1.1

#include<iostream>

int main(){

return 0;

}

1.2

#include<iostream>

int main(){

return -1;

}

1.3

#include<iostream>

int main(){

std::cout<<"Hello,world。" << std::endl;

system("pause");

}

1.4

#include<iostream>

int main(){

std::cout<<"Enter two numbers:" << std::endl;

int v1=0,v2=0;

std::cin>>v1>>v2;

std::cout<<"The product of "<<v1<<" and "<< v2<<" is "<<v1\*v2<< std::endl;

system("pause");

}

1.5

#include<iostream>

int main(){

std::cout<<"Enter two numbers:" << std::endl;

int v1=0,v2=0;

std::cin>>v1>>v2;

std::cout<<"The product of ";

std::cout<<v1;

std::cout<<" and ";

std::cout<<v2;

std::cout<<" is ";

std::cout<<v1\*v2<< std::endl;

system("pause");

}

1.6

不合法，因为后两行的输出运算符缺少ostream左侧运算对象。在后两行添加ostream对象即可修正

#include<iostream>

int main(){

std::cout<<"Enter two numbers:" << std::endl;

int v1=0,v2=0;

std::cin>>v1>>v2;

std::cout<<"The sum of "<<v1;

std::cout<<" and "<<v2;

std::cout<<" is "<<v1+v2<< std::endl;

system("pause");

}

1.7

#include<iostream>

int main(){

std::cout<<"Enter two numbers:" << std::endl;

/\*

\*对/\* \*/嵌套

\*导致匹配第一个“\*/”注释界定符

\*其余认为是源码

\*/

int v1=0,v2=0;

std::cin>>v1>>v2;

std::cout<<"The sum of "<<v1;

std::cout<<" and "<<v2;

std::cout<<" is "<<v1+v2<< std::endl;

system("pause");

}

1.8

#include<iostream>

int main(){

std::cout<<"/\*";

system("pause");

}

#include<iostream>

int main(){

std::cout<<"\*/";

system("pause");

}

#include<iostream>

int main(){

std::cout<</\*"\*/"\*/";

system("pause");

}

#include<iostream>

int main(){

std::cout<</\*"\*/"/\*"/\*"\*/;;

system("pause");

}

1.9

#include<iostream>

int main(){

int sum=0,val=50;

while(val<101){

sum+=val;

val++;

}

std::cout<<sum<<std::endl;

system("pause");

}

1.10

#include<iostream>

int main(){

int val=10;

while(val>=0){

std::cout<<val--<<std::endl;

}

system("pause");

}

1.11

#include<iostream>

int main(){

int n1,n2,max,min;

std::cin>>n1>>n2;

if(n1>n2){

max=n1,min=n2;

}

else{

max=n2,min=n1;

}

while(min<=max){

std::cout<<min++<<std::endl;

}

system("pause");

}

1.12

将-100到100间的所有数相加，结果为0

1.13

最简单的方法是while改为for并在判断语句前后加上；

略

1.14

for循环多出了初始化语句与表达式，能很直观的看出关键变量每次循环的变化

但缺点是对于不初始化和不适用表达式的语句则显得略微多余

1.15

略

1.16

#include<iostream>

int main(){

int sum=0,val;

while(std::cin>>val){

sum+=val;

}

std::cout<<sum<<std::endl;

system("pause");

}

1.17

若有n个输入值

所有值相等则只会有一行，表示输入值的次数为n

没有重复值则会有n行，每个输入值的次数为1

1.18

略

1.19

（本来就能。。。）略

1.20/21/22/23/24/25

（没网下头文件）略

2.3

32

-32取模后的数

32

-32

0

0

2.4

#include<iostream>

int main() {

unsigned u = 10, u2 = 42;

int i = 10, i2 = 42;

std::cout << i - u << std::endl;

std::cout << u - i << std::endl;

system("pause");

}

2.5

（a）字符 宽字符 字符串 宽字符串

（b）整型 无符号整型 长整型 无符号长整型 八进制12 十六进制c

（c）浮点型 单精度浮点型 双精度浮点型

（d）整型 无符号整型 不知道 浮点型

2.6

有，后者为八进制

2.7

字符串

long double

float

long double

2.8

#include<iostream>

int main() {

std::cout << "2\tM" << std::endl;

system("pause");

}

2.9

（b）i使用列表初始化导致未执行

（c）未先定义wage导致无法编译

2.10

global\_str 空

global\_int 0

local\_int 未赋值

local\_str 空

2.11

a 定义

b 定义

c 声明

2.12

a c d

2.13

100

2.14

100 45

2.15

b 引用的不是对象

d 引用必须初始化

2.16

a 合法，将3.14159赋值给了d

b合法，将0赋值给了d

c合法，丢失精度后0赋值给i

d合法，丢失精度后0赋值给i

2.17

10 10

2.18

略

2.19

引用只是将一个新的名字赋予对象，不会产生新的对象，所有对引用的操作都是对其初始化时绑定的对象的操作，并且初始化后其所绑定对象不可更改

指针本身是一个对象，存放特定类型对象的内存地址，只要对象的类型匹配，那么其内存地址都可以被赋值给指针

2.20

//定义了一个整型i，初始化为42

//定义整型指针p1，初始化为i的内存地址

//将i\*i赋值给i

2.21

a 指针类型与指向对象类型不匹配

b 初始化值应为地址值

c 无

2.22

if（p）当p为非空指针时为true

if（\*p）当p所指向对象为非0时成立

2.23

不能，有可能该指针指向的特定地址或其下一地址，这两个地址可能并没有存放对象

int a[2];

int \*p=a;

2.24

p为void所以存放任意类型地址，lp则进行了不匹配的类型地址赋值

2.25

a ip为整型指针，i为整型，r为对i的引用

b i为整型，ip为空指针

c ip为指针，ip2为整型

2.26

a 未初始化

d 对const对象sz赋值

2.27

a r非法，非常量引用必须为对象

c r非法，非常量引用必须为对象

d 若i2非常量则非法

e 若i2非常量则非法

f 引用未初始化

2.28

ic cp p2 p3 未初始化

2.29

b不合法因为指向非常量的指针不能接受指向常量的指针赋值

c不合法因为指向非常量的指针不能接受指向常量的指针赋值

d不合法因为常量指针不能被赋值

e不合法因为常量指针不能被赋值

f不合法因为常量不能被赋值

2.30

v2顶层

p2底层

p3顶层+底层

r2底层

2.31

p1=p2不合法 p1不是底层const而p2是，p2是常量不能转化为非常量

p2=p1合法 常量转换为非常量

p1=p3不合法 p1不是底层const而p3是，p3的对象是常量不能转化为非常量

p2=p3合法 顶层const不影响

2.32

int null=0,\*p=&null;

2.33

abc为整型变量 被赋值42

d e为指针，赋值非法

g为常量引用，对其赋值非法

2.34

#include<iostream>

int main() {

int i = 0, &r = i;

auto a = r;

const int ci = i, &cr = ci;

auto b = ci;

auto c = cr;

auto d = &i;

auto e = &ci;

auto &g = ci;

a = 42;

b = 42;

c= 42;

d = 42;

e = 42;

g = 42;

getchar();

}

2.35

j为整型，k为整型常量引用，p为指向整型常量的非常量指针

j2为整型常量，k2为整型常量引用

2.36

a int 4

b int 4

c int 4

d int& 4

2.37

a int 3

b int 4

c int e

d int& 3

2.38

int i=0;

auto j=i;

decltype(i)k=i;

int i=0;

auto j=i;

decltype((i))k=i;

2.39

略

2.40

struct Sales\_data {

std::string book\_No ;

int units\_sold;

double amount\_of\_money;

};

2.41

#include<string>

#include<iostream>

struct Sales\_data {

std::string book\_No ;

int units\_sold;

double amount\_of\_money;

};

int main() {

Sales\_data data1, data2;

double unit\_price = 0;

std::cin >> data1.book\_No >> data1.units\_sold >> unit\_price;

data1.amount\_of\_money = data1.units\_sold\*unit\_price;

std::cin >> data2.book\_No >> data2.units\_sold ;

data2.amount\_of\_money = data2.units\_sold\*unit\_price;

if (data1.book\_No == data2.book\_No) {

unsigned total\_sales = data1.units\_sold + data2.units\_sold;

double total\_AOM = data1.amount\_of\_money + data2.amount\_of\_money;

std::cout << data1.book\_No << ' ' << total\_sales << ' ' << total\_AOM;

if (total\_sales != 0) {

std::cout <<' '<<total\_AOM / total\_sales;

}

std::cout << std::endl;

}

else {

std::cerr << "Data must refer to the same ISBN" << std::endl;

}

getchar();

}

3.2

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s;

while (getline(cin, s))

cout << s << endl;

getchar();

}

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

char s;

while (cin>>s)

cout << s << endl;

getchar();

}

3.3

cin无视第一个非空白字符前的所有空白字符，并在开始输入后遇到第一个空白字符时停止输入，即输入中不存在空白符（空格换行制表）

getline将所有字符包括空白符输入，在遇到第一个换行符时停止输入，即输入中不存在换行符

3.4

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1,s2;

cin >> s1 >> s2;

if (s1 != s2) {

if (s1 > s2)cout << s1 << endl;

else cout << s2 << endl;

}

getchar();

}

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1,s2;

cin >> s1 >> s2;

if (s1.size()!= s2.size()) {

if (s1.size() >s2.size())cout << s1 << endl;

else cout << s2 << endl;

}

getchar();

}

3.5

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1,s2;

while (cin>>s1){

s2 = s2+s1;

}

cout << s2 << endl;

getchar();

}

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1,s2;

while (cin>>s1){

s2 = s2+s1 + ' ';

}

cout << s2 << endl;

getchar();

}

3.6

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1;

cin >> s1;

for (auto &c:s1){

c='X';

}

cout << s1 << endl;

getchar();

}

3.7

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1;

cin >> s1;

for (char &c:s1){

c='X';

}

cout << s1 << endl;

getchar();

}

3.8

#include<string>

#include<iostream>

using std::cin;

using std::cout;

using std::endl;

using std::string;

int main() {

string s1;

cin >> s1;

for (decltype(s1.size())n=0;n<s1.size();n++) {

s1[n]='X';

}

cout << s1 << endl;

getchar();

}

3.14

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

int num;

vector<int > vi;

while (cin >> num) {

vi.push\_back(num);

}

return 0;

}

3.15

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

string s;

vector<string> vi;

while (cin >> s) {

vi.push\_back(s);

}

return 0;

}

3.16

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int>v2(10);

for (auto num : v2) {

cout << num << endl;

}

return 0;

}

3.17

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <string> words;

string word;

while (cin >> word) {

words.push\_back(word);

}

for (string &word : words) {

for (char &alpha : word) {

if (islower(alpha)) alpha = alpha -32;

}

cout << word << endl;

}

return 0;

}

3．18

ivec.push\_back(42);

3.19

Vector<int>vec(10,42);

Vector<int>vec=(10,42);

Vector<int>vec{42,42,42,42,42,42,42,42,42,42,42};

3.20

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int> nums;

int num;

vector<int>::size\_type n=0;

while (cin >> num) {

nums.push\_back(num);

n++;

}

int add = 0;

int count = 0;

for (int num:nums) {

add += num;

count++;

if(count%2==0){

cout << add << endl;

add = 0;

}

}

return 0;

}

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int> nums;

int num;

vector<int>::size\_type n=0;

while (cin >> num) {

nums.push\_back(num);

n++;

}

for (int twin = 0; twin < n / 2; twin++) {

cout << nums[twin] + nums[n - 1 - twin] << endl;

}

return 0;

}

3.21

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int>v2{10,2,5,6,6,3,2};

for (auto it = v2.begin(); it != v2.end() ;it++) {

cout << \*it << endl;

}

return 0;

}

3.23

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int>v2(10,3);

for (auto it = v2.begin(); it != v2.end() ;it++) {

\*it \*= 2;

cout << \*it << endl;

}

return 0;

}

3.24

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int> nums;

int num;

while (cin >> num) {

nums.push\_back(num);

}

for (auto it = nums.begin(), eit = nums.end() - 1; it < (nums.begin() + nums.size() / 2); it = it + 1, eit = eit - 1) {

cout << (\*it + \*eit) << endl;

}

return 0;

}

3.25

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector <int> scores(11);

auto it = scores.begin();

int num;

while (cin >> num) {

it += num / 10;

(\*it)++;

it = scores.begin();

}

for (auto s : scores) {

cout << s<<' ';

}

return 0;

}

3.26

对于iterator没有/运算符

3.27

A buf\_size不是常量

B 返回值不是常量

D 没有空间存放空字符

3.29

大小固定

3.30

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

int a[10];

for (int i = 0; i <= 9; i++) {

a[i] = i;

}

return 0;

}

3.31

#include<string>

#include<iostream>

#include<vector>

using std::cin;

using std::cout;

using std::endl;

using std::string;

using std::vector;

int main() {

vector<int> a;

for (int i = 0; i <= 9; i++) {

a.push\_back(i);

}

return 0;

}

3.35

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int nums[10];

for (int \*p =begin(nums); p != end(nums); p++) {

\*p = 0;

}

for (int s : nums) {

cout << s << endl;

}

return 0;

}

3.40

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

char a1[]{ "i am a" }, a2[]{ "big head" }, a3[400];

strcpy(a3, a1);

strcat(a3, a2);

cout << a3;

return 0;

}

3.41

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int a[]{ 1,2,3,4 };

vector<int>b(begin(a), end(a));

for (int c : b) { cout << c; }

return 0;

}

3.42

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int a[4];

vector<int>b{ 1,2,3,4 };

int n = 0;

for (int c : b) { a[n++]=c; }

for (int d : a) {

cout << d << endl;

}

return 0;

}

3.43

//范围版

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int ia[3][4]{ 1,2,3,4,5,6,7,8,1,2,3,4 };

for (int(&p)[4] : ia) {

for (int q : p) {

cout << q;

}

cout << endl;

}

return 0;

}

//下标版

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int ia[3][4]{ 1,2,3,4,5,6,7,8,1,2,3,4 };

for (int p = 0; p < 3;p++){

for (int q = 0; q < 4;q++){

cout << ia[p][q];

}

cout << endl;

}

return 0;

}

//至臻版

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int ia[3][4]{ 1,2,3,4,5,6,7,8,1,2,3,4 };

for (int (\*p)[4] = begin(ia); p!=end(ia);p++){

for (int \*q =begin(\*p) ; q!=end(\*p);q++){

cout <<\*q;

}

cout << endl;

}

return 0;

}

3.44

//至臻版

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

using apple = int[4];

int main() {

int ia[3][4]{ 1,2,3,4,5,6,7,8,1,2,3,4 };

for (apple \*p= begin(ia); p!=end(ia);p++){

for (int \*q =begin(\*p) ; q!=end(\*p);q++){

cout <<\*q;

}

cout << endl;

}

return 0;

}

//范围版

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

typedef int apple[4];

int main() {

int ia[3][4]{ 1,2,3,4,5,6,7,8,1,2,3,4 };

for (apple &p: ia) {

for (int q : p) {

cout << q;

}

cout << endl;

}

return 0;

}

4.1

105

4.2

\*（vec.begin()）

(\*(vec.begin()))+1

4.4

(（（12/3）\*4）+（5\*15）)+((24%4)/2)=91

4.5

-86

-18

0

-2

4.6

N%2==0

4.7

赋值超出该变量最大表示范围

4.10

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<int> vec;

int i;

while (cin >> i) {

if (i == 42)break;

vec.push\_back(i);

}

return 0;

}

4.11

If (a>b&&b>c&&c>d)

