编程练习题

8.8 编程练习

1．编写通常接受一个参数（字符串的地址），并打印该字符串的函数。然而，如果提供了第二个参数（int 类型），且该参数不为 0，则该函数打印字符串的次数将为该函数被调用的次数（注意，字符串的打印次数不等于第二个参数的值，而等于函数被调用的次数）。是的，这是一个非常可笑的函数，但它让您能够使用本章介绍的一些技术。在一个简单的程序中使用该函数，以演示该函数是如何工作的。

2．CandyBar 结构包含 3 个成员。第一个成员存储 candy bar 的品牌名称；第二个成员存储 candy bar 的重量（可能有小数）；第三个成员存储 candy bar 的热量（整数）。请编写一个程序，它使用一个这样的函数，即将 CandyBar 的引用、char 指针、double 和 int 作为参数，并用最后 3 个值设置相应的结构成员。最后 3 个参数的默认值分别为“Millennium Munch”、2.85 和 350。另外，该程序还包含一个以 CandyBar 的引用为参数，并显示结构内容的函数。请尽可能使用 const。

3．编写一个函数，它接受一个指向 string 对象的引用作为参数，并将该 string 对象的内容转换为大写，为此可使用表 6.4 描述的函数 toupper( )。然后编写一个程序，它通过使用一个循环让您能够用不同的输入来测试这个函数，该程序的运行情况如下：

```
Enter a string (q to quit): go away
GO AWAY
Next string (q to quit): good grief!
GOOD GRIEF!
Next string (q to quit): q
Bye.
```

4. 下面是一个程序框架：

```
#include <iostream>
using namespace std;
#include <cstring>      // for strlen(), strcpy()
struct stringy {
    char * str;         // points to a string
    int ct;             // length of string (not counting '\0')
    };

// prototypes for set(), show(), and show() go here
int main()
{
    stringy beany;
    char testing[] = "Reality isn't what it used to be.";

    set(beany, testing);    // first argument is a reference,
                // allocates space to hold copy of testing,
                // sets str member of beany to point to the
                // new block, copies testing to new block,
                // and sets ct member of beany
    show(beany);      // prints member string once
    show(beany, 2);   // prints member string twice
    testing[0] = 'D';
    testing[1] = 'u';
    show(testing);    // prints testing string once
    show(testing, 3); // prints testing string thrice
    show("Done!");
    return 0;
}
```

请提供其中描述的函数和原型，从而完成该程序。注意，应有两个 show( )函数，每个都使用默认参数。请尽可能使用 cosnt 参数。set( )使用 new 分配足够的空间来存储指定的字符串。这里使用的技术与设计和实现类时使用的相似。（可能还必须修改头文件的名称，删除 using 编译指令，这取决于所用的编译器。）

5. 编写模板函数 max5( )，它将一个包含 5 个 T 类型元素的数组作为参数，并返回数组中最大的元素（由于长度固定，因此可以在循环中使用硬编码，而不必通过参数来传递）。在一个程序中使用该函数，将 T 替换为一个包含 5 个 int 值的数组和一个包含 5 个 dowble 值的数组，以测试该函数。

6. 编写模板函数 maxn( )，它将由一个 T 类型元素组成的数组和一个表示数组元素数目的整数作为参数，并返回数组中最大的元素。在程序对它进行测试，该程序使用一个包含 6 个 int 元素的数组和一个包含 4 个 double 元素的数组来调用该函数。程序还包含一个具体化，它将 char 指针数组和数组中的指针数量作为参数，并返回最长的字符串的地址。如果有多个这样的字符串，则返回其中第一个字符串的地址。使用由 5 个字符串指针组成的数组来测试该具体化。

7. 修改程序清单 8.14，使其使用两个名为 SumArray()的模板函数来返回数组元素的总和，而不是显示数组的内容。程序应显示 thing 的总和以及所有 debt 的总和。

1.

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

void show(const char * str, int n = 0);
```

```
int main(void)
{
    show("Hello World!");
    show("Good Morning!");
    show("I love you, Rick.",8);

    return 0;
}

void show(const char * str, int n)
{
    // 定义一个静态变量，它的可以使得在函数中定义的局部变量相当于一个"全局变量"，它只会被初始化1次.
    static int num = 0;      // 这个语句只会被调用1次
    num++;                   // 每次调用函数都会执行一次

    if (0 == n){
        cout << str << endl;
    }
    else{
        for(int i = 0; i < num; i++){
            cout << str << endl;
        }
    }
}
```

2.

```
#include <iostream>
#include <cstring>

using namespace std;
const int SIZE = 30;
struct CandyBar
{
    char brand[SIZE];
    double weight;
    int heat;
};

void setValue(CandyBar & candybar, const char * str = "Millennium Munch", const double w = 2.85,
const int h = 350);

void show(const CandyBar & bar);

int main(void)
{
    CandyBar bar;
    setValue(bar);
    show(bar);
    return 0;
}

void setValue(CandyBar & candybar, const char * str, const double w, const int h)
{
    strcpy(candybar.brand, str);
```

```
    candybar.weight = w;
    candybar.heat = h;
}

void show(const CandyBar & bar)
{
    cout << "Brand: " << bar.brand << endl;
    cout << "Weight: " << bar.weight << endl;
    cout << "Heat: " << bar.heat << endl;
}
```

3.

