《C++ Primer》5th
IDE:Visual Studio 2015
20th 毛金勇

目录

第 1 章 开始 1
第 3 章 字符串、向量和数组 5
第 4 章 表达式 15
第 5 章 语句 16
第 6 章 函数 23
第 7 章：类 32
第 8 章：IO 库 32
第 9 章 顺序容器 38
第 10 章：泛型算法 53
第 11 章：关联容器 68

第 1 章 开始

P8

练习 1.3

```
#include<iostream>
int main()
{
	std::cout << "Hello,World." << std::endl;
	return 0;
}
```

练习 1.4

```
#include<iostream>
int main()
{
	int v1,v2;
	std::cin >> v1 >> v2;
	std::cout << "The sum of "<< v1 <<" and "<< v2 << " is " << v1*v2 << std::endl;
	return 0;
}
```

练习 1.6

```
#include<iostream>
int main()
{
	int v1,v2;
```

```
    std::cin >> v1 >> v2;
    std::cout << "The sum of "<< v1;
                <<" and "<< v2;
               << " is " << v1+v2 << std::endl;
    return 0;
}
```

以上程序是**不合法**的。原因在于：<<运算符接受两个运算对象，左侧的运算对象必须是一个 ostream 对象，右侧的运算对象是要打印的值。

因此应该修改为：

```
#include<iostream>
int main()
{
    int v1,v2;
    std::cin >> v1 >> v2;
    std::cout << "The sum of "<< v1;
    std::cout <<" and "<< v2;
    std::cout << " is " << v1+v2 << std::endl;
    return 0;
}
```

或者，**去掉分号，只保留最后一个**

```
#include<iostream>
int main()
{
    int v1,v2;
    std::cin >> v1 >> v2;
    std::cout << "The sum of "<< v1
                <<" and "<< v2
               << " is " << v1+v2 << std::endl;
    return 0;
}
```

P9

练习 1.9

```
#include<iostream>
int main()
{
    int sum = 0, val = 50;
    while(val <= 100){
        sum += val;
        ++val;
    }
    std::cout << "Sum of 50 to 100 is " << sum << std::endl;
    return 0;
}
```

练习 1.10

```
#include<iostream>
int main()
{
	int val = 10;
	while(val >= 0){
		std::cout << val << " ";
		--val;
	}
	return 0;
}
```

练习 1.11

```
#include<iostream>
int main()
{
	int v1,v2;
	std::cout << "Please enter two numbers" << std::endl;
	std::cin >> v1 >> v2;
	if(v1 > v2){
		std::cout << "ERROR!" << std::endl;
	}
	while(v1 <= v2){
		std::cout << v1 << " ";
		++v1;
	}
	return 0;
}
```

P15

练习 1.16

```
/*读取数量不定的输入数据*/
#include<iostream>
int main()
{
	int sum = 0,val;
	while(std::cin >> val){ //输入文件结束符的方法是：Ctrl+Z 然后按 Enter
		sum += val;
	}
	std::cout << "Sum is: " << sum << std::endl;
	return 0;
}
```

P16

练习 1.18

```
/*统计在输入中每个值连续出现的次数*/
#include<iostream>
```

```
int main()
{
	int currVal = 0,val = 0;
	if(std::cin >> currVal){
		int cnt = 1;
		while(std::cin >> val){
			if(val == currVal) ++cnt;
			else{
				std::cout << currVal << " occurs " << cnt << " times." << std::endl;
				currVal = val;//记住新值
				cnt = 1;//重置计数器
			}
		}
		//打印文件最后一个值的个数
		std::cout << currVal << " occurs " << cnt << " times." << std::endl;
	}
	return 0;
}
```

P20

练习 1.20（先下载头文件 Sales_item.h）

```
#include<iostream>
#include"Sales_item.h"
int main()
{
	Sales_item book;
	while(std::cin >> book){
		std::cout << book << std::endl;
	}
	return 0;
}
```

练习 1.21

```
#include<iostream>
#include"Sales_item.h"
int main()
{
	Sales_item item1,item2;
	std::cin >> item1 >> item2;
	std::cout << item1+item2 << std::endl;

	return 0;
}
```

练习 1.22

```
#include<iostream>
#include"Sales_item.h"
```

```
int main()
{
	Sales_item book,total;
	while(std::cin >> book){
		total += book;
	}
	std::cout << total << std::endl;
	return 0;
}
```

P21

练习 1.23 & 1.24（思路和练习 1.18 很像的）

```
#include<iostream>
#include"Sales_item.h"
int main()
{
	Sales_item total;
	if(std::cin >> total){
		Sales_item temp;
		while(std::cin >> temp){
			if(total.isbn() == temp.isbn()){
				total += temp;
			}else{
				std::cout << total << std::endl;
				total = temp;
			}
		}
		std::cout << total << std::endl;
	}
	return 0;
}
```

第 3 章 字符串、向量和数组

P81

练习 3.2

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string s1;
	getline(cin,s1);//读入一整行，包括空白符，遇到换行符则停止
	cout << s1 << endl;
	string s2;
```

```
	while(cin >> s2){//读取数量未知的 string 对象
		cout << s2 << endl;
	}
	return 0;
}
```

练习 3.4

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string s1,s2;
	cin >> s1 >> s2;
	if(s1 != s2){//输出较大的字符串
		if(s1 > s2) cout << s1 << endl;
		if(s1 < s2) cout << s2 << endl;
	}
	if(s1.size() != s2.size()){//输出较长的字符串
		if(s1.size() > s2.size()) cout << s1 << endl;
		if(s1.size() < s2.size()) cout << s2 << endl;
	}
	return 0;
}
```

练习 3.5

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string strPart,strTotal;
	while(cin >> strPart){
		strTotal += strPart + " ";
	}
	cout << strTotal << endl;
	return 0;
}
```

P86

练习 3.6/3.7/3.8

```
#include "stdafx.h"
#include<iostream>
#include<string>
using namespace std;
int main()
{
```

```
    string s;
    cin >> s;
    //使用范围 for 语句
    for (auto &c : s) {
        c = 'X';
    }
    cout << s << endl;
    for (char &c : s) {
        c = 'X';
    }
    cout << s << endl;
    //使用下标迭代
    for (decltype(s.size()) index = 0; index != s.size(); ++index) {
        s[index] = 'X';
    }
    cout << s << endl;
    return 0;
}
```

练习 3.10

```
#include "stdafx.h"
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string s;
    getline(cin, s);
    for (decltype(s.size()) i = 0; i != s.size(); ++i) {
        if (!ispunct(s[i])) cout << s[i];//如果不是标点符号，就输出
    }
    cout << endl;
    return 0;
}
```

P90

练习 3.13（验证）

```
#include "stdafx.h"
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int main()
{
    vector<int> v1;
```

```
    vector<int> v2(10);
    vector<int> v3(10, 42);
    vector<int> v4{ 10 };
    vector<int> v5{ 10,42 };
    vector<string> v6{ 10 };
    vector<string> v7{ 10,"hi" };
    cout << v1.size() << endl;
    for (int i = 0; i != v1.size();++i) {
        cout << v1[i] << " ";
    }
    cout << endl;
    cout << v2.size() << endl;
    for (int i = 0; i != v2.size(); ++i) {
        cout << v2[i] << " ";
    }
    cout << endl;
    cout << v3.size() << endl;
    for (int i = 0; i != v3.size(); ++i) {
        cout << v3[i] <<" ";
    }
    cout << endl;
    cout << v4.size() << endl;
    for (int i = 0; i != v4.size(); ++i) {
        cout << v4[i] << " ";
    }
    cout << endl;
    cout << v5.size() << endl;
    for (int i = 0; i != v5.size(); ++i) {
        cout << v5[i] << " ";
    }
    cout << endl;
    cout << v6.size() << endl;
    for (int i = 0; i != v6.size(); ++i) {
        cout << v6[i] << " ";
    }
    cout << endl;
    cout << v7.size() << endl;
    for (int i = 0; i != v7.size(); ++i) {
        cout << v7[i] << " ";
    }
    cout << endl;
    return 0;
}
```

输出为：

```
0

10
0 0 0 0 0 0 0 0 0 0
10
42 42 42 42 42 42 42 42 42 42
1
10
2
10 42
10

10
hi hi hi hi hi hi hi hi hi hi
请按任意键继续. . .
```

P91

练习 3.14

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> vi;
	int number;
	while(cin >> number){
		vi.push_back(number);
	}
	for(int i=0;i!=vi.size();++i){
		cout << vi[i] << " ";
	}
	cout << endl;
	return 0;
}
```

练习 3.15

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
int main()
{
	vector<string> vs;
	string str;
	while(cin >> str){
		vs.push_back(str);
	}

	for(int i=0;i!=vs.size();++i){
		cout << vs[i] << " ";
	}
	cout << endl;
```

```
	return 0;
}
```

P94

练习 3.17

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
int main()
{
	vector<string> text;
	string word;
	while(cin >> word){
		text.push_back(word);
	}

	for(int i=0;i!=text.size();++i){
		for(int j=0;j!=text[i].size();++j){
			text[i][j] = toupper(text[i][j]);
		}
	}
	for(int i=0;i!=text.size();++i){
		cout << text[i] << endl;
	}
	return 0;
}
```

练习 3.19

```
vector<int> vi1(10,42);
vector<int> vi2 = {42,42,42,42,42,42,42,42,42,42};
vector<int> vi3{42,42,42,42,42,42,42,42,42,42};
```

练习 3.20

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> vi;
	int number;
	while(cin >> number){
		vi.push_back(number);
	}

	for(int i=0;i!=vi.size();i+=2){
		cout << vi[i] + vi[i+1] << " ";
```

```
	}
	cout << endl;
	for(int i=0,j=vi.size()-1;i < j;++i,--j){
		cout << vi[i] + vi[j] << " ";
	}
	return 0;
}
```

P99

练习 3.23

```
#include "stdafx.h"
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> vi = { 1,2,3,4,5,6 };
	auto end = vi.end();
	for (auto it = vi.begin(); it != end; ++it) {
		*it *= 2;
	}
	for (auto it = vi.begin(); it != end; ++it) {
		cout << *it << " ";
	}
	return 0;
}
```

P101

练习 3.25

```
#include "stdafx.h"
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<unsigned> scores(11,0);
	unsigned grade;
	auto beg = scores.begin();
	while (cin >> grade) {
		if (grade <= 100) {
			++*(beg + grade / 10);
		}
	}
	auto end = scores.end();
	for (auto it = scores.begin(); it != end; ++it) {
		cout << *it << " ";
```

