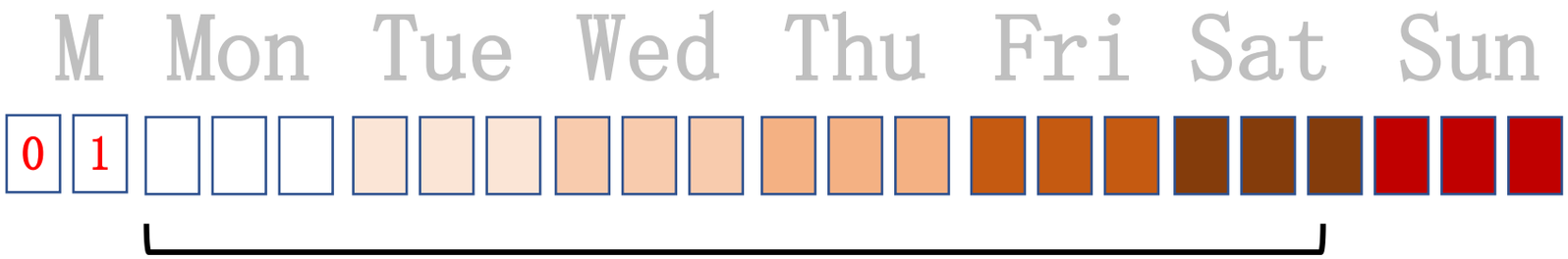
（本年1月1日星期几？）

以2023年为例：




```
cout << "01";
for (int i=0; i<7-1; i++) cout << "   ";
printf("%3d", 1);
cout << endl << "  "; // 2 spaces
printf("%3d", 2);
```

输出换行，输出2日

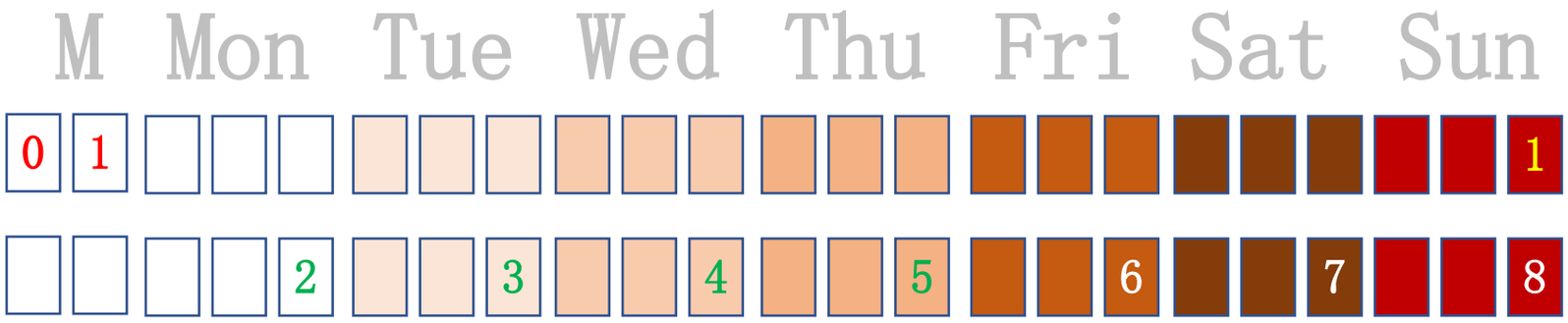



```
cout << "01";
```

上一年所占第一周的天数？

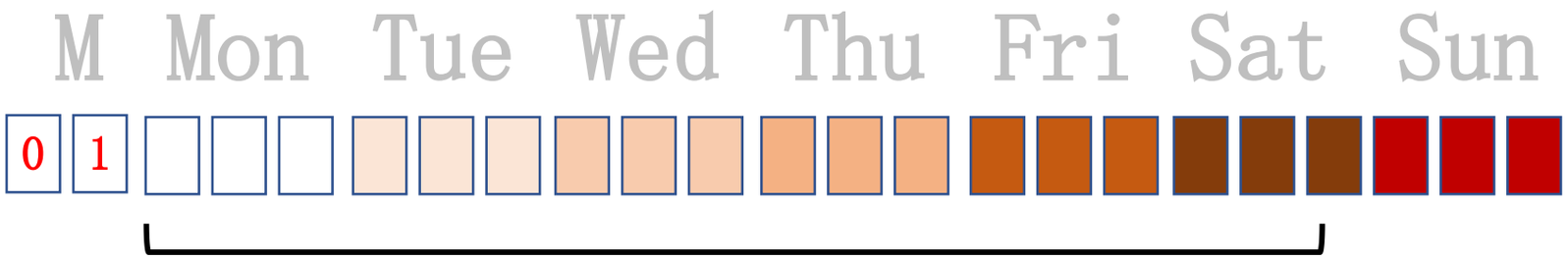
（本年1月1日星期几？）

以2023年为例：




```
cout << "01";
for (int i=0; i<7-1; i++) cout << "   ";
printf("%3d", 1);
cout << endl << "  "; // 2 spaces
for (int d=2; d<=8; d++)
    printf("%3d", d);
```

输出2日那一周（行）

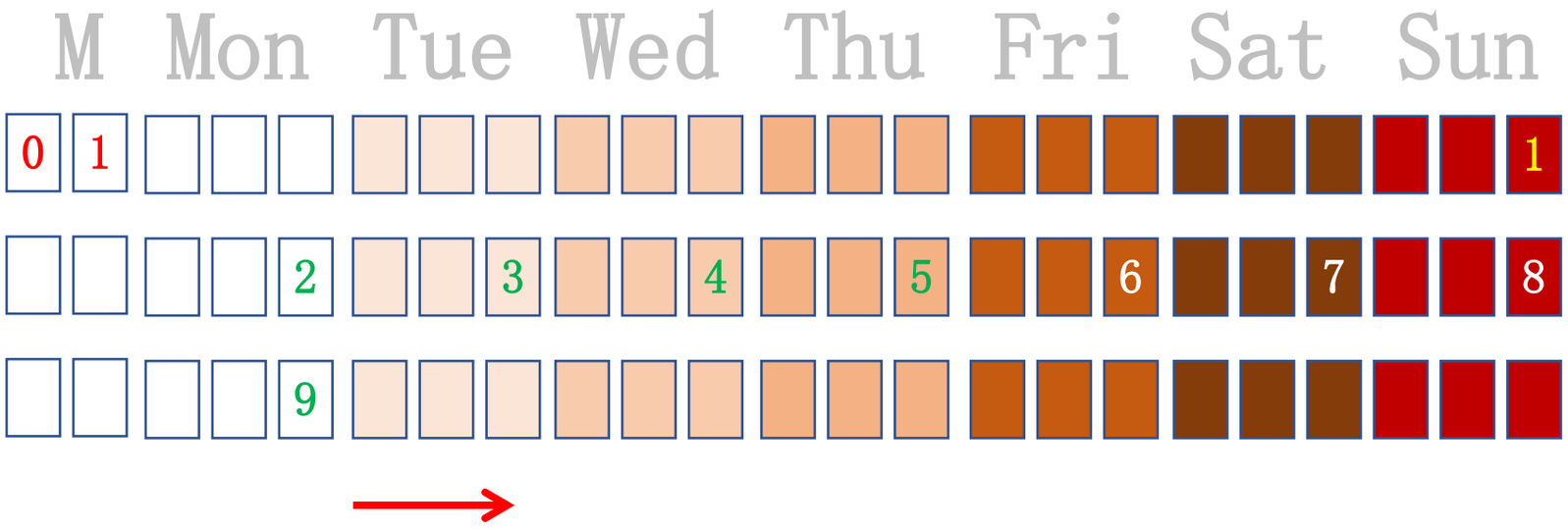



```
cout << "01";
```

上一年所占第一周的天数？

（本年1月1日星期几？）

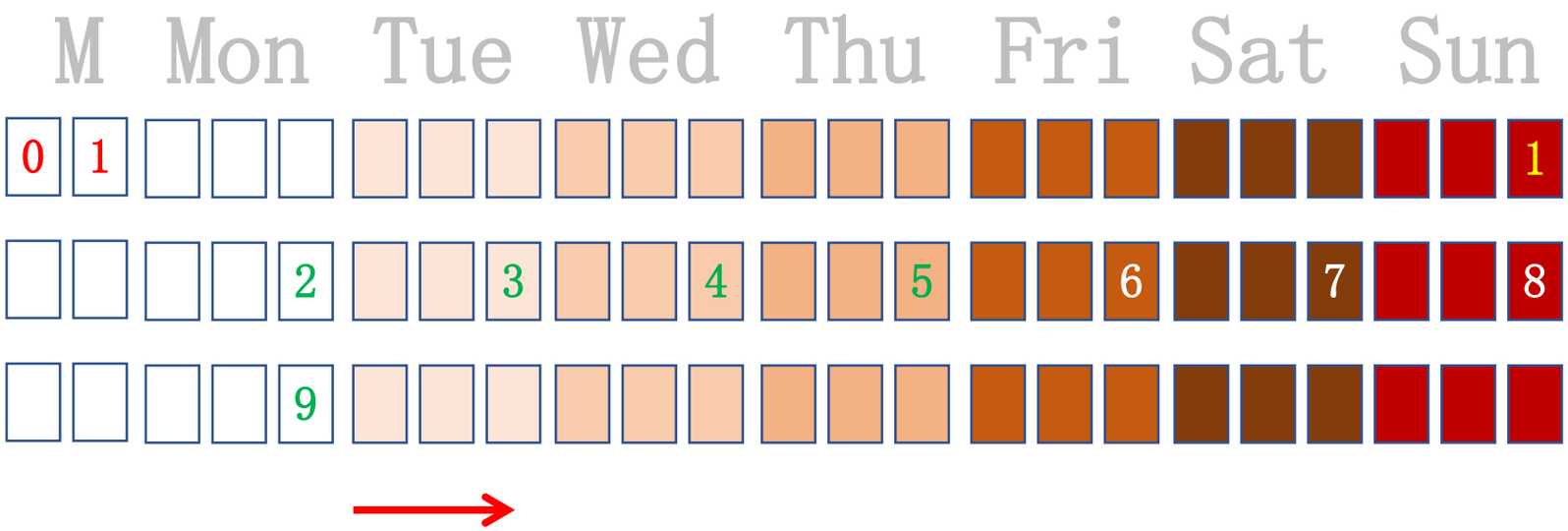
以2023年为例：




```
cout << "01";
for (int i=0; i<7-1; i++) cout << "   ";
printf("%3d", 1);
cout << endl << "  "; // 2 spaces
for (int d=2; d<=8; d++)
    printf("%3d", d);
cout << endl << "  "; // 2 spaces
printf("%3d", 9);
```

