小技巧1:如何处理实现部分仅微小差异不同类型的数据

例: 画色块 - 合成十 - 打印指定宽度的数字(int) - 消灭星星 - 打印"★"(char *)

```
//先实现打印char* 数据的draw block函数
                                    //再实现打印 int 数据的draw block函数
                                    int draw_block(..., int num, ...
int draw_block(..., char *value, ...)
                 //位置颜色等参数略
                                        cout << "画边框1" << endl;
                                        cout << setw(2) > value << end1 | 5
   cout << "画边框1" << endl:
                                        cout << "画边框n" << endl;
   cout << value << endl; //填充框内
   cout << "画边框n" << endl;
                                        cout << end1;
   cout << endl:
                                                                       10
                                        return 0;
   return 0:
                                    //再实现打印 int 数据的draw block函数
                                    int draw block(..., int num, ...)
两个函数除num/value的输出外,其余均相同
                                        char tmp[80];
int main()
                                        sprintf(tmp, "%2d", num); // %-2d/%02d
                                        //重载调用另一个函数,不再有具体实现
   draw block ("\star");
                                        draw_block(..., tmp, ...);
   draw block (5);
   return 0:
                                        return 0;
```

例: 数组初始化 - 合成十 - array[8][10]

- 消灭星星 - array[10][10] => 更通用, 假设array[7][12], 即行列均不同

```
//90-b0.h
                                           //90-b1.h
#pragma once
                                           #pragma once
                                           #include "../common/common.h"
#include "../common/common.h"
#define MAX ROW
                                           #define MAX ROW 7
#define MAX COL 10
                                           #define MAX COL 12
                                           //90-b1-main. cpp
//90-b0-main. cpp
#include <iostream>
                                           #include <iostream>
#include "90-b0.h"
                                           #include "90-b1.h"
using namespace std:
                                           using namespace std:
int main()
                                           int main()
                                               int array[MAX_ROW][MAX_COL];
    int array[MAX_ROW][MAX_COL];
    init array (array, row, col);
                                                init array (array, row, col);
   return 0;
                                               return 0:
```

例:数组初始化 - 合成十 - array[8][10]
- 消灭星星 - array[10][10] => 更通用, 假设array[7][12],即行列均不同

```
//common. h
#pragma once
template \(\text{typename T, int rowsize, int colsize}\)
int init array(T(&array)[rowsize][colsize], int row, intcol);
//common base.cpp
template <typename T, int rowsize, int colsize>
int init array(T(&array)[rowsize][colsize], int row, int col);
   int i, j;
   cout << sizeof(array) << ' ' << sizeof(array[0]) << endl;</pre>
   for (i = 0; i < row; i++)
                                                 编译错: C++规定函数模板的实现与
       for (i = 0: i < col: i++)
                                                        调用必须在一个源文件中
           arrav[i][i] = 0:
   return 0:
```

例:数组初始化 - 合成十 - array[8][10] - 消灭星星 - array[10][10] => 更通用,假设array[7][12],即行列均不同

```
//common. h
#pragma once
template \(\text{typename T, int rowsize, int colsize}\)
int init array(T(&array)[rowsize][colsize], int row, int col);
    int i, j;
    cout << sizeof(array) << ' ' << sizeof(array[0]) << endl;</pre>
                                                  //90-b1-main. cpp
    for (i = 0; i < row; i++)
                                                  #include <iostream>
        for (j = 0; j < col; j++)
                                                  #include "90-b1.h"
            arrav[i][i] = 0:
                                                  using namespace std:
    return 0;
                                                  int main()
     //90-b1.h
     #pragma once
                                                      int array[MAX ROW][MAX COL];
                                                      init array(array, row, col);
     #include "../common/common.h"
     #define MAX ROW
                                                      return 0;
     #define MAX COL 12
```

例:数组初始化 - 合成十 - array[8][10] - 消灭星星 - array[10][10] => 更通用,假设array[7][12],即行列均不同

```
//common base.cpp
template \(\text{typename T, int rowsize, int colsize}\)
int init array(T(&array)[rowsize][colsize], int row, int col);
   int i, j;
   cout << sizeof(array) << ' ' << sizeof(array[0]) << endl;</pre>
   for (i = 0; i < row; i++)
                                                //90-b1-main. cpp
       for (i = 0: i < col: i++)
                                                #include <iostream>
           arrav[i][i] = 0:
                                                #include "90-b1.h"
                                                using namespace std:
                    问:是否可以?
   return 0:
                    答:可以,但不建议,因为
                                                int main()
                        common base.cpp中可能
     //90-b1.h
                        还有非模板共用函数
     #pragma once
                                                    int array[MAX ROW][MAX COL];
                                                    init array(array, row, col):
     #include "../common/common base.cpp"
     #define MAX ROW
                                                   return 0;
     #define MAX COL 12
```

小技巧3: 如何在通用函数中调用专用函数

例:数组初始化 - 合成十 - array[8][10] - 合成十 - 1~10,不同概率 - 消灭星星 - array[7][12] - 消灭星星 - 1~5,等概率

```
//common.h
#pragma once
template <typename T, int rowsize, int colsize>
int init array(T(&array)[rowsize][colsize], int row, int col, int (*fun getrand)(int, int), int min, int max)
   int i, j;
   cout << sizeof(array) << ' ' << sizeof(array[0]) << endl:</pre>
   for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j] = fun getrand(min, max);
                                                           //90-b0-main. cpp
                                                           #include "90-b0, h"
   return 0:
                                                           using namespace std:
                                                           int getrand 10(int min, int max)
//90-b1-main. cpp
                                                               if (\max \le 3)
#include "90-b1.h"
                                                                   return min~max等概率:
using namespace std:
                                                               else
int getrand star(int min, int max)
                                                                   return min~max不等概率:
   return min~max等概率:
                                                           int main()
int main()
                                                               int array[MAX ROW][MAX COL];
   int array[MAX ROW][MAX COL];
                                                               srand((unsigned int)time(0)):
   srand((unsigned int)time(0));
                                                               init array (array, row, col, getrand 10, 1, 10);
   init array (array, row, col, getrand star, 1, 5);
                                                               return 0;
  return 0:
```