```
    }
    return 0;
}
```

P104

练习 3.31/3.32

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int ai[10],copy[10];
    for(int i = 0;i < 10;++i){
        ai[i] = i;
    }
    //数组拷贝
    for(int i = 0;i < 10;++i){
        copy[i] = ai[i];
    }
    vector<int> vi,vcopy;
    for(int i = 0;i < 10;++i){
        vi.push_back(i);
    }
    vcopy = vi;//vector 拷贝
    return 0;
}
```

练习 3.33

```
#include<iostream>
using namespace std;
int main()
{
    unsigned scores[11];//未初始化
    unsigned grade;
    while(cin >> grade){
        if(grade <= 100){
            ++scores[grade/10];
        }
    }
    for(int i=0;i<11;++i){
        cout << scores[i] << " ";
    }
    return 0;
}
```

当未初始化时，会出现如下状况：

```
42 65 95 100 39 67 95 76 88 76 83 92 76 93^Z
4294967295 4294967295 4253557 1 2 0 4254491 3 2 4 76
```

P108

练习 3.35

```
#include<iostream>
using namespace std;
int main()
{
	int a[5] = {1,2,3,4,5};
	for(int* p = a; p < a + 5; ++p){
		*p = 0;
	}
	return 0;
}
```

练习 3.36

```
#include "stdafx.h"
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	bool isSame = true;//表示两个数组一样
	int a1[5] = { 1,2,3,4,5 }, a2[5] = { 1,2,3,4,6};
	int *ptr1 = a1, *ptr2 = a2;
	int *endPtr1 = a1 + 4, *endPtr2 = a2 + 4;
	for (; ptr1 <= endPtr1 && ptr2 <= endPtr2; ++ptr1, ++ptr2) {
		if (*ptr1 != *ptr2) {
			isSame = false;
			break;
		}
	}
	if (!isSame) {
		cout << "a1 is not the same as a2." << endl;
	}
	else {
		cout << "a1 is the same as a2." << endl;
	}
	vector<int> v1 = { 1,2,3,4,5 }, v2 = { 1,2,3,4,6 };
	auto p1 = v1.begin(), p2 = v2.begin();
	auto endP1 = v1.end(), endP2 = v2.end();
	for (; p1 < endP1 && p2 < endP2; ++p1, ++p2) {
		if (*p1 != *p2)
			break;
	}
	if (p1 != endP1) {
		cout << "v1 is not the same as v2." << endl;
```

```
	}
	else {
		cout << "v1 is the same as v2." << endl;
	}
	return 0;
}
```

P110

练习 3.40

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
	char str1[] = "hello,";
	char str2[] = "world";
	char str3[100];//该数组的大小要大于 str1  与 str2  之和
	strcat(str1,str2);//把 str2 附加到 str1,返回 str1
	strcpy(str3,str1);//把 str1 拷贝给 str3，返回 str3
	cout << str3 << endl;
	return 0;
}
```

P112

练习 3.41

```
#include "stdafx.h"
#include<iostream>
#include<vector>
using namespace std;
int main() {
	int a[6] = { 1,2,3,4,5,6 };
	vector<int> vi(begin(a), end(a));//利用数组初始化 vi
	for (int i = 0; i<6; ++i) {
		cout << vi[i] << " ";
	}
	return 0;
}
```

P116

练习 3.45

```
#include "stdafx.h"
#include<iostream>
using namespace std;
int main() {
	int arr[3][4] = { 1,2,3,4,5,6,7,8,9,10,11,12 };
	//范围 for 语句迭代
	for (auto &row : arr) {
```

```
        for (auto &col : row) {
            cout << col << " ";
        }
        cout << endl;
    }
    //下标迭代
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 4; ++j) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    //指针迭代
    for (auto p = arr; p != arr + 3; ++p) {
        for (auto q = *p; q != *p + 4; ++q) {
            cout << *q << " ";
        }
        cout << endl;
    }
    return 0;
}
```

第 4 章 表达式

P140

练习 4.28

```
#include<iostream>
using namespace std;
int main()
{
    cout << "类型名称\t" << "所占空间" << endl;
    cout << "bool\t\t" << sizeof(bool) << endl;
    cout << "char\t\t" << sizeof(char) << endl;
    cout << "wchar_t\t\t" << sizeof(wchar_t) << endl;
    cout << "char16_t\t" << sizeof(char16_t) << endl;
    cout << "char32_t\t" << sizeof(char32_t) << endl;
    cout << "short\t\t" << sizeof(short) << endl;
    cout << "int\t\t" << sizeof(int) << endl;
    cout << "long\t\t" << sizeof(long) << endl;
    cout << "long long\t" << sizeof(long long) << endl;
    cout << "float\t\t" << sizeof(float) << endl;
    cout << "double\t\t" << sizeof(double) << endl;
    cout << "long double\t" << sizeof(long double) << endl;
    return 0;
}
```

输出结果：

```
类型名称        所占空间
bool            1
char            1
wchar_t         2
char16_t        2
char32_t        4
short           2
int             4
long            4
long long       8
float           4
double          8
long double     8
请按任意键继续. . .
```

第 5 章 语句

P159

练习 5.5

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	int grade;
	cout << "请输入一个分数：";
	cin >> grade;
	if (grade < 0 || grade>100) {
		cout << "输入无效！" << endl;
		return -1;
	}
	if (grade < 60) {
		cout << "F" << endl;
		return -1;
	}
	if (grade == 100) {
		cout << "A++" << endl;
		return -1;
	}
	string letter, rank, letterRank;
	int unit_digit = grade % 10, tens_digit = grade / 10;
	if (tens_digit == 9) {
		letter = "A";
	}
	else if (tens_digit == 8) {
		letter = "B";
	}
	else if (tens_digit == 7) {
```

```
            letter = "C";
        }
        else {
            letter = "D";
        }
        if (unit_digit > 7) {
            rank = "+";
        }
        else if (unit_digit < 3) {
            rank = "-";
        }
        else {
            rank = "";
        }
        letterRank = letter + rank;
        cout << letterRank << endl;
        return 0;
}
```

练习 5.6

```
        int unit_digit = grade % 10, tens_digit = grade / 10;//个位数，十位数
        letter = tens_digit == 9 ? "A" : (tens_digit == 8 ? "B" : (tens_digit == 7 ? "C" : "D"));
        rank = unit_digit > 7 ? "+" : (unit_digit < 3 ? "-" : "");
        letterRank = letter + rank;
```

p164

练习 5.10

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
        unsigned int aCnt = 0, eCnt = 0, iCnt = 0, oCnt = 0, uCnt = 0;
        string str;
        cin >> str;
        for (auto elem : str) {
            if (elem == 'a' || elem == 'A')
                ++aCnt;
            if (elem == 'e' || elem == 'E')
                ++eCnt;
            if (elem == 'i' || elem == 'I')
                ++iCnt;
            if (elem == 'o' || elem == 'O')
                ++oCnt;
            if (elem == 'u' || elem == 'U')
                ++uCnt;
```

```
    }
    cout << "a 或 A 的个数：" << aCnt << endl;
    cout << "e 或 E 的个数：" << eCnt << endl;
    cout << "i 或 I 的个数：" << iCnt << endl;
    cout << "o 或 O 的个数：" << oCnt << endl;
    cout << "u 或 U 的个数：" << uCnt << endl;
    return 0;
}
```

练习 5.11

```
#include<iostream>
using namespace std;
int main()
{
    unsigned int aCnt = 0, eCnt = 0, iCnt = 0, oCnt = 0, uCnt = 0;
    unsigned int spaceCnt = 0, tabCnt = 0, newlineCnt = 0;
    char elem;
    while (cin.get(elem)) {//可接收空格符，换行符，制表符
        if (elem == 'a' || elem == 'A')
            ++aCnt;
        if (elem == 'e' || elem == 'E')
            ++eCnt;
        if (elem == 'i' || elem == 'I')
            ++iCnt;
        if (elem == 'o' || elem == 'O')
            ++oCnt;
        if (elem == 'u' || elem == 'U')
            ++uCnt;
        if (elem == ' ')
            ++spaceCnt;
        if (elem == '\t')
            ++tabCnt;
        if (elem == '\n')
            ++newlineCnt;
    }
    cout << "a 或 A 的个数：" << aCnt << endl;
    cout << "e 或 E 的个数：" << eCnt << endl;
    cout << "i 或 I 的个数：" << iCnt << endl;
    cout << "o 或 O 的个数：" << oCnt << endl;
    cout << "u 或 U 的个数：" << uCnt << endl;
    cout << "空格符的个数：" << spaceCnt << endl;
    cout << "制表符的个数：" << tabCnt << endl;
    cout << "换行符的个数：" << newlineCnt << endl;
    return 0;
}
```

练习 5.12

```
#include<iostream>
using namespace std;
int main()
{
	int ffCnt = 0, fiCnt = 0, flCnt = 0;
	char ch, prech = '\0';
	while (cin >> ch) {
		bool flag = true;
		if (prech == 'f') {
			switch (ch)
			{
			case 'f':
				++ffCnt;
				flag = false;
				break;
			case 'i':
				++fiCnt;
				break;
			case 'l':
				++flCnt;
				break;
			}
		}
		if (!flag)
			prech = '\0';
		else
			prech = ch;
	}
	cout << "ff 的数量：" << ffCnt << endl;
	cout << "fi 的数量：" << fiCnt << endl;
	cout << "fl 的数量：" << flCnt << endl;
	return 0;
}
```

P166

练习 5.14

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
int main()
{
	vector<string> vs;
	string word, maxWord;
```

```
	int nowCnt = 1, maxCnt = 0;
	while (cin >> word) {
		vs.push_back(word);
	}
	for (int i = 1; i != vs.size(); ++i) {
		if (vs[i] == vs[i - 1]) {
			++nowCnt;
			if (nowCnt > maxCnt) {
				maxWord = vs[i];
				maxCnt = nowCnt;
			}
		}
		else {
			nowCnt = 1;
		}
	}
	if (maxCnt == 0) {
		cout << "没有连续出现过的单词！ " << endl;
	}
	else {
		cout << maxWord << "连续出现了" << maxCnt << "次" << endl;
	}
	return 0;
}
```

P168

练习 5.17

```
#include<iostream>
#include<vector>
using namespace std;
bool compare(const vector<int> v1, const vector<int> v2)
{
	vector<int> shortVi, longVi;
	shortVi = (v1.size() < v2.size()) ? v1 : v2;
	longVi= (v1.size() > v2.size()) ? v1 : v2;
	int size = shortVi.size();
	for (int i = 0; i != size; ++i) {
		if (shortVi[i] != longVi[i]) {
			return false;
		}
	}
	return true;
}
int main()
{
```

```
	vector<int> v1 = { 0,1,1,2 }, v2 = { 0,1,1,2,3,5,8 };
	if (compare(v1, v2))
		cout << "是前缀！" << endl;
	else
		cout << "不是前缀！" << endl;
	return 0;
}
```