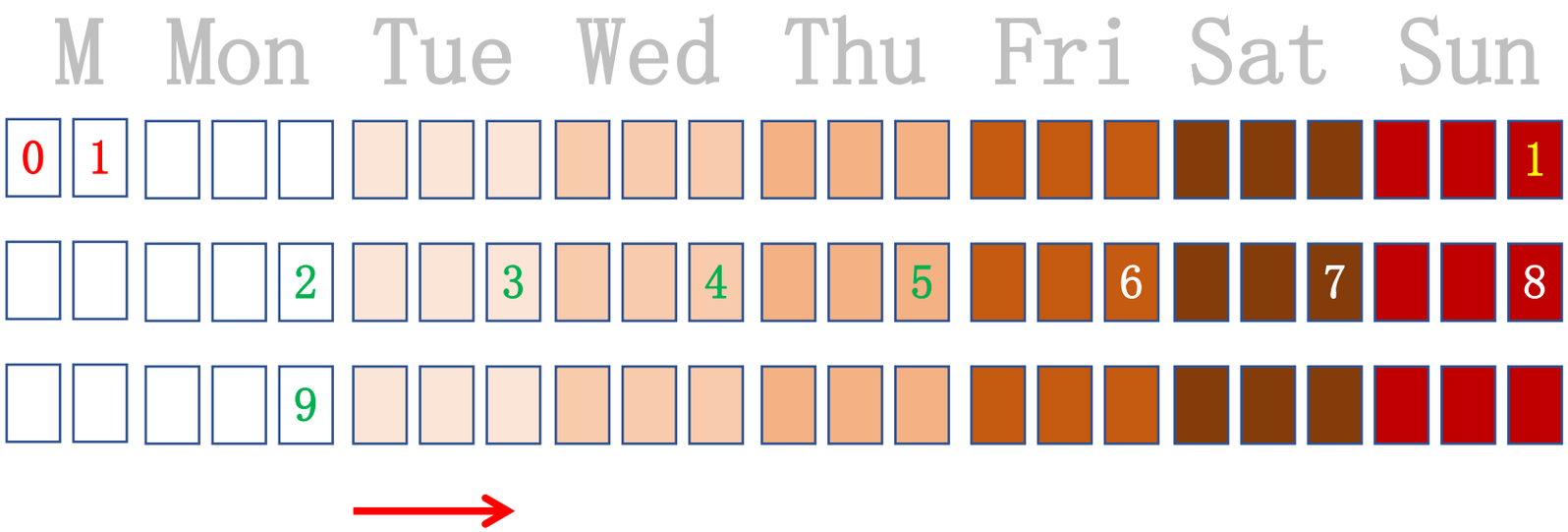
换行，开始新一周（行）

以2023年为例：




```
cout << "01";
for (int i=0; i<7-1; i++) cout << "   ";
printf("%3d", 1);
cout << endl << "  "; // 2 spaces
for (int d=2; d<=8; d++)
    printf("%3d", d);
cout << endl << "  "; // 2 spaces
printf("%3d", 9);
```

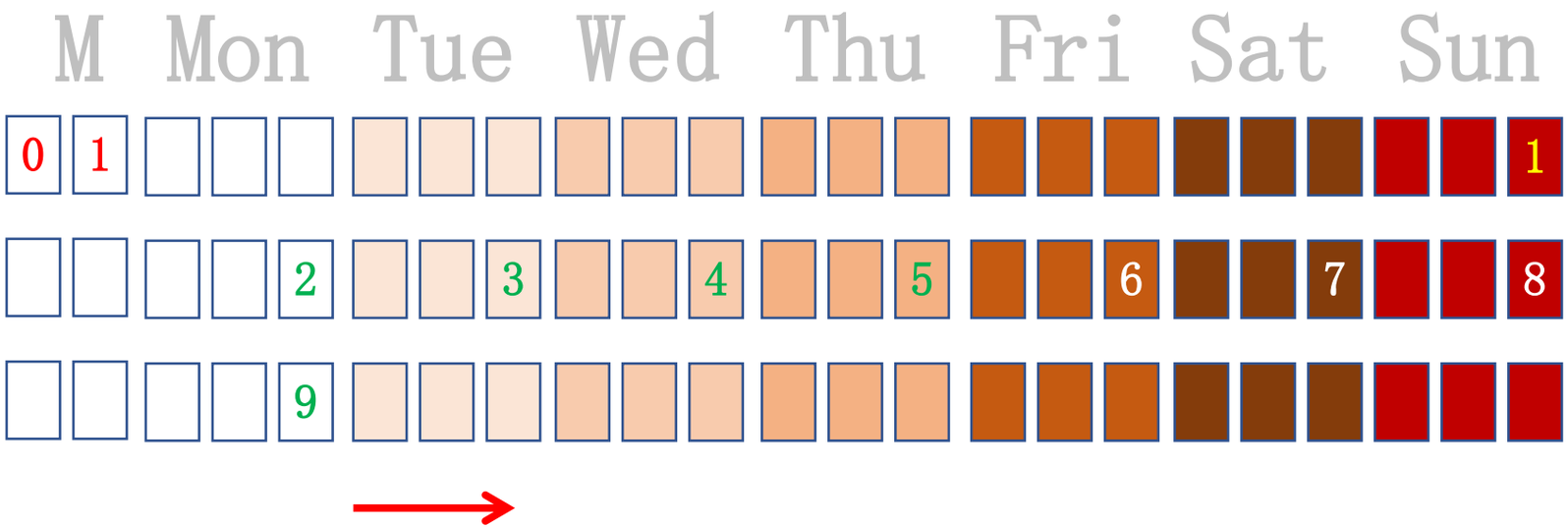
以2023年为例：




```
cout << "01";

for (int i=0; i<7-1; i++) cout << "   ";
int day = 1;
printf("%3d", day);
cout << endl << "  "; // 2 spaces
for (day=2; day<=8; day++)
    printf("%3d", day);
cout << endl << "  "; // 2 spaces
printf("%3d", day); // day == 9
```

优化代码实现

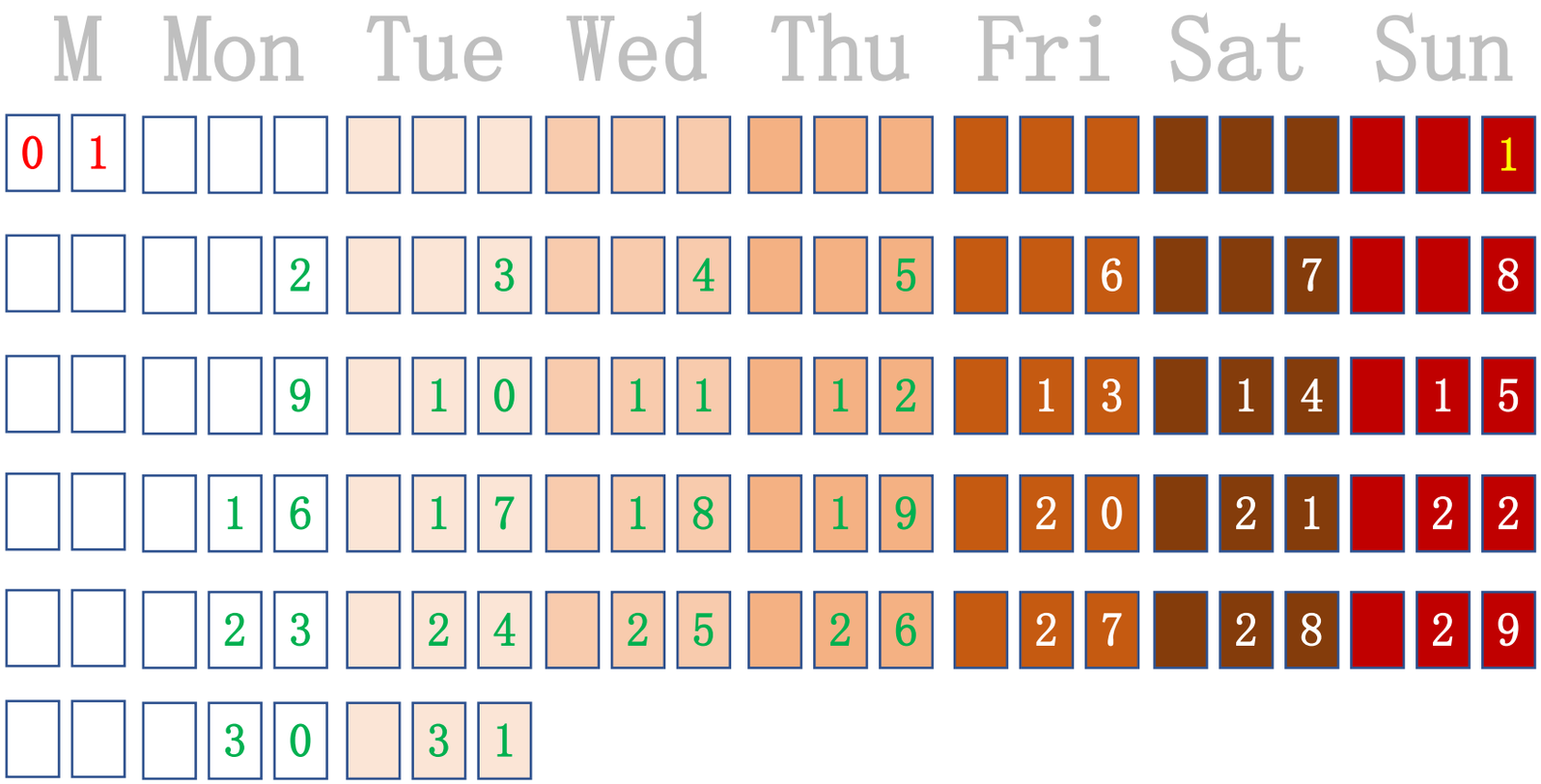
以2023年为例：




```
cout << "01";

for (int i=0; i<7-1; i++) cout << "   ";
for (int day=1; day<=9; day++) {
    printf("%3d", day);
    if (day == 1 || day == 8)
        cout << endl << "  ";
}
```

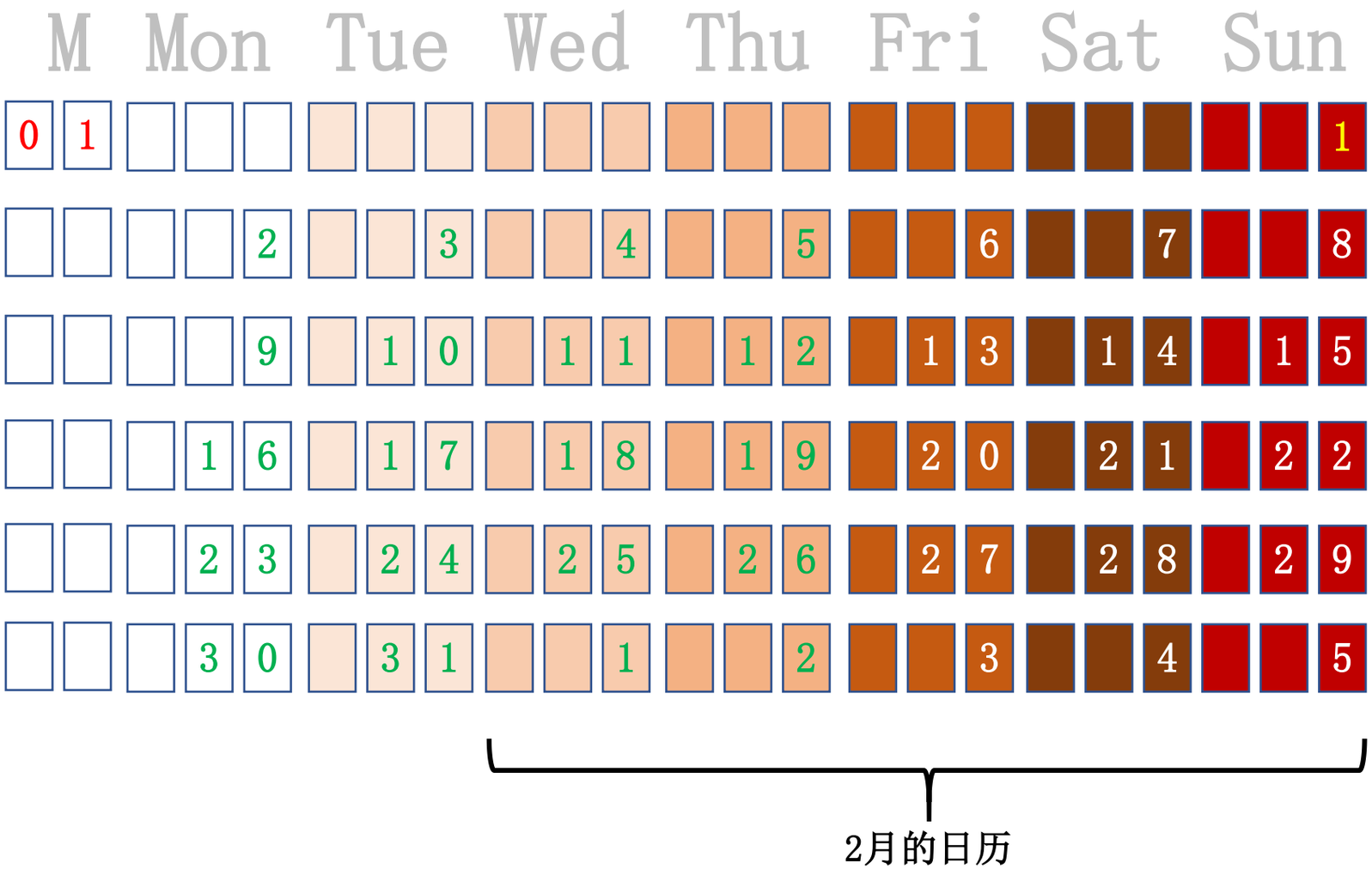
```
int day = 1;
printf("%3d", day);
cout << endl << "  "; // 2 spaces
for (day=2; day<=8; day++)
    printf("%3d", day);
cout << endl << "  "; // 2 spaces
printf("%3d", day); // day == 9
```