4.12

先判断j与k的大小关系，再将其布尔值与i比较

4.13

I为3，d为3

D为3.5，i为3

4.14

42不能作为左值

42被赋值42

4.15

赋值的左右值类型不同

4.16

不管getPtr（）结果如何，都会后执行赋值作为判断条件

4.21

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<int> v{ 1,2,3,4,5,6,7,8,9, };

for (int &i : v) {

if (i % 2 != 0) { i \*= 2; }

cout << i << endl;

}

return 0;

}

4.22

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<int>scores{ 65,75,85,95 };

for (int score : scores) {

string grade = (score >= 90) ? "high pass" : (score >= 75) ? "pass" : (score > =60) ? "low pass" : "fail";

cout << grade << "\t";

}

return 0;

}

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<int>scores{ 65,75,85,95 };

for (int score : scores) {

string grade;

if (score >= 0) {

if (score >= 60) {

if (score >= 75) {

if (score >= 90) {

grade = "high pass";

}

else grade = "pass";

}

else grade = "low pass";

}

else grade = "fail";

}

cout << grade << "\t";

}

return 0;

}

4.23

string s = "word";

string pl = (s[s.size() - 1]=='s')? "" : "s";

4.25

11111111 11111111 11111111 10001110

11111111 11111111 11100011 10000000

4.26

如果环境定义的unsigned int 短于32，则先提升成 int

如果unsigned int是32位，则可正常执行

4.28

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

cout << sizeof(int) << ' ' << sizeof(unsigned int) << " " << sizeof(int\*) << ' ' << sizeof(int[3]) << ' ' << sizeof(string) << ' ' << sizeof(vector<int>) << ' ' << sizeof(vector<string>) << ' ' << sizeof(vector<vector<int>>);

return 0;

}

4.29

40/4 数组大小/首元素大小

4/4 指针大小/首元素大小

4.30

(sizeof x)+y

(sizeof a)<b

sizeof (p->mem[i])

sizeof (f())

4.32

定义并初始化了int常量size

列表初始化整型数组ia

指向ia的指针ptr，整型ix=0，ix与ptr每次循环均递增

执行循坏体直至ix=5或ptr指向队尾

4.33

表达式分为两部分

Somevalue？++x，++y？--x；

--y；

4.34

A fval转换为bool型

B ival转换为float，相加后再将结果转换为double

C cval转换成int，相乘后结果提升为double执行加法

4.35

A ‘a’转换为int，相加后转换为char

B ival转换为double，相乘后ui也转换为double

C ui转换为float，相乘后结果转换为double

D ival转换为float，相加结果转换为double，最后结果转换为char

4.36

i\*=static\_cast<int>(d)

4．37

pv = reinterpret\_cast<void\*>ps;

i = static\_cast<int>\*pc;

pv = reinterpret\_cast<void\*>&d;

pc = reinterpret\_cast<char\*>pv;

4.38

j/i的结果强制转换为double赋值给slope

5.1

没有任何语句仅有一个分号，填充语法上需要语句但实际不需要任何操作的地方

5.2

一对花括号及内部的语句

逻辑需要多个语句但语法仅需要一个语句，或者需要界定作用域

5.3

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

int sum = 0, val = 1;

while (val <= 10)

sum += val,++val;

cout << "Sum " << sum << endl;

return 0;

}

可读性降低

5.4

while (bool status == find(word)) { ; }

5.5

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<string> alpha{ "p","n","r","sr","ssr","ur" };

int score;

while (cin >> score) {

string grade{""};

if (score < 60)grade += alpha[0];

else {

grade += alpha[(score / 10) - 5];

if (score % 10 < 2)grade += "--";

else if (score % 10 < 5)grade += "-";

else if (score % 10 < 8)grade += "+";

else grade += "++";

}

cout << grade << endl;

}

return 0;

}

5.6

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

vector<string> alpha{ "p","n","r","sr","ssr","ur" };

int score;

while (cin >> score) {

string grade = score < 60 ? "p" : score < 70 ? "n" : score < 80 ? "r" : score < 90 ? "sr" : score < 100 ? "ssr" : "ur";

cout << grade << endl;

}

return 0;

}

5.7

A 第二行加；

B 第二行；改为，

C 第一行=改为==

D 第一行=改为==

5.8

Else与if有多种可能的匹配情况

与最近的一个if相匹配

5.9

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

char ch;

vector<int>v(5, 0);

while (cin >> ch) {

switch (ch) {

case'a':case'A':

v[0]++;

break;

case'e':case'E':

v[1]++;

break;

case'i':case'I':

v[2]++;

break;

case'o':case'O':

v[3]++;

break;

case'u':case'U':

v[4]++;

break;

}

}

cout << "a/A " << v[0] << endl;

cout << "e/E " << v[1] << endl;

cout << "i/I " << v[2] << endl;

cout << "o/O " << v[3] << endl;

cout << "u/U " << v[4] << endl;

return 0;

}

5.11

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

char ch;

vector<int>v(8, 0);

while (cin.get(ch)) {

switch (ch) {

case'a':case'A':

v[0]++;

break;

case'e':case'E':

v[1]++;

break;

case'i':case'I':

v[2]++;

break;

case'o':case'O':

v[3]++;

break;

case'u':case'U':

v[4]++;

break;

case'\t':

v[5]++;

break;

case'\n':

v[6]++;

break;

case' ':

v[7]++;

break;

}

}

cout << "a/A " << v[0] << endl;

cout << "e/E " << v[1] << endl;

cout << "i/I " << v[2] << endl;

cout << "o/O " << v[3] << endl;

cout << "u/U " << v[4] << endl;

cout << "换行 " << v[6] << endl;

cout << "空格 " << v[7] << endl;

cout << "制表 " << v[5] << endl;

return 0;

}

5.12

#include<string>

#include<iostream>

#include<vector>

#include<iterator>

using namespace std;

int main() {

char ch,prech='\0';

vector<int>v(11, 0);

while (cin.get(ch)) {

switch (ch) {

case'a':case'A':

v[0]++, prech = ch;

break;

case'e':case'E':

v[1]++, prech = ch;

break;

case'i':case'I':

v[2]++;

if (prech == 'f') {

v[10]++;

}

prech = ch;

break;

case'o':case'O':

v[3]++, prech = ch;

break;

case'u':case'U':

v[4]++, prech = ch;

break;

case'\t':

v[5]++, prech = ch;

break;

case'\n':

v[6]++, prech = ch;

break;

case' ':

v[7]++, prech = ch;

break;

case'f': {

if (prech == 'f') {

v[8]++, prech = '\0';

}

else prech = 'f';

}

break;

case'l': {

if (prech == 'f') {

v[9]++;

}

prech = ch;

break;

}

default:

prech = ch;

}

}

cout << "a/A " << v[0] << endl;

cout << "e/E " << v[1] << endl;

cout << "i/I " << v[2] << endl;

cout << "o/O " << v[3] << endl;

cout << "u/U " << v[4] << endl;

cout << "换行 " << v[6] << endl;

cout << "空格 " << v[7] << endl;

cout << "制表 " << v[5] << endl;

cout << "ff " << v[8] << endl;

cout << "fl " << v[9] << endl;

cout << "fi " << v[10] << endl;

return 0;

}

5.13

A 漏写break；

B default下未声明变量

C 格式应为case 1：case2：case 3:~~~

D ival等未声明为常量

5.14

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

vector<string>v;

vector<int> c;

string s, pres{ "\0" };

int t=0;

while (cin >> s) {

//若与上一词匹配

if (s == pres) {

t++;

int cc = 0;

bool flag = false;

//遍历输出容器查找是否已存在

for (string ss : v) {

if (s == ss) {

flag = true;

if (c[cc] < t)c[cc] = t;

}

cc++;

}

//不存在则存入

if (flag == false) {

v.push\_back(s), c.push\_back(t);

}

}

//若不匹配则重置

else {

pres = s;

t = 1;

}

}

for (int i = 0; i < v.size(); i++) {

cout << v[i] << " appears " << c[i] << " times" << endl;

}

if (v.size() == 0)cout << "There is no repeated words." << endl;

return 0;

}

5.15

A ix跳出循环时必然=sz，所以后面的if条件必不成立，多余

B ix未初始化且for语句头缺少初始化语句

C ix与sz同步递增，若statement内没有改变二者值则循环会不执行或持续执行

5.16

For循环与while循环可以转换，for循环的initializer与expression均可以放入其他部分，如果为了这两部分便于编写检测可以使用for，如果这两部分简单则适合使用while

5.17

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

vector<int>v1{ 1,1,1,22,2 };

vector<int>v2{ 1,1,1 };

vector<int>shortv;

vector<int>longv;

if (v1.size() < v2.size())shortv = v1, longv = v2;

else shortv = v2, longv = v1;

bool flag= true;

for (int i = 0; i < shortv.size(); i++) {

if (shortv[i] != longv[i]) {

flag = false;

break;

}

}

cout <<( flag?"true":"false") << endl;

return 0;

}

5.18

A 循环体无{}

B 在条件部分定义

5.19

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

string s1, s2, s;

cout << "please input two strings." << endl;

cin >> s1>>s2;

do {

if (s1.size() < s2.size())s = s1;

else s = s2;

cout << s << endl;

cout << "please input two strings." << endl;

} while (cin >> s1 >> s2);

return 0;

}

5.20

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

string word, preword{ "\0" };

bool flag{ true };

while (cin >> word) {

if (word == preword) {

cout << word << endl;

flag = false;

break;

}

preword = word;

}

if (flag == true)cout << "There is no repeated word" << endl;

return 0;

}

5.21

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

string word, preword{ "\0" };

bool flag{ true };

while (cin >> word) {

if (word == preword) {

if (isupper(word[0])) {

cout << word << endl;

flag = false;

break;

}

else {

preword = word;

continue;

}

}

preword = word;

}

if (flag == true)cout << "There is no repeated word" << endl;

return 0;

}

5.22

do{

int sz = get\_size();

}while(sz<=0)

5.23/24/25

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

int i1, i2;

while (cin >> i1 >> i2) {

if (i2 == 0) {

throw runtime\_error("Divisor must not be 0.");

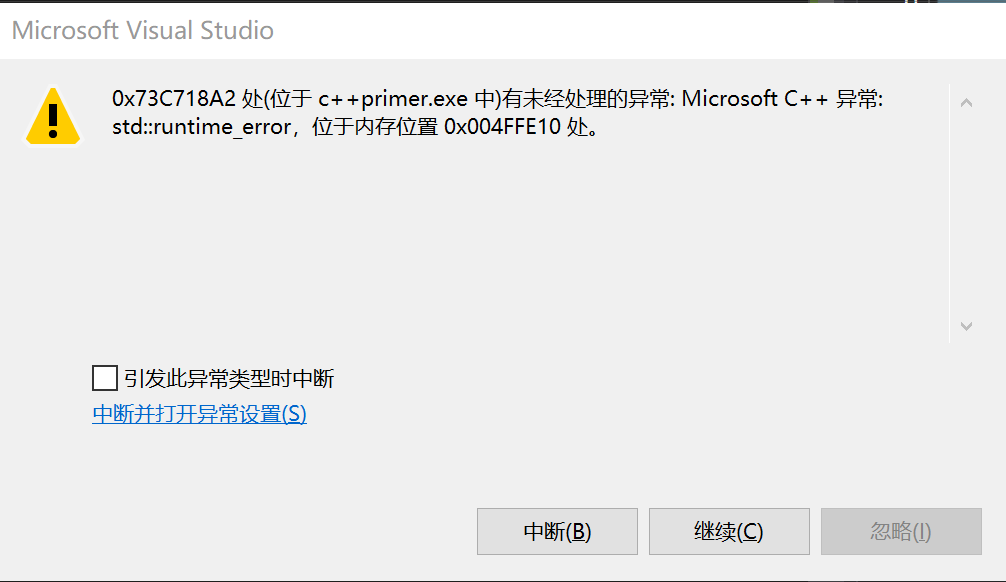
}

cout << i1 / i2 << endl;

}

return 0;

}



#include<string>

#include<iostream>

#include<vector>

using namespace std;

int main() {

int i1, i2;

while (cin >> i1 >> i2) {

try {

if (i2 == 0) {

throw runtime\_error("Divisor must not be 0.");

}

cout << i1 / i2 << endl;

}

catch (runtime\_error err) {

cout << err.what() << "\nTry again? Please enter y or n" << endl;

char a;

cin >> a;

if (a == 'y')continue;

else break;

}

}

return 0;

}

6.1

实参用于被调用函数的形参的初始化，实参必须可以被赋值给相应的形参

6.2

A 返回值类型未int实际返回string sting f（）

B 缺少返回值类型 void

C 形参名字重复 v2

D 函数体缺少花括号

6.3/6.4

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int fact(int x) {

int ret{ 1 };

for (; x > 0; x--) {

ret \*= x;

}

return ret;

}

int main() {

int i;

while (cin >> i) {

cout << fact(i) << endl;

}

return 0;

}

6.5

void o(int v) {

cout << v << endl;

}

6.6

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int o(int v) {

int t = v;

t++;

static int x=0;

x++;

return x\*t;

}

int main() {

int i=0;

for (int b = 1; b < 4; b++) {

cout << o(i) << endl;

}

return 0;

}

6.7

#include<string>

#include<iostream>

#include<vector>

using namespace std;

int plus1() {

static int x=-1;

x++;

return x;

}

int main() {

for (int b = 1; b < 4; b++) {

cout << plus1 ()<< endl;

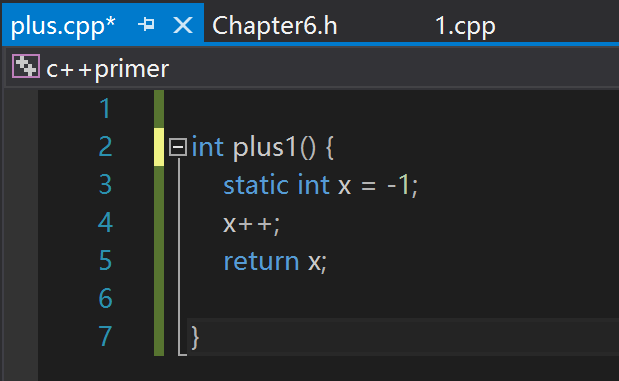
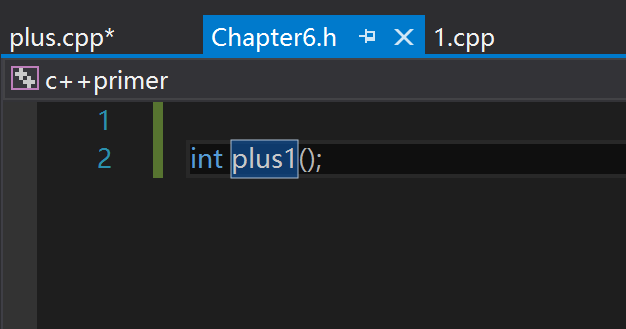
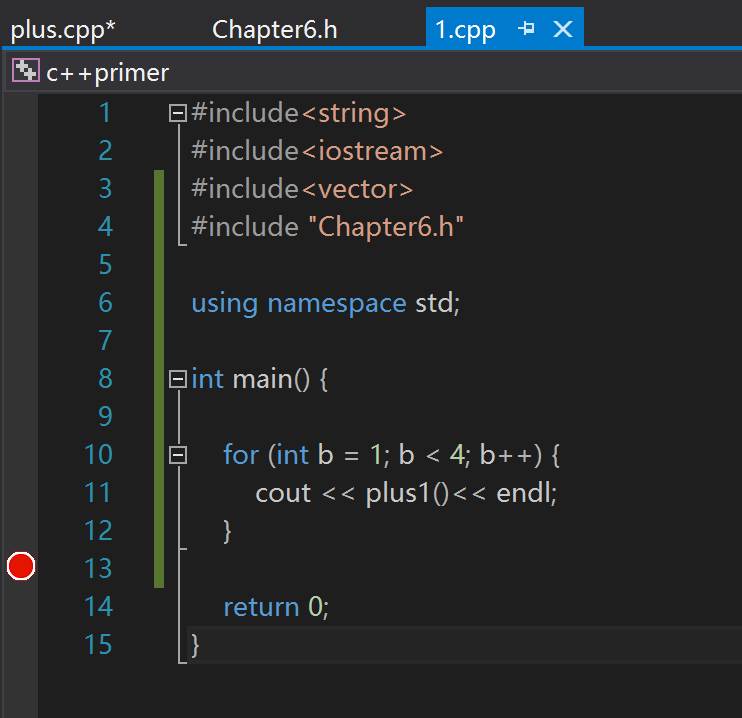
}

return 0;