P170

练习 5.19

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	do {
		string str1, str2;
		cout << "请输入两个字符串：";
		cin >> str1 >> str2;
		if (str1.size() < str2.size())
			cout << "较短的字符串是：" << str1 << endl;
		else if (str1.size() > str2.size())
			cout << "较短的字符串是：" << str2 << endl;
		else
			cout << "两个字符串相等" << endl;

	} while (cin);
	return 0;
}
```

P171

练习 5.20

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	bool isContinue = false;//表示不连续
	string word, preWord = "";
	while (cin >> word) {
		if (word == preWord) {
			isContinue = true;
			cout << word << endl;
			break;
		}
		preWord = word;
```

```
    }
    if (!isContinue)
        cout << "没有任何单词是连续重复的！" << endl;
    return 0;
}
```

P177

练习 5.23

```
#include<iostream>
using namespace std;
int main()
{
    int val1, val2;
    cout << "请输入两个数：" << endl;
    cin >> val1 >> val2;
    if (val2 == 0) {
        cout << "除数不能为 0！" << endl;
        return -1;
    }
    cout << val1 / val2 << endl;
    return 0;
}
```

练习 5.25

```
#include<iostream>
#include<stdexcept>
using namespace std;
int main()
{
    int val1, val2;
    cout << "请输入两个数：" << endl;
    while (cin >> val1 >> val2) {
        try {
            if (val2 == 0) {
                throw runtime_error("除数不能为 0！");
            }
            cout << val1 / val2 << endl;
        }
        catch (runtime_error err) {
            cout << err.what() << endl;
            cout << "要继续吗? Y or N" << endl;
            char ch;
            cin >> ch;
            if (ch == 'N')
                break;
        }
```

```
    }
    return 0;
}
```

第 6 章 函数

P184

练习 6.4

```
#include "stdafx.h"
#include<iostream>
using namespace std;
int fact(int n)
{
    if (n < 0) return -1;
    if (n == 0 || n == 1) return 1;
    return n*fact(n - 1);
}
int main()
{
    int number;
    cout << "please enter a number: ";
    cin >> number;
    cout << number << "! is " << fact(number) << endl;
    return 0;
}
```

练习 6.5

```
#include<iostream>
#include<cmath>
using namespace std;
double myABS(double val)
{
    if (val < 0) return -val;
    else return val;
}
double myABS2(double val)
{
    return abs(val);
}
int main()
{
    double num;
    cout << "请输入一个数：";
    cin >> num;
    cout << num << "的绝对值是：  " << myABS(num) << endl;
```

```
	cout << num << "的绝对值是：  " << myABS2(num) << endl;
	return 0;
}
```

P185

练习 6.6

```
#include<iostream>
using namespace std;
int add(int v1, int v2)//形参
{
	int sum = v1 + v2;
	static unsigned cnt = 0;//静态局部变量
	++cnt;
	cout << "该函数已经累计执行了" << cnt << "次" << endl;
	return sum;
}
int main()
{
	int v1, v2;
	cout << "请输入两个整数：  ";
	while (cin >> v1 >> v2) {
		cout << v1 << "  和  " << v2 << "的和为：" << add(v1, v2) << endl;
	}
	return 0;
}
```

练习 6.7

```
#include<iostream>
using namespace std;
int cnt_call()
{
	static int cnt = -1;
	return ++cnt;
}
int main()
{
	for (int i = 1; i <= 10; ++i) {
		cout << "函数第  " << cnt_call() << "  次被调用" << endl;
	}
	return 0;
}
```

P188

练习 6.10

```
#include<iostream>
using namespace std;
void swap(int* p1, int* p2)
```

```
{
	int temp;
	temp = *p1;
	*p1 = *p2;
	*p2 = temp;
}
int main()
{
	int a = 1, b = 2;
	int *p1 = &a, *p2 = &b;
	swap(p1, p2);
	cout << "*p1 = " << *p1 << " *p2 = " << *p2 << endl;
	cout << "a = " << a << " b = " << b << endl;
	return 0;
}
```

P190

练习 6.12

```
#include<iostream>
using namespace std;
void swap(int &v1, int &v2)
{
	int temp;
	temp = v1;
	v1 = v2;
	v2 = temp;
}
int main()
{
	int a = 1, b = 2;
	swap(a, b);
	cout << "a = " << a << " b = " << b << endl;
	return 0;
}
```

与使用指针相比，使用引用交换变量的内容更简单，无需额外声明指针变量，也避免了拷贝指针的值。

练习 6.13

```
#include<iostream>
using namespace std;
void f1(int);//传值参数
void f2(int&);//传引用参数

void f1(int a)
{
	++a;
	cout << a << endl;
```

```
}
void f2(int &a)
{
	++a;
	cout << a << endl;
}
int main()
{
	int s = 0, t = 10;
	f1(s);				//输出 1
	cout << s << endl;		//输出 0
	f2(t);				//输出 11
	cout << t << endl;		//输出 11
	return 0;
}
```

P192

练习 6.17

```
#include<iostream>
#include<string>
using namespace std;
bool hasUpper(const string& str)//使用常量引用
{
	for (auto c : str) {
		if (isupper(c))
			return true;
	}
	return false;
}
void toLower(string& str)
{
	for (auto &c : str) {
		if (isupper(c))
			c = tolower(c);
	}
}
int main()
{
	string str;
	cout << "请输入一个字符串：";
	cin >> str;
	if (hasUpper(str)) {
		toLower(str);
		cout << "转换后的字符串是：  " << str << endl;
	}
```

```
	else {
		cout << "该字符串不含大写字母，无需转换" << endl;
	}
	return 0;
}
```

P196

练习 6.21

```
#include<iostream>
using namespace std;
int compare(const int a, const int *p)//最好别忘了 const
{
	return a > *p ? a : *p;
}
int main()
{
	int v1, v2, v3;
	cout << "请输入两个整数：";
	cin >> v1 >> v2;
	v3 = compare(v1, &v2);
	cout << "较大的数是："<< v3 << endl;
	return 0;
}
```

练习 6.22

```
#include<iostream>
using namespace std;
//该函数既不交换指针，也不交换指针的内容
void swap1(int *p1,int *p2)
{
	int *temp;
	temp = p1;
	p1 = p2;
	p2 = temp;
}
//该函数交换指针所指的内容
void swap2(int *p1, int *p2)
{
	int temp;
	temp = *p1;
	*p1 = *p2;
	*p2 = temp;
}
//该函数交换指针本身的值，即交换指针所指的内存地址
void swap3(int* &p1, int* &p2)//表示该参数是一个引用，引用的是一个 int 型的指针
{
```

```
    int *temp;
    temp = p1;
    p1 = p2;
    p2 = temp;
}
int main()
{
    int a = 5, b = 10;
    int *p = &a, *q = &b;
    cout << "Before：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl;
    swap1(p, q);
    cout << "After：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl << endl;
    a = 5, b = 10;
    p = &a, q = &b;
    cout << "Before：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl;
    swap2(p, q);
    cout << "After：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl << endl;
    a = 5, b = 10;
    p = &a, q = &b;
    cout << "Before：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl;
    swap3(p, q);
    cout << "After：" << endl;
    cout << "p = " << p << " q = " << q << endl;
    cout << "*p = " << *p << " *q = " << *q << endl;
    return 0;
}
```

输出：

```
Before:
p = 010FFC5C q = 010FFC50
*p = 5 *q = 10
After:
p = 010FFC5C q = 010FFC50
*p = 5 *q = 10

Before:
p = 010FFC5C q = 010FFC50
*p = 5 *q = 10
After:
p = 010FFC5C q = 010FFC50
*p = 10 *q = 5

Before:
p = 010FFC5C q = 010FFC50
*p = 5 *q = 10
After:
p = 010FFC50 q = 010FFC5C
*p = 10 *q = 5
请按任意键继续. . .
```

练习 6.23

```
#include<iostream>
using namespace std;
void print1(const int*p)
{
	cout << *p << endl;
}
void print2(const int *p, const int size)//参数为整形指针和数组容量
{
	int i = 0;
	while (i != size) {
		cout << *p++ << " ";
		i++;
	}
	cout << endl;
}
void print3(const int *beg, const int *end)//参数为数组的首尾边界
{
	while (beg != end) {
		cout << *beg++ <<" ";
	}
	cout << endl;
}
int main()
{
	int i = 0, j[2] = { 1,2 };
	print1(&i);
	print2(j, sizeof(j)/sizeof(*j));
	print3(begin(j), end(j));
	return 0;
}
```

P199

练习 6.27

```
#include<iostream>
using namespace std;
int iSum(initializer_list<int> il)//参数的个数是未知的
{
	int sum = 0;
	for (auto ele : il) {
		sum += ele;
	}
	return sum;
}
int main()
{
	cout << "1 + 2 + 3 = " << iSum({ 1,2,3 }) << endl;
	cout << "1 + 2 + 3 + 4 + 5 = " << iSum({ 1,2,3,4,5 }) << endl;
	return 0;
}
```

P204

练习 6.32

```
#include<iostream>
using namespace std;
int &get(int* array, int index)//引用返回左值
{
	return array[index];
}
int main()
{
	int ia[10];
	for (int i = 0; i != 10; ++i) {
		get(ia, i) = i;//为返回类型是非常量引用的函数的结果赋值
	}
	for (const auto elem : ia) {
		cout << elem << " ";//输出 0,1,2,3,4,5,6,7,8,9
	}
	return 0;
}
```

练习 6.33

```
#include<iostream>
#include<vector>
using namespace std;
//递归输出 vector<int>
void print(const vector<int> iVec, unsigned index)
{
	unsigned sz = iVec.size();
	if (!iVec.empty() && index < sz) {
```

```
            cout << iVec[index] << endl;
            print(iVec, index + 1);
        }
}
int main()
{
        vector<int> vi = { 1,2,3,4,5 };
        print(vi, 0);
        return 0;
}
```

P213

练习 6.42

```
#include<iostream>
#include<string>
using namespace std;
string make_plural(size_t ctr, const string &word, const string &ending = "s")
{
        return (ctr > 1) ? word + ending : word;
}
int main()
{
        cout << "success 的单数形式是：" << make_plural(1, "success", "es") << endl;
        cout << "success 的复数形式是：" << make_plural(2, "success", "es") << endl;
        //使用默认实参调用函数
        cout << "failure 的单数形式是：" << make_plural(1, "failure") << endl;
        cout << "failure 的复数形式是：" << make_plural(2, "failure") << endl;
        return 0;
}
```