以2023年为例：




```
cout << "01";

for (int i=0; i<7-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day == 1  || day == 8  ||
        day == 15 || day == 22 ||
        day == 29)
        cout << endl << "   ";
}
```

以2023年为例：

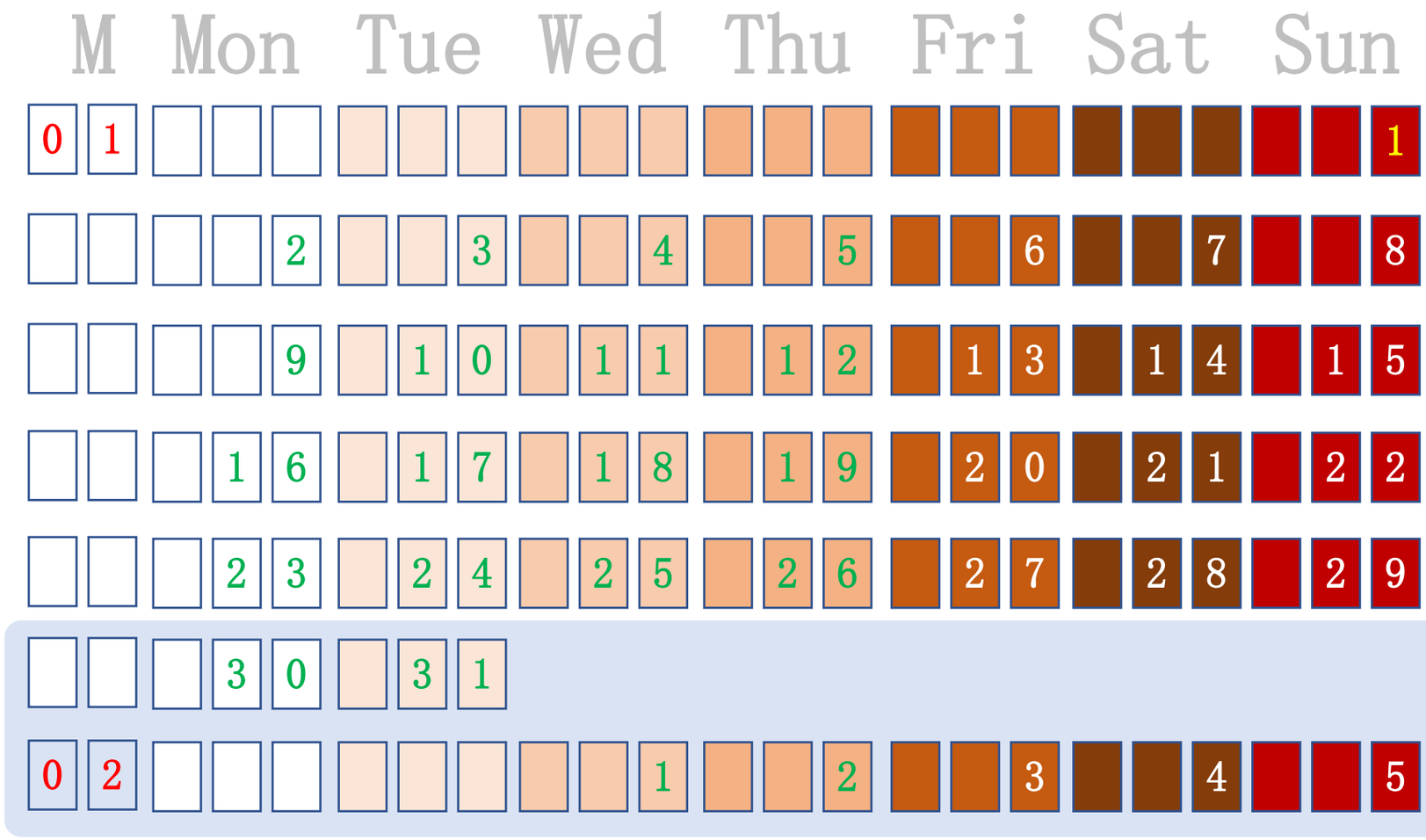



```
cout << "01";

for (int i=0; i<7-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1)
        cout << endl << "  ";
}
/// Feb.
for (int day=1; day<=28; day++) {
    printf("%3d", day);
    if (day % 7 == 1)
        cout << endl << "  ";
}
```

WRONG

方案一

以2023年为例：




```
cout << "01";
int w = 7; // From Sun.
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1)
        cout << endl << "  ";
}
/// Feb.
cout << endl << "02";
w = 3; // From Wed.
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=29; day++) {
    printf("%3d", day);
    if (day == 5 ...)
        cout << endl << "  ";
}
```

