复习题

1．哪种函数适合定义为内联函数？

2．假设 song( )函数的原型如下：

```
void song(const char * name, int times);
```

a．如何修改原型，使 times 的默认值为 1？

b．函数定义需要做哪些修改？

c．能否为 name 提供默认值“O. My Papa”？

3．编写 iquote( )的重载版本——显示其用双引号括起的参数。编写 3 个版本：一个用于 int 参数，一个用于 double 参数，另一个用于 string 参数。

4．下面是一个结构模板：

```
struct box
{
    char maker[40];
    float height;
    float width;
    float length;
    float volume;
};
```

a．请编写一个函数，它将 box 结构的引用作为形参，并显示每个成员的值。

b．请编写一个函数，它将 box 结构的引用作为形参，并将 volume 成员设置为其他 3 边的乘积。

5．为让函数 fill()和 show()使用引用参数，需要对程序清单 7.15 做哪些修改？

6．指出下面每个目标是否可以使用默认参数或函数重载完成，或者这两种方法都无法完成，并提供合适的原型。

a．mass(density, volume)返回密度为 density、体积为 volume 的物体的质量，而 mass(denstity)返回密度为 density、体积为 1.0 立方米的物体的质量。这些值的类型都为 double。

b．repeat(10, “I'm OK”)将指定的字符串显示 10 次，而 repeat(“But you're kind of stupid”)将指定的字符串显示 5 次。

c．average(3, 6)返回两个 int 参数的平均值（int 类型），而 average(3.0, 6.0)返回两个 double 值的平均值（double 类型）。

d．mangle(“I'm glad to meet you”)根据是将值赋给 char 变量还是 char*变量，分别返回字符 I 和指向字符串“I'm mad to gleet you”的指针。

7．编写返回两个参数中较大值的函数模板。

8．给定复习题 6 的模板和复习题 4 的 box 结构，提供一个模板具体化，它接受两个 box 参数，并返回体积较大的一个。

9．在下述代码（假定这些代码是一个完整程序的一部分）中，v1、v2、v3、v4 和 v5 分别是哪种类型？

```
int g(int x);
...
float m = 5.5f;
float & rm = m;
decltype(m) v1 = m;
decltype(rm) v2 = m;
decltype((m)) v3 = m;
decltype (g(100)) v4;
decltype (2.0 * m) v5;
```

1.

内联函数是直接把函数代码插入到调用的位置, 节省了调用函数时进行跳转的时间, 但同事会增加代码的占用空间.
使用内敛函数的准则是:

1. 函数代码简单, 不长
2. 函数不是递归函数

2.

```
// a.
void song(const char * name, int times = 1);    // 注意是函数 原型 中增加

// b.
// ans: 只在在函数原型(声明)中添加默认参数, 在函数定义中 不写 任何默认参数

// c.
// ans: 能, 但是此时 times 也需要对应设置默认值. 或者把 name 放在右侧
```

3.

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

// v1:
void iquote(int n)
{
    cout << "\"" << n << "\"" << endl;      // 需要转义字符
}

// v2:
void iquote(double d)
{
    cout << "\"" << d << "\"" << endl;      // 需要转义字符
}

// v3:
void iquote(string s)
{
    cout << "\"" << s << "\"" << endl;      // 需要转义字符
}
```

4.

```
struct box
{
    char maker[40];
    float height;
    float width;
    float length;
    float volume;
};

// a:
void ShowBox(const box & a)
{
    cout << "maker: " << a.maker << endl;
    cout << "height: " << a.height << endl;
    cout << "width: " << a.width << endl;
    cout << "length: " << a.length << endl;
    cout << "volume: " << a.volume << endl;
}

// b:
void SetVolume(box & a)
{
    a.volume = a.height * a.width * a.length;
}
```

5.

- 修改函数声明
- 修改函数体(函数定义)

修改后的代码:

```
#include <iostream>
#include <array>
#include <string>

using namespace std;
const int SEASONS = 4;

void fill(array<double, SEASONS> &pa);
void show(const array<double, SEASONS> &da);

const array<string, SEASONS> Snames = {"Spring", "Summer", "Fall", "Winter"};  // 初始化四个季节的
名字

int main(void)
{
    array<double, SEASONS> expenses;  // expenses 用来存放四个季节的开销
    fill(expenses);                   // 防止拷贝，传指针
    show(expenses);

    return 0;
}
```

```
void fill(array<double, SEASONS> &pa)           // 要修改数组，所以不加 const 做保护
{
    for(int i = 0; i < SEASONS; i++){
        cout << "Enter " << Snames[i] << " expenses: ";
        cin >> pa[i];                          // (*pa) 先将指针转换为 array 对象，只有 array 对象可以
像数组一样操作，但是array指针是不可以直接像指针一样的操作的，因为array指针"并不表示第数组的第一个元素的地址"！
    }
}

void show(const array<double, SEASONS> &da)
{
    double total = 0.0;
    cout << "EXPENSES: " << endl;
    for (int i = 0; i < SEASONS; i++){
        cout << Snames[i] << "\t: $" << da[i] << endl;
        total += da[i];
    }
    cout << "Total: " << total << endl;
}
```

6.

```
// a. 可以直接使用默认值的方式来完成
double mass(double density, double volume = 1.0);

// b. 由于指定显示次数位于左边，而字符串位于右边，此时无法使用默认参数，需要用函数重载
void repeat(const char * str, int times = 5);
void repeat(const char *); // time在函数里写死就行了

// c. 传入参数的类型不同，需要使用重载的函数来完成
int average(int a, int b);
double average(double a, double b);

// d. 注意此时传入一个字符数组和传递一个字符串都是一样的，它们实际传入的就是一个指针，也就是说下面这两个函数的特
征标是相同的，因此 **无法使用重载** 来完成，也 **无法使用默认值** 来完成
char mangle(const char str[]);      // 特征标与下面的函数相同
char * mangle(const char * str);
```

7.

```
template <typename T>
T Max(T a, T b)
{
    return a > b ? a : b;
}
```

8.

```
struct box
{
    char maker[40];
    float height;
    float width;
    float length;
    float volume;
};

template <typename T>
T Max(T a, T b)
{
    return a > b ? a : b;
}

template <> box Max(box b1, box b2)           // 注意返回的类型还是 box
{
    return b1.volume > b2.volume ? b1 : b2;   // 返回的是 box 类型的 b1 或 b2
}
```

9.

```
int g(int x);
...
float m = 5.5f;
float & rm = m;
decltype(m) v1 = m;     // float
decltype(rm) v2 = m;    // float &
decltype((m)) v3 = m;   // float &
decltype(g(100)) v4;    // int
decltype(2.0 * m) v5;   // double   (m 自动从 float 提升为 double)
```