P224

练习 6.56

```
#include<iostream>
#include<vector>
using namespace std;
int f1(int v1, int v2)
{
        return v1 + v2;
}
int f2(int v1, int v2)
{
        return v1 - v2;
}
int f3(int v1, int v2)
{
```

```
    return v1 / v2;
}
int f4(int v1, int v2)
{
    return v1 * v2;
}
void compute(int a, int b, int(*p)(int, int))
{
    cout << p(a, b) << endl;
}
int main()
{
    int i = 10, j = 5;
    decltype(f1) *p1 = f1, *p2 = f2, *p3 = f3, *p4 = f4;
    vector<decltype(f1) *> vPtr = { p1,p2,p3,p4 };
    for (auto ptr : vPtr) {
        compute(i, j, ptr);
    }
    return 0;
}
```

第 7 章：类

P230

练习 7.1

第 8 章：IO 库

P281

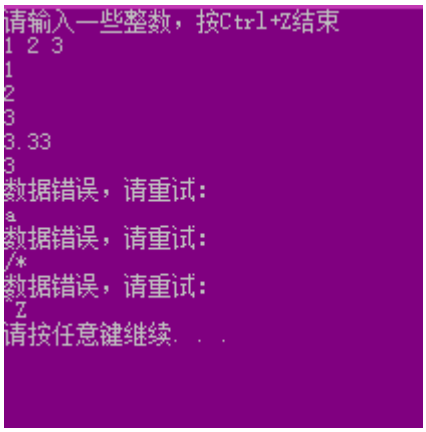
练习 8.1

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

istream& func(istream& in)
{
    int val;
    while (in >> val , !in.eof()) {
        if (in.bad())
            throw runtime_error("IO流错误");
        if (in.fail()) {
            cerr << "数据错误，请重试：" << endl;
            in.clear();
            in.ignore(100, '\n');
```

```
            continue;
        }
        cout << val << endl;
    }
    in.clear();//复位
    return in;
}
int main()
{
    cout << "请输入一些整数，按Ctrl+Z结束" << endl;
    func(cin);
    return 0;
}
```

练习 8.2

测试 8.1 函数如下：



练习 8.4

```
#include<iostream>
#include<fstream>
#include<string>
#include<vector>
using namespace std;
int main()
{
    //打开文件
    ifstream in("F:/origin.txt");
    if (!in) {
        cout << "error!can't open!" << endl;
        return -1;
    }
    string line;
    vector<string> words;
    //读文件，并将其内容存于words中
    while (getline(in,line) ) {//将每一行作为一个独立的元素
```

```
        words.push_back(line);
    }
    //关闭文件
    in.close();
    //将words中的内容打印
    //迭代器遍历
/*  auto iter = words.cbegin();
    while (iter != words.cend()) {
        cout << *iter << endl;
        ++iter;
    }
*/
    //范围for语句遍历
    for (auto elem : words) {
        cout << elem << endl;
    }
    return 0;
}
```

练习 8.5

将 8.4 的程序稍作修改：

```
while (in >> line ) {//将每一个单词作为一个独立的元素
        words.push_back(line);
    }
```

练习 8.6/8.7/8.8

练习 8.9（暂不懂）

```
#include<iostream>
#include<sstream>
#include<string>
#include<stdexcept>
using namespace std;

istream& func(istream& in)
{
    string val;
    while (in >> val, !in.eof()) {
        if (in.bad())
            throw runtime_error("IO流错误");
        if (in.fail()) {
            cerr << "数据错误，请重试：" << endl;
            in.clear();
            in.ignore(100, '\n');
            continue;
        }
        cout << val << endl;
```

```
    }
    in.clear();//复位
    return in;
}
int main()
{
    ostringstream msg;
    msg << "C++ Primer 第5版" << endl;
    istringstream in(msg.str());
    func(cin);
    return 0;
}
```

练习 8.10

```
#include<iostream>
#include<sstream>
#include<fstream>
#include<string>
#include<vector>
using namespace std;

int main()
{
    ifstream in("F:/origin.txt");//打开文件
    if (!in) {
        cerr << "error!can't open!" << endl;
        return -1;
    }
    string line;
    vector<string> words;
    while (getline(in, line)) {//将文件内容按行读取并存入words中
        words.push_back(line);
    }
    in.close();//别忘了关闭文件

    auto iter = words.cbegin();
    while (iter != words.cend()) {
        istringstream line_str(*iter);
        string word;
        while (line_str >> word) {//从line_str中读入到word
            cout << word << " ";
        }
        cout << endl;
        ++iter;
    }
```

```
    return 0;
}
```

练习 8.11

```
#include<iostream>
#include<sstream>
#include<string>
#include<vector>
using namespace std;

struct PersonInfo {
    string name;
    vector<string> phones;
};

int main()
{
    string line, word;
    vector<PersonInfo> people;
    istringstream record;
    while (getline(cin, line)) {
        PersonInfo info;
        record.clear();//重复使用字符流，每次都需要调用clear()
        record.str(line);//将record绑定到刚读入的行
        record >> info.name;
        while (record >> word) {
            info.phones.push_back(word);
        }
        people.push_back(info);
    }
    return 0;
}
```

练习 8.13

```
#include<iostream>
#include<fstream>
#include<sstream>
#include<string>
#include<vector>
using namespace std;

struct PersonInfo {
    string name;
    vector<string> phones;
};
```

```
bool valid(const string &s) {
    //暂时还写不出来，简化处理
    return true;
}

string format(const string &s) {
    //暂时还写不出来，简化处理
    return s;
}

int main()
{
    //打开文件
    ifstream in("F:/origin.txt");
    if (!in) {
        cerr << "error!" << endl;
        return -1;
    }
    string line, word;
    vector<PersonInfo> people;
    istringstream record;
    while (getline(in, line)) {
        PersonInfo info;
        record.clear();//重复使用字符流，每次都需要调用clear()
        record.str(line);//将record绑定到刚读入的行
        record >> info.name;
        while (record >> word) {
            info.phones.push_back(word);
        }
        people.push_back(info);
    }
    in.close();
    ostringstream os;
    for (const auto &entry : people) {
        ostringstream formatted, badNums;
        for (const auto &nums : entry.phones) {
            if (!valid(nums)) {
                badNums << " " << nums;//如果号码无效，则将数的字符串形式存到badNums
中
            }
            else {
                //将格式化字符串写入formatted中
                formatted << " " << format(nums);
```

```
            }
            if (badNums.str().empty()) {
                os << entry.name << " " << formatted.str() << endl;
            }
            else {
                cerr << "input error:  " << entry.name << " invald number(s) " <<
badNums.str() << endl;
            }
        }
    }
    cout << os.str() << endl;

    return 0;
}
```

第 9 章 顺序容器

P297

练习 9.4

```
#include<iostream>
#include<vector>
using namespace std;
bool findNum(vector<int>::iterator beg, vector<int>::iterator end, int num) {
    for (vector<int>::iterator it = beg; it != end; ++it) {
        if (*it == num)
            return true;
    }
    return false;
}
int main()
{
    int num = 12;
    vector<int> vi = { 0,1,2,3,4,5,6,7,8,9 };
    vector<int>::iterator beg = vi.begin(), end = vi.end();
    if (findNum(beg, end, num))
        cout << num << " has founded!" << endl;
    else
        cout << num << " has not founded!" << endl;
    return 0;
}
```

练习 9.5

```
#include<iostream>
#include<vector>
using namespace std;
//函数返回迭代器
vector<int>::iterator findNum(vector<int>::iterator beg, vector<int>::iterator end, int num) {
	for (vector<int>::iterator it = beg; it != end; ++it) {
		if (*it == num)
			return it;
	}
	return end;//查找不成功，返回尾后迭代器
}

int main()
{
	vector<int> vi = { 0,1,2,3,4,5,6,7,8,9 };
	vector<int>::iterator beg = vi.begin(), end = vi.end();
	cout << findNum(beg, end, 8)-beg << endl;//查找成功，返回该值（通过迭代器实现的
	cout << find(beg, end, 12)-beg << endl;//查找不成功，返回 vi 的元素个数
	return 0;
}
```

P302

练习 9.13

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

int main()
{
	vector<int> ivec = { 7,6,5,4,3,2,1 };
	list<int> ilist = { 1,2,3,4,5,6,7 };
	//容器类型不同，不能使用拷贝初始化；
	//vector<double> dvec1(ivec);
	//元素类型相容，可用范围初始化
	vector<double> dvec2(ivec.begin(), ivec.end());
	//同上
	//vector<double> dvec3(ilist);
	vector<double> dvec4(ilist.begin(), ilist.end());
	cout << dvec2[0] << endl;
	cout << dvec4[0] << endl;
	return 0;
}
```

P304

练习 9.14

```
#include<iostream>
#include<string>
#include<vector>
#include<list>
using namespace std;

int main()
{
	list<char*> slist = { "hello","world","!!!" };
	vector<string> svec;
	//容器类型不同，不可通过拷贝赋值
	//svec = slist;
	//元素类型相容，可采用范围赋值
	svec.assign(slist.begin(), slist.end());
	cout << svec.capacity() << " " << svec.size() << endl;
	cout << svec[0] << endl;
	return 0;
}
```

P305

练习 9.15

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

int main()
{
	vector<int> ivec = { 1,2,3,4,5 };
	vector<int> ivec1 = { 1,2,3,4,5 };
	vector<int> ivec2 = { 1,2,3,4 };
	vector<int> ivec3 = { 1,2,3,4,6 };
	vector<int> ivec4 = { 1,2,3,5,4 };
	cout << (ivec == ivec1) << endl;//输出 1
	cout << (ivec > ivec2) << endl;//输出 1
	cout << (ivec < ivec3) << endl;//输出 1
	cout << (ivec < ivec4) << endl;//输出 1
	return 0;
}
```

练习 9.16

```
#include<iostream>
#include<list>
#include<vector>
using namespace std;
bool l_v_equal(vector<int> &ivec, list<int> &ilist) {
```

```
	if (ivec.size() != ilist.size())
		return false;
	auto lb = ilist.cbegin();
	auto le = ilist.cend();
	auto vb = ivec.cbegin();
	for (; lb != le; ++lb, ++vb) {
		if (*lb != *vb)
			return false;
	}
	return true;
}

int main()
{
	vector<int> ivec = { 1,2,3,4,5 };
	list<int> ilist1 = { 1,2,3,4,5 };
	list<int> ilist2 = { 1,2,3,4 };
	list<int> ilist3 = { 1,2,3,4,6 };
	list<int> ilist4 = { 1,2,3,5,4 };
	cout << l_v_equal(ivec, ilist1) << endl;
	cout << l_v_equal(ivec, ilist2) << endl;
	cout << l_v_equal(ivec, ilist3) << endl;
	cout << l_v_equal(ivec, ilist4) << endl;
	return 0;
}
```

P309

练习 9.18

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

int main()
{
	deque<string> sdeq;
	string str;
	while (cin >> str) {
		sdeq.push_back(str);
	}
	auto beg = sdeq.cbegin(), end = sdeq.cend();
	while (beg != end) {
		cout << *beg++ << endl;
	}
	return 0;
```

```
}
```