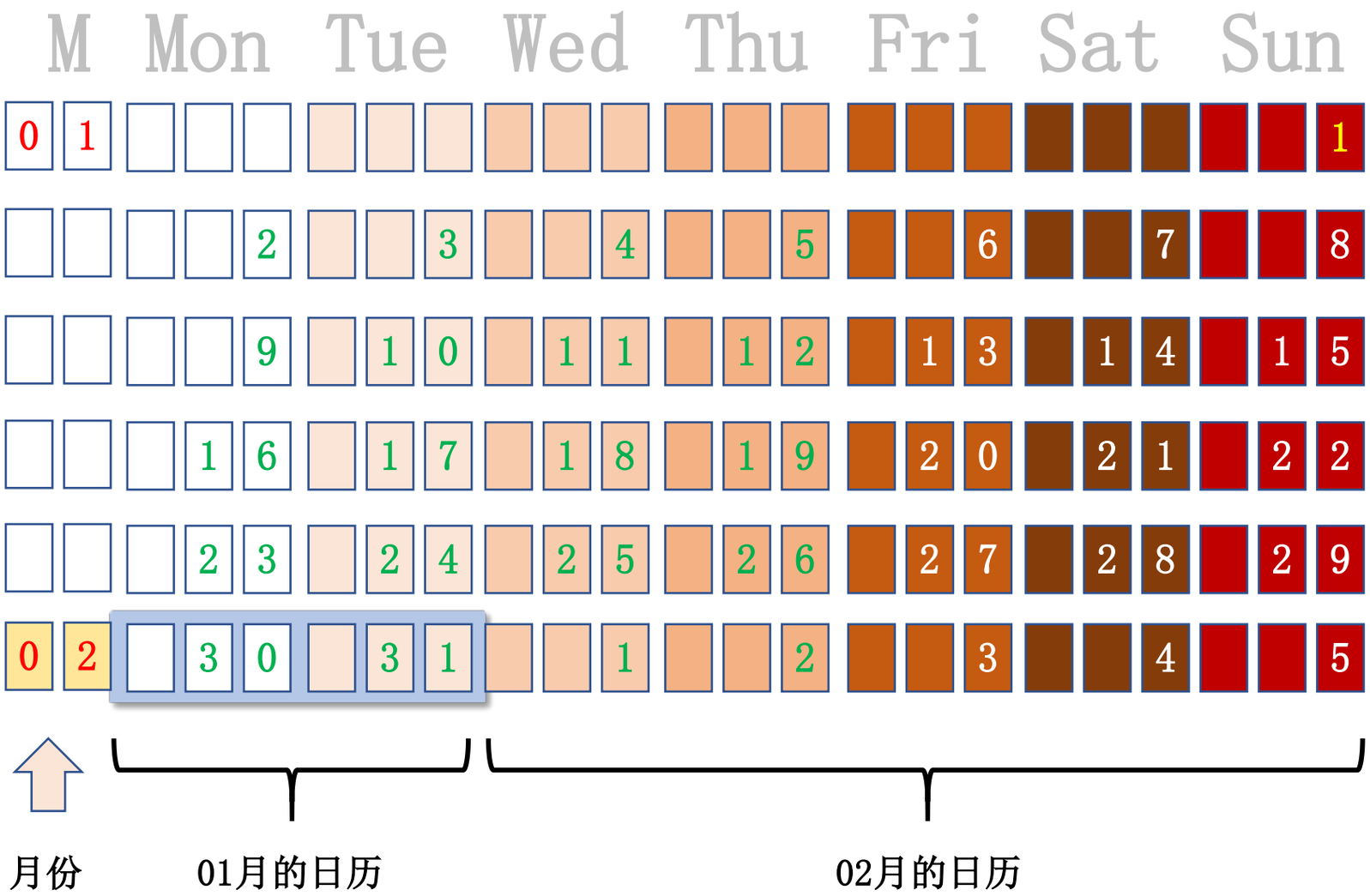
WRONG

方案二

沿用1月的代码，试图输出2月的日历，

也是不对的（与样例要求的格式不符）！

以2023年为例：

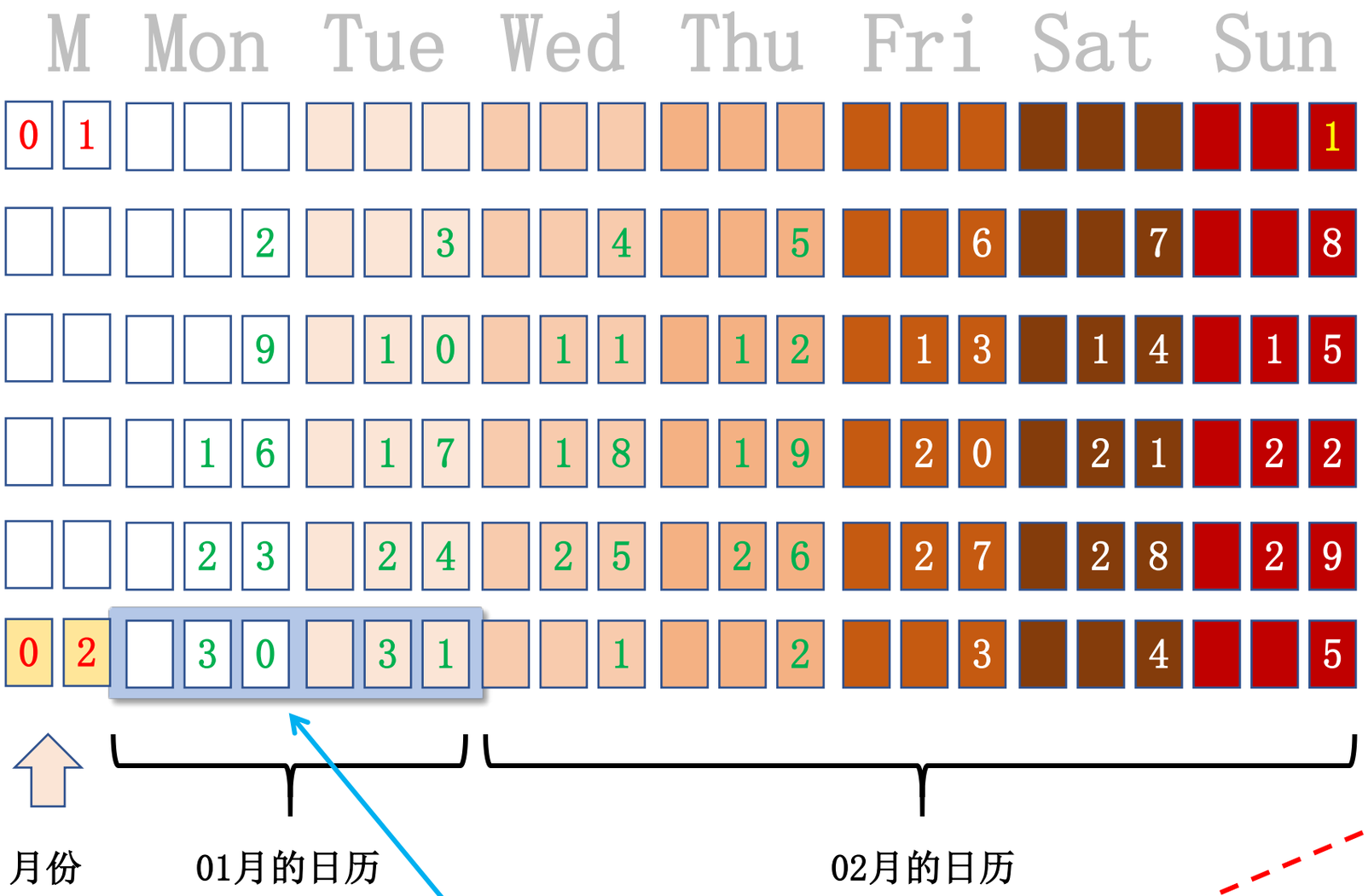



```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1)
        cout << endl << "  ";
}
```

需要在输出1月份30、31前，输出"02"。

如何做到这一点呢？是退回去修改吗？

以2023年为例：




```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1) {
        if (31 - day >= 7)
            cout << endl << "   ";
    else
        cout << endl << "02";
    }
}
```

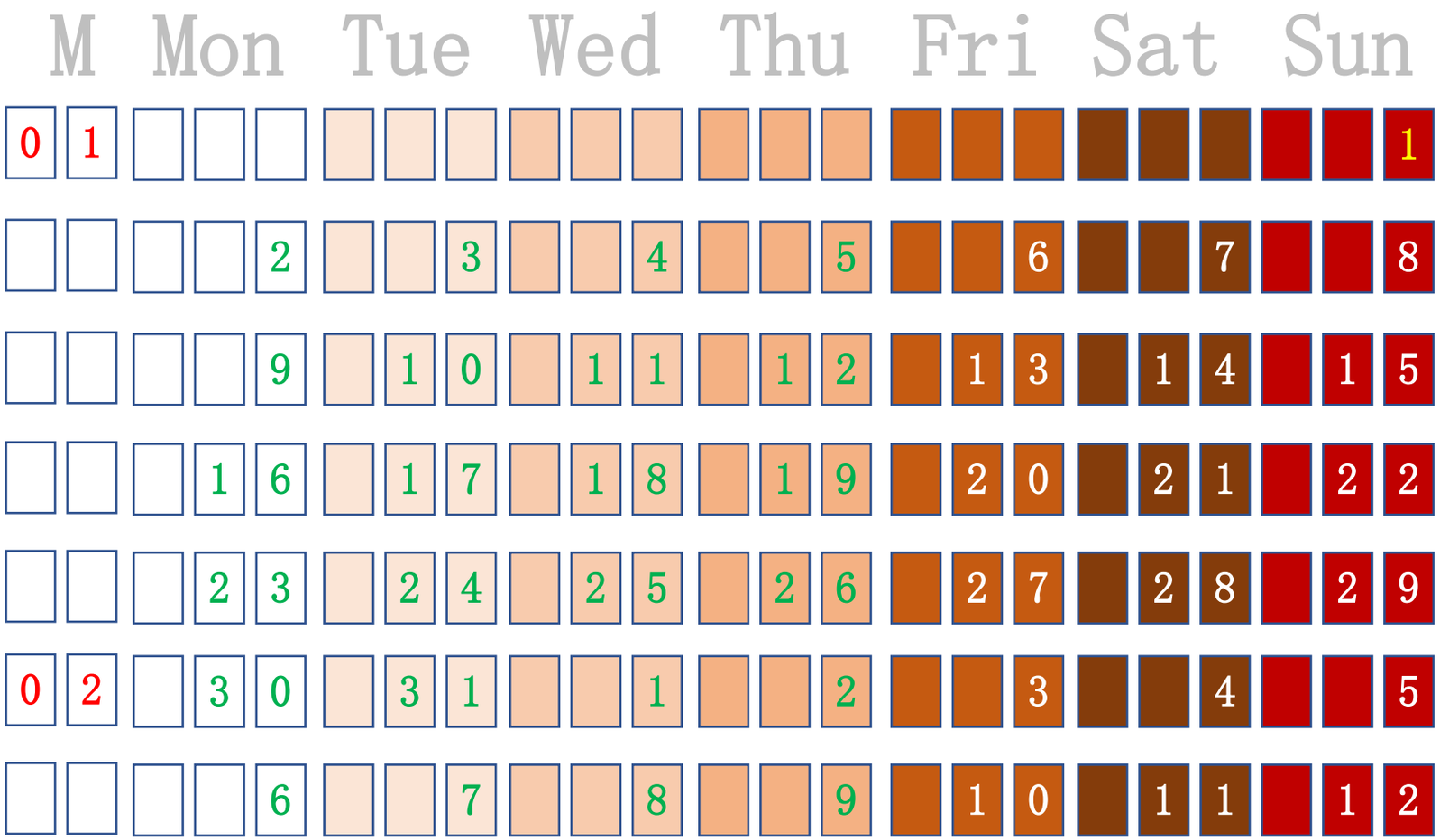
当01月剩余天数不足7天时，
意味着02月要开始。

未雨绸缪

凡事豫则立，不豫则废。言前定，则不跲；事前定，则不困；行前定，则不疚；道前定，则不穷。

—— 《中庸 问政》

以2023年为例：




```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
```

```
for (int day=1; day<=28; day++) {
    printf("%3d", day);
    if (day % 7 == 5) {
        if (28 - day >= 7)
            cout << "\n  ";
        else
            cout << "\n03"; // Mar.
    }
}
```

以2023年为例：

M	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	1						1
			2	3	4	5	6	7	8
			9	10	11	12	13	14	15
			16	17	18	19	20	21	22
			23	24	25	26	27	28	29
0	2		30	31	1	2	3	4	5
			6	7	8	9	10	11	12

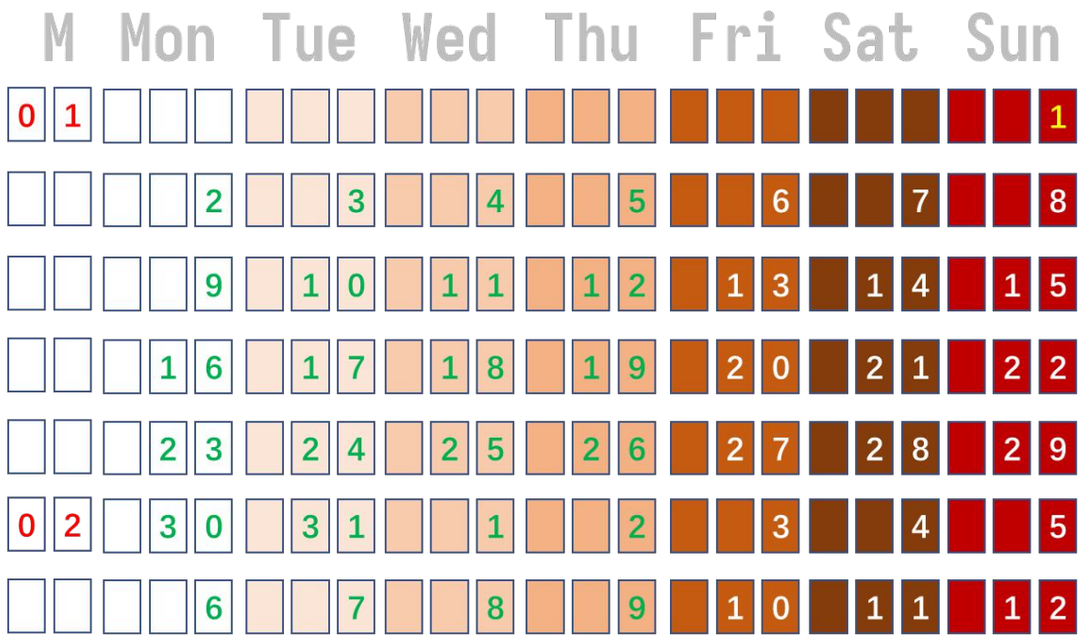
```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
```

```
for (int day=1; day<=28; day++) {
    printf("%3d", day);
    if (day % 7 == 5) {
        if (28 - day >= 7)
            cout << "\n  ";
        else
            cout << "\n03"; // Mar.
    }
}
```

但，2023-2050每年2月不一定总是28天啊，闰年怎么办？

每周换行也好像有问题，其他月份如何处理每周的换行？

以2023年为例：

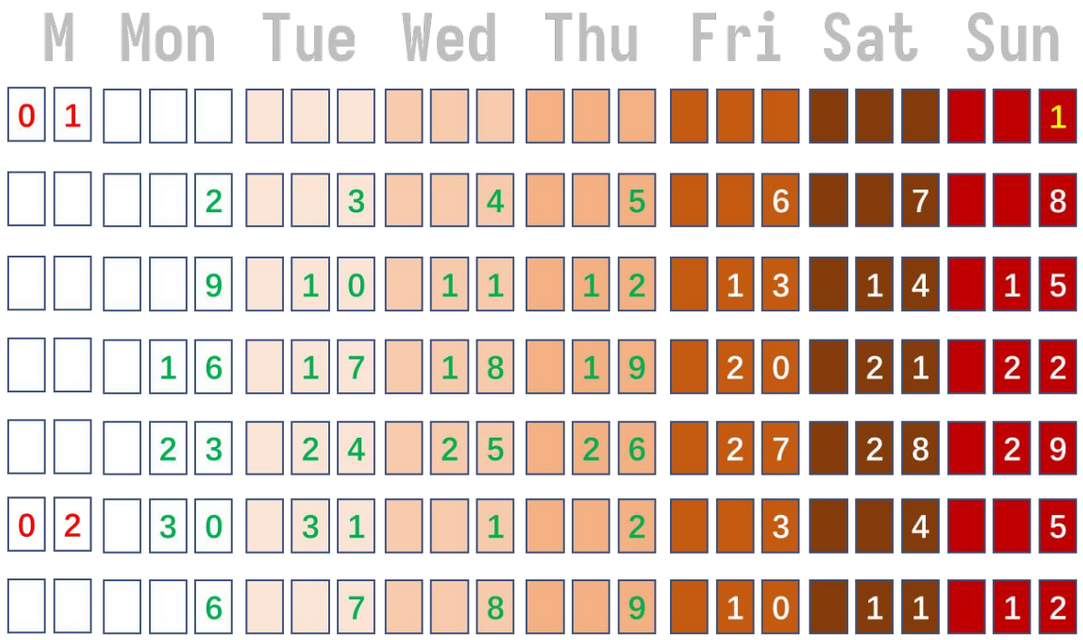


但，2023-2050每年2月不一定总是28天啊，闰年怎么办？

每周换行也好像有问题，其他月份如何处理每周的换行？

```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++) {
    printf("%3d", day);
    if (day % 7 == 1) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
int lastday = (year % 400 == 0 ||
               year % 4 == 0 && year % 100 != 0) ? 29 : 28;
for (int day=1; day<=lastday; day++) {
    printf("%3d", day);
    if (day % 7 == 5) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n03"; // Mar.
    }
}
```