```
#include <iostream>
#include <string>
#include <cctype>       // 为了使用 toupper 函数

using namespace std;

void to_upper(string &str);

int main(void)
{
    cout << "Enter a string (q to quit): ";
    string str;

    getline(cin, str);
    while( str != "q"){
        to_upper(str);
        cout << str << endl;
        cout << "Enter a string (q to quit): ";
        getline(cin, str);
    }
    cout << "Bye!" << endl;

    return 0;
}

void to_upper(string &str)
{
    for (int i = 0; i < str.size(); i ++){
        str[i] = toupper(str[i]);
    }
}
```

4.

```
#include <iostream>
#include <cstring>
```

```
using namespace std;
struct stringy
{
    char *str;
    int ct;
};

void set(stringy &str, const char *source);
void show(const stringy & beany, int n = 1);
void show(const char *str, int n = 1);

int main(void)
{
    stringy beany;
    char testing[] = "Reality isn't what it used to be.";

    set(beany, testing);
    show(beany);
    show(beany, 2);
    cout << "----" << endl;
    testing[0] = 'D';
    testing[1] = 'u';
    show(testing);
    show(testing, 3);
    show("Done!");

    delete beany.str;
    return 0;
}

void set(stringy &str, const char *source)
{
    str.ct = strlen(source) + 1;        // 记录字符串的大小
    str.str = new char[str.ct];         // 注意应该使用方括号!!
    strcpy(str.str, source);
}

void show(const stringy & beany, int n)
{
    for (int i = 0; i < n; i++){
        cout << beany.str << endl;
    }
}

void show(const char *str, int n)
{
    for (int i = 0; i < n; i++){
        cout << str << endl;
    }
}
```

5.

```
#include <iostream>

using namespace std;
```

```
template <typename T>
T max5(T t[]);

int main(void)
{
    int arr_i[5] = {1,3,5,7,9};
    double arr_d[5] = {1.3, 22.2, 13.8, 17.9, 14.2};

    cout << "The max value of arr_i: " << max5(arr_i) << endl;
    cout << "The max value of arr_d: " << max5(arr_d) << endl;

    return 0;
}

template <typename T>
T max5(T t[])
{
    T max = t[0];
    for (int i = 1; i < 5; i++){
        if (max < t[i]){
            max = t[i];
        }
    }
    return max;
}
```

6.

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

using namespace std;

template <typename T>
T maxn(T t[], int n);

template <>
string maxn <string> (string str[], int n);

int main(void)
{
    int arr_i[6] = {1,3,5,7,9,11};
    double arr_d[4] = {1.3, 22.2, 13.8, 17.9};

    string str[5] = {"hello world", "good morning", "I love you Rick", "What is this?", "Bye"};

    cout << "The max value of arr_i: " << maxn(arr_i, 6) << endl;
    cout << "The max value of arr_d: " << maxn(arr_d, 4) << endl;
    cout << "The max value of str: " << maxn(str, 5) << endl;

    return 0;
}

template <typename T>
T maxn(T t[], int n)
```

```
{
    T max = t[0];
    for (int i = 1; i < n; i++){
        if (max < t[i]){
            max = t[i];
        }
    }
    return max;
}

// 针对 char * 做具体化
template <>
string maxn <string>(string str[], int n)       // 注意 str 是数组，别漏了 []
{
    int pos = 0;
    for (int i = 0; i < n; i++){
        if (str[pos].size() < str[i].size()){
            pos = i;
        }
    }
    return str[pos];
}
```

7.

```
#include <iostream>

using namespace std;

template <typename T>
void ShowArray(T arr[], int n);      // #1

template <typename T>
void ShowArray(T * arr[], int n);    // #2

template <typename T>
T SumArray(T arr[], int n);

template <class T>
T SumArray(T *arr[], int n);

struct debts
{
    char name[50];
    double amount;
};

int main(void)
{
    int things[6] = {13, 31, 03, 301, 310, 130};

    ShowArray(things, 6);

    struct debts mr_E[3] =      // 这个 struct 加不加都可以
    {
        {"Rick", 2400.00},
```

```
        {"Jack", 1300.00},
        {"Rose", 1800.00}
    };

    double *pd[3];   // 指针数组，数组有3个元素，每个元素都是double类型的指针

    for(int i = 0; i < 3; i++){
        pd[i] = &mr_E[i].amount;        // 传递数据的指针
    }

    ShowArray(pd, 3);       // 此时若匹配到 #1，打印出来的是地址；如果匹配到 #2，则正常输出.
                            // 从实验结果来看，会匹配到#2，而且 #2 确实是"最省事"的

    cout << "The sum of things: " << SumArray(things, 6) << endl;
    cout << "The sum of pd: " << SumArray(pd, 3) << endl;

    return 0;
}

template <typename T>
void ShowArray(T arr[], int n)      // #1
{
    cout << "Template A: " << endl;
    for (int i = 0; i < n; i++){
        cout << arr[i] << " " ;
    }
    cout << endl;
}

template <typename T>
void ShowArray(T * arr[], int n)    // #2
{
    cout << "Template B: " << endl;
    for (int i = 0; i < n; i++){
        cout << *arr[i] << " " ;
    }
    cout << endl;
}

template <typename T>
T SumArray(T arr[], int n)
{
    T Sum = 0;
    for(int i = 0; i < n; i++){
        Sum += arr[i];
    }
    return Sum;
}

template <class T>
T SumArray(T *arr[], int n)
{
    T Sum = 0;
    for (int i = 0; i < n; i++){
        Sum += *arr[i];
    }
    return Sum;
}
```