练习 9.19

```
#include<iostream>
#include<list>
#include<string>
using namespace std;
int main()
{
	list<string> slist;
	string str;
	while (cin >> str) {
		slist.push_back(str);
	}
	auto beg = slist.cbegin(), end = slist.cend();
	while (beg != end) {
		cout << *beg++ << endl;
	}
	return 0;
}
```

练习 9.20

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

int main()
{
	list<int> ilist = { 0,1,2,3,4,5,6,7,8,9 };
	deque<int> ideq_odd, ideq_even;
	for (auto it = ilist.cbegin(); it != ilist.cend(); ++it) {
		if (*it % 2 == 0)
			ideq_even.push_back(*it);
		else
			ideq_odd.push_back(*it);
	}
	for (auto it = ideq_even.cbegin(); it != ideq_even.cend(); ++it) {
		cout << *it << " ";
	}
	cout << endl;
	for (auto it = ideq_odd.cbegin(); it != ideq_odd.cend(); ++it) {
		cout << *it << " ";
	}
	cout << endl;
	return 0;
```

```
}
```

练习 9.22

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> iv = { 1,1,2,1 };
	int some_val = 1, new_ele = 0;
	int orginal_size = iv.size();
	auto iter = iv.begin();
	while (iter != (iv.begin() + orginal_size / 2 + new_ele)) {
		if (*iter == some_val) {
			iter = iv.insert(iter, 2 * some_val);
			++new_ele;
			iter += 2;
		}
		else
			++iter;
	}
	for (iter = iv.begin(); iter != iv.end(); ++iter) {
		cout << *iter << " ";
	}
	cout << endl;
	return 0;
}
```

P311

练习 9.24

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> vi;
	cout << vi[0] << endl;
	cout << vi.front() << endl;
	cout << *(vi.begin()) << endl;
	cout << vi.at(0) << endl;
	return 0;
}
```

注：程序会异常终止

P312

练习 9.26

```
#include<iostream>
```

```
#include<vector>
#include<list>
using namespace std;
int main()
{
	int ia[] = { 0,1,1,2,3,5,8,13,21,55,89 };
	vector<int> ivec;
	list<int> ilist;
	//范围初始化
	ivec.assign(ia, ia + 11);
	ilist.assign(ia, ia + 11);
	auto iter1 = ivec.begin();
	auto iter2 = ilist.begin();
	while (iter1 != ivec.end()) {//删除 ivec 中的偶数
		if (*iter1 % 2 == 0)
			iter1 = ivec.erase(iter1);
		else
			++iter1;
	}
	while (iter2 != ilist.end()) {//删除 ilist 中的奇数
		if (*iter2 % 2 == 1)
			iter2 = ilist.erase(iter2);
		else
			++iter2;
	}
	for (auto beg = ivec.cbegin(); beg != ivec.end(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	for (auto beg = ilist.cbegin(); beg != ilist.end(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	return 0;
}
```

P314

练习 9.27

```
#include<iostream>
#include<forward_list>
using namespace std;
int main()
{
	forward_list<int> flist = { 0,1,2,3,4,5,6,7,8,9 };
	auto prev = flist.before_begin();//前驱结点
```

```
    auto curr = flist.begin();//当前结点
    while (curr != flist.end()) {
        if (*curr % 2 == 1) {
            curr = flist.erase_after(prev);//删除奇数
        }
        else
        {
            prev = curr;
            ++curr;
        }
    }
    for (auto beg = flist.cbegin(); beg != flist.cend(); ++beg) {
        cout << *beg << " ";
    }
    cout << endl;
    return 0;
}
```

练习 9.28

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

void GetNewStr(forward_list<string> &flist, const string &str1, const string &str2) {
    bool isInserted = false;
    auto prev = flist.before_begin();
    auto curr = flist.begin();
    while (curr != flist.end()) {
        if (*curr == str1) {
            curr = flist.insert_after(curr, str2);
            isInserted = true;
            break;
        }
        else {
            prev = curr;
            ++curr;
        }
    }
    if (!isInserted) {//未找到给定字符，插入链表末尾
        flist.insert_after(prev, str2);
    }
}

int main()
```

```
{
	forward_list<string> flist1 = { "C","C++","Python","Java","Ruby" };
	string s1 = "Python", s2 = "hello";
	GetNewStr(flist1, s1, s2);
	for (auto beg = flist1.cbegin(); beg != flist1.cend(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	forward_list<string> flist2 = { "C","C++","Python","Java","Ruby" };
	string s3 = "kkk", s4 = "hello";
	GetNewStr(flist2, s3, s4);
	for (auto beg = flist2.cbegin(); beg != flist2.cend(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	return 0;
}
```

P317

练习 9.31

```
//双向链表
#include<iostream>
#include<list>
using namespace std;
int main()
{
	list<int> ilist = { 0,1,2,3,4,5,6,7,8,9 };
	auto iter = ilist.begin();
	while (iter != ilist.end()) {
		if (*iter % 2 == 0) {//删除偶数
			iter = ilist.erase(iter);
		}
		else {
			iter = ilist.insert(iter, *iter);//复制奇数
			++iter;
			++iter;//注意：链表写成 iter += 2;是错的
		}
	}
	for (auto beg = ilist.cbegin(); beg != ilist.cend(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	return 0;
}
```

//单向链表

```
#include<iostream>
#include<forward_list>
using namespace std;
int main()
{
	forward_list<int> flist = { 0,1,2,3,4,5,6,7,8,9 };
	auto prev = flist.before_begin();
	auto curr = flist.begin();
	while (curr != flist.end()) {
		if (*curr % 2 == 0) {
			curr = flist.erase_after(prev);
		}
		else {
			curr = flist.insert_after(curr, *curr);
			prev = curr;
			++curr;
		}
	}
	for (auto beg = flist.cbegin(); beg != flist.cend(); ++beg) {
		cout << *beg << " ";
	}
	cout << endl;
	return 0;
}
```

练习 9.34

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	vector<int> vi = { 1,2,3,4,5,6,7,8,9 };
	auto beg = vi.begin();
	while (beg != vi.end()) {
		if (*beg % 2) {
			beg = vi.insert(beg, *beg);
			beg += 2;
		}
		else {
			++beg;
		}

	}
	for (auto beg = vi.cbegin(); beg != vi.cend(); ++beg) {
```

```
		cout << *beg << " ";
	}
	cout << endl;
	return 0;
}
```

P322

练习 9.41

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
int main()
{
	vector<char> vc = { 'a','b','c','d','e' };
	string s(vc.data(),vc.size());
	cout << s << endl;
	return 0;
}
```

练习 9.42

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string s;
	s.reserve(100);
	char c;
	while (cin >> c) {
		s.push_back(c);
	}
	cout << s << endl;
	return 0;
}
```

P324

练习 9.43

```
#include<iostream>
#include<string>
using namespace std;
void replaceStr(string &s, const string &oldVal, const string &newVal) {
	auto len = oldVal.size();
	if (len == 0) {//要查找的字符串为空，直接退出
		return;
	}
	auto iter = s.begin();
```

```
    while (iter <= s.end() - len) {
        auto iter1 = iter;
        auto iter2 = oldVal.begin();
        while (iter2 != oldVal.end() && *iter1 == *iter2) {
            ++iter1;
            ++iter2;
        }
        if (iter2 == oldVal.end()) {//在字符串 s 中找到了 oldVal
            iter = s.erase(iter, iter1);//返回指向被删元素之后的一个元素的迭代器
            if (newVal.size() != 0) {
                auto beg = newVal.begin(), end = newVal.end();
                iter = s.insert(iter, beg, end);//在 iter 指向的元素之前插入 beg~end 范围间的元素，
返回指向第一个新插入值的的迭代器
                iter += newVal.size();
            }
        }
        else {
            ++iter;
        }
    }
}

int main()
{
    string s = "tho thru hello tho";
    replaceStr(s, "tho", "though");
    cout << s << endl;
    return 0;
}
```

练习 9.45

```
#include<iostream>
#include<string>
using namespace std;
void replaceStr(string &s, const string &oldVal, const string &newVal) {
    auto len = oldVal.size();
    if (len == 0) {//要查找的字符串为空，直接退出
        return;
    }
    int index1 = 0;
    while (index1 <= s.size() - len) {
        int index1_temp = index1;
        int index2 = 0;
        while (index2 < oldVal.size() && s[index1_temp] == oldVal[index2]) {
            ++index1_temp;
```

```
                ++index2;
            }
            if (index2 == oldVal.size()) {//匹配成功
                s = s.replace(index1, oldVal.size(), newVal);
                index1 += oldVal.size();
            }
            else {
                ++index1;
            }
        }

}

int main()
{
    string s = "tho thru hello tho";
    replaceStr(s, "tho", "though");
    cout << s << endl;
    return 0;
}
```

上面这种做法虽然正确，但确实毫无意义的，因为用下标做法的话，压根不用自己去匹配字符串，而是直接用 find 函数。如下：

```
#include<iostream>
#include<string>
using namespace std;
void replaceStr(string &s, const string &oldVal, const string &newVal) {
    string::size_type p = 0;
    while ((p = s.find(oldVal, p)) != string::npos) {
        s.replace(p, oldVal.size(), newVal);
        p += newVal.size();
    }
}

int main()
{
    string s = "tho thru hello tho";
    replaceStr(s, "tho", "though");
    cout << s << endl;
    return 0;
}
```

练习 9.45

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

```
string combineName(string &name, const string &pre, const string &suffix) {
	auto iter = name.begin();
	name.insert(iter, pre.begin(), pre.end());
	name.append(" ");
	name.append(suffix);
	return name;
}
int main()
{
	string name = "Bill";
	cout << combineName(name, "Mr.", "Jr.") << endl;
	return 0;
}
```

练习 9.46