以2023年为例：



但，2023-2050每年2月不一定总是28天啊，闰年怎么办？

每周换行也好像有问题，其他月份如何处理每周的换行？

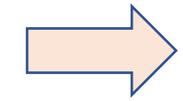
可以将 **w** 理解为代表日历表位置的绝对值，

而将 **day** 理解为各月份内部日期的相对值。

```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
int lastday = (year % 400 == 0 ||
               year % 4 == 0 && year % 100 != 0) ? 29 : 28;
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n03"; // Mar.
    }
}
```

```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
for (int day=1; day<=31; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
int lastday = (year % 400 == 0 ||
               year % 4 == 0 && year % 100 != 0) ? 29 : 28;
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n03"; // Mar.
    }
}
```

```
lastday = 31; // Mar.
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n04"; // Apr.
    }
}
```



```
lastday = 30; // Apr.
for
    lastday = 31; // May
    for
        lastday = 30; // Jun.
        for
            lastday = 31; // Jul.
            for
                lastday = 31; // Aug.
                for
                    lastday = 30; // Sep.
                    for
                        lastday = 31; // Oct.
                        for
                            lastday = 30; // Nov.
                            for
                                lastday = 31; // Dec.
                                for (int day=1; day<=lastday; day++, w++) {
                                    printf("%3d", day);
                                    if (w % 7 == 0) cout << "\n  ";
                                    // 最后一周输出时，没有后续月份
                                }
                                cout << endl; // last week

                            }
                        return 0; // END OF main()
                    }
                }
            }
        }
    }
}
```

VERSION 2.0

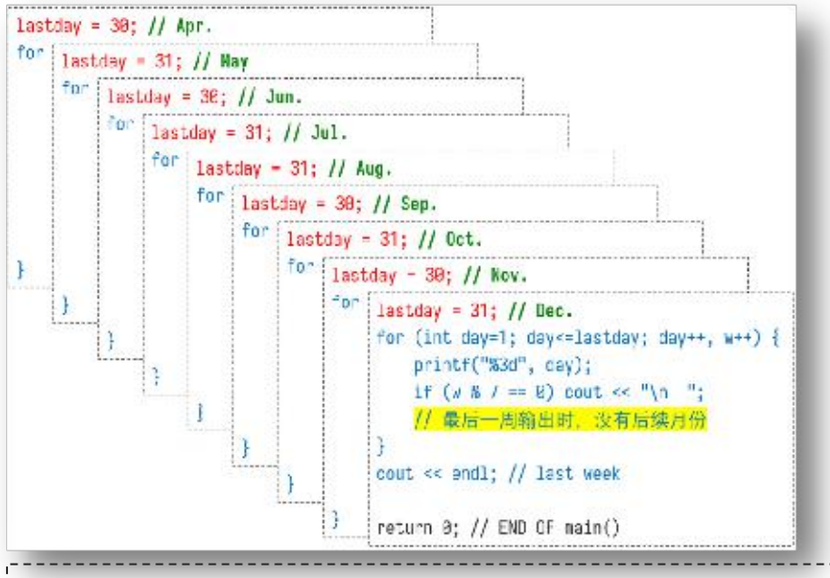
```
lastday = 30; // Apr.
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n05"; // May
    }
}
```

```
lastday = 31; // May
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n06"; // Jun.
    }
}
```

```
lastday = 30; // Jun.
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (lastday - day >= 7)
            cout << "\n  ";
        else
            cout << "\n07"; // Jul.
    }
}
```

仔细观察
总结规律

```
lastday = 31; // Dec.
for (int day=1; day<=lastday; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) cout << "\n  ";
    // 最后一周输出时，没有后续月份
}
cout << endl; // last week

return 0; // END OF main()
```



找出规律，重构代码！

```
for (int month=1; month<=12; month++) {
    int lastday = (month == 4 || month == 6 || month == 9 || month == 11) ? 30 : 31;
    if (month == 2) // overwrite
        lastday = (year % 400 == 0 || year % 4 == 0 && year % 100 != 0) ? 29 : 28;
    for (day=1; day<=lastday; day++, w++) {
        printf("%3d", day);
        if (w % 7 == 0) {
            if (lastday - day < 7 && month < 12)
                printf("\n%02d", month+1); // 1~11月最后一周要输出月份（含前导0）
            else
                printf("\n  "); // 各月前几周及12月各周，总是输出空格
        }
    }
}
```

体会这个技巧

VERSION 3.0

且慢！以上只是以2023为例，其他年份呢？

```
cout << "01";
int w = 7;
for (int i=0; i<w-1; i++) cout << "   ";
```

```
for (int day=1; day<=31; day++, w++) {
    printf("%3d", day);
    if (w % 7 == 0) {
        if (31 - day >= 7)
            cout << endl << "  ";
        else
            cout << endl << "02";
    }
}
```

问题：对于用户指定的 20XX 年，如何计算该年1月1日是星期几？

已知 2023.01.01 是星期日，请问 20XX.01.01 是星期几?

因为每7天一周，第8天与第1天是星期几相同，所以，若2023.01.01到20XX.01.01共有N天（不计20XX.01.01），则20XX.01.01的星期是：

W = N % 7（因为2023.01.01是星期日）

如2024.01.01的星期是：

W = 365 % 7 = 1，星期一

2023.01.01 到 20XX.01.01 ，共有多少天？

```
int year;
cin >> year;
int cnt = 0;
for (int y = 2023; y < year; y++)
    cnt += (y % 400 == 0 || y % 4 == 0 && y % 100 != 0) ? 366 : 365;
```

输出 20XX.01.01 所在周次（20XX年的第1周）

```
// 已知2023.01.01是星期日，计算1月1日是星期几
int w = cnt % 7; // 1: Mon, 2: Tue, 3: Wed, 4: Thu, 5: Fri, 6: Sat, 0: Sun
if (w == 0) w = 7; // 调整一下，方便后续处理前导空白天
```

```
// 输出指定年份的各月
cout << "01"; // Jan.
// 输出年前空白（上一年在第一周里所占天数）
for (int day=0; day<w-1; day++) cout << "    ";
```

完整源程序（仅供参考）

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

int main()
{
    // 用户输入年份
    cout << "Please input year (2023-2050): ";
    int year;
    cin >> year;
    if (year < 2023 || year > 2050) {
        cout << "Error! Not in range [2023..2050]\n";
        return 1;
    }

    // 计算天数
    int cnt = 0;
    for (int y = 2023; y < year; y++)
        cnt += (y % 400 == 0 || y % 4 == 0 && y % 100 != 0) ? 366 : 365;

    // 已知2023.01.01是星期日，计算1月1日是星期几
    int w = cnt % 7; // 1: Mon, 2: Tue, 3: Wed, 4: Thu, 5: Fri, 6: Sat, 0: Sun
    if (w == 0) w = 7; // 调整一下，方便后续处理1月份的前导空白

    // 开始输出指定年份的日历
    cout << "01"; // Jan.
    // 输出年前空白（上一年在第一周里所占天数）
    for (int day=0; day<w-1; day++) cout << "    ";
```

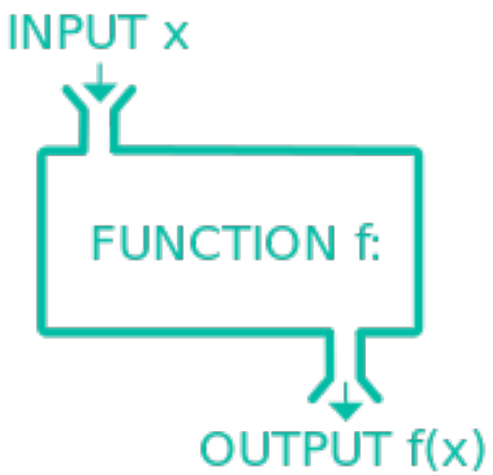
完整源程序（仅供参考）

```
    // 输出各月日历
    for (int month=1; month<=12; month++) {
        int lastday = (month == 4 || month == 6 || month == 9 || month == 11) ? 30 : 31;
        if (month == 2) lastday = (year % 400 == 0 || year % 4 == 0 && year % 100 != 0) ? 29 : 28;

        for (int day=1; day<=lastday; day++, w++) {
            printf("%3d", day); // 右对齐，3字符宽，不足补空格
            if (w % 7 == 0) {
                if (lastday - day < 7 && month < 12)
                    printf("\n%02d", month+1); // 01 ~ 11 月日历最后一行要特殊处理，右对齐，2字符宽，不足补0
                else
                    printf("\n   "); // 一般情况是显示空格
            }
        }
    }

    // 至此，已完成所有日期的输出
    cout << endl;