}

6.8/6.9

Vs2015：



与c++primer略有不同，我使用的ide不需要在存放被调用函数的文件内添加#include“Chapter6.h“

一样可以完成编译

6.10

#include<string>

#include<iostream>

#include<vector>

#include "Chapter6.h"

using namespace std;

void exchange1(int\* x, int\* y) {

int tmp = \*y;

\*y = \*x;

\*x = tmp;

}

int main() {

int n = 1, m = 3;

exchange1(&n, &m);

cout << n << endl << m << endl;

return 0;

}

6.11

#include<string>

#include<iostream>

#include<vector>

using namespace std;

void reset(int& i) {

i = 0;

}

int main() {

int t = 9999;

reset(t);

cout << t << endl;

return 0;

}

6.12

#include<string>

#include<iostream>

#include<vector>

#include "Chapter6.h"

using namespace std;

void exchange1(int& x, int& y) {

int tmp = y;

y = x;

x = tmp;

}

int main() {

int n = 1, m = 3;

exchange1(n, m);

cout << n << endl << m << endl;

return 0;

}

引用略去了被调函数内部的解引用过程，较为便捷

6.13

前者为值传递，实参不会被改变

后者为引用传递，值有可能改变

6.14

需要函数改变实参值使用引用传递

实参为常量则要避免使用引用传递

6.15

避免被引用的同时被改变，所以s是常量引用；occurs为了记录次数则必须可以被修改且能被主函数读取

C不会被find\_char修改也不需要被主函数读取；

如果s是普通引用则有可能使原string被改变，如果occurs是常量则无法记录出现次数

6.16

不能判断const/字面值/需要类型转换的对象

bool is\_empty(const string& s) {

return s.empty();

}

6.17

bool is\_upper(const string& s) {

for (char c : s) {

if(isupper(c));

return true;

}

return false;

}

void to\_upper( string& s) {

for (char &c : s) {

if(islower(c));

c -=32;

}

}

不一致，后者必须修改string故不能使用const string&

6．18

bool compare(matrix& x, matrix& y);

比较两个matrix变量的大小

vector<int>::iterator change\_val(int x, vector<int>::iterator y);

将容器指针指向的元素改为参数值

6.19

A 参数过多

6.20

实参本身是常量或不想改变实参

可能导致传递出错或改变实参

6.21

int compare(int x,int \*y) {

if (x > \*y) {

return x;

}

else return \*y;

}

6.22

void exchange(int\* &x,int\* &y) {

int \*tmp = x;

x = y;

y = tmp;

}

6.23

void print(const int\* x, size\_t size) {

for (; size > 0; size--) {

cout << \*(x++) << endl;

}

}

void print(const int\*x,const int\* y) {

while (x != y) {

cout << \*(x++) << endl;

}

}

void print(const int\*x) {

while (\*x&& x) {

cout << \*x << endl;

}

}

6.24

输出从ia[0]到ia[9]的元素

如果ia实际的元素个数不足10则会出错

6.25

6.26

#include <stdio.h>

#include<iostream>

int main(int argc, char \*\* argv)

{

for (int i = 0; i < argc;i++)

std::cout << argv[0] << std::endl;

return 0;

}

6.25/6.26

#include<iostream>

#include<string>

using namespace std;

int main(int argc, char \*\* argv)

{

for (int i=0; i < argc;i++)

cout << argv[i]<< endl;

return 0;

}

6.27

#include<iostream>

#include<string>

using namespace std;

int ad(initializer\_list<int> il) {

int ret = 0;

for (int i : il) {

ret += i;

}

return ret;

}

int main(int argc, char \*\* argv,int in)

{

cout << ad({ 1, 3, 4, 5, 6 }) << endl;

return 0;

}

6.28

const string&

6.29

需要，形参传递的是字面值不是变量

6.30

编译失败，函数必须返回值

6.31

引用所绑定的对象在函数结束后失效

引用绑定的常量被函数结束后失效或被用于左值赋值

6.32

合法 将下标赋值给数组arry中该下标对应的元素

6.33

#include<iostream>

#include<string>

#include<vector>

using namespace std;

void putsvector (vector<int>::iterator beg, vector<int>::iterator end ) {

if (beg != end) {

cout << \*beg << endl;

putsvector(++beg, end);

}

}

int main(int argc, char \*\* argv,int in)

{

vector<int> vi{ 1,2,3,4,5,6,7 };

putsvector(vi.begin(), vi.end());

return 0;

}

6.34

输入大于0的数正常输出阶乘

输入小于0的数陷入无限递归

6.35

如果是val—则会先执行factorial（val）后执行val-=1，导致实际以同样值调用陷入无限递归

6.36/6.37/6.38

尝试编写返回数组引用的函数，报错：不能返回数组

编译器会将引用看作是数组，试了很多这题实在莫得办法

如果一定要返回数组，我会使用指向数组的指针

6.39

B 非法，两个函数仅返回值不同且名字相同

6.40

B错误 在有默认值的形参后仍有没有默认值的形参

6.41

A非法，ht没有默认实参也没有提供实参

C中的wd被赋予42

6.42

string make\_plural(size\_t ctr, const string &word, const string ending = { 's' }) {

return (ctr > 1) ? word + ending : word;

}

6.43

A定义放在头文件

B声明放在头文件

6.44

inline bool isShorter(const string &s1, const string &s2) {

return s1.size() < s2.size();

}

6.46

不能 该函数功能必须对两个形参进行比较，必须含有除常量返回值的语句

哪怕形如下列语句依然不被认为是常量

//错误，返回值不是常量

const bool t{ true };

const bool f{ false };

constexpr bool isShorter(const string &s1, const string &s2) {

return (s1.size() < s2.size())?t:f;

}

6.47

#include<iostream>

#include<string>

#include<vector>

using namespace std;

void putsvector(vector<int>::iterator beg, vector<int>::iterator end, vector<int> vi) {

if (beg != end) {

cout << \*beg <<sizeof(vi)<< endl;

putsvector(++beg, end, vi);

}

}

int main(int argc, char \*\* argv, int in)

{

vector<int> vi{ 1,2,3,4,5,6,7 };

putsvector(vi.begin(), vi.end(),vi);

return 0;

}

6.48

不合理，只要输入结束才会执行assert命令，则永远会终止程序

6.49

候选函数：重载函数集内的同名函数

可行函数：能被调用提供的实参正确调用的函数

6.50

A 二义性

B void f（）

C void f(int ,int)

D void f(double,double)

6.51

#include<iostream>

#include<string>

#include<vector>

#include<cassert>

using namespace std;

int f(int) {

return 0;

}

int f(double, double = 3.14) {

return 1;

}

int main(int argc, char \*\* argv, int in)

{

cout << f(42)<< endl;

return 0;

}

6.52

A 类型提升

B 强制类型转换

6.53

ab 增加一个以常量指针或引用为形参的函数，如果实参是常量优先匹配常量函数，如果实参是非常量优先匹配非常量参数

c 如果实参是指向非常量的常量指针会优先匹配下方函数（匹配顶层const）

6.54

typedef int(f)(int, int);

vector<f> v;

6.55

typedef int(\*f)(int, int);

vector<f> v{plus1,dif,x,chu };

int plus1(int x, int y) {

return x + y;

}

int dif(int x, int y) {

return x - y;

}

int x(int x, int y) {

return x\*y;

}

int chu(int x, int y) {

return x / y;

}

6.56

#include<iostream>

#include<string>

#include<vector>

#include<cassert>

using namespace std;

typedef int(\*f)(int, int);

int plus1(int x, int y) {

return x + y;

}

int dif(int x, int y) {

return x - y;

}

int x(int x, int y) {

return x\*y;

}

int chu(int x, int y) {

return x / y;

}

vector<f> v{ plus1,dif,x,chu };

int main(int argc, char \*\* argv, int in)

{

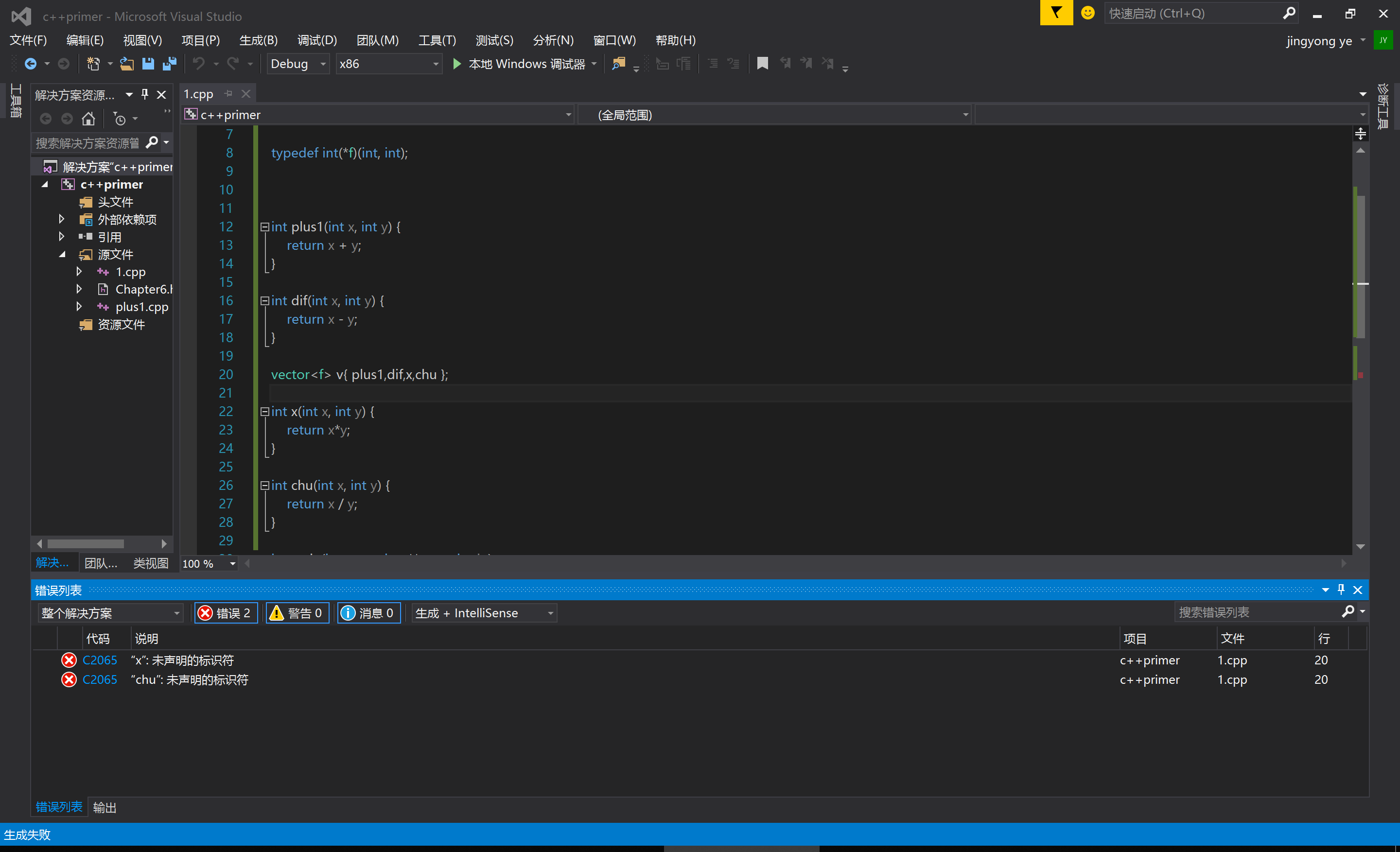
for (auto i : v) {

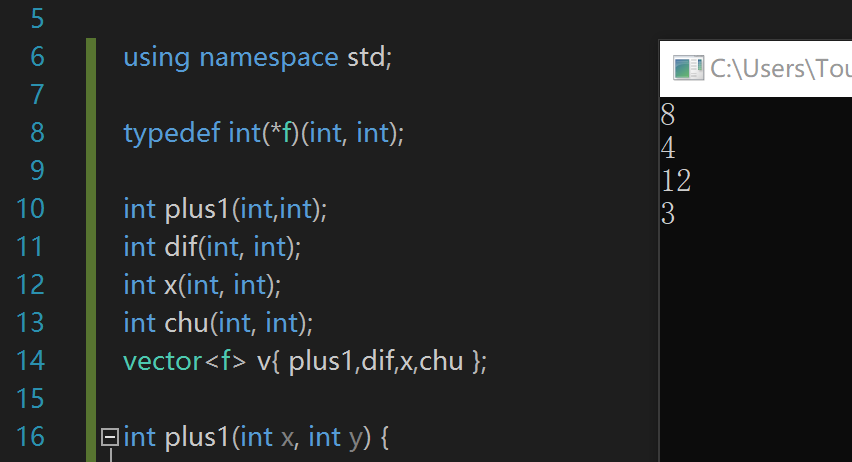
cout << i(6, 2) << endl;

}

return 0;

}





可以得出如果缺少函数声明/定义或者声明省略了形参会导致不能识别为该类型函数

7.1

#include<iostream>

#include<string>

#include<vector>

#include<cassert>

using namespace std;

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

};

int main() {

Sales\_data pre;

if (cin >> pre.bookNo >> pre.units\_sold >> pre.revenue) {

Sales\_data tmp;

while (cin >> tmp.bookNo >> tmp.units\_sold >> tmp.revenue) {

if (tmp.bookNo == pre.bookNo) {

pre.units\_sold += tmp.units\_sold; pre.revenue += tmp.revenue;

}

else {

cout << pre.bookNo <<" "<<pre.units\_sold << " " << pre.revenue << endl;

pre.units\_sold = tmp.units\_sold; pre.revenue = tmp.revenue; pre.bookNo = tmp.bookNo;

}

}

cout << pre.bookNo << " " << pre.units\_sold << " " << pre.revenue << endl;

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.2

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

string isbn() {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

};

7.3

int main() {

Sales\_data pre;

if (cin >> pre.bookNo >> pre.units\_sold >> pre.revenue) {

Sales\_data tmp;

while (cin >> tmp.bookNo >> tmp.units\_sold >> tmp.revenue) {

if (tmp.isbn() == pre.isbn()) {

pre.combine(tmp);

}

else {

cout << pre.bookNo <<" "<<pre.units\_sold << " " << pre.revenue << endl;

pre.units\_sold = tmp.units\_sold; pre.revenue = tmp.revenue; pre.bookNo = tmp.bookNo;

}

}

cout << pre.bookNo << " " << pre.units\_sold << " " << pre.revenue << endl;

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.4/7.5

struct Person {

string names;

string addresss;

string name() {

return names;

}

string address() {

return addresss;

}

};

7.6

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

7.7

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

Sales\_data pre;

if (read(cin,pre)) {

Sales\_data tmp;

while (read(cin,tmp) ){

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.8

Read本身需要对Sales\_data对象写操作，如果使用const则无法写入

Print不需要修改Sales\_data对象，使用const可以减少出错

7.9

#include<iostream>

#include<string>

#include<vector>

#include<cassert>

using namespace std;

struct Person {

string names;

string addresss;

string name() {

return names;