```
#include<iostream>
#include<string>
using namespace std;
string combineName(string &name, const string &pre, const string &suffix) {
	name.insert(0, pre);//插入前缀
	name.insert(name.size(), " ");
	name.insert(name.size(), suffix);//插入后缀
	return name;
}

int main()
{
	string name = "Bill";
	cout << combineName(name, "Mr.", "Jr.") << endl;
	return 0;
}
```

P327

练习 9.47

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

void find_char(const string &s, const string &str) {
    string::size_type pos = 0;
    while ((pos = s.find_first_of(str, pos)) != string::npos) {
        cout << "pos: " << pos << ",char:" << s[pos] << endl;
        ++pos;
    }
}
int main()
```

```
{
    string s = "ab2c3d7R4E6", number = "0123456789";
    string letter = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string::size_type pos = 0;
    cout << "查找s中的数字" << endl;
    find_char(s, number);//
    cout << "查找s中的字母" << endl;
    find_char(s, letter);//
    return 0;
}
```

练习 9.49

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

void find_longest_word(ifstream &in) {
    string s, longest_word;
    int maxLength = 0;
    while (in >> s) {
        if (s.find_first_of("bdfghjkpqty") != string::npos) {
            continue;
        }
        if (s.size() > maxLength) {
            maxLength = s.size();
            longest_word = s;
        }
    }
    cout << "满足要求的最长的单词是： " << longest_word << endl;
}

int main()
{
    ifstream in("F:/origin.txt");
    if (!in) {
        cerr << "error!can't open!" << endl;
        return -1;
    }
    find_longest_word(in);
    return 0;
}
```

P328

练习 9.50

计算整型值

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

int main()
{
    vector<string> vs = {"10","20","30","40"};
    int sum = 0;
    for (auto elem : vs) {
        sum += stoi(elem);
    }
    cout << sum << endl;
    return 0;
}
```

计算浮点值

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

int main()
{
    vector<string> vs = {"10.6","-9.6","+7.8e-2"};
    double sum = 0;
    for (auto elem : vs) {
        sum += stod(elem);
    }
    cout << sum << endl;
    return 0;
}
```

练习 9.51/9.52（暂不会）!!!

第 10 章：泛型算法

P337

练习 10.1

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> vi = { 1,2,2,2,3,4,5,5,5,5,6 };
    auto beg = vi.cbegin(), end = vi.cend();
    int num = 2;
```

```
    int cnt = count(beg, end, num);
    cout << num << " 出现了 " << cnt << " 次" << endl;
    return 0;
}
```

练习 10.2

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

int main()
{
    list<string> ls = { "aaa","bbb","ccc","ddd","ddd","ddd" };
    auto beg = ls.cbegin(), end = ls.cend();
    string s1 = "ddd", s2 = "aaa";
    int cnt1 = count(beg, end, s1);
    int cnt2 = count(beg, end, s2);
    cout << s1 << "出现了 " << cnt1 << " 次" << endl;
    cout << s2 << "出现了 " << cnt2 << " 次" << endl;
    return 0;
}
```

P339

练习 10.3

```
#include<iostream>
#include<numeric>
#include<vector>
using namespace std;
int main()
{
    vector<int> vi = { 1,2,3 };
    auto beg = vi.cbegin(), end = vi.cend();
    int sum = accumulate(beg, end, 0);
    cout << sum << endl;
    return 0;
}
```

练习 10.5

```
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;
int main()
{
    char *p[] = { "hello","world","!" };
    char *q[] = { _strdup(p[0]),_strdup(p[1]),_strdup(p[2]) };
```

```
    char *r[] = { p[0],p[1],p[2] };
    cout << "数组p[]的地址：" << &p[0] << endl;
    cout << "数组q[]的地址：" << &q[0] << endl;
    cout << "数组r[]的地址：" << &r[0] << endl;
    cout << equal(begin(p), end(p), q) << endl;//结果返回 0
    cout << equal(begin(p), end(p), r) << endl;//结果返回 1
    return 0;
}
```

？疑问？：既然 equal(begin(p), end(p), r)返回值为 1，那&r[0]和&p[0]的值怎么不相等呢？？

P342

练习 10.6

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
int main()
{
    vector<int> vi = {1,2,3,4,5,6,7,8,9};
    fill_n(vi.begin(), vi.size(), 0);
    fill(vi.begin(), vi.end(), 10);
    return 0;
}
```

P344

练习 10.9

```
#include<iostream>
#include<algorithm>
#include<vector>
#include<string>
using namespace std;
int main()
{
    vector<string> vs =
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    cout << "原始序列：";
    for (auto elem : vs) {
        cout << elem << " ";
    }
    cout << endl;
    sort(vs.begin(), vs.end());//字典序排序
    auto end_unique = unique(vs.begin(), vs.end());
    cout << "调用unique后：";
    for (auto elem : vs) {
        cout << elem << " ";
    }
```

```
    cout << endl;
    vs.erase(end_unique, vs.end());//删除重复元素
    cout << "删除重复元素后：";
    for (auto elem : vs) {
        cout << elem << " ";
    }
    cout << endl;
    return 0;
}
```

注！在调用 unique 之后，内存是这样的：

名称	值
[allocator]	allocator
[0]	"fox"
[1]	"jumps"
[2]	"over"
[3]	"quick"
[4]	"red"
[5]	"slow"
[6]	"the"
[7]	"turtle"
[8]	"the"
[9]	""
[原始视图]	{...}

自动窗口　监视 1

在调用 erase 后，变成了：

名称	值
vs	{ size=8 }
[capacity]	10
[allocator]	allocator
[0]	"fox"
[1]	"jumps"
[2]	"over"
[3]	"quick"
[4]	"red"
[5]	"slow"
[6]	"the"
[7]	"turtle"
[原始视图]	{...}

自动窗口　监视 1

P345

练习 10.11

```
#include<iostream>
#include<algorithm>
#include<vector>
#include<string>
using namespace std;
```

```
bool isShorter(const string& s1, const string& s2) {
    return s1.size() < s2.size();
}

//删除重复元素的函数
void elimDups(vector<string> &words) {
    sort(words.begin(), words.end());//字典序排序
    auto end_unique = unique(words.begin(), words.end());//将重复元素移至末尾
    words.erase(end_unique, words.end());//删除重复元素
}

int main()
{
    vector<string> words =
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    elimDups(words);
    stable_sort(words.begin(), words.end(), isShorter);
    for (auto elem : words) {
        cout << elem << " ";
    }
    cout << endl;
    return 0;
}
```

练习 10.13

```
#include<iostream>
#include<algorithm>
#include<vector>
#include<string>
using namespace std;

bool isShorter_5(const string& s) {
    return s.size() < 5;
}

int main()
{
    vector<string> words = { "eeeeeeee","ffffff","aaa","bbbb","ccccc","dddddd","ggg" };
    auto iter = partition(words.begin(), words.end(), isShorter_5);
    while (iter != words.end()) {
        cout << *iter << " ";
        ++iter;
    }
    cout << endl;
```

```
    return 0;
}
```

P349

练习 10.14

```
#include<iostream>
using namespace std;
int main()
{
    auto f = [](const int a, const int b) {return a + b; };
    cout << f(1,1) << endl;
    return 0;
}
```

练习 10.15

```
#include<iostream>
using namespace std;
int add(int a) {
    auto sum = [a](const int b) {return a + b; };//a为局部变量
    return sum(1);//求1 + a 的和
}
int main()
{
    cout << add(4) << endl;
    return 0;
}
```

练习 10.16（练习 lambda 表达式）

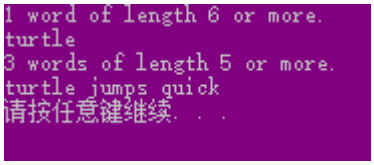
```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
using namespace std;

//删除重复的单词
void elimDups(vector<string> &words) {
    sort(words.begin(), words.end());//按字典序排序
    auto unique_end = unique(words.begin(), words.end());
    words.erase(unique_end, words.end());
}

//确定单词的单复数形式
string make_plural(size_t cnt, const string &word, const string &ending) {
    return (cnt > 1) ? word + ending : word;
}

//找出vector中长度超过sz的单词，并打印
```

```
void biggies(vector<string> &words, vector<string>::size_type sz) {
    elimDups(words);//删除重复元素
    //按长度排序，长度相同的维持字典序
    stable_sort(words.begin(), words.end(), [](const string &s1, const string &s2)
{return s1.size() < s2.size(); });
    //找到第一个满足长度要求的单词所对应的迭代器
    auto iter = find_if(words.begin(), words.end(), [sz](const string &word) {return
word.size() >= sz; });
    auto cnt = words.end() - iter;//满足要求的单词个数
    cout << cnt << " " << make_plural(cnt,"word","s") << " of length " << sz << " or
more." << endl;
    for_each(iter, words.end(), [](const string &s) {cout << s << " "; });
    cout << endl;
}

int main()
{
    vector<string> words=
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    biggies(words, 3);
    return 0;
}
```

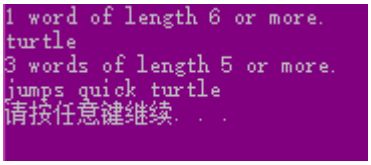
练习 10.18/10.19（比较 partition 和 stable_partition 函数的区别）

```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
using namespace std;

//删除重复的单词
void elimDups(vector<string> &words) {
    sort(words.begin(), words.end());//按字典序排序
    auto unique_end = unique(words.begin(), words.end());
    words.erase(unique_end, words.end());
}

//确定单词的单复数形式
string make_plural(size_t cnt, const string &word, const string &ending) {
    return (cnt > 1) ? word + ending : word;
}

//找出vector中长度超过sz的单词，并打印
void biggies(vector<string> &words, vector<string>::size_type sz) {
```

```
    elimDups(words);//删除重复元素
//注意与find_if函数的区别，调用find_if函数前必须保证words按长度先排好序；
//而partition则不需要先有序
    //partition写法
    auto iter = partition(words.begin(), words.end(), [sz](const string &word) {return
word.size() >= sz; });
    //stable_partition写法
//  auto iter = stable_partition(words.begin(), words.end(), [sz](const string &word)
{return word.size() >= sz; });
    auto cnt = iter - words.begin();//满足要求的单词个数
    cout << cnt << " " << make_plural(cnt,"word","s") << " of length " << sz << " or
more." << endl;
    for_each(words.begin(), iter, [](const string &s) {cout << s << " "; });
    cout << endl;
}

int main()
{
    vector<string> words=
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    biggies(words, 6);
    biggies(words, 5);
    return 0;
}
```



partition



stable_partition

p354

练习 10.20

```
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
using namespace std;
int isLonger_sz(vector<string> &words,vector<string>::size_type sz) {
    auto cnt = count_if(words.begin(), words.end(), [sz](const string &word) {return
word.size() > sz; });
    return cnt;
}
int main()
{
    vector<string> words=
```

```
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    cout << isLonger_sz(words, 5) << endl;//1
    cout << isLonger_sz(words, 3) << endl;//5
    return 0;
}
```

