§.大作业综合+公共函数提取

例1：菜单函数的通用处理

```
char menu(const char *menu[], const char *choice)
{
    char sel, **pitem;
    //打印菜单并给出选择提示
    cout << "[请选择:] ";
    /* 只有按了choice中的字符才能退出内循环(字母大小写是否敏感由调用者决定) */
    int march=0;
    while (!march) {
        sel = _getch();
        if (strchr(choice, sel) != NULL)     //原始匹配
            march = 1;
        if (march) {
            putchar(sel);  //打印选择，如果getche则不需要
            putchar('\n');
            Sleep(200); //停顿，为了看清楚选择
            }
        }
    return sel;
}
```

```
const char *menuitem[] = {
   "1.***",
   "2.***",
   "5.***",
   "7.***"
   "0.退出",
    NULL
};
const char *choice = "12750";

char sel = menu(menuitem, choice);
```

```
const char *menuitem[] = {
   "A.***",
   "B.***",
   "D.***",
   "F.***"
   "Q.退出",
    NULL
};
const char *choice = "ABDFQ";

char sel = menu(menuitem, choice);
```

```
通过传入的choise+调用者决定：
"ABC" //仅大写
"abc" //仅小写
"abcABC" //大小写表示不同选项

调用者：
char sel = menu(…, "abcABC");
switch(sel) {
    case 'a':
        菜单处理1;
        break;
    case 'A':
        菜单处理2;
        break;
    ...
}
```

```
通过传入的choise+调用者决定：
"ABC"    //仅大写
"abc"    //仅小写
"abcABC" //大小写不敏感

调用者：
char sel = menu(…, "abcABC");
switch(sel) {
    case 'a':
    case 'A':
        菜单处理1;
        break;
    ...
}
```

§.大作业综合+公共函数提取

例2：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
};
```

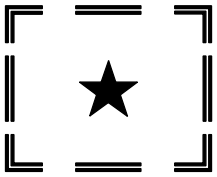
§.大作业综合+公共函数提取

例3：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 *)

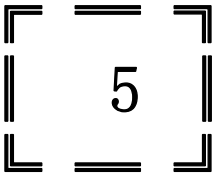
```
/* 首先实现打印 char* 数据的draw_block函数 */

int draw_block(..., char *value, ...) //位置颜色等参数略
{
    //画边框等其它语句
    cout << value << endl; //value正好填满框内位置

    return 0;
}
```



其它数据类型也相似处理

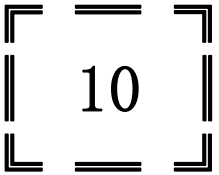
```
int main()
{
    draw_block(..., "★", ...);
    draw_block(..., 5, ...);
    return 0;
}
```

```
/* 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
{
    //画边框等其它语句（与左侧完全相同）
    cout << setw(2) << value << endl; //num要占宽度2

    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4：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]

- 消灭星星 - array[10][10]

=> 更通用，假设两个数组行列均不相同

方法1：不同程序各自独立实现(传入指针+行+列)

//扫雷.h #pragma once #include "../common/common.h" #define MAX_ROW 12 #define MAX_COL 26	//消灭星星.h #pragma once #include "../common/common.h" #define MAX_ROW 10 #define MAX_COL 10

#include <iostream>
#include "扫雷.h"
using namespace std;
 	
int main()
{
int array[MAX_ROW][MAX_COL];
init_array(array, row, col);
return 0;
}
	#include <iostream>
	#include "消灭星星.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]

- 消灭星星 - array[10][10]

=> 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

```
//扫雷.h
#pragma once

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

#define MAX_ROW          12
#define MAX_COL          26
```

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

```
//消灭星星.h
#pragma once

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

#define MAX_ROW          10
#define MAX_COL          10
```

```
#include <iostream>
#include "消灭星星.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]

- 消灭星星 - array[10][10]

=> 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

建议：

1、普通头文件(*.h)
放宏定义、常变量、全局变量声明、函数声明等

2、模板函数头文件(*.hpp)
放模板函数的定义与实现
=> 右侧common_base.cpp
改名为 common)base.hpp

3、源程序文件(*.c/*.cpp)
正常的源程序
注：(1) c/cpp之间不要互相include
(2) 可以include需要的h/hpp

```cpp
//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

```cpp
//扫雷.h
#pragma once

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

#define MAX_ROW          12
#define MAX_COL          26
```

```cpp
//消灭星星.h
#pragma once

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

#define MAX_ROW          10
#define MAX_COL          10
```

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

```cpp
#include <iostream>
#include "消灭星星.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5：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]

- 消灭星星 - array[7][12]

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 "合成十.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 "消灭星星.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;
}
```