}

string address() {

return addresss;

}

};

istream &read(istream& a, Person &b) {

a >> b.names >> b.addresss;

return a;

}

ostream& print(ostream &a, const Person &b) {

a << b.names << b.addresss;

return a;

}

int main() {

return 0;

}

7.10

先将输入写入data1再写入data2，写入成功则执行下面的语句

7.11

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

Sales\_data ()= default;

Sales\_data(const string &a):bookNo(a){}

Sales\_data(unsigned a):units\_sold(a){}

Sales\_data(double a,unsigned b):units\_sold(a),revenue(a\*b){}

Sales\_data(const string&a,unsigned b,double c):bookNo(a),units\_sold(b),revenue(b\*c){}

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream & a);

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

Sales\_data::Sales\_data(istream& a) {

read(a, \*this);

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

Sales\_data s2("wt");

Sales\_data s3(12);

Sales\_data s4(12.0,3);

Sales\_data s5("魔法少女郭德纲",998,12);

Sales\_data pre;

Sales\_data gg(cin);

print(cout,gg);

if (read(cin,pre)) {

Sales\_data tmp;

while (read(cin,tmp) ){

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

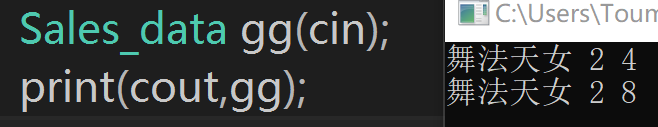
cerr << "?" << endl;

return -1;

}

return 0;

}



7.12

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

Sales\_data ()= default;

Sales\_data(const string &a):bookNo(a){}

Sales\_data(unsigned a):units\_sold(a){}

Sales\_data(double a,unsigned b):units\_sold(a),revenue(a\*b){}

Sales\_data(const string&a,unsigned b,double c):bookNo(a),units\_sold(b),revenue(b\*c){}

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

Sales\_data s2("wt");

Sales\_data s3(12);

Sales\_data s4(12.0,3);

Sales\_data s5("魔法少女郭德纲",998,12);

Sales\_data pre;

Sales\_data gg(cin);

print(cout,gg);

if (read(cin,pre)) {

Sales\_data tmp;

while (read(cin,tmp) ){

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.13

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

struct Sales\_data {

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

Sales\_data ()= default;

Sales\_data(const string &a):bookNo(a){}

Sales\_data(unsigned a):units\_sold(a){}

Sales\_data(double a,unsigned b):units\_sold(a),revenue(a\*b){}

Sales\_data(const string&a,unsigned b,double c):bookNo(a),units\_sold(b),revenue(b\*c){}

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

if (cin) {

Sales\_data pre(cin);

while (cin){

Sales\_data tmp(cin);

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.14

Sales\_data(const string&a = 0, unsigned b = 0 , double c=0) :bookNo(a), units\_sold(b), revenue(b\*c) {}

7.15

#include<iostream>

#include<string>

using namespace std;

struct Person;

istream &read(istream& a, Person &b);

ostream& print(ostream &a, const Person &b);

struct Person {

string names;

string addresss;

string name() {

return names;

}

string address() {

return addresss;

}

Person(const string&nam,const string& addr):names(nam),addresss(addr){}

Person() = default;

Person(const string&nam) :names(nam) {};

Person(istream &is) {

read(is, \*this);

}

};

istream &read(istream& a, Person &b) {

a >> b.names >> b.addresss;

return a;

}

ostream& print(ostream &a, const Person &b) {

a << b.names <<" "<< b.addresss;

return a;

}

int main() {

Person ps(cin);

print(cout, ps);

return 0;

}

7.16

没有，允许外部访问，仅允许内部成员访问

7.17

有，默认访问权限

7.18

仅提供端口隐藏实现细节

7．19

Private：names addresss，具体数据隐藏，只提供访问端口

7.20

需要允许非成员访问private成员

7.21

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

class Sales\_data {

friend istream &read(istream& a, Sales\_data &b);

friend ostream &print(ostream &a, const Sales\_data &b);

friend Sales\_data add(const Sales\_data &a, const Sales\_data &b);

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

public:

Sales\_data ()= default;

Sales\_data(const string &a):bookNo(a){}

Sales\_data(unsigned a):units\_sold(a){}

Sales\_data(double a,unsigned b):units\_sold(a),revenue(a\*b){}

Sales\_data(const string&a , unsigned b , double c) :bookNo(a), units\_sold(b), revenue(b\*c) {}

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

if (cin) {

Sales\_data pre(cin);

while (cin){

Sales\_data tmp(cin);

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.22

#include<iostream>

#include<string>

using namespace std;

struct Person;

istream &read(istream& a, Person &b);

ostream& print(ostream &a, const Person &b);

class Person {

friend istream &read(istream& a, Person &b);

friend ostream& print(ostream &a, const Person &b);

string names;

string addresss;

public:

string name() {

return names;

}

string address() {

return addresss;

}

Person(const string&nam,const string& addr):names(nam),addresss(addr){}

Person() = default;

Person(const string&nam) :names(nam) {};

Person(istream &is) {

read(is, \*this);

}

};

istream &read(istream& a, Person &b) {

a >> b.names >> b.addresss;

return a;

}

ostream& print(ostream &a, const Person &b) {

a << b.names <<" "<< b.addresss;

return a;

}

int main() {

Person ps(cin);

print(cout, ps);

return 0;

}

7.23/7.24

#include<string>

#include<iostream>

using namespace std;

struct Screen {

using sst = string::size\_type;

Screen() = default;

Screen(const sst&wid, const sst&heig, int num):width(wid),height(heig),contents(num,' ') {};

Screen(const sst&wid, const sst&heig, char ch) :width(wid), height(heig), contents({ ch }) {};

inline Screen& move(sst r, sst c) {

sst row = r\*width;

pos = row + c;

return \*this;

}

inline char get(sst r, sst c)const {

sst row =r\* width;

return contents[row + c];

}

inline char get()const{

return contents[pos];

}

void some\_memeber()const {

++ctr;

}

private:

sst pos = 0;

sst width = 0, height = 0;

string contents;

mutable size\_t ctr=0;

};

7.26

#include<iostream>

#include<string>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

class Sales\_data {

friend istream &read(istream& a, Sales\_data &b);

friend ostream &print(ostream &a, const Sales\_data &b);

friend Sales\_data add(const Sales\_data &a, const Sales\_data &b);

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

public:

Sales\_data ()= default;

Sales\_data(const string &a):bookNo(a){}

Sales\_data(unsigned a):units\_sold(a){}

Sales\_data(double a,unsigned b):units\_sold(a),revenue(a\*b){}

Sales\_data(const string&a , unsigned b , double c) :bookNo(a), units\_sold(b), revenue(b\*c) {}

string isbn () const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

inline double avg\_price() {

return revenue / units\_sold;

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a,const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

if (cin) {

Sales\_data pre(cin);

cout << pre.avg\_price() << endl;

while (cin){

Sales\_data tmp(cin);

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout,pre);

pre = tmp;

}

}

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

7.27

#include<string>

#include<iostream>

using namespace std;

struct Screen {

using sst = string::size\_type;

Screen() = default;

Screen(const sst&wid, const sst&heig):width(wid),height(heig),contents(wid\*heig,' ') {};

Screen(const sst&wid, const sst&heig, char ch) :width(wid), height(heig), contents(wid\*heig,ch) {};

inline Screen& move(sst r, sst c) {

sst row = r\*width;

pos = row + c;

return \*this;

}

inline char get(sst r, sst c)const {

sst row =r\* width;

return contents[row + c];

}

inline char get()const{

return contents[pos];

}

void some\_memeber()const {

++ctr;

}

inline Screen& set(char c) {

contents[pos] = c;

return \*this;

}

inline Screen&set(sst row, sst col, char c) {

move(row, col);

contents[pos] = c;

return \*this;

}

Screen &display(ostream& os) {

do\_display(os);

return \*this;

}

const Screen &display(ostream& os) const{

do\_display(os);

return \*this;

}

private:

sst pos = 0;

sst width = 0, height = 0;

string contents;

mutable size\_t ctr=0;

void do\_display(ostream& os)const {

os << contents;

}

};

int main() {

Screen myScreen(5, 5, 'x');

myScreen.move(4, 0).set('#').display(cout);

cout << endl;

myScreen.display(cout);

return 0;

}

7.30

容易忘记取址或解引用引发错误

7.31

class X {

Y\* cxk;

};

class Y {

X wyf;

};

7.32/7.33

#include<string>

#include<iostream>

#include<vector>

using namespace std;

struct Screen {

using sst = string::size\_type;

Screen() = default;

Screen(const sst& wid, const sst& heig):width(wid),height(heig),contents(wid\*heig,' ') {};

Screen(const sst& wid, const sst& heig, char ch) :width(wid), height(heig), contents(wid\*heig,ch) {};

inline Screen& move(sst r, sst c) {

sst row = r\*width;

pos = row + c;

return \*this;

}

inline char get(sst r, sst c)const {

sst row =r\* width;

return contents[row + c];

}

inline char get()const{

return contents[pos];

}

void some\_memeber()const {

++ctr;

}

inline Screen& set(char c) {

contents[pos] = c;

return \*this;

}

inline Screen& set(sst row, sst col, char c) {

move(row, col);

contents[pos] = c;

return \*this;

}

Screen& display(ostream& os) {

do\_display(os);

return \*this;

}

const Screen& display(ostream& os) const{

do\_display(os);

return \*this;

}

sst size() {

return height\*width;

}

private:

sst pos = 0;

sst width = 0, height = 0;

string contents;

mutable size\_t ctr=0;

void do\_display(ostream& os)const {

os << contents;

}

friend class Windows\_mgr;

};

struct Windows\_mgr {

using Screenst = vector<Screen>::size\_type;

private:

vector<Screen>screens{ Screen(24,80,' ') };

void clear(Screenst sno) {

Screen &s = screens[sno];

s.contents = string(s.height\*s.width, ' ');

}

};

int main() {

Screen myScreen(5, 5, 'x');

myScreen.move(4, 0).set('#').display(cout);

cout << endl;

myScreen.display(cout);

return 0;

}

7.34

类内所有pos均为未定义

7.35

将Type作为string别名

声明一个全局函数initVal及一个Exercise类的成员函数Exercise. initVal()

7.36

struct X {

X(int i,int j):base(i),rem(base%j){}

int base, rem;

};

7.37

Next使用Sales\_data(string=’s’),bookNo为空，units\_sold/revenue为0

Last使用Sales\_data(string=’s’),bookNo为”9-999-99999-9”，units\_sold/revenue为0

7.38

Sales\_data(istream& a=cin) {

read(a, \*this);

}

7.39

不合法，二义性

7.40

class Employee {

string name;

unsigned workNo;

unsigned Phone;

double salary;

public:

Employee():name(""){}

Employee(string &a,unsigned wn=0,unsigned ph=0,double sa=0):name(a),workNo(wn),Phone(ph),salary(sa){}

};

7.42

#include<iostream>

#include<string>

using namespace std;

class Employee {

friend void read(istream &is, Employee &em);

string name;

unsigned workNo;

unsigned Phone;

double salary;

public:

Employee():Employee("",0,0,0){}

Employee(const string &a,unsigned wn,unsigned ph,double sa):name(a),workNo(wn),Phone(ph),salary(sa){}

Employee(istream& is) {

read(is, \*this);

}

Employee(const string &a) :Employee(a,0,0,0){}

};

void read(istream &is, Employee &em) {

is >> em.name >> em.workNo >> em.Phone >> em.salary;

}

7.43

struct NoDefault {

int idont;

NoDefault(const int& i):idont(i){}

};

class C {

NoDefault nd;

C():nd(0){}

};

7.44

不合法，该声明为含有十个NoDefault元素的容器，不包含元素初始化，但NoDefault没有默认构造函数

7.45

合法，C有默认构造函数

7.46

a不正确，可能会有自动合成的默认构造函数

c不提供默认值初始化可能导致出错

d某些情况无法为部分数据成员提供默认初始化

7.48

通过形参为一个string的构造函数创建item1

先将字符串隐式转换为临时string对象，再初始化item2

无影响

7.51

防止单参数被误转换为其他类型

7.52

bookNo初始化为字符串，units\_sold初始化为25，revenue初始化为15.99

7.53

class Debug {

bool hw;

bool io;

bool other;

public:

constexpr Debug(bool b=true):hw(b),io(b),other(b){}

constexpr Debug(bool h,bool i,bool o) : hw(h), io(i), other(o) {}

constexpr bool any() {

return hw || io || other;

}

void set\_io(bool b) { io = b; }

void set\_hw(bool h) { hw = h; }

void set\_other(bool o) { other = o; }

};

7.54

不声明constexpr不会影响代码实现，视具体需求决定要不要声明constexpr

7.55

不是，不含constexpr构造函数

7.57

struct Account {

void calculate() { amount += amount\*interestRate; }

static double rate() { return interestRate; }

static void rate(double newRate) {

interestRate = newRate;

}

private:

string owner;

double amount;

static double initRate() {

return 0;

}

static double interestRate;

};

double Account::interestRate = initRate();

7.58

错误，static不可在类内初始化

8.1/8.2

#include<vector>

#include<iostream>

#include<string>

using namespace std;

istream& wt(istream& in) {

auto olds = in.rdstate();

while (!in.eof()) {

string s;

in>>s;

cout << s;

}

in.setstate(olds);

cin.clear();

return in;

}

int main() {

wt(cin);

return 0;

}

8.3

输入的字面值类型错误或cin.fail（）为true

8.4

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

using namespace std;

int main()

{

ifstream in("C:/test.txt");

if (!in) {

cerr<< "Can't open" << endl;

return -1;

}

string line;

vector<string> text;

while (getline(in, line)) {

text.push\_back(line);

}

in.close();

for (auto elem : text) {

cout << elem << endl;

}

return 0;

}

8.5

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

using namespace std;

int main()

{

ifstream in("C:/test.txt");

if (!in) {

cerr<< "Can't open" << endl;

return -1;

}

string word;

vector<string> words;

while (in>>word) {

words.push\_back(word);

}

in.close();

for (auto w : words) {

cout << w << endl;

}

return 0;

}

8.6

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

class Sales\_data {

friend istream &read(istream& a, Sales\_data &b);

friend ostream &print(ostream &a, const Sales\_data &b);

friend Sales\_data add(const Sales\_data &a, const Sales\_data &b);

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

public:

Sales\_data(string a=" ") :bookNo(a) {}

Sales\_data(unsigned a) :units\_sold(a) {}

Sales\_data(double a, unsigned b) :units\_sold(a), revenue(a\*b) {}

Sales\_data(const string&a, unsigned b, double c) :bookNo(a), units\_sold(b), revenue(b\*c) {}

string isbn() const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

inline double avg\_price() {

return revenue / units\_sold;

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a, const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main(int argc,char\*argv[]) {

ifstream in(argv[1]);

if (in) {

Sales\_data pre(in);

while (in) {

Sales\_data tmp(in);

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(cout, pre);

pre = tmp;

}

if (in.eof())break;

}

in.close();

print(cout, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

8.7/8.8

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

struct Sales\_data;

istream &read(istream& a, Sales\_data &b);

ostream &print(ostream &a, const Sales\_data &b);

Sales\_data add(const Sales\_data &a, const Sales\_data &b);

class Sales\_data {

friend istream &read(istream& a, Sales\_data &b);

friend ostream &print(ostream &a, const Sales\_data &b);

friend Sales\_data add(const Sales\_data &a, const Sales\_data &b);

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;

public:

Sales\_data(string a=" ") :bookNo(a) {}

Sales\_data(unsigned a) :units\_sold(a) {}

Sales\_data(double a, unsigned b) :units\_sold(a), revenue(a\*b) {}

Sales\_data(const string&a, unsigned b, double c) :bookNo(a), units\_sold(b), revenue(b\*c) {}

string isbn() const {

return bookNo;

}

Sales\_data& combine(const Sales\_data &a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

inline double avg\_price() {

return revenue / units\_sold;

}

};

istream &read(istream& a, Sales\_data &b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price\*b.units\_sold;

return a;

}

ostream &print(ostream &a, const Sales\_data &b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data &a, const Sales\_data &b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main(int argc,char\*argv[]) {

ifstream in(argv[1]);

ofstream out(argv[2], ofstream::app);

if (in) {

Sales\_data pre(in);

while (in) {

Sales\_data tmp(in);

if (tmp.isbn() == pre.isbn()) {

pre = add(pre, tmp);

}

else {

print(out, pre);

pre = tmp;

}

if (in.eof())break;

}

in.close();

print(out, pre);

}

else {

cerr << "?" << endl;

return -1;

}

return 0;

}

8.9

#include<vector>

#include<iostream>

#include<string>

#include<sstream>

using namespace std;

istream& wt(istream& in) {

auto olds = in.rdstate();

while (!in.eof()) {

string s;

in >> s;

cout << s;

}

in.setstate(olds);

cin.clear();

return in;

}

int main() {

string example{ "hello world" };

istringstream iss(example);

wt(iss);

return 0;

}

8.10

#include<vector>

#include<iostream>

#include<string>

#include<sstream>

#include<fstream>

using namespace std;

int main(int argc,char\* argv[]) {

ifstream in(argv[1]);

while (in) {

string line;

getline(in, line);

istringstream in(line);

string put;

while (in) {

in >> put;

cout << put << ' ';

}

}

in.close();

return 0;

}

8.11

#include<vector>

#include<iostream>

#include<string>

#include<sstream>

#include<fstream>

using namespace std;

struct PersonInfo {

string name;

vector<string>phones;

};

int main(int argc,char\* argv[]) {

string line, word;

vector<PersonInfo>people;

istringstream record;

while (getline(cin, line)) {

PersonInfo info;

record.clear();

record.str(line);

record >> info.name;

while (record >> word) {

info.phones.push\_back(word);

}

people.push\_back(info);

}

return 0;

}

8.12

默认初始化已足够

8.13

#include<vector>

#include<iostream>

#include<string>

#include<sstream>

#include<fstream>

using namespace std;

struct PersonInfo {

string name;

vector<string>phones;

};

int main(int argc,char\* argv[]) {

ifstream in("example.txt");

string line, word;

vector<PersonInfo>people;

istringstream record;

while (getline(in, line)) {

PersonInfo info;

record.clear();

record.str(line);

record >> info.name;

while (record >> word) {

info.phones.push\_back(word);

}

people.push\_back(info);

}

in.close();

return 0;

}

9.1

A list

B deque

C list

9.2

list<deque<int>>;

9.4

bool search(vector<int>::iterator begin, vector<int>::iterator end, int i) {

while (begin != end) {

if (\*begin == i)return true;

begin++;

}

return false;

}

9.5

vector<int>::iterator search(vector<int>::iterator begin, vector<int>::iterator end, int i) {

while (begin != end) {

if (\*begin == i)return begin;

begin++;

}

return end;

}

9.6

没想明白

9.7

vector<int>::iterator

9.8

const list<string>::iterator

list<string>::iterator

9.9

Cbegin返回的是const迭代器

9.10

V1是以int作为元素的vector

9.13

list<int>li{ 1,2,3,4,5 };

vector<int>vi{ 6,7,8,9 };

vector<double>vd(li.begin(), li.end());//使用list初始化

vector<double>vd(vi.begin(), vi.end());//使用vector<int>初始化，不能直接使用vi因为类型不匹配

9.14

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

int main() {

char ca[]{ 'i','a','m' };

char\* p = ca;

list<char\*>lc{p};

vector<string>vs;

vs.assign(lc.begin(), lc.end());

return 0;

}

9.15

bool compare(vector<int>v1, vector<int>v2) {

if (v1 == v2)return true;

else return false;

}

9.16

bool compare(vector<int>&v1, list<int>&l1) {

if (v1.size() != l1.size())return false;

auto vi = v1.begin();

auto li = l1.begin();

while (vi != v1.end()) {

if (\*vi != \*li)return false;

vi++, li++;

}

return false;

}

9.18

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

int main() {

string bufs;

deque<string>ds;

while (cin >> bufs) {

ds.push\_back(bufs);

}

auto dit = ds.begin();

while (dit != ds.end())

cout << \*dit++ << endl;

return 0;

}

9.19

//更改容器及迭代器即可

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

int main() {

string bufs;

list<string>ls;

while (cin >> bufs) {

ls.push\_back(bufs);

}

auto dit = ls.begin();

while (dit != ls.end())

cout << \*dit++ << endl;

return 0;

}

9.20

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

int main() {

list<int>li{1, 2, 3, 4, 5, 6};

deque<int> odd, even;

for (auto i = li.begin(); i != li.end(); i++) {

if (\*i % 2 == 0)even.push\_back(\*i);

else odd.push\_back(\*i);

}

return 0;

}

9.22

Iter无自增语句

9.23

均为1

9.24

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

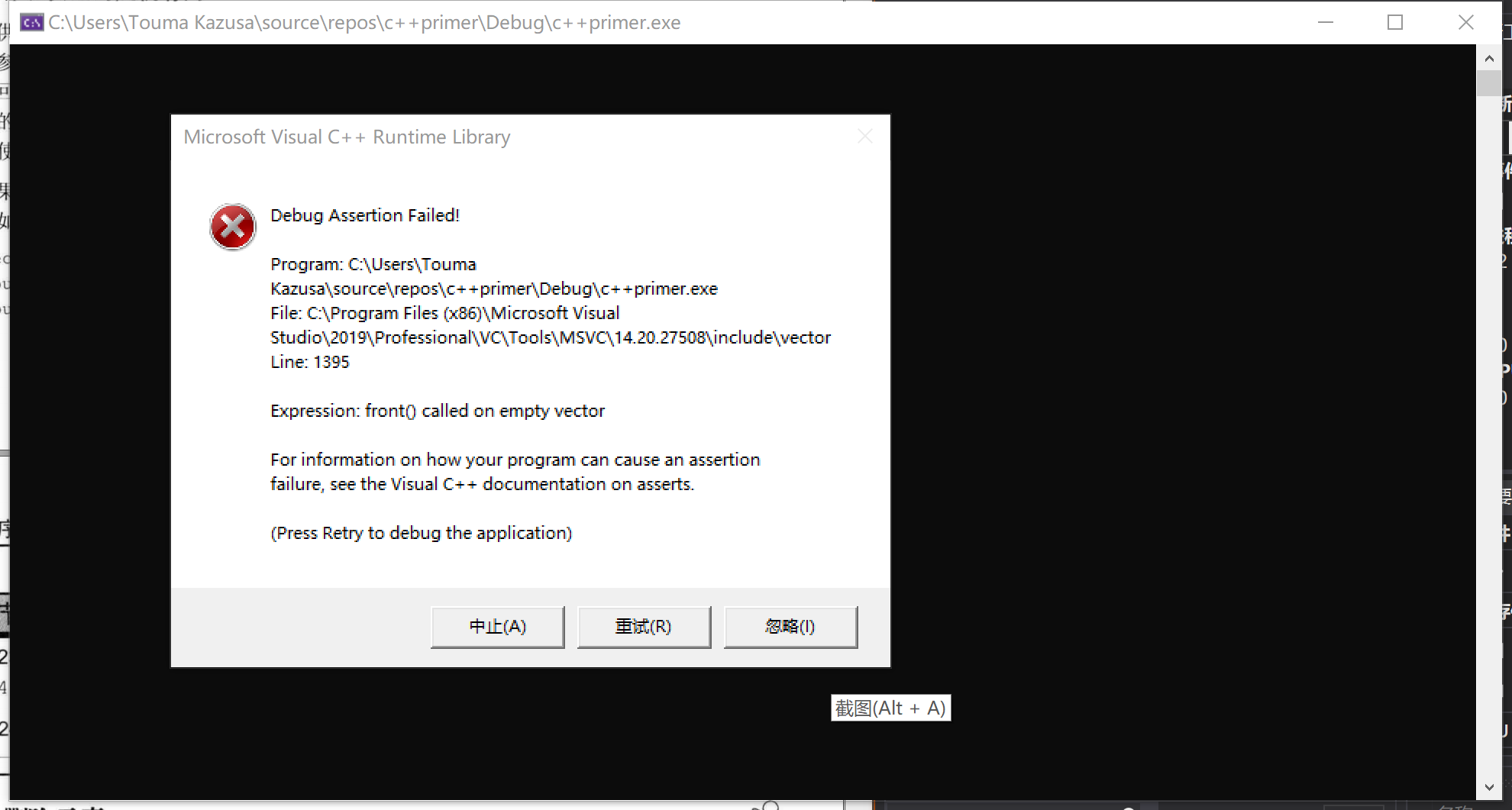
int main() {

vector<int>li;

cout << " "<<li.at(0) << " " << \*li.begin() << " "<< li.front();

return 0;

}



9.25

相等则删除这一个元素

返回为后迭代器

未定义

9.26

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

using namespace std;

int main() {

int ia[]{ 0,1,1,2,3,5,8,13,21,55,89 };

vector<int>vi;

list<int>li;

for (int i = 0;i<11 ; i++) {

vi.push\_back(ia[i]);

li.push\_back(ia[i]);

}

for (auto vib = vi.begin(); vib != vi.end(); ) {

if (\*vib % 2==0)vib=vi.erase(vib);

else vib++;

}

for (auto lib = li.begin(); lib != li.end(); ) {

if (\*lib % 2==1)lib=li.erase(lib);

else lib++;

}

return 0;

}

9.27

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

int main() {

int ia[]{ 0,1,1,2,3,5,8,13,21,55,89 };

forward\_list<int>fli;

auto flib = fli.before\_begin();

for (int i = 0;i<11 ; i++) {

fli.insert\_after(flib,ia[i]);

flib++;

}

auto pre = fli.before\_begin(), tmp = fli.begin();

while (tmp != fli.end()) {

if (\*tmp % 2 == 1)tmp = fli.erase\_after(pre);

else pre = tmp, tmp++;

}

return 0;

}

9.28

void insert(forward\_list<string> fls,const string& s1,const string& s2) {

bool isfind =false;

for (auto pre = fls.before\_begin(), tmp = fls.begin(); tmp != fls.end;) {

if (\*tmp == s1) {

fls.insert\_after(tmp, s2);

isfind = true;

break;

}

else pre = tmp, tmp++;

}

if(isfind==false)fls.insert\_after(fls.end(), s2);

}

9.29

尾部新增75个元素

仅保留头部10元素

9.30

如果比原大小大，则元素类型需要有默认构造函数

9.31

Forward\_list需要对删除/插入的前一节点更改指针

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

int main() {

forward\_list<int>fli{ 0,1,2,3,4,5,6,7,8,9 };

auto pre=fli.before\_begin(),tmp=fli.begin();

while (tmp != fli.end()) {

if (\*tmp % 2) {

pre = fli.insert\_after(tmp, \*tmp);

tmp++;

tmp++;

}

else {

tmp=fli.erase\_after(pre);

}

}

return 0;

}

9.32

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

int main() {

vector<int>vi{ 0,1,2,3,4,5,6,7,8,9 };

auto tmp = vi.begin();

while (tmp != vi.end()) {

if (\*tmp % 2) {

tmp= vi.insert(tmp, \*tmp++);

tmp++;

}

else {

tmp=vi.erase(tmp);

}

}

return 0;

}

9.33

反复在第一和第二个元素间添加元素

9.34

若为奇数则在其之前反复添加一个同样大小元素并将其指针赋值给iter，不会执行iter自增，死循环

（暂时不会记录循环）

9.39

先为一个空vector分配至少能容纳1024个元素的内存空间

输入一定的元素后，将现有元素个数的1/4从尾部去除，内存空间在这次操作不变

9.40

1024及以上

1048个词有可能会超出capacity，导致重新分配空间

9.41

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

int main() {

vector<char> vc;

string s{ "You know nothing John Snow" };

for (auto c : s) {

vc.push\_back(c);

}

auto const vcb = &vc[0];

string s1(vcb,6,8);

cout << s1;

return 0;

}

9.42

s.reserve(100);

9.43

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

void myreplace(string& s,const string& oldVal, const string& newVal) {

auto si = s.begin();

while (si < s.end()) {

if (\*si == \*oldVal.begin()) {

bool flag =true;

auto st = si;

auto ot = oldVal.begin();

while (st<s.end()&&ot < oldVal.end()) {

if (\*st != \*ot){

flag = false;

break;

}

st++;

ot++;

}

if (flag == true&&ot==oldVal.end()) {

si=s.erase(si, si+oldVal.size()-1);

s.insert(si, newVal.begin(),newVal.end());

}

}

si++;

}

}

int main() {

string s{ "this is a simple example" };

myreplace(s, "sim", "exam");

cout << s;

return 0;

}

9.44

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

void myreplace(string& s,const string& oldVal, const string& newVal) {

int si = 0;

while (si < s.size()) {

if (s[si] == oldVal[0]) {

bool flag =true;

int st =si;

int ot = 0;

while (st<s.size()&&ot < oldVal.size()) {

if (s[st] != oldVal[ot]){

flag = false;

break;

}

st++;

ot++;

}

if (flag == true&&ot==oldVal.size()) {

s.replace(si, oldVal.size(), newVal);

si += newVal.size();

}

}

si++;

}

}

int main() {

string s{ "this is a simple example" };

myreplace(s, "sim", "exam");

cout << s;

return 0;

}

9.45

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

void myreplace(string& s,const string& oldVal, const string& newVal) {

int si = 0;

while (si < s.size()) {

if (s[si] == oldVal[0]) {

bool flag =true;

int st =si;

int ot = 0;

while (st<s.size()&&ot < oldVal.size()) {

if (s[st] != oldVal[ot]){

flag = false;

break;

}

st++;

ot++;

}

if (flag == true&&ot==oldVal.size()) {

s.replace(si, oldVal.size(), newVal);

si += newVal.size();

}

}

si++;

}

}

string cname(string name, const string& pre, const string& beh) {

string ret{ name };

ret.insert(ret.begin(), pre.begin(), pre.end());

ret.insert(ret.end(), beh.begin(), beh.end());

return ret;

}

int main() {

string s{ "this is a simple example" };

s=cname(s, "sim", "exam");

cout << s;

return 0;

}

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

void myreplace(string& s,const string& oldVal, const string& newVal) {

int si = 0;

while (si < s.size()) {

if (s[si] == oldVal[0]) {

bool flag =true;

int st =si;

int ot = 0;

while (st<s.size()&&ot < oldVal.size()) {

if (s[st] != oldVal[ot]){

flag = false;

break;

}

st++;

ot++;

}

if (flag == true&&ot==oldVal.size()) {

s.replace(si, oldVal.size(), newVal);

si += newVal.size();

}

}

si++;

}

}

string cname(string name, const string& pre, const string& beh) {

string ret{ name };

ret.insert(ret.begin(), pre.begin(), pre.end());

ret.append( beh);

return ret;

}

int main() {

string s{ "this is a simple example" };

s=cname(s, "sim", "exam");

cout << s;

return 0;

}

9.47

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

using namespace std;

int main() {

string s{ "ab2c3d7R4E6" };

int pos = 0;

const char \* p= "1234567890";

while (pos < s.size()) {

pos = s.find\_first\_of(p, pos);

cout << pos << ' ' << s[pos]<<endl;

pos++;

}

return 0;