练习 10.21

```
#include<iostream>
#include<algorithm>
using namespace std;
void mutable_lambda() {
    int val = 5;
    auto f = [val]()mutable->bool {if (val > 0) { val--; return false; } else return
true; };
    for (int j = 0; j < 6; ++j) {
        cout << f() << " ";
    }
    cout << endl;
}
int main()
{
    mutable_lambda();//输出0 0 0 0 0 1
    return 0;
}
```

练习 10.22

```
#include<iostream>
#include<string>
#include<vector>
#include<functional>//bind函数
#include<algorithm>
using namespace std;
using namespace std::placeholders;

bool check_size(const string &s, string::size_type sz) {
    return s.size() >= sz;
}

string make_plural(size_t cnt, const string &word, const string &ending) {
    return (cnt > 1) ? word + ending : word;
}

void biggies(vector<string> &words, vector<string>::size_type sz) {
    auto cnt = count_if(words.begin(), words.end(), bind(check_size, _1, sz));
    cout << cnt << " " << make_plural(cnt, "word", "s") << " length of " << sz << " or
more." << endl;
```

```
}
int main()
{
    vector<string> words =
{ "the","quick","red","fox","jumps","over","the","slow","red","turtle" };
    biggies(words, 6);//输出  1 word length of 6 or more.
    biggies(words, 5);//输出  3 words length of 5 or more.
    return 0;
}
```

练习 10.24/10.25

P359

练习 10.27

```
#include<iostream>
#include<vector>
#include<list>
#include<iterator>
#include<algorithm>
using namespace std;

int main()
{
    vector<int> vi = { 1,2,2,3,3,3,3,4,5,6,7,8,9 };
    list<int> lst,lst2;
    unique_copy(vi.begin(), vi.end(), inserter(lst, lst.begin()));
    unique_copy(vi.begin(), vi.end(), back_inserter(lst2));
    for (auto elem : lst) {
        cout << elem << " ";//输出1 2 3 4 5 6 7 8 9
    }
    cout << endl;
    for (auto elem : lst2) {
        cout << elem << " ";//输出1 2 3 4 5 6 7 8 9
    }
    cout << endl;
    return 0;
}
```

练习 10.28

```
#include<iostream>
#include<vector>
#include<list>
#include<iterator>
#include<algorithm>
using namespace std;
```

```
int main()
{
    vector<int> vi = { 1,2,2,3,3,3,3,4,5,6,7,8,9 };
    list<int> lst, lst2, lst3;
    unique_copy(vi.begin(), vi.end(), inserter(lst, lst.begin()));
    for (auto elem : lst) {
        cout << elem << " ";//输出1 2 3 4 5 6 7 8 9
    }
    cout << endl;

    unique_copy(vi.begin(), vi.end(), back_inserter(lst2));
    for (auto elem : lst2) {
        cout << elem << " ";//输出1 2 3 4 5 6 7 8 9
    }
    cout << endl;

    unique_copy(vi.begin(), vi.end(), front_inserter(lst3));
    for (auto elem : lst3) {
        cout << elem << " ";//输出9 8 7 6 5 4 3 2 1
    }
    cout << endl;
    return 0;
}
```

P362

练习 10.29

```
#include<iostream>
#include<string>
#include<vector>
#include<iterator>
#include<fstream>
using namespace std;

int main()
{
    ifstream in("F:/data.txt");
    if (!in) {
        cerr << "ERROR!" << endl;
        exit(1);
    }
    vector<string> words;
    istream_iterator<string> in_iter(in), eof;
    while (in_iter != eof) {
        words.push_back(*in_iter++);
```

```
    }
    for (auto elem : words) {
        cout << elem << " ";
    }
    cout << endl;
    return 0;
}
```

练习 10.30/10.31

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<iterator>
using namespace std;

int main()
{
    istream_iterator<int> in_iter(cin), eof;
    ostream_iterator<int> out_iter(cout, " ");
    vector<int> vec(in_iter, eof);//标准输入初始化
    sort(vec.begin(), vec.end());
    copy(vec.begin(), vec.end(), out_iter);//标准输出
    cout << endl;
    unique_copy(vec.begin(), vec.end(), out_iter);//标准输出（输出序列不重复）
    cout << endl;
    return 0;
}
```

练习 10.33

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<iterator>
#include<fstream>
using namespace std;

int main()
{
    ifstream in("F:/data.txt");//打开输出文件
    if (!in) {
        cerr << "ERROR!Can't open origin file!" << endl;
        exit(1);
    }
    ofstream out_odd("F:/odd.txt");//打开输入文件，原先可以不存在，语句执行时自动创建
    if (!out_odd) {
        cerr << "ERROR!Can't open odd file!" << endl;
```

```
        exit(1);
    }
    ofstream out_even("F:/even.txt");//打开输入文件，原先可以不存在，语句执行时自动创建
    if (!out_even) {
        cerr << "ERROR!Can't open even file!" << endl;
        exit(1);
    }
    //创建迭代器
    istream_iterator<int> in_iter(in), eof;
    ostream_iterator<int> out_odd_iter(out_odd, " ");
    ostream_iterator<int> out_even_iter(out_even, "\n");
    while (in_iter != eof) {
        if (*in_iter % 2 == 0) {
            *out_even_iter = *in_iter;
            ++out_even_iter;
        }
        else {
            *out_odd_iter = *in_iter;
            ++out_odd_iter;
        }
        ++in_iter;
    }
    return 0;
}
```

P365

练习10.34/10.35

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

int main()
{
    vector<int> vec = { 0,1,2,3,4,5,6,7,8,9 };
    //使用反向迭代器逆向输出
    for (auto iter = vec.crbegin(); iter != vec.crend(); ++iter) {
        cout << *iter << " ";
    }
    cout << endl;
    //使用普通迭代器逆向输出
    for (auto iter = vec.cend(); iter != vec.cbegin();  ) {
        cout << *(--iter) << " ";
    }
    cout << endl;
    //正向输出
```

```
    for (auto iter = vec.cbegin(); iter != vec.cend(); ++iter) {
        cout << *iter << " ";
    }
    cout << endl;
    return 0;
}
```

练习10.36

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

int main()
{
    list<int> lst = { 0,1,2,3,4,5,6,7,8,9,0,10 };
    auto last_0 = find(lst.crbegin(), lst.crend(), 0);//用反向迭代器找到末尾的最后一个0
    //将迭代器向表头位置移一位，为了在将反向迭代器转换成普通迭代器时(即调用base函数时)
可以使其正好对应0
    ++last_0;//这一步很关键
    int pos = 1;
    //记录末尾最后一个0的位置
    for (auto iter = lst.begin(); iter != last_0.base(); ++iter) {
        ++pos;
    }
    if (pos >= lst.size()) {
        cout << "未找到0！" << endl;
    }
    else {
        cout << "最后一个0在第 " << pos << " 个位置" << endl;
    }
    return 0;
}
```

练习10.37

```
#include<iostream>
#include<list>
#include<vector>
#include<algorithm>
#include<iterator>
using namespace std;

int main()
{
    vector<int> vec = { 0,1,2,3,4,5,6,7,8,9 };
    list<int> lst1, lst2;
```

```
    auto iter1 = find(vec.crbegin(), vec.crend(), 3);//返回指向3的迭代器
    ++iter1;//这一步很重要，此时迭代器指向2
    auto iter2 = find(vec.crbegin(), vec.crend(), 7);//返回指向7的迭代器
    //将3-7之间的数拷贝至list中
    copy(iter2,iter1, inserter(lst1, lst1.begin()));//逆向拷贝
    for (auto elem : lst1) {
        cout << elem << " ";//输出7 6 5 4 3
    }
    cout << endl;

    //注意下面这种写法时错误的，iter1和iter2都是反向迭代器，不能正向使用
    //copy(iter1, iter2, inserter(lst2, lst2.begin()));//错误写法

    //要想输出3 4 5 6 7，应该这样写
    copy(iter1.base(), iter2.base(), inserter(lst2, lst2.begin()));//正向拷贝
    for (auto elem : lst2) {
        cout << elem << " ";//输出3 4 5 6 7
    }
    cout << endl;
    return 0;
}
```

P370

练习10.42

```
#include<iostream>
#include<list>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
//vector版删除重复单词
void elimDups1(vector<string> &words) {
    sort(words.begin(), words.end());
    auto unique_end = unique(words.begin(), words.end());
    words.erase(unique_end, words.end());
}
//list版删除单词，需要体会的是：链表类型的特定容器算法是以成员函数的形式定义的
//与通用版区别开来
void elimDups2(list<string> &words) {
    words.sort();
    words.unique();//unique内部会直接调用erase删除重复的单词
}

int main()
```

```
{
    vector<string> words1 = { "aaa","bbb","ccc","aaa","bbb","ccc" };
    list<string> words2 = { "aaa","bbb","ccc","aaa","bbb","ccc" };
    elimDups1(words1);
    for (auto elem : words1) {
        cout << elem << " ";
    }
    cout << endl;
    elimDups2(words2);
    for (auto elem : words2) {
        cout << elem << " ";
    }
    cout << endl;
    return 0;
}
```

第 11 章：关联容器

P376

练习 11.3

```
#include<iostream>
#include<string>
#include<map>
#include<set>
#include<algorithm>
using namespace std;
int main()
{
    map<string, size_t> word_count;
    string word;
    while (cin >> word) {
        ++word_count[word];
    }
    for (auto &elem : word_count) {
        cout << elem.first << " occurs " << elem.second << (elem.second > 1 ? "
times" : " time") << endl;
    }
    return 0;
}
```

练习 11.4

```
#include<iostream>
#include<string>
#include<map>
#include<set>
#include<algorithm>
```

```
using namespace std;

string &trans(string &s) {
    for (int p = 0; p < s.size(); p++) {
        if (s[p] >= 'A'&&s[p] <= 'Z') {
            s[p] -= ('A' - 'a');//将大写变成小写
        }
        else if (s[p] == '.' || s[p] == ',') {
            s.erase(p, 1);//删除从p位置开始的一个字符
        }
    }
    return s;
}
int main()
{
    map<string, size_t> word_count;
    string word;
    while (cin >> word) {
        ++word_count[trans(word)];
    }
    for (auto &elem : word_count) {
        cout << elem.first << " occurs " << elem.second << (elem.second > 1 ? "
times" : " time") << endl;
    }
    return 0;
}
```

