

# 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**

```
int draw_block(..., char *value, ...)
    //位置颜色等参数略
{
    cout << "画边框1" << endl;
    cout << value << endl; //填充框内
    cout << "画边框n" << endl;
    cout << endl;

    return 0;
}
```

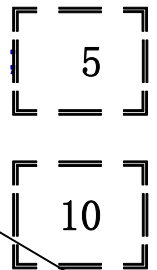
**两个函数除num/value的输出外, 其余均相同**

```
int main()
{
    draw_block("★");
    draw_block(5);
    return 0;
}
```

**//再实现打印 int 数据的draw\_block函数**

```
int draw_block(..., int num, ...)
{
    cout << "画边框1" << endl;
    cout << setw(2) << value << endl;
    cout << "画边框n" << endl;
    cout << endl;

    return 0;
}
```



**//再实现打印 int 数据的draw\_block函数**

```
int draw_block(..., int num, ...)
{
    char tmp[80];
    sprintf(tmp, "%2d", num); // %-2d/%02d
    //重载调用另一个函数, 不再有具体实现
    draw_block(..., tmp, ...);

    return 0;
}
```

## 小技巧2:如何共用使用了函数模板的通用函数

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
#pragma once

#include "../common/common.h"

#define MAX_ROW    8
#define MAX_COL    10
```

```
//90-b0-main.cpp
#include <iostream>
#include "90-b0.h"
using namespace std;

int main()
{
    int array[MAX_ROW][MAX_COL];

    init_array(array, row, col);

    return 0;
}
```

```
//90-b1.h
#pragma once

#include "../common/common.h"

#define MAX_ROW    7
#define MAX_COL    12
```

```
//90-b1-main.cpp
#include <iostream>
#include "90-b1.h"
using namespace std;

int main()
{
    int array[MAX_ROW][MAX_COL];

    init_array(array, row, col);

    return 0;
}
```

## 小技巧2:如何共用使用了函数模板的通用函数

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 <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;

    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j] = 0;

    return 0;
}
```

编译错: C++规定函数模板的实现与调用必须在一个源文件中

## 小技巧2:如何共用使用了函数模板的通用函数

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 <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;

    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j] = 0;

    return 0;
}
```

```
//90-b1.h
#pragma once

#include "../common/common.h"
#define MAX_ROW 7
#define MAX_COL 12
```

```
//90-b1-main.cpp
#include <iostream>
#include "90-b1.h"
using namespace std;

int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);

    return 0;
}
```

## 小技巧2:如何共用使用了函数模板的通用函数

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 <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;

    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j] = 0;

    return 0;
}
```

```
//90-b1.h
#pragma once
```

```
#include "../common/common_base.cpp"
#define MAX_ROW    7
#define MAX_COL    12
```

问: 是否可以?  
答: 可以, 但不建议, 因为  
common\_base.cpp中可能  
还有非模板共用函数

```
//90-b1-main.cpp
#include <iostream>
#include "90-b1.h"
using namespace std;
```

```
int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);

    return 0;
}
```

## 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;
    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j] = fun_getrand(min, max);

    return 0;
}
```

```
//90-b1-main.cpp
#include "90-b1.h"
using namespace std;
int getrand_star(int min, int max)
{
    return min~max等概率;
}
int main()
{
    int array[MAX_ROW][MAX_COL];
    srand((unsigned int)time(0));
    init_array(array, row, col, getrand_star, 1, 5);
    return 0;
}
```

```
//90-b0-main.cpp
#include "90-b0.h"
using namespace std;
int getrand_10(int min, int max)
{
    if (max<=3)
        return min~max等概率;
    else
        return min~max不等概率;
}

int main()
{
    int array[MAX_ROW][MAX_COL];
    srand((unsigned int)time(0));
    init_array(array, row, col, getrand_10, 1, 10);
    return 0;
}
```