    return 0;
}
```

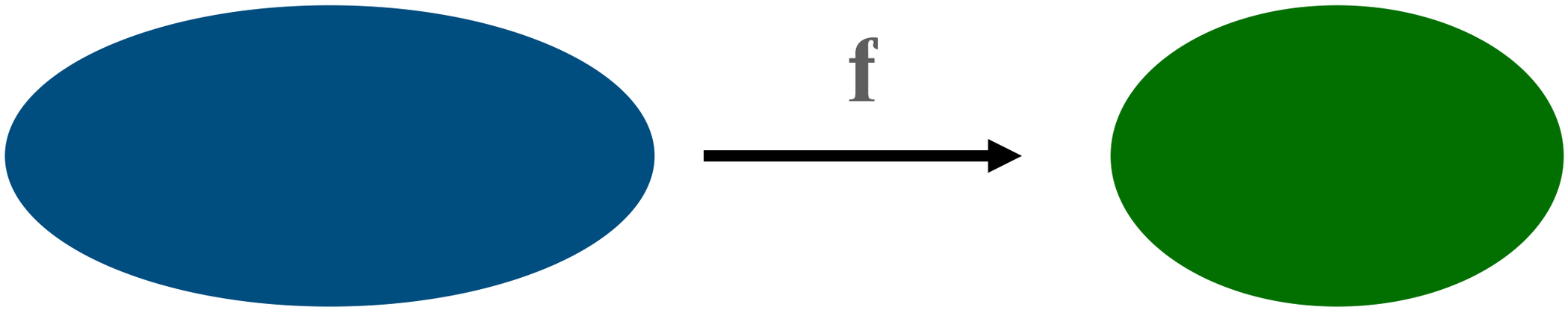
•Laziness: The quality that ***makes you go to great effort to reduce*** overall energy expenditure. It ***makes you write labor-saving programs*** that other people will find useful and document what you wrote so you don't have to answer so many questions about it.

程序员的“懒惰”美德绝不是“懒得思考”！

第4讲 函数思维与计算机解题



回顾数学中函数的定义

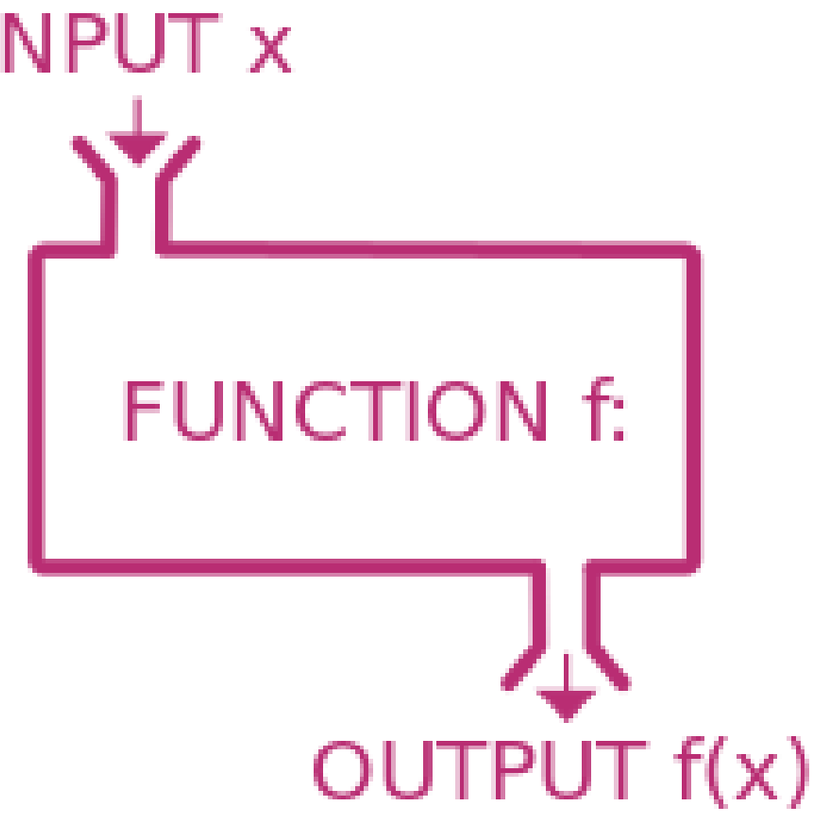


映射 f : x --> y

函数这个数学名词是**莱布尼兹**在1694年引入的，用来描述跟曲线相关的一个量。中文“函数”一词由清朝数学家**李善兰**译出。李善兰《代数学》中解释：“凡此变量中函（包含）彼变量者，则此为彼之函数”。

什么是函数？

映射 f : x --> y



换个角度，可以理解为：

f : INPUT --> OUTPUT

或 **output = f(input)**

什么是函数？

$$y(x) = a * x \wedge 2 + b * x + c$$

函数的名称

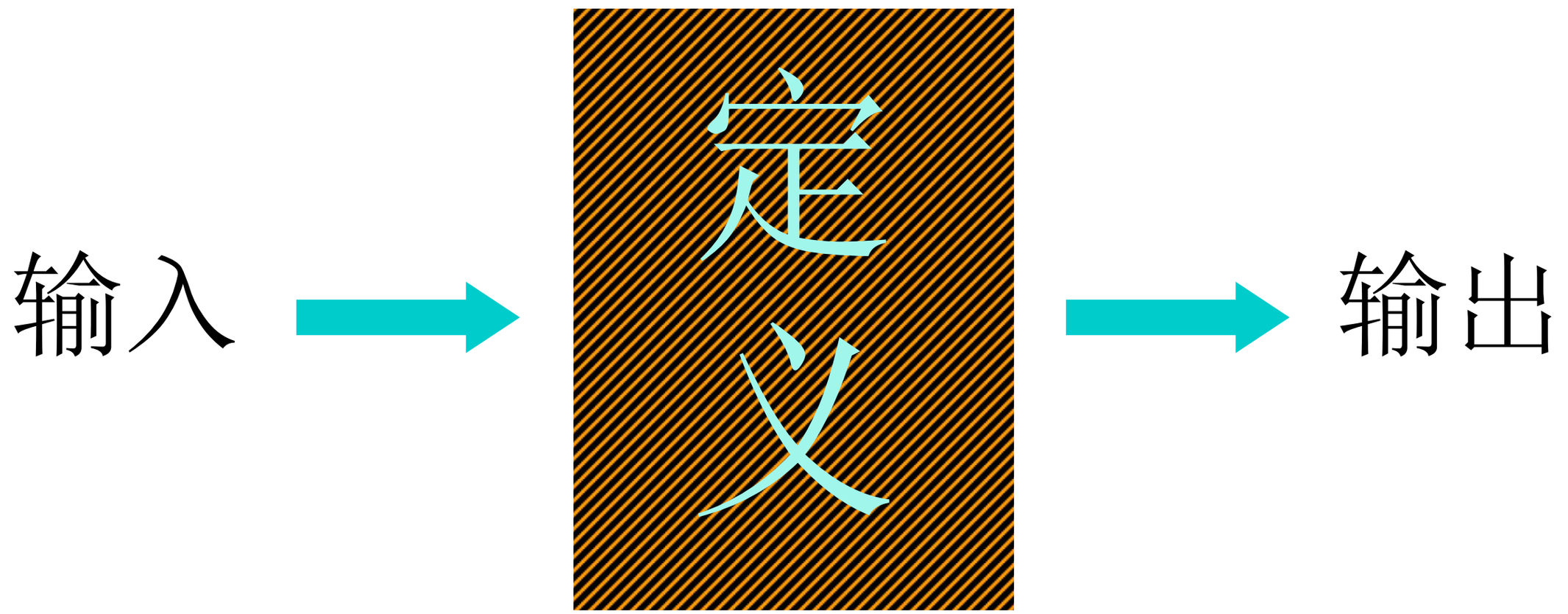
函数的定义

函数名称则给出了如何使用函数的标识（记号）

函数定义给出了如何计算函数值的方法（公式）

函数的四大要素：名称-输入-输出-函数体（定义）

（函数的）名称



定义函数的步骤 —— 要回答的问题

- **STEP 1：确定函数的输出【想要什么】**
  - 函数调用结束后，产生哪些效果？
  - 得到哪些结果？
- **STEP 2：确定函数的输入【提供什么】**
  - 完成函数功能，需要哪些前提条件？
- **STEP 3：确定函数的实现语句体【如何做到】**
  - 如何处理输入的数据，得到所需要的结果?
  - 如何将所需要的结果返回？

有自定义函数的源程序结构

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

int add(int a, int b)
{
   return a + b;
}
int main()
{
   cout << add(3, 4);
   return 0;
} // E04-01.cpp
```

头文件与编译指令

辅助函数定义

函数调用（与使用标准库函数方式相同）须在定义或声明之后！

主函数定义

函数声明、函数定义、函数调用

函数声明：形式参数名称可以省略，但类型名不能省略

函数返回值的类型　　函数名

(类型名　形式参数变量1，类型名　形式参数变量2，．．．

　　);

```
// 格式 1
int add(int, int);

// 格式 2
int add(int a, int b);
```

这个分号是必须的！

函数命名应符合实际定义且容易理解。函数声明要放在所有调用它的函数之前（包括main函数），即“先声明，后使用”。调用方式与使用标准库函数相同。

函数声明、函数定义、函数调用

函数定义的语法格式：

```
函数返回值的类型      函数名
(类型名    形式参数变量1，类型名    形式参数变量2，．．．
    )
{
        // 函数体
}
```

```
int add(int a, int b)
{
    return a + b; // 返回表达式的值
}
```

函数体中的 **return** 语句要返回相应类型的值（或能自动转换成相应类型的值），可以返回表达式的值，也可以返回变量的值 。

没有返回值、没有参数的函数

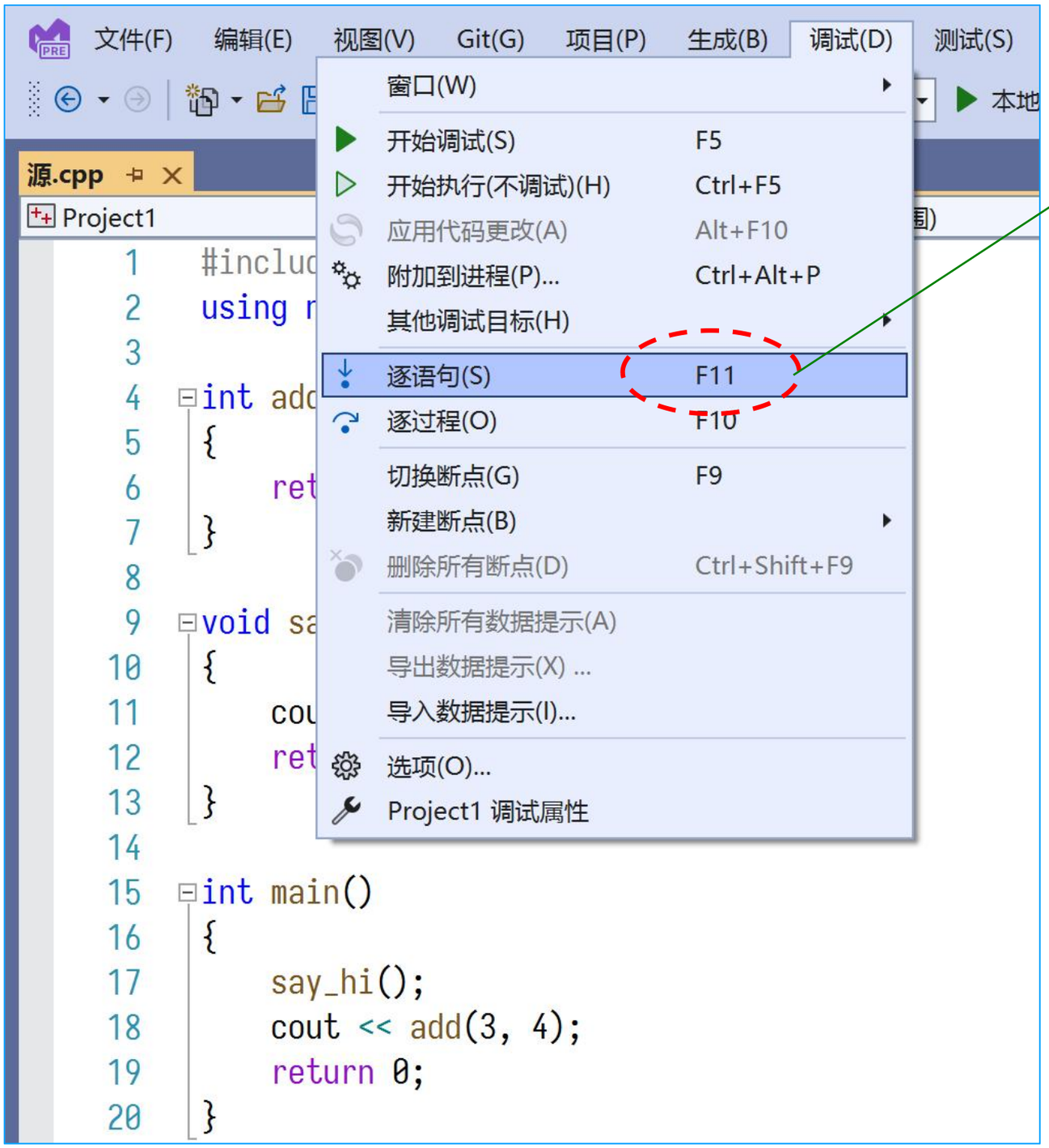
若无返回值，则“返回值类型”应定义为**void**类型。**void**类型一般只用于定义“函数返回值类型”和“指针类型变量”（如 void* ptr)。

```
void say_hi()
{
    cout << "Hi!\n";

}
```

```
void say_hi()
{
    cout << "Hi!\n";
    return;
}
```