}

9.49

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

#include<fstream>

using namespace std;

int main(int argc,char\*argv[]) {

ifstream in(argv[1]);

const char\* o = "bdfghijklpqty";

string word;

vector<string>rw;

while (in >> word) {

if (word.find(o) == string::npos)rw.push\_back(word);

}

string maxword;

size\_t max=0;

for (auto it :rw) {

if (it.size() > max) {

max = it.size();

maxword = it;

}

}

cout << maxword << endl;

return 0;

}

9.50

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

#include<fstream>

using namespace std;

int main(int argc,char\*argv[]) {

vector<string>vs{ "12","34","45","45" };

int sum = 0;

for (auto s : vs) {

sum += stoi(s);

}

cout << sum << endl;

return 0;

}

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

#include<fstream>

using namespace std;

int main(int argc,char\*argv[]) {

vector<string>vs{ "12.89","34.36","45.71","45.88" };

double sum = 0;

for (auto s : vs) {

sum += stod(s);

}

cout << sum << endl;

return 0;

}

9.51

#include<vector>

#include<iostream>

#include<string>

#include <deque>

#include<list>

#include<forward\_list>

#include<fstream>

using namespace std;

struct date {

unsigned year, month, day;

date(const string& ins) {

string s = ins;

const string ar = "0123456789";

if (isupper(s[0]) ){

if (s.find("January")!=string::npos|| s.find("Jan") != string::npos)month = 1;

if (s.find("February") != string::npos)month = 2;

if (s.find("March") != string::npos)month = 3;

if (s.find("April") != string::npos)month = 4;

if (s.find("May") != string::npos)month = 5;

if (s.find("June" )!= string::npos)month = 6;

if (s.find("July") != string::npos)month = 7;

if (s.find("August") != string::npos)month = 8;

if (s.find("September") != string::npos)month = 9;

if (s.find("October") != string::npos)month = 10;

if (s.find("November") != string::npos)month = 11;

if (s.find("December") != string::npos)month = 12;

string sday;

auto sn = s.find\_first\_of(ar);

while (isdigit(s[sn])&&sn<s.size()) {

sday.push\_back(s[sn++]);

}

day = stoi(sday);

string syear;

s = s.substr(sn);

sn = s.find\_first\_of(ar);

while (isdigit(s[sn]) && sn < s.size()) {

syear.push\_back(s[sn++]);

}

year = stoi(syear);

}

else {

string smonth, sday, syear;

auto sn = s.find\_first\_of(ar);

while (isdigit(s[sn]) && sn < s.size()) {

smonth.push\_back(s[sn++]);

}

month= stoi(smonth);

s = s.substr(sn);

sn = s.find\_first\_of(ar);

while (isdigit(s[sn]) && sn < s.size()) {

sday.push\_back(s[sn++]);

}

day = stoi(sday);

s = s.substr(sn);

sn = s.find\_first\_of(ar);

while (isdigit(s[sn]) && sn < s.size()) {

syear.push\_back(s[sn++]);

}

year = stoi(syear);

}

}

};

int main(int argc,char\*argv[]) {

string s{"Jan / 12 /3"};

date d(s);

cout << d.month << " " << d.day << " " << d.year << endl;

return 0;

}

9.52

//默认字符串为括号内仅包含两个数和一个+-\*/且不含负数，否则涉及到优先级匹配算法工作量较大

//较好的方法应为数字与操作符分别建栈

#include<vector>

#include<iostream>

#include<string>

#include<stack>

using namespace std;

stack<char> sc;

int getnum() {

string buf;

while (isdigit(sc.top())) {

buf.insert(buf.begin(),sc.top());

sc.pop();

}

return stoi(buf);

}

string s{"(((16/8)+(2\*94))-(76+92))"};

int main() {

for (char c : s) {

if (c == ')') {

int n2 = getnum();

char ope = sc.top();

sc.pop();//弹出操作符

int n1 = getnum();

sc.pop();//弹出左括号

int rep = 0;

if (ope == '+')rep = (n1 + n2);

else if (ope == '-')rep = (n1 - n2);

else if (ope == '\*')rep = (n1 \* n2);

else if (ope == '/')rep = (n1 / n2);

string re = to\_string(rep);

for (char r : re)sc.push(r);

}

else if (c == ' ') continue;

else sc.push(c);

}

string out;

while (!sc.empty()) {

out.insert(out.begin(), sc.top());

sc.pop();

}

cout << out;

return 0;

}

10.1/10.2

#include<algorithm>

#include<vector>

#include<iostream>

using namespace std;

int main() {

vector<int> vi;

int i;

while (cin >> i) {

vi.push\_back(i);

}

i=6;

auto o = count(vi.begin(), vi.end(), i);

cout << o << endl;

return 0;

}

#include<algorithm>

#include<vector>

#include<iostream>

using namespace std;

int main() {

vector<string> vs;

string s;

while (cin >> s) {

vs.push\_back(s);

}

s="areyoukidding";

auto o = count(vs.begin(), vs.end(),s);

cout << o << endl;

return 0;

}

10.3

#include<algorithm>

#include<vector>

#include<iostream>

#include<numeric>

using namespace std;

int main() {

vector<int>vi{ 1,2,4,5,7,9 };

double sum = accumulate(vi.begin(), vi.end(), 0);

cout << sum << endl;

return 0;

}

10.5

不受影响功能正常

10.6

#include<algorithm>

#include<vector>

#include<iostream>

#include<numeric>

using namespace std;

int main() {

vector<int>vi(10);

fill\_n(vi.begin(), 10, 0);

for (int i : vi) {

cout << i << endl;

}

}

10.7

A vec为空，使用back\_insert()

10.8

严格来说并不是算法导致的大小变化，而是插入迭代器调用了push\_back()

10.9

#include<algorithm>

#include<vector>

#include<iostream>

#include<numeric>

using namespace std;

void elimdups(vector<string>& vs) {

sort(vs.begin(), vs.end());

auto vsi=unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

}

int main() {

vector<string> vs;

string word;

while (cin >> word)vs.push\_back(word);

elimdups(vs);

for (string s : vs)cout << s << endl;

}

10.10

只对迭代器操作，不调用容器函数

10.11

#include<algorithm>

#include<vector>

#include<iostream>

#include<numeric>

using namespace std;

bool sshorter(const string& s1, const string& s2) {

return s1.size() < s2.size();

}

void elimdups(vector<string>& vs) {

sort(vs.begin(), vs.end(),sshorter);

auto vsi=unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

stable\_sort(vs.begin(), vs.end(), sshorter);

}

int main() {

vector<string> vs;

string word;

while (cin >> word)vs.push\_back(word);

elimdups(vs);

for (string s : vs)cout << s << endl;

}

10.12

bool compareIsbn(Sales\_data sd1, Sales\_data sd2) {

return sd1.isbn() < sd2.isbn();

}

int main(int argc, char\* argv[]) {

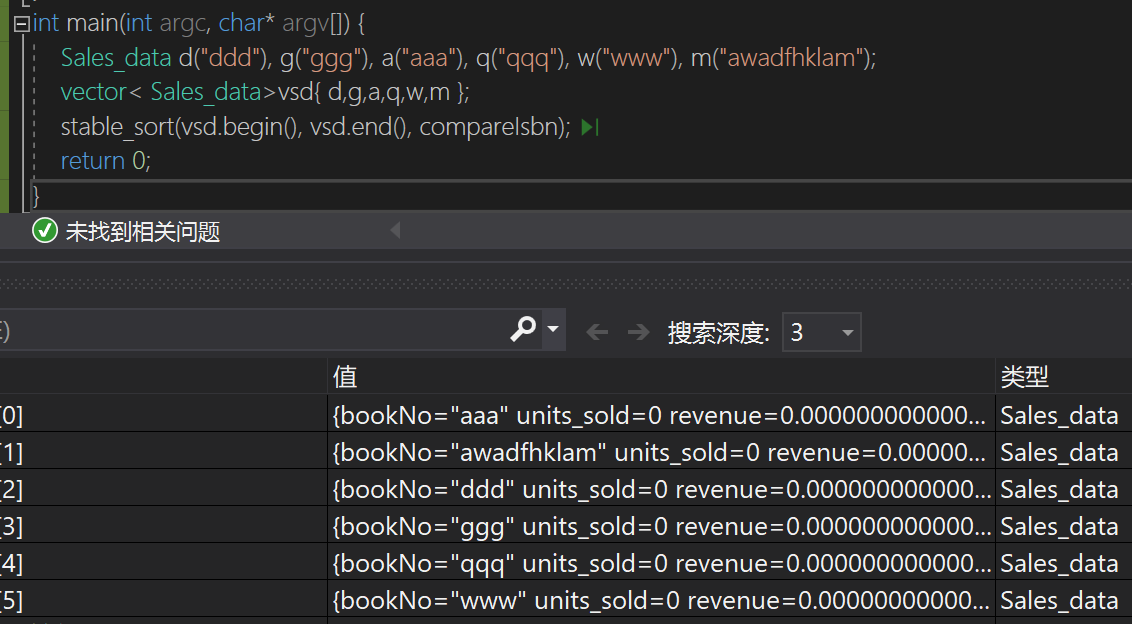
Sales\_data d("ddd"), g("ggg"), a("aaa"), q("qqq"), w("www"), m("awadfhklam");

vector< Sales\_data>vsd{ d,g,a,q,w,m };

stable\_sort(vsd.begin(), vsd.end(), compareIsbn);

return 0;

}



10.13

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

using namespace std;

bool size5(string s) {

return s.size() >= 5 ? true : false;

}

int main(int argc, char\* argv[]) {

vector<string>words{ "you","me","complicate","main","winter","is","english" };

auto it=partition(words.begin(), words.end(), size5);

auto be = words.begin();

while (be < it)cout << \*be++ << endl;

}

10.14

[](const int& i1, const int i2) {

return i1 + i2;

}

10.15

void example(int ad) {

[ad](const int& i1) {

return ad + i2;

}

}

10.16

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

using namespace std;

string make\_plural(size\_t ct, const string& s1, const string& s2) {

return ct <= 1 ? s1 : s1 + s2;

}

void elimdups(vector<string>& vs) {

sort(vs.begin(), vs.end(), [](const string & s1, const string & s2) {

return s1.size() < s2.size(); });

auto vsi = unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

stable\_sort(vs.begin(), vs.end(), [](const string & s1, const string & s2) {

return s1.size() < s2.size(); });

}

void bigges(size\_t ct, vector<string>& vs) {

elimdups(vs);

auto fi = find\_if(vs.begin(), vs.end(), [ct](const string & s) {

return s.size() >= ct;

});

auto co = vs.end() - fi;

cout << co << " " << make\_plural(co, "word", "s") << " of length " << ct << " or longer " << endl;

for\_each(fi, vs.end(), [](const string & s) {

cout << s << endl;

});

}

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","english" };

bigges(5, words);

}

10.17

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

using namespace std;

struct Sales\_data;

istream& read(istream& a, Sales\_data& b);

ostream& print(ostream& a, const Sales\_data& b);

Sales\_data add(const Sales\_data& a, const Sales\_data& b);

class Sales\_data {

friend istream& read(istream& a, Sales\_data& b);

friend ostream& print(ostream& a, const Sales\_data& b);

friend Sales\_data add(const Sales\_data& a, const Sales\_data& b);

string bookNo;

unsigned units\_sold = ;

double revenue = ;

public:

Sales\_data(string a = " ") :bookNo(a) {}

Sales\_data(unsigned a) :units\_sold(a) {}

Sales\_data(double a, unsigned b) :units\_sold(a), revenue(a\* b) {}

Sales\_data(const string& a, unsigned b, double c) :bookNo(a), units\_sold(b), revenue(b\* c) {}

string isbn() const {

return bookNo;

}

Sales\_data& combine(const Sales\_data& a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

inline double avg\_price() {

return revenue / units\_sold;

}

};

istream& read(istream& a, Sales\_data& b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price \* b.units\_sold;

return a;

}

ostream& print(ostream& a, const Sales\_data& b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data& a, const Sales\_data& b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

int main() {

Sales\_data d("ddd"), g("ggg"), a("aaa"), q("qqq"), w("www"), m("awadfhklam");

vector< Sales\_data>vsd{ d,g,a,q,w,m };

stable\_sort(vsd.begin(), vsd.end(), [](Sales\_data sd1, Sales\_data sd2) {

return sd1.isbn() < sd2.isbn(); });

return 0;

}

10.18

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

using namespace std;

string make\_plural(size\_t ct, const string& s1, const string& s2) {

return ct <= 1 ? s1 : s1 + s2;

}

void elimdups(vector<string>& vs) {

auto vsi = unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

}

void bigges(size\_t ct, vector<string>& vs) {

elimdups(vs);

auto fi = partition(vs.begin(), vs.end(), [ct](const string & s) {return s.size() < ct; });

stable\_sort(vs.begin(), vs.end(), [](const string & s1, const string & s2) {

return s1.size() < s2.size(); });

auto co = vs.end() - fi;

cout << co << " " << make\_plural(co, "word", "s") << " of length " << ct << " or longer " << endl;

for\_each(fi, vs.end(), [](const string & s) {

cout << s << endl;

});

}

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","english" };

bigges(5, words);

return 0;

}

10.19

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

using namespace std;

string make\_plural(size\_t ct, const string& s1, const string& s2) {

return ct <= 1 ? s1 : s1 + s2;

}

void elimdups(vector<string>& vs) {

auto vsi = unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

sort(vs.begin(), vs.end());

stable\_sort(vs.begin(), vs.end(), [](const string & s1, const string & s2) {

return s1.size() < s2.size(); });

}

void bigges(size\_t ct, vector<string>& vs) {

elimdups(vs);

auto fi = stable\_partition(vs.begin(), vs.end(), [ct](const string & s) {return s.size() < ct; });

auto co = vs.end() - fi;

cout << co << " " << make\_plural(co, "word", "s") << " of length " << ct << " or longer " << endl;

for\_each(fi, vs.end(), [](const string & s) {

cout << s << endl;

});

}

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","coming","english","ridcus","aaaaaa","bbbbbb" };

bigges(5, words);

return 0;

}

10.20

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","coming","english","ridcus","aaaaaa","bbbbbb" };

auto c=count\_if(words.begin(), words.end(), [](const string & s) {

return s.size() > 5 ? true : false; });

cout << c;

return 0;

}

10.21

[&i]()->bool {

if (i = 0)return true;

else { i--; return false; }

10.22

#include<iostream>

#include<string>

#include<algorithm>

#include <functional>

#include<vector>

using namespace std;

using namespace std::placeholders;

string make\_plural(size\_t ct, const string& s1, const string& s2) {

return ct <= 1 ? s1 : s1 + s2;

}

bool sizejudge(const string& s, size\_t sz) {

return s.size() < sz;

}

bool sizecom(const string& s1, const string& s2) {

return s1.size() < s2.size();

}

void pri(const string& s,ostream& os) {

cout << s << endl;

}

void elimdups(vector<string>& vs) {

auto vsi = unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

sort(vs.begin(), vs.end());

stable\_sort(vs.begin(), vs.end(), bind(sizecom, \_1, \_2));

}

void bigges(size\_t ct, vector<string>& vs) {

elimdups(vs);

auto fi = stable\_partition(vs.begin(), vs.end(), bind(sizejudge,\_1,ct));

auto co = vs.end() - fi;

cout << co << " " << make\_plural(co, "word", "s") << " of length " << ct << " or longer " << endl;

for\_each(fi, vs.end(), bind(pri, \_1, ref(cout)));

}

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","coming","english","ridcus","aaaaaa","bbbbbb" };

bigges(5, words);

return 0;

}

10.23

视其调用对象而定

10.24

#include<iostream>

#include<string>

#include<algorithm>

#include <functional>

#include<vector>

using namespace std;

using namespace std::placeholders;

bool check\_size(int i, size\_t sz) {

return i > sz;

