



§. 大作业综合+公共函数提取

例1: 菜单函数返回后的处理 (常规的switch/if => 函数指针形式)

```
char sel = menu(****);

switch(sel) {
    case '1':
        fun_1(int);
        break;
    case '2':
        fun_2();
        break;
    ...
}
```

```
char sel = menu(****);

switch(sel) {
    case 'A':
        fun_a();
        break;
    case 'B':
        fun_b(int, int);
        break;
    ...
}
```

```
Fun_Call fc[]={
    {'1', NULL, fun_1, NULL, ...},
    {'2', fun_2, NULL, NULL, ...},
    ...
    {'\0', NULL, NULL, NULL, ...}
}; //保证函数指针仅一个非NULL

char sel = menu(****);

for(i=0; fc[i].sel; i++)
    if (fc[i].sel == sel) {
        if (fc[i].pf1 != NULL)
            fc[i].pf1();
    }
```

```
Fun_Call fc[]={
    {'A', fun_a, NULL, NULL, ...},
    {'B', NULL, NULL, fun_b, ...},
    ...
    {'\0', NULL, NULL, NULL, ...}
}; //保证函数指针仅一个非NULL

char sel = menu(****);

for(i=0; fc[i].sel; i++)
    if (fc[i].sel == sel) {
        if (fc[i].pf1 != NULL)
            fc[i].pf1();
    }
```

```
struct Fun_Call {
    char selected;
    void (*pf1)();
    void (*pf2)(int);
    int (*pf3)(int, int);
    ...//可多种
};
```



§. 大作业综合+公共函数提取

例2: 程序逻辑的处理 (如何处理实现部分仅微小差异不同类型的数据)

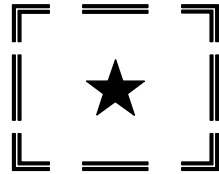
例: 画色块 - 2048/合成十 - 打印指定宽度的数字(int)

- 消灭星星 - 打印“★” (char *)

/* 首先实现打印 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; //value正好填满框内位置
    cout << "画边框n" << endl;
    cout << endl;

    return 0;
}
```



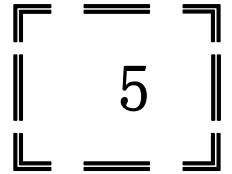
其它数据类型也相似处理

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

/* 重载, 除num/value的输出外, 其余均相同 */

```
int draw_block(..., int num, ...) //位置颜色等参数略
{
    cout << "画边框1" << endl;
    cout << setw(2) << value << endl; //num要占宽度2
    cout << "画边框n" << endl;
    cout << endl;

    return 0;
}
```



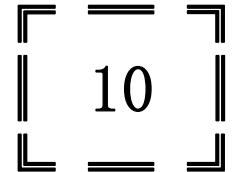
方法1: 直接重载

/* 重载, 直接调用 char* */

```
int draw_block(..., int num, ...) //位置颜色等参数略
{
    char tmp[80];
    sprintf(tmp, "%2d", num); // %-2d/%02d

    draw_block(..., tmp, ...); //重载调用另一个函数
                                //不需要再有实际实现部分

    return 0;
}
```



方法2: 重载中调同名函数



§. 大作业综合+公共函数提取

例3: 如何共用使用了函数模板的通用函数

例: 数组初始化 - 扫雷 - array[12][26] (90-01-b2)

- 消灭星星 - array[10][10] (90-02-b1)

=> 更通用, 假设两个数组行列均不相同

方法1: 各自独立实现(传入指针+行+列)

<pre>//90-01-b2.h #pragma once #include "../common/common.h" #define MAX_ROW 12 #define MAX_COL 26 #include <iostream> #include "90-01-b2.h" using namespace std; int main() { int array[MAX_ROW][MAX_COL]; init_array(array, row, col); return 0; }</pre>	<pre>//90-02-b1.h #pragma once #include "../common/common.h" #define MAX_ROW 10 #define MAX_COL 10 #include <iostream> #include "90-02-b1.h" using namespace std; int main() { int array[MAX_ROW][MAX_COL]; init_array(array, row, col); return 0; }</pre>
---	---



§. 大作业综合+公共函数提取

例3: 如何共用使用了函数模板的通用函数

例: 数组初始化 - 扫雷 - array[12][26] (90-01-b2)
- 消灭星星 - array[10][10] (90-02-b1)

=> 更通用, 假设两个数组行列均不相同

方法2: 用数组的引用统一实现

```
//common.h
#pragma once

template <typename T, int rowsize, int colsize>
int init_array(T(&array)[rowsize][colsize], int row, int col);
```

```
//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

```
//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 < rows; i++)
        for (j = 0; j < col; j++)
            array[i][j] = 0;
    return 0;
}
```

正确, 将模板函数放在 .h 中

```
//90-01-b2.h
#pragma once

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

#define MAX_ROW 12
#define MAX_COL 26
```

```
#include <iostream>
#include "90-01-b2.h"
using namespace std;
int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);
    return 0;
}
```

```
//90-02-b1.h
#pragma once

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

#define MAX_ROW 10
#define MAX_COL 10

#include <iostream>
#include "90-02-b1.h"
using namespace std;
int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);
    return 0;
}
```



§. 大作业综合+公共函数提取

例3: 如何共用使用了函数模板的通用函数

例: 数组初始化 - 扫雷 - array[12][26] (90-01-b2)

- 消灭星星 - array[10][10] (90-02-b1)

=> 更通用, 假设两个数组行列均不相同

方法2: 用数组的引用统一实现

```
//common_base.cpp
```

```
#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 < rows; i++)
        for (j = 0; j < col; j++)
            array[i][j] = 0;
    return 0;
}
```

可以将模板函数放在cpp中, 但不建议, 因为common_base.cpp中可能还有非模板共用函数

```
//90-01-b2.h
```

```
#pragma once
```

```
#include "../common/common_base.cpp"
```

```
#define MAX_ROW 12
```

```
#define MAX_COL 26
```

```
#include <iostream>
#include "90-01-b2.h"
using namespace std;
int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);
    return 0;
}
```

```
//90-02-b1.h
```

```
#pragma once
```

```
#include "../common/common_base.cpp"
```

```
#define MAX_ROW 10
```

```
#define MAX_COL 10
```

```
#include <iostream>
#include "90-02-b1.h"
using namespace std;
int main()
{
    int array[MAX_ROW][MAX_COL];
    init_array(array, row, col);
    return 0;
}
```



§. 大作业综合+公共函数提取

例4: 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] (90-02-b2)

- 消灭星星 - array[7][12] (90-02-b1)

用随机数初始化 - 合成十 - 1~10, 不同概率

消灭星星 - 1~5, 等概率

```
#include "90-02-b2.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;
}
```

```
#include "90-02-b1.h"
using namespace std;
int getrand_star(int min, int max)
{
    return min~max之间的等概率;
}
int main()
{
    int array[MAX_ROW][MAX_COL];
    srand((unsigned int)time(0));
    init_array(array, row, col,
               getrand_star, 1, 5);
    return 0;
}
```

```
//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;
}
```