如果函数返回值类型为**void**，一般也可以称为“函数没有返回值”，这时，函数体中可以不写 **return** 语句。如果要写的话，不能用**return**返回值，只能写为：**return ;**

函数调用的执行过程（以Visual Studio为例）



使用Visual Studio的“调试丨逐语句”菜单项，观察函数的“调用、返回”的程序执行过程。

建议使用键盘功能键 F11来提高效率。

【任务4.1】定义计算三角形面积的函数

三角形面积 ＝ 0.5 x 底 x 高

A ＝ 0.5 * W * H

A ：W, H →0.5 * W * H

A（W，H）＝ 0.5 * W * H

【任务4.1】计算三角形面积的函数 V0

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

float TriangleArea(float base, float height)
{
    float area;
    area = 0.5 * base * height;
    return area;
} /// V0：先计算并保存，最后返回结果

int main()
{
    int base, height;
    cin >> base >> height;
    cout << TriangleArea(base, height) << endl;
    return 0;
} // L04-01.cpp
```

重点概念：实在参数（变量）和形式参数（变量）

```
cout << TriangleArea(base, height) << endl;
```

base，height是调用处定义的变量，被称为“实在参数”，简称“实参”。

```
float  TriangleArea( float base, float height) { …. }
```

在函数定义中使用的变量base，height，被称为“形式参数”，其特点：

1. 在定义函数时放在函数名后括号中；
2. 函数被调用之前，不会被分配内存单元；
3. 函数被调用时，系统为其分配内存单元；
4. 函数调用结束后，释放占用的内存单元；
5. 仅在函数内有效，属于局部变量；
6. 形式参数与实在参数可以使用相同的名字（当然也可以不同）

重点概念：实在参数（变量）和形式参数（变量）

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

float TriangleArea(float base, float height)
{
    cout << "In TriangleArea(): " << endl;
    cout << " base @ " << &base << ", height @ " << &height << endl;
    float area;
    area = 0.5 * base * height;
    return area;
} /// V0: 先计算并保存，最后返回结果

int main()
{
    int base, height;
    cout << "In main(): " << endl;
    cout << " base @ " << &base << ", height @ " << &height << endl;

    cin >> base >> height;
    cout << TriangleArea(base, height) << endl;
    return 0;
} // L04-01A.cpp
```

程序运行结果（注意：地址与运行环境有关）

```
In main():
 base @ 0x61fe1c, height @ 0x61fe18
6 4
In TriangleArea():
 base @ 0x61fdf0, height @ 0x61fdf8
12
```

单选题 1分

设置

如下源程序中定义了函数并对它进行了调用，调用处的实参名称与函数定义处的形参名称完全不同，这种写法是否有错误？

A **没有错误**

B **有编译错误**

C **计算结果错误**

D **程序运行错误（崩溃）**

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

float TriangleArea(float base, float height)
{
    float area;
    area = 0.5 * base * height;
    return area;
} /// V0: 先计算并保存，最后返回结果

int main()
{
    int w, h;
    cin >> w >> h;
    cout << TriangleArea(w, h) << endl;
    return 0;
} // E04-02.cpp
```

提交

【任务4.1】计算三角形面积的函数 V1

V1：在函数实现时，不借助临时变量，而是直接返回表达式的值。

```
float TriangleArea (float base, float height)
{
    return 0.5 * base * height;
} /// V1：直接返回计算结果
```

【任务4.1】计算三角形面积的函数 V2

V2：定义全局变量，通过改变全局变量，实现函数结果的返回。

```
float area;

void TriangleArea (float base, float height)
{
      area = 0.5 * base * height;
} /// V2: 使用全局变量保存（返回）计算结果
```

虽然语法上允许你这么做，有时也很方便，但建议你尽量不要这么做！

【任务4.1】计算三角形面积的函数 V3

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

float area, base, height;

/// 调用此函数前，要给base,height赋值
void TriangleArea()
{
    area = 0.5 * base * height;
} /// V3: 使用全局变量作参数和返回结果

int main()
{
    cin >> base >> height;
    TriangleArea();
    cout << area << endl;

    return 0;
} // L04-01-V3.cpp
```

【任务4.1】计算三角形面积的函数 V4*

“名字空间”不作要求

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

namespace xmx {
    float area, base, height;

    /// 调用此函数前，要给base,height赋值
    void TriangleArea()
    {
        area = 0.5 * base * height;
    }
} // 可以使用“名字空间”将“全局变量”局部化

int main()
{
    cin >> xmx::base >> xmx::height;
    xmx::TriangleArea();
    cout << xmx::area << endl;
    return 0;
} // L04-01-V4.cpp
```

应尽量避免使用全局变量

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

float area, base, height;

/// 调用此函数前，要给base,height赋值
void TriangleArea()
{
    area = 0.5 * base * height;
} /// V3：使用全局变量作参数和返回结果

int main()
{
    cin >> base >> height;
    TriangleArea();
    cout << area << endl;

    return 0;
} // L04-01-V3.cpp
```

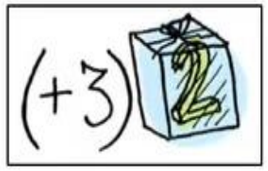
【任务4.1】计算三角形面积的函数 V5

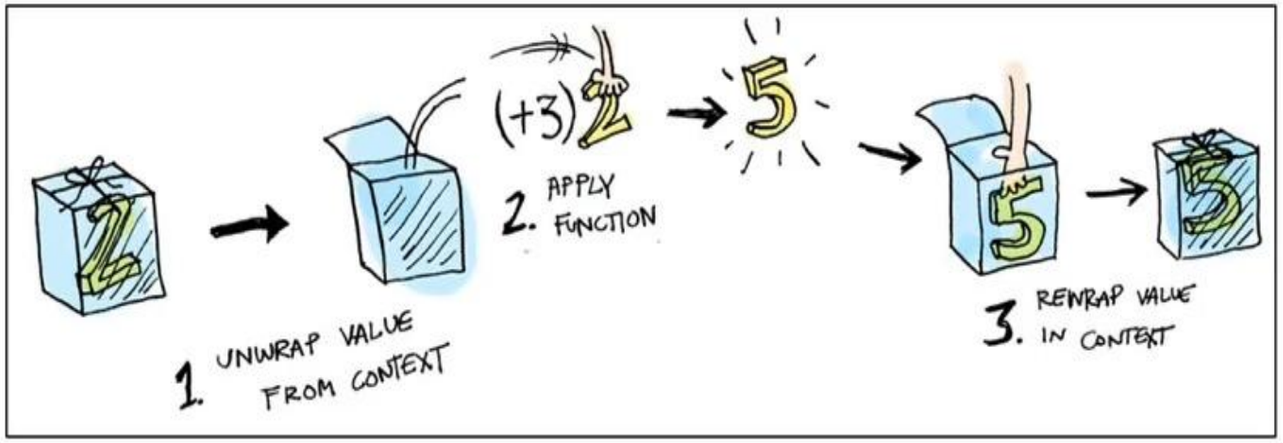
设计准则，努力领悟遵守

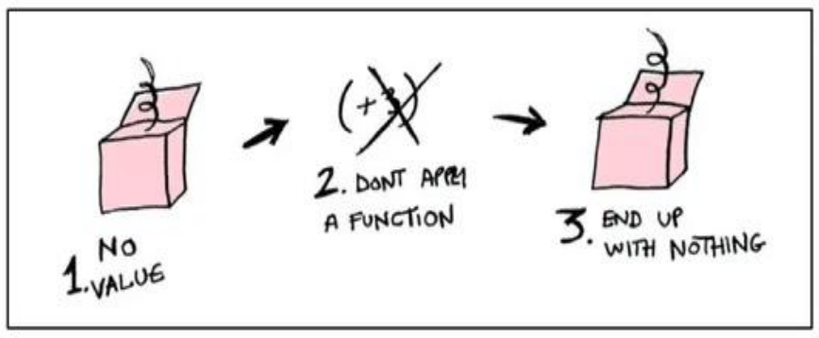
```
void TriangleArea(float base, float height)
{
    float area;
    area = 0.5 * base * height;
    cout << area;
    //或cout << 0.5 * base * height;
} /// V5：函数直接输出结果，也改变了 “系统”的状态
```

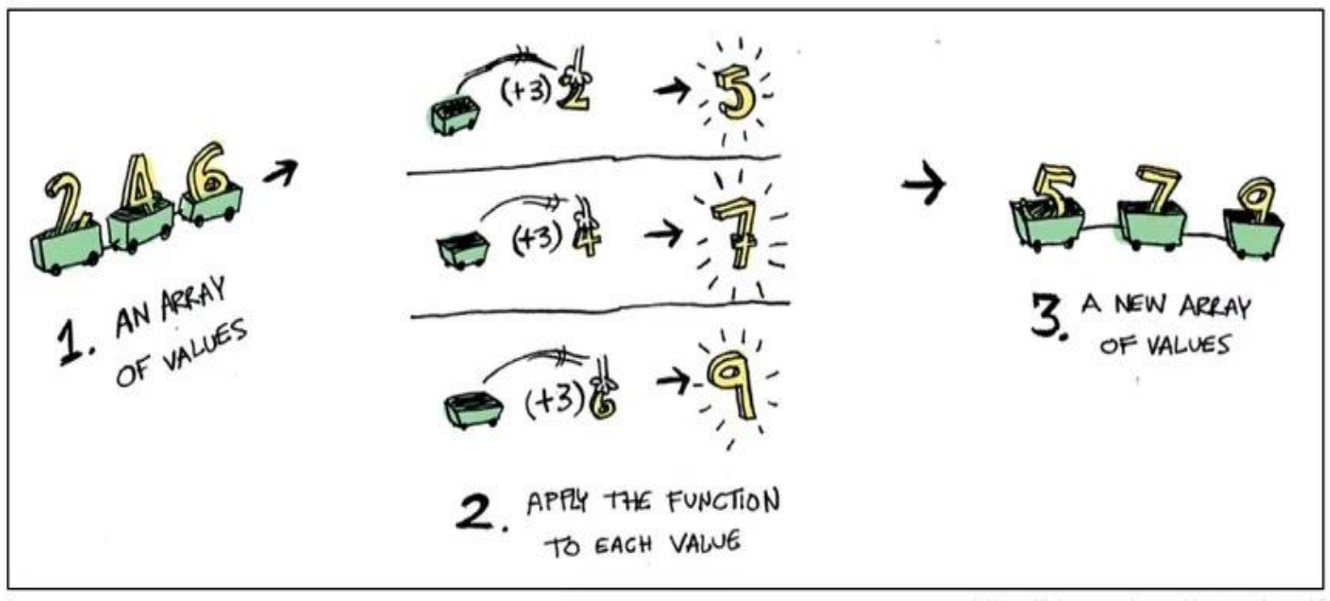
从“设计”角度看，在这种实现方法中，函数实际上完成了两个“独立”的任务，一个是“计算”，一个是“输出”，差异很大。一个函数应该只做“一件事”。