}

int lookfor(const string& s, vector<int>v) {

auto sz = s.size();

auto it = find\_if(v.begin(), v.end(), bind(check\_size,\_1,sz));

return \*it;

}

int main() {

vector<int>vi{ 1,2,3,4,5,6,7,8,9,11 };

string si;

cin >> si;

int i=lookfor(si, vi);

cout << i;

return 0;

}

10.25

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

#include <functional>

using namespace std;

using namespace std::placeholders;

void print(const string& s) {

cout << s << endl;

}

bool comp(const string& s1, const string& s2) {

return s1.size() < s2.size();

}

bool issh(size\_t st, const string& s) {

return s.size() < st;

}

string make\_plural(size\_t ct, const string& s1, const string& s2) {

return ct <= 1 ? s1 : s1 + s2;

}

void elimdups(vector<string>& vs) {

auto vsi = unique(vs.begin(), vs.end());

vs.erase(vsi, vs.end());

sort(vs.begin(), vs.end());

stable\_sort(vs.begin(), vs.end(), bind(comp, \_1, \_2));

}

void bigges(size\_t ct, vector<string>& vs) {

elimdups(vs);

auto fi = stable\_partition(vs.begin(), vs.end(), bind(issh, ct, \_1));

auto co = vs.end() - fi;

cout << co << " " << make\_plural(co, "word", "s") << " of length " << ct << " or longer " << endl;

for\_each(fi, vs.end(), bind(print, \_1));

}

int main() {

vector<string>words{ "you","me","complicate","main","winter","is","coming","english","ridcus","aaaaaa","bbbbbb" };

bigges(5, words);

return 0;

}

10.27

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

#include <functional>

#include<list>

using namespace std;

using namespace std::placeholders;

int main() {

vector<int>nums{1,2,3,4,5,6,7,88,8,8,8,8,8,8,8,8,8,8,8,8,8,8,91,1,2,2,2,2 };

list<int> li;

sort(nums.begin(), nums.end());

unique\_copy(nums.begin(), nums.end(), back\_inserter(li));

return 0;

}

10.28

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

#include <functional>

#include<list>

#include<deque>

#include<iterator>

using namespace std;

using namespace std::placeholders;

int main() {

vector<int>nums{1,2,3,4,5,6,7,8,9,10};

list<int> li;

list<int>li2;

deque<int>di;

auto lit = inserter(li, li.begin());

auto li2i = front\_inserter(li2);

auto dii = back\_inserter(di);

for (int i : nums) {

\*lit = i;

\*li2i = i;

\*dii = i;

}

for (int i : li)cout << i << " ";

cout << endl;

for (int i : li2)cout << i << " ";

cout << endl;

for (int i : di)cout << i << " ";

cout << endl;

return 0;

}

10.29

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

#include <functional>

#include<list>

#include<deque>

#include<iterator>

#include<fstream>

using namespace std;

using namespace std::placeholders;

int main() {

vector<string>vs;

ifstream ifs ("input.txt" );

istream\_iterator<string> iii = (ifs), eof;

while (iii != eof) {

vs.push\_back(\*iii++);

}

ostream\_iterator<string>ois =(cout);//无法添加空格，数组也不行？？？

for (string s : vs) {

\*ois = s;

\*ois = " ";

}

cout << endl;

return 0;

}

10.30

#include<iostream>

#include<string>

#include<algorithm>

#include<vector>

#include <functional>

#include<list>

#include<deque>

#include<iterator>

#include<fstream>

using namespace std;

using namespace std::placeholders;

int main() {

vector<int>vi;

istream\_iterator<int> iii = (cin), eof;

ostream\_iterator<int>oii(cout," ");

while (iii != eof) {

vi.push\_back(\*iii++);

}

sort(vi.begin(), vi.end());

copy(vi.begin(), vi.end(), oii++);

cout << endl;

return 0;

}

10.31

//不知道怎么用accumulate求和

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

using namespace std;

struct Sales\_data;

istream& read(istream& a, Sales\_data& b);

ostream& print(ostream& a, const Sales\_data& b);

Sales\_data add(const Sales\_data& a, const Sales\_data& b);

class Sales\_data {

friend istream& read(istream& a, Sales\_data& b);

friend ostream& print(ostream& a, const Sales\_data& b);

friend Sales\_data add(const Sales\_data& a, const Sales\_data& b);

string bookNo;

unsigned units\_sold ;

double revenue ;

public:

Sales\_data(string a = " ") :bookNo(a) {}

Sales\_data(unsigned a) :units\_sold(a) {}

Sales\_data(double a, unsigned b) :units\_sold(a), revenue(a\* b) {}

Sales\_data(const string& a, unsigned b, double c) :bookNo(a), units\_sold(b), revenue(b\* c) {}

string isbn() const {

return bookNo;

}

Sales\_data& combine(const Sales\_data& a) {

units\_sold += a.units\_sold, revenue += a.revenue;

return \*this;

}

Sales\_data(istream& a) {

read(a, \*this);

}

inline double avg\_price() {

return revenue / units\_sold;

}

};

istream& read(istream& a, Sales\_data& b) {

double price{ 0 };

a >> b.bookNo >> b.units\_sold >> price;

b.revenue = price \* b.units\_sold;

return a;

}

ostream& print(ostream& a, const Sales\_data& b) {

a << b.isbn() << " " << b.units\_sold << " " << b.revenue << endl;

return a;

}

Sales\_data add(const Sales\_data& a, const Sales\_data& b) {

Sales\_data ret = a;

ret.combine(b);

return ret;

}

bool compareIsbn(Sales\_data sd1, Sales\_data sd2) {

return sd1.isbn() < sd2.isbn();

}

int main() {

ifstream ifsd("sdinput.txt");

istream\_iterator<Sales\_data> iisd=(ifsd),eof;

vector< Sales\_data>vsd{};

while (iisd != eof) {

Sales\_data sd= \*iisd++;

vsd.push\_back(sd);

}

sort(vsd.begin(), vsd.end(), compareIsbn);

auto it = vsd.begin();

while(it!=vsd.end()){

string s = (\*it).isbn();

auto itp1 = find\_if\_not(it, vsd.end(), [s](Sales\_data & sd) {return sd.isbn() == s; });

auto itp = itp1-1;

accummulate(it, itp, 0);

}

if(itp==vsd.end()){

cout << (\*it).isbn();

return 0;

}

10.33

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

using namespace std;

void pick(const string in,const string outodd,const string outeven) {

ifstream inf(in);

istream\_iterator<int> iii(inf), eof;

ofstream out1(outodd), out2(outeven);

ostream\_iterator<int>odd(out1, " "), even(out2, " ");

while (inf) {

int i = \*iii++;

if (i % 2 == 1) \* odd = i;

else \*even = i;

}

}

int main() {

pick("numin.txt", "odd.txt", "even.txt");

return 0;

}

10．34

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

using namespace std;

int main() {

vector<int>vi{ 1,2,3,4,5,6,7,8 };

for\_each(vi.rbegin(), vi.rend(), [](int i) {

cout << i << endl;

});

return 0;

}

10.35

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

using namespace std;

int main() {

vector<int>vi{ 1,2,3,4,5,6,7,8 };

sort(vi.rbegin(), vi.rend());

for\_each(vi.begin(), vi.end(), [](int i) {

cout << i << endl;

});

return 0;

}

10.36

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

#include<list>

using namespace std;

int main() {

list<int>vi{ 1,2,3,0,4,5,0,6,7,8 };

auto it = find(vi.rbegin(), vi.rend(), 0);

for\_each(it, vi.rend(), [](int i) {

cout << i << endl;

});

return 0;

}

10.37

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

#include<list>

using namespace std;

int main() {

list<int>vi{ 1,2,3,0,4,5,0,6,7,8 };

list<int>li;

auto ri1 = vi.rend();

for (int i = 1; i < 3; i++)ri1--;

auto ri2 = vi.rend();

for (int i = 1; i < 8; i++)ri2--;

for(;ri2!=ri1;ri2++) {

cout << \*ri2 << endl;

li.push\_back(\*ri2);

}

return 0;

}

10.39

List 双向迭代器

Vector 随机访问迭代器

10.40

Copy 输出迭代器

Reverse 双向迭代器

Unique 随机访问迭代器

10.42

#include<algorithm>

#include<vector>

#include<iostream>

#include<numeric>

#include<list>

using namespace std;

void elimdups(list<string>& vs) {

vs.sort();

vs.unique();

}

int main() {

list<string> vs;

string word;

while (cin >> word)vs.push\_back(word);

elimdups(vs);

for (string s : vs)cout << s << endl;

}

11.4

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

string word;

map < string, int>msi;

while (cin >> word) {

auto it = word.begin();

while (it != word.end()) {

if (ispunct(\*it)) {

word.erase(it);

break;

}

tolower(\*it++);

}

++msi[word];

}

for (auto w : msi) {

cout << w.first << " appears " << w.second << ((w.second > 1) ? " times " : " time " )<< endl;

}

}

11.7

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

string family;

map < string, vector<string>>msv;

while (cin >>family) {

string name;

while (cin >> name) {

msv[family].push\_back(name);

}

}

return 0;

}

11.8

Set可以快速查找关键字

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

vector<string>words;

string word;

while (cin >>word) {

words.push\_back(word);

}

return 0;

}

11.9

map <string, list<size\_t>>msl;

11.11

multiset<Sales\_data, bool(\*)(const Sales\_data, const Sales\_data)>bookstore(&compareIsbn);

11.12

#include<iostream>

#include<string>

#include<vector>

#include <algorithm>

#include<fstream>

#include<iterator>

#include<set>

using namespace std;

int main() {

pair<string, int>psi;

vector<pair<string, int>>vp;

while (cin >> psi.first) {

cin >> psi.second;

vp.push\_back(psi);

}

}

11.14

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

string family;

map < string, vector<pair<string, int>>>msv;

while (cin >> family) {

string name;

int birthday;

while (cin >> name >> birthday) {

pair<string, int> psi{ name,birthday };

msv[family].push\_back(psi);

}

}

return 0;

}

11.15

Mappen\_type vector<int>

Key\_type int

Value\_type pair<const int,vector<int>>

11.16

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

map<int, int>mii{ {1,0} };

auto it = mii.begin();

it->second = 1;

return 0;

}

11.18

map<string,size\_t>::iterator

11.19

multiset<Sales\_data, bool(\*)(const Sales\_data, const Sales\_data)>::iterator

11.20

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

string word;

map < string, int>msi;

while (cin >> word) {

auto it = word.begin();

while (it != word.end()) {

if (ispunct(\*it)) {

word.erase(it);

break;

}

tolower(\*it++);

}

auto fb = msi.insert({word, 1});

if (!fb.second)(\*fb.first).second++;

}

for (auto w : msi) {

cout << w.first << " appears " << w.second << ((w.second > 1) ? " times " : " time ") << endl;

}

}

11.23

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

string family;

string name;

multimap < string, string>msv;

while (cin >> family>>name) {

msv.insert({ family,name });

}

return 0;

}

11.26

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

map<int, int>mii;

int pci = mii[100]++;

cout << pci;

return 0;

}

11.28

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

map<string, vector<int>>mii;

map<string, vector<int>>::iterator m = mii.find("leibs");

return 0;

}

11.29

均會返回一個不影響容器内順序的插入該關鍵字位置的迭代器

11.31

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

multimap<string, string>mss{ {"a","a"},{"a","b"},{"a","c"},{"b","a"},{"b","b"} };

auto it = mss.find("c");

auto ct = mss.count("c");

while (ct) {

mss.erase(it++);

ct--;

}

return 0;

}

11.32

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

using namespace std;

int main() {

multimap<string, string>mss{ {"a","a"},{"a","b"},{"a","c"},{"b","a"},{"b","b"} };

auto it = mss.begin();

string pres = (\*it).first;

cout << pres << ":" << endl;

while (it!=mss.end()) {

if ((\*it).first != pres) {

pres = (\*it).first;

cout << pres <<":"<< endl;

}

cout << " "<<(\*it++).second << endl;

}

return 0;

}

11.33

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<fstream>

#include<sstream>

using namespace std;

map<string, string> buildmap(ifstream& ifs) {

map<string, string> ret;

string key, towords;

while (ifs>>key&&getline(ifs, towords)) {

ret[key] = towords.substr(1);

}

return ret;

}

string wordtrans(map<string, string>rules, string word) {

auto it = rules.find(word);

if (it == rules.end())return word;

else return (\*it).second;

}

void trans(ifstream& rulefile, ifstream& inputfile) {

auto rules = buildmap(rulefile);

string line;

while (getline(inputfile, line)) {

istringstream l(line);

string word;

bool fw = true;

while (l>> word) {

if (!fw) {

cout << " " << wordtrans(rules,word);

}

else {

cout << wordtrans(rules, word);

fw = false;

}

}

cout << endl;

}

}

int main() {

ifstream rule{ "transrules.txt" }, input{ "senin.txt" };

trans(rule, input);

return 0;

}

11.34

不存在的關鍵詞會被創建

11.38

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<fstream>

#include<sstream>

#include<unordered\_map>

using namespace std;

unordered\_map<string, string> buildmap(ifstream& ifs) {

unordered\_map<string, string> ret;

string key, towords;

while (ifs>>key&&getline(ifs, towords)) {

ret[key] = towords.substr(1);

}

return ret;

}

string wordtrans(unordered\_map<string, string>rules, string word) {

auto it = rules.find(word);

if (it == rules.end())return word;

else return (\*it).second;

}

void trans(ifstream& rulefile, ifstream& inputfile) {

auto rules = buildmap(rulefile);

string line;

while (getline(inputfile, line)) {

istringstream l(line);

string word;

bool fw = true;

while (l>> word) {

if (!fw) {

cout << " " << wordtrans(rules,word);

}

else {

cout << wordtrans(rules, word);

fw = false;

}

}

cout << endl;

}

}

int main() {

ifstream rule{ "transrules.txt" }, input{ "senin.txt" };

trans(rule, input);

return 0;

}

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<unordered\_map>

using namespace std;

int main() {

string word;

unordered\_map < string, int>msi;

while (cin >> word) {

auto it = word.begin();

while (it != word.end()) {

if (ispunct(\*it)) {

word.erase(it);

break;

}

tolower(\*it++);

}

++msi[word];

}

for (auto w : msi) {

cout << w.first << " appears " << w.second << ((w.second > 1) ? " times " : " time ") << endl;

}

}

12.1

B2/b1的成员data指向相同的vector对象，其中含有4个string元素

12.2

class StrBlob {

using size\_type=vector<string>::size\_type;

shared\_ptr<vector<string>> data;

void check(size\_type i, const string& msg)const {

if (i >= data->size())throw out\_of\_range(msg);

}

public:

StrBlob():data(make\_shared<vector<string>>()) {}

StrBlob(initializer\_list<string> il):data(make\_shared<vector<string>>(il)) {}

size\_type size() const {

return data->size();

}

bool empty()const {

return data->empty();

}

void push\_back(const string& s) {

data->push\_back(s);

}

void pop\_back() {

check(0, "pop\_back on empty StrBlob");

data->pop\_back();

}

string& front() {

check(0, "front on empty StrBlob");

return data->front();

}

string& back() {

check(0, "back on empty StrBlob");

return data->back();

}

const string& front()const {

check(0, "pop\_back on empty StrBlob");

data->pop\_back();

}

const string& back()const {

check(0, "pop\_back on empty StrBlob");

data->pop\_back();

}

};

12.3

不需要，对于常量对象不需要修改

12.4

调用该成员函数的所有操作都是可以由程序员定义

12.5

优点：不会误读

缺点：显式转换会略微麻烦一些

12.6

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<unordered\_map>

using namespace std;

void inp(vector <int>& v) {

int i;

while (cin >> i)v.push\_back(i);