P378

练习 11.7

```
#include<iostream>
#include<vector>
#include<string>
#include<map>
#include<set>
using namespace std;

//添加新的家庭
void add_family(map<string, vector<string>> &families, const string &family_name) {
    families[family_name] = vector<string>();
}
//添加孩子
void add_child(map<string, vector<string>> &families, const string &family_name, const
string &child) {
    families[family_name].push_back(child);
}
```

```
int main()
{
    map<string, vector<string> > families;
    add_family(families, "张");
    add_family(families, "李");
    add_child(families, "张", "一一");
    add_child(families, "张", "二二");
    add_child(families, "李", "白");
    for (auto family : families) {
        for (auto child : family.second) {
            cout << family.first << child << endl;
        }
    }
    return 0;
}
```

练习 11.8

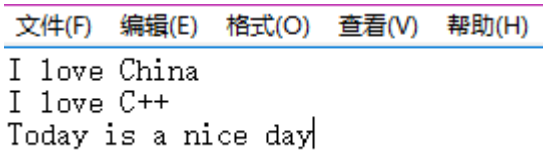
```
#include<iostream>
#include<vector>
#include<string>
#include<set>
#include<algorithm>
using namespace std;

int main()
{
    vector<string> vec_words;
    set<string> set_words;
    string word;
    while (cin >> word) {
        //在vector中存储不重复的单词，首先要判断当前存入的单词是否已经存在了，用find函数来判断
        if (find(vec_words.begin(), vec_words.end(), word) == vec_words.end()) {
            vec_words.push_back(word);
        }
        set_words.insert(word);
    }
    for (auto elem : vec_words) {
        cout << elem << " ";
    }
    cout << endl;
    for (auto elem : set_words) {
        cout << elem << " ";
    }
```

```
    cout << endl;
    return 0;
}
```

P379

练习 11.9

首先在 F 盘下有一份 data.txt 文件，存有数据如下：



执行代码：

```
#include<iostream>
#include<fstream>
#include<sstream>
#include<map>
#include<list>
#include<string>
#include<utility>
using namespace std;
int main()
{
    ifstream in("F:/data.txt");
    if (!in) {
        cerr << "ERROR!" << endl;
        exit(1);
    }
    string line, word;
    int lineNo = 0;//记录行号
    map<string, list<int>> wordLineNo;//存放单词对应的行号
    while (getline(in, line)) {
        lineNo++;
        istringstream in_line(line);
        while (in_line >> word) {
            wordLineNo[word].push_back(lineNo);
        }

    }
    for (const auto &elem1 : wordLineNo) {
        //cout << left <<setw(5) << elem1.first << "所在的行：";
        cout << elem1.first << "所在的行：";
        for (const auto &elem2 : elem1.second) {
            cout << elem2 << " ";
        }
```

```
        cout << endl;
    }
    return 0;
}
```

输出：

```
C++所在的行: 2
China所在的行: 1
I所在的行: 1 2
Today所在的行: 3
a所在的行: 3
day所在的行: 3
is所在的行: 3
love所在的行: 1 2
nice所在的行: 3
请按任意键继续. . .
```

文字不对齐，是不是很丑！？！？那就想办法对齐啊！！！

添加头文件<iomanip>，使用 setw 函数，cout 语句修改如下：

```
cout << left << setw(5) << elem1.first << "所在的行：";//即将 elem1.first 左对齐 5 个字符
```

输出变成如下效果：

```
C++  所在的行: 2
China所在的行: 1
I    所在的行: 1 2
Today所在的行: 3
a    所在的行: 3
day  所在的行: 3
is   所在的行: 3
love 所在的行: 1 2
nice 所在的行: 3
请按任意键继续. . .
```

是不是看着舒服多了：）

P381

练习 11.12/11.13

```
#include<iostream>
#include<string>
#include<utility>
#include<vector>
using namespace std;
int main()
{
    vector<pair<string, int>> data;
    string str;
    int val;
    while (cin >> str && cin >> val) {
        //三种创建pair的方法
        data.push_back({ str,val });
        //data.push_back(pair<string, int>(str, val));
        //data.push_back(make_pair(str, val));
    }
    for (auto elem : data) {
        cout << elem.first << " " << elem.second << endl;
```

```
    }
    return 0;
}
```

练习 11.14

```
#include<iostream>
#include<vector>
#include<string>
#include<map>
#include<set>
#include<utility>//pair
using namespace std;

//添加新的家庭
void add_family(map<string, vector<pair<string, string>>> &families, const string
&family_name) {
    families[family_name];
}
//添加孩子
void add_child(map<string, vector<pair<string, string>>> &families,
    const string &family_name, const string &child,const string &birthday) {
    families[family_name].push_back({child,birthday});
}

int main()
{
    map<string, vector<pair<string,string>>> families;
    add_family(families, "张");
    add_family(families, "李");
    add_child(families, "张", "三","1994-11-17");
    add_child(families, "张", "强","1998-10-05");
    add_family(families, "王");
    for (auto elem : families) {
        cout << elem.first << "家的孩子:  " << endl;
        for (auto child : elem.second) {
            cout << elem.first << child.first << "(生日:" << child.second << ")" <<
endl;
        }
    }
    return 0;
}
```

P386

练习 11.20

```
#include<iostream>
#include<map>
```

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

int main()
{
    map<string, size_t> word_count;
    string word;
    //下标版
/*  while (cin >> word) {
        ++word_count[word];
    }
*/
    //调用insert版
    while (cin >> word) {
        auto ret = word_count.insert({ word,1 });
        if (!ret.second) {
            ++ret.first->second;
        }
    }
    for (auto elem : word_count) {
        cout << elem.first << " occurs " << elem.second << " times." << endl;
    }
    return 0;
}
```

练习 11.23

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

void add_family(multimap<string, string> &families, const string &family, const string
&child) {
    families.insert({ family,child });
}

int main()
{
    multimap<string, string> famlies;
    add_family(famlies, "张", "三");
    add_family(famlies, "张", "全蛋");
    add_family(famlies, "李", "四");
    add_family(famlies, "王", "五");
    for (auto elem : famlies) {
        cout << elem.first << elem.second << endl;
```

```
    }
    return 0;
}
```

P391

练习 11.28

```
#include<iostream>
#include<map>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;

int main()
{
    map<string, vector<int>> mp;
    mp.insert({ "aaa",{1,2,3} });//列表初始化
    map<string, vector<int>>::iterator ret = mp.find("aaa");
    //find返回的类型就是 map<string, vector<int>>::iterator
    //auto ret = mp.find("aaa");
    for (auto elem : ret->second) {
        cout << elem << " ";//输出1 2 3
    }
    return 0;
}
```

练习 11.31/11.32

```
#include<iostream>
#include<map>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;

void printBooks(multimap<string, string> &bookList) {
    cout << "当前目录：" << endl;
    for (auto &book : bookList) {
        cout << book.first << ", 《" << book.second << "》" << endl;
    }
}
//删除某一作者的作品
void remove_author(multimap<string, string> &bookList, const string &author) {
    auto pos = bookList.equal_range(author);
    if (pos.first == pos.second) {
        cout << "没有" << author << "这个作者" << endl;
    }
```

```
    else {
        bookList.erase(pos.first, pos.second);
    }
}

int main()
{
    multimap<string, string> books;
    books.insert({ "wandao","Operating System" });
    books.insert({ "wandao","Networks" });
    books.insert({ "wandao","Data Structure" });
    books.insert({ "wandao","Computer Oganization and Architecture" });
    books.insert({ "Barth,John","Sot-Weed Factor" });
    books.insert({ "Barth,John","Lost in the funhouse" });
    books.insert({ "aaa","朝花夕拾" });
    books.insert({ "aaa","三味书屋" });

    //打印原始目录
    printBooks(books);
    //删除wandao的作品
    remove_author(books, "wandao");
    printBooks(books);
    remove_author(books, "李白");

/*
    //使用find和count函数
    auto iter = authors.find("wandao");//找到第一个指向wandao的迭代器
    auto cnt = authors.count("wandao");//确定wandao的个数
    while (cnt) {
        cout << iter->second << endl;
        iter++;
        cnt--;
    }
    //使用upper_bound函数和lower_bound函数
    auto beg = authors.lower_bound("鲁迅");
    auto end = authors.upper_bound("鲁迅");
    for (; beg != end; ++beg) {
        cout << beg->second << endl;
    }
    //使用equal_range函数
    auto pos = authors.equal_range("Barth,John");//返回迭代器pair
    for (; pos.first != pos.second; pos.first++) {
        cout << pos.first->second << endl;
    }
```

```
*/
    return 0;
}
```

P394

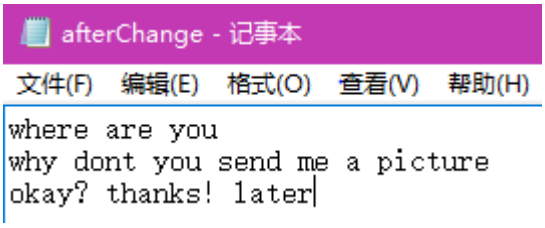
练习 11.33

```
#include<iostream>
#include<map>
#include<string>
#include<fstream>
#include<sstream>
#include<algorithm>
using namespace std;

//打开转换规则文件，产生映射map
map<string,string> buildMap(ifstream &in) {
    map<string, string> trans_map;
    string key;
    string value;
    while (in >> key && getline(in,value)){
        if (value.size() > 1) {
            trans_map[key] = value.substr(1);//除去空格号
        }
        else {
            throw runtime_error("No rule for " + key);
        }

    }
    return trans_map;
}
//单词转换函数
const string& transform(const string &s,const map<string,string> &mp) {
    auto it = mp.find(s);
    if (it != mp.cend()) {
        return it->second;//返回转换后的字符串
    }
    else {
        return s;
    }
}
//文本转换函数，最后写入out所指的文件
void word_transform(ifstream &in_rule, ifstream &in_beforeChange, ostream &out) {
    auto trans_map = buildMap(in_rule);//根据打开的规则文件创建映射map
    string line;
    while (getline(in_beforeChange, line)) {//从待修改的文件中读取一行
```

```
        string word;
        istringstream stream(line);//读入单个单词
        bool firstword = true;//用于控制是否输出格式
        while (stream >> word) {
            if (firstword)
                firstword = false;
            else
                out << " " ;
            out << transform(word, trans_map);
        }
        out << endl;
    }
}

int main()
{
    ifstream in_rule("F:/rule.txt");//rule文件存放转换规则
    if (!in_rule) {
        cerr << "error!" << endl;
        exit(1);
    }
    ifstream in_before("F:/beforeChange.txt");//打开待修改的文件
    if (!in_before) {
        cerr << "error!" << endl;
        exit(1);
    }
    ofstream out("F:/afterChange.txt");//afterChange文件存放改变后的单词
    if (!out) {
        cerr << "error!" << endl;
        exit(1);
    }
    word_transform(in_rule, in_before, out);
    return 0;
}
```

执行效果：

Before



After