函数式编程思想









重点概念：全局变量、局部变量、函数参数

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

void f4(int num) {
    cout << "f4(): num = " << num << endl;
}

int num = 123;

void f5(int num) {
    cout << "f5(): num = " << num << endl;
}

void f6() {
    cout << "f6(): num = " << num << endl;
}

void f7() {
    int num = 789;
    cout << "f7(): num = " << num << endl;
}

int main() {
    cout << "@LINE: " << __LINE__
         << " num = " << num << endl;

    f4(num);
    f5(num);
    f6();
    f7();

    cout << "\n======================\n\n";

    int num = 456;
    cout << "@LINE: " << __LINE__
         << " num = " << num << endl;
    f4(num);
    f5(num);
    f6();
    f7();

    return 0;
} // L04-02.cpp
```

重点概念：全局变量、局部变量、函数参数

```
27  int main() {
28
29      cout << "@LINE: " << __LINE__ << " num = " << num << endl;
30
31      f4(num);
32      f5(num);
33      f6();
34      f7();
35
36      cout << "\n=========================\n\n"
37
38      int num = 456;
39      cout << "@LINE: " << __LINE__ << " num =
40
41      f4(num);
42      f5(num);
43      f6();
44      f7();
45
46      return 0;
47  } // L04-02.cpp
```

预编译指令，编译时会被代码行号替换掉

```
@LINE: 29 num = 123
f4(): num = 123
f5(): num = 123
f6(): num = 123
f7(): num = 789

=================================

@LINE: 39 num = 456
f4(): num = 456
f5(): num = 456
f6(): num = 123
f7(): num = 789
Program ended with exit code: 0
```

重点概念：局部变量（FOR语句、语句块、函数体语句块）

```
D:\Work>type D04-02.cpp
#include <iostream>
using namespace std;

int main()
{
    {
        int i = 123;
    }
    cout << "i = " << i << endl;

    for (int k = 0; k < 3; k++)
    {
        int m = 456;
    }
    cout << "k = " << k << endl;
    cout << "m = " << m << endl;

    return 0;
} // D04-02.cpp

D:\Work>g++ D04-02.cpp
D04-02.cpp: In function 'int main()':
D04-02.cpp:9:23: error: 'i' was not declared in this scope
     cout << "i = " << i << endl;

D04-02.cpp:15:23: error: 'k' was not declared in this scope
     cout << "k = " << k << endl;

D04-02.cpp:16:23: error: 'm' was not declared in this scope
     cout << "m = " << m << endl;
```

语句块中的局部变量

for 语句中的两种局部变量

一对花括号中的语句，
被称为“语句块”

变量 i 和 m 都是语句块中的局部变量

变量 k 是 for 语句块中的局部变量

main函数语句块没有直接定义变量k, m，导致输出出错！

【任务4.2】判断质数

要求：从键盘输入一个正整数a，编一个程序判断a是否为质数。

思路：设计一个函数

```
bool IsPrime( int n)
```

来负责检查n是否为质数：如果是，该函数返回true；否则，返回false。

```cpp
#include <iostream> // cout
#include <cmath>    // sqrt
using namespace std;

bool IsPrime(int); // 子函数声明。相对于主函数main，用户自定义函数均可被称为“子”函数

int main() {
    int n = 0;
    cout << "请输入一个整数：n=";
    cin >> n;

    // 用实参n调用子函数，子函数返回值作为if语句的分支条件
    if (IsPrime(n))
        cout << n << "是质数" << endl;
    else
        cout << n << "不是质数" << endl;
    return 0;
}
```

这里不用考虑（顾虑）质数判断的实现细节，直接调用函数完成任务即可

```cpp
bool IsPrime(int n) {
    for (int k = 2; k <= sqrt(n); k++)
        if (n % k == 0) return false; // n 能被k整除，返回false
    return true; // n 不能被k整除，返回true
} // L04-04.cpp
```

为何要在程序中设计和使用函数？

(1) 实现层面：在解决不同的实际问题时，常常需要用到许多相同的方法或技术。当它们对于不同问题可能只是选择的参数不同，内部功能完全相同时，如果能将这些方法或技术形成一个固定的函数，让用户在解决问题时直接使用，则程序实现会更加快捷。

(2) 设计层面：在解决一些较复杂问题时，如果先将问题划分或化解成一些子问题，然后再分别加以解决，则能显著降低问题的复杂度，有利于提高程序设计的效率。

【任务4.3】字符与ASCII码输出

此示例任务与“日历输出”上机实验相似

- 输出所有ASCII码对应的“字符”及其内存中的二进制表示（由0和1组成的串）。
- 一行输出4个字符，字符之间用一个空格隔开
- 在输出各字符时，格式要求如下：
  - ASCII码的十进制值□二进制串□字符
  - □表示输出空格。例如：
    - 75□□01001011□K
    - 十进制数占三个字符宽，不够宽时在后面补空格（如□ ）。

模块化程序设计：自顶向下 + 逐步求精

先建框架

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

int main()
{
    return 0;
}
```

v1.0

模块化程序设计：自顶向下 + 逐步求精

逐步扩充

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

int main()
{
        for (int i=0; i<128; i++)
        {
                cout << i;
                cout << ' ' << char(i);
        }
        return 0;
}
```

v2.0

模块化程序设计：自顶向下 + 逐步求精

不断细化

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

int main()
{
        for (int i=0; i<128; i++)
        {
                out_int_with_sp(i);
                cout << ' ' << char(i);
        }
        return 0;
}
```

v3.0

模块化程序设计：自顶向下 + 逐步求精

层层深入

```
void out_int_with_sp(int i)
{
        cout << i;
        if (i < 10)
                cout << "  "; // 双引号是字符串
        else if (i < 100)
                cout << ' ';  // 单引号是字符
}
```

模块化程序设计：自顶向下 + 逐步求精

逐步完善

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

int main()
{
        for (int i=0; i<128; i++)
        {
                out_int_with_sp(i);
                cout << ' ' << char(i);
                if (i % 4 == 3)
                        cout << endl;
                else
                        cout << ' ';
        }
        return 0;
}
```

v4.0

模块化程序设计：自顶向下 + 逐步求精

逐步完善

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

int main() {
        for (int i=0; i<128; i++) {
                out_int_with_sp(i);
                cout << ' ';
                out_char_bin(i);
                cout << ' ' << char(i);
                if (i % 4 == 3)
                        cout << endl;
                else
                        cout << ' ';
        }
        return 0;
}
```

v5.0

模块化程序设计：自顶向下 + 逐步求精

```
void out_char_bin(int n)
{
        for (int i=7; i>=0; i--)
        {
                output_bit(n, i);
        }
}
```

模块化程序设计：自顶向下 + 逐步求精

```
void output_bit(int n,  int pos)
{
        int bit = get_bit(n,  pos);
        cout << bit;
}
```

如何得到整数二进制表示的某个比特位的值呢？

【编程技巧】获取某整数某个二进制位上的值

```
int get_bit(int n, int pos)
{
        int mask = (1 << pos);
        int bit = (n & mask) >> pos;
        // int bit = (n & (1 << pos)) >> pos;
        return bit;
        // return (n & (1 << pos)) >> pos;
}

void output_bit(int n, int pos)
{
        int bit = get_bit(n, pos);
        cout << bit;
}
```

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

void out_int_with_sp(int i)
{
    cout << i;
    if (i < 10)
        cout << "  ";
    else if (i < 100)
        cout << ' ';
}

int get_bit(int n, int pos)
{
    int mask = (1 << pos);
    int bit = (n & mask) >> pos;
    return bit;
}

void output_bit(int n, int pos)
{
    int bit = get_bit(n, pos);
    cout << bit;
}

void out_char_bin(int n)
{
    for (int i=7; i>=0; i--)
        output_bit(n, i);
}

int main()
{
    for (int i=0; i<128; i++)
    {
        out_int_with_sp(i);
        cout << ' ';
        out_char_bin(i);
        cout << ' ' << char(i);
        if (i % 4 == 3)
            cout << endl;
        else
            cout << ' ';
    }
    return 0;
} // L04-03.cpp
```

【任务4.4】输出4n+1型质数

编写程序，找出2－100以内的全部4n＋1型质数，输出这些质数并统计输出它们的数目，其中n为正整数。

算法思路（1）

真代码

```
// 设置计数器初值为0
int Count = 0;
// 枚举 2，3，… ，100 中的所有数，
for (int i = 2; i <= 100; i++) {
    // 逐一进行检验：
    // 如果（不是质数），检测下一个数；
    if (!IsPrime(i)) continue;
    // 如果（不可表示为4n+1），检测下一个数；
    if ((i-1) % 4 != 0) continue;
    // 输出满足条件的数；
    cout << i << endl;
    // 让计数器（变量）加 1；
    Count ++;
}
// 输出计数器的值；
cout << "Count = " << Count << endl;
```

IsPrime(i)是什么？

continue;是什么？

算法实现的完整源程序

```
#include <iostream>
#include <cmath>
using namespace std;
bool IsPrime(int n) {
    bool ans = true;
    for (int k=2; k<=sqrt(n); k++)
        if (n % k == 0) {
            ans = false;
            break;
        }
    return ans;
}
int main() {
    int Count = 0;
    for (int i = 2; i <= 100; i++) {
        if (!IsPrime(i)) continue;
        if ((i-1) % 4 != 0) continue;
        cout << i << endl;
        Count ++;
    }
    cout << "Count = " << Count << endl;
    return 0;
} // L04-05.cpp
```

continue 语句

- 可以在循环语句的循环体中出现。
- **continue**：不继续执行当前循环体中后续的语句，直接跳转到当前所在循环语句循环体的结尾，开始进入下一轮新的循环。因此，若是在 **for** 循环语句中使用，则接下来会执行表达式3、2、循环体，进行新一轮的循环。

【对比】break语句是无条件结束当前所在的最内层循环语句，跳转到该循环语句后的语句，接着执行之后的语句。

算法思路（2）

伪代码

枚举 2，3，… ，100 中的所有数，

逐一进行检验：

如果（ 不是质数 ）

检测下一个数；

如果（ 不可表示为4n＋1 ）

检测下一个数；

输出满足条件的数；

让计数器（变量）加 1；

输出计数器的值；

思路1

枚举 2，3，… ，100 中的所有数，

逐一进行检验：

如果（ 是质数 且可表示为4n＋1 ）

就输出满足条件的数；

让计数器（变量）加 1；

输出计数器的值；

思路2

算法实现的局部代码

```
bool Is4n1(int n) { return (n-1) % 4 == 0; }

int main() {
    int Count = 0;
    for (int i = 2; i <= 100; i++)
        if (IsPrime(i) && Is4n1(i)) {
            cout << i << endl;
            Count ++;
        }
    cout << "Count=" << Count << endl;
    return 0;
}
```

课后思考题

两个程序“重复”定义了判断质数的函数，能否避免重复劳动？

```cpp
#include <iostream> // cout
#include <cmath>    // sqrt
using namespace std;

bool IsPrime(int);

int main() {
    int n = 0;
    cout << "请输入一个整数：n=";
    cin >> n;
    if (IsPrime(n))
        cout << n << "是质数" << endl;
    else
        cout << n << "不是质数" << endl;
    return 0;
}

bool IsPrime(int n) {
    for (int k = 2; k <= sqrt(n); k++)
        if (n % k == 0) return false;
    return true; // n 不能被k整除，返回true
}
```

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

bool IsPrime(int n) {
    bool ans = true;
    for (int k=2; k<=sqrt(n); k++)
        if (n % k == 0) {
            ans = false;
            break;
        }
    return ans;
}
bool Is4n1(int n) { return (n-1) % 4 == 0; }

int main() {
    int Count = 0;
    for (int i = 2; i <= 100; i++)
        if (IsPrime(i) && Is4n1(i)) {
            cout << i << endl;
            Count ++;
        }
    cout << "Count=" << Count << endl;
    return 0;
} // L04-05-V2.cpp
```

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

void say_bye()
{
    cout << "END. See you later!" << endl;
}

int main()
{
    say_bye();
    return 0;
}
```