}

void outp(vector <int> v) {

for (int i : v) cout << i << " ";

}

int main() {

vector<int>\* vi = new vector<int>;

inp(\*vi);

outp(\*vi);

delete vi;

}

12.7

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<unordered\_map>

using namespace std;

void inp(vector <int>& v) {

int i;

while (cin >> i)v.push\_back(i);

}

void outp(vector <int> v) {

for (int i : v) cout << i << " ";

}

int main() {

shared\_ptr <vector<int>> vi = make\_shared< vector<int>>();

inp(\*vi);

outp(\*vi);

}

12.8

返回值类型为bool，实际返回为指针，且未释放动态内存

12.9

R与q均为指向动态内存int（42）的指针

R2 q2均为指向动态内存int（42）的shared\_ptr

12.10

正确

12.11

调用结束后p所指向的动态内存被delete

12.13

Sp悬空

12.14

void f(destination& d) {

connection c = connect(d);

shared\_ptr<connection> sp(make\_shared(&c,end\_connection);

}

12.15

void f(destination& d) {

connection c = connect(d);

shared\_ptr<connection> sp(make\_shared(&c, [](connection \* sp) {disconnect(\*sp); });

}

12.17

A 非法，不支持拷贝

B 非法，非new创建的动态对象

C 合法，但要注意delete

D 非法，非new创建的动态对象

E 合法，但要注意delete

F 合法，但p5释放时p2也会失效

12.18

Shared\_ptr本身的计数机制非常合理，如果在还有指向该内存的指针存在时释放会导致指针失效，而所有指针都销毁时自动释放内存省去了delete的烦恼，所以不需要release

12.19

#pragma once

#include<algorithm>

#include<vector>

#include<iostream>

#include<map>

#include<set>

#include<unordered\_map>

using namespace std;

class StrBlob {

friend class StrBlobPtr;

using size\_type=vector<string>::size\_type;

shared\_ptr<vector<string>> data;

void check(size\_type i, const string& msg)const {

if (i >= data->size())throw out\_of\_range(msg);

}

public:

StrBlobPtr begin() {

return StrBlobPtr(\*this);

}

StrBlobPtr end() {

auto ret = StrBlobPtr((\*this), data->size());

return ret;

}

StrBlob() :data(make\_shared<vector<string>>()) {}

StrBlob(initializer\_list<string> il) :data(make\_shared<vector<string>>(il)) {}

size\_type size() const {

return data->size();

}

bool empty()const {

return data->empty();

}

void push\_back(const string & s) {

data->push\_back(s);

}

void pop\_back() {

check(0, "pop\_back on empty StrBlob");

data->pop\_back();

}

string& front() {

check(0, "front on empty StrBlob");

return data->front();

}

string& back() {

check(0, "back on empty StrBlob");

return data->back();

}

const string& front()const {

check(0, "front on empty StrBlob");

return data->front();

}

const string& back()const {

check(0, "back on empty StrBlob");

return data->back();

}

};

class StrBlobPtr {

shared\_ptr<vector<string>>check(size\_t i, const string&msg)const {

shared\_ptr<vector<string>> ret = wptr.lock();

if (!ret)

throw runtime\_error("unbound StrBlobPtr");

if (i >= ret->size())

throw out\_of\_range(msg);

return ret;

}

weak\_ptr<vector<string>> wptr;

size\_t curr;

public:

StrBlobPtr() :curr(0) {}

StrBlobPtr(StrBlob& a, size\_t sz = 0) : wptr(a.data), curr(sz) {}

string& deref() const{

auto p = check(curr, "dereference past end");

return (\*p)[curr];

}

StrBlobPtr& incr(){

check(curr, "increment past end of StrBlob Ptr");

++curr;

return \*this;

}

};

12.20

#include<algorithm>

#include<vector>

#include<iostream>

#include<fstream>

#include<string>

#include<memory>

#include<initializer\_list>

using namespace std;

class StrBlobPtr;

class StrBlob {

public:

using size\_type=vector<string>::size\_type;

StrBlob() :data(make\_shared<vector<string>>()) {}

StrBlob(initializer\_list<string> il) :data(make\_shared<vector<string>>(il)) {}

size\_type size() const {

return data->size();

}

bool empty()const {

return data->empty();

}

void push\_back(const string& s) {

data->push\_back(s);

}

void pop\_back() {

check(0, "pop\_back on empty StrBlob");

data->pop\_back();

}

string& front() {

check(0, "front on empty StrBlob");

return data->front();

}

string& back() {

check(0, "back on empty StrBlob");

return data->back();

}

const string& front()const {

check(0, "front on empty StrBlob");

return data->front();

}

const string& back()const {

check(0, "back on empty StrBlob");

return data->back();

}

private:

friend class StrBlobPtr;

StrBlobPtr begin();

StrBlobPtr end();

shared\_ptr<vector<string>> data;

void check(size\_type i, const string& msg)const {

if (i >= data->size())throw out\_of\_range(msg);

}

};

class StrBlobPtr {

shared\_ptr<vector<string>>check(size\_t i, const string& msg)const;

weak\_ptr<vector<string>> wptr;

size\_t curr;

public:

StrBlobPtr(StrBlob & a, size\_t sz = 0) : wptr(a.data), curr(sz) { }

StrBlobPtr() :curr(0) {}

string& deref() const;

StrBlobPtr& incr();

};

StrBlobPtr StrBlob::begin() { return StrBlobPtr(\*this); }

StrBlobPtr StrBlob::end() { return StrBlobPtr(\*this, data->size()); }

shared\_ptr<vector<string>> StrBlobPtr::check(size\_t i, const string& msg)const {//检查curr未越界与有无shared\_ptr，返回shared\_ptr

shared\_ptr<vector<string>> ret = wptr.lock();//当前无shared\_ptr则返回空

if (!ret)

throw runtime\_error("unbound StrBlobPtr");

if (i >= ret->size())

throw out\_of\_range(msg);

return ret;

}

StrBlobPtr& StrBlobPtr::incr() {//自增并返回StrBlobPtr

check(curr, "increment past end of StrBlob Ptr");

++curr;

return \*this;

}

string& StrBlobPtr::deref() const {

auto p = check(curr, "dereference past end");

return (\*p)[curr];

}

int main() {

StrBlob a;

ifstream ifs{ "input.txt" };

string s;

while (getline(ifs, s)) {

a.push\_back(s);

}

StrBlobPtr ap(a);

for (int i = 0; i <= 3; i++) {

ap = ap.incr();

cout << ap.deref() << endl;

}

return 0;

}

12.22

ConstStrBlobPtr(const StrBlob & a, size\_t sz = 0) : wptr(a.data), curr(sz) { }

12.23

int main() {

string con = "abc" ;

con += "ccc";

unique\_ptr<char[]> upc{ new char[100] };

size\_t st = 0;

for (char c : con) {

upc[st++] = c;

}

}

12.24

#include<iostream>

#include<new>

#include<string>

using namespace std;

int main() {

unique\_ptr<char[]> upc{ new char[20] };

string buf;

size\_t st = 0;

while (cin >> buf) {

for (char c : buf) {

upc[st++] = c;

}

}

return 0;

}

12.25

delete []pa;

12.26

#include<iostream>

#include<new>

#include<string>

#include<memory>

using namespace std;

int main() {

allocator<string> as;

auto ias = as.allocate(10);

auto f = ias;

string s;

while (cin >> s) {

as.construct(ias++, s);

}

cout << \*f;

return 0;

}

12.27

#include<iostream>

#include<new>

#include<string>

#include<memory>

#include<vector>

#include<set>

#include<map>

#include<fstream>

#include<sstream>

using namespace std;

class TextQuery {

vector<string> textbuf;//文本

map<string, pair<int,set<int>>> word2line;//词及其出现次数与行号集合

public:

//建立文本构造函数

TextQuery(ifstream& in) {

string s;

size\_t i = 0;

while (getline(in, s)) {//保存一行到s

textbuf.push\_back(s);//保存行

for (auto& w : s)w = tolower(w);//统一小写

istringstream ins(s);

string word;

while(ins>>word){

word2line[word].first++;//出现次数

word2line[word].second.insert(i);//行号记录

}

i++;//行号+1

}

}

//取出某行号对应一行

string oneline(size\_t i) {

return textbuf[i];

}

//查询并保存结果

QueryResult query(string word) {

QueryResult ans(\*this,word,word2line[word].first, word2line[word].second);

return ans;

}

};

class QueryResult {

friend class TextQuery;

TextQuery txtbuf;//文本暂存

string desword;//查找目标词

int times;//出现次数

set<size\_t> linnum;//行号记录

public:

QueryResult(TextQuery tq, string s, int i, set<size\_t> si) :txtbuf(tq), desword(s), times(i), linnum(si) {}

//返回目标文本

TextQuery text() {

return txtbuf;

}

//返回目标词

string answord() {

return desword;

}

//返回出现次数

int counts() {

return times;

}

//返回出现行号set

set<size\_t> rowlist() {

return linnum;

}

};

//输出

void print(ostream out, QueryResult ans) {

out << ans.answord() << " occurs " << ans.counts() << ((ans.counts() > 1 )? " times " : " time " )<< endl;

auto seti = ans.rowlist().begin();

while (seti != ans.rowlist().end()) {

out << "\t(line " << \*seti << ") " << ans.text().oneline(\*seti)<<endl;

}

}

12.30

#include<iostream>

#include<new>

#include<string>

#include<memory>

#include<vector>

#include<set>

#include<map>

#include<fstream>

#include<sstream>

using namespace std;

class TextQuery;

class QueryResult;

ostream& print(ostream& out,const QueryResult& ans);

void runQueries(ifstream& infile);

class TextQuery {

friend class QueryResult;

shared\_ptr<vector<string>> textbuf;//文本

map<string, pair<int, shared\_ptr<set<size\_t>>>> word2line;//词,出现次数与行号集合的pair

public:

//建立文本构造函数

TextQuery(ifstream& in):textbuf(new vector<string>){

string s;

size\_t i = 0;

while (getline(in, s)) {//保存一行到s

textbuf->push\_back(s);//保存行

for (auto& w : s)w = tolower(w);//统一小写

istringstream ins(s);

string word;

while (ins >> word) {

word2line[word].first++;//出现次数

auto & spss=word2line[word].second;//行号记录

if (!spss)spss.reset(new set<size\_t>);

spss->insert(i);

}

i++;//行号+1

}

}

//查询并保存结果

QueryResult query(const string& word);

};

class QueryResult {

public:

friend class TextQuery;

shared\_ptr<vector<string>> txtbuf;//文本

string desword;//查找目标词

int times;//出现次数

shared\_ptr<set<size\_t>> linnum;//行号记录

QueryResult(shared\_ptr<vector<string>> tq, string s, int i, shared\_ptr<set<size\_t>> si) :txtbuf(tq), desword(s), times(i), linnum(si) {}

//返回目标文本

};

QueryResult TextQuery::query(const string& word) {

static shared\_ptr <set<size\_t>>nodata(new set<size\_t>);

auto loc = word2line.find(word);

if (loc == word2line.end())

return QueryResult(textbuf, word, 0, nodata);

else

return QueryResult(textbuf, word, word2line[word].first, word2line[word].second);

}

//输出

ostream& print(ostream& out,const QueryResult& ans) {

out << ans.desword<< " occurs " << ans.times << ((ans.times > 1 )? " times " : " time " )<< endl;

for (auto w : \*ans.linnum) {

out << "\t( line " << w + 1 << " ) " << \*(ans.txtbuf->begin()+ w )<< endl;

}

return out;

}

void runQueries(ifstream& infile) {

TextQuery tq(infile);

while (true) {

cout << "Enter word to look for,or q to quit:";

string s;

if (!(cin >> s) || s == "q")break;

print(cout, tq.query(s))<< endl;

}

}

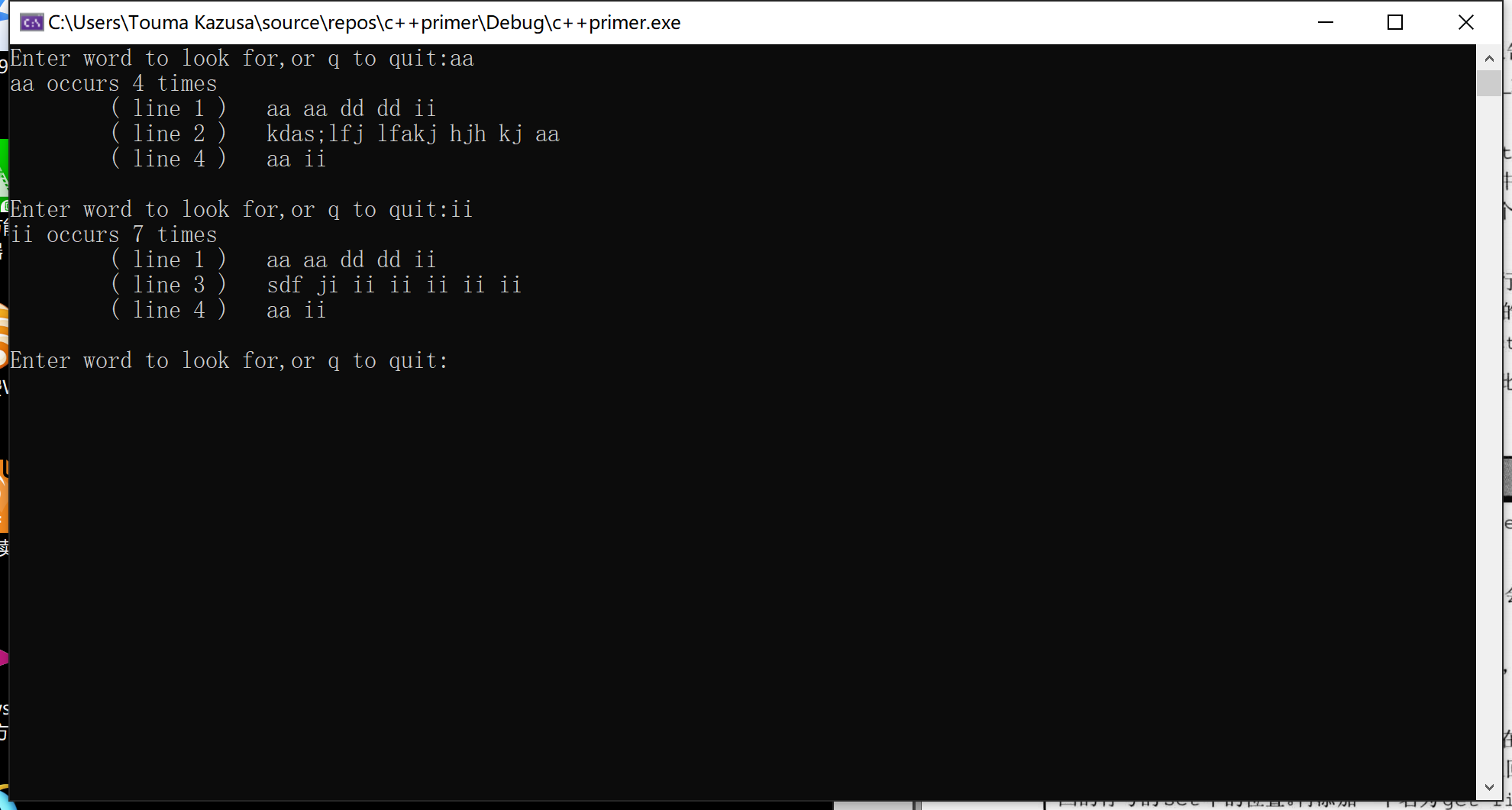
int main() {

ifstream inf("text.txt");

runQueries(inf);

return 0;

}



12.33

#include<iostream>

#include<new>

#include<string>

#include<memory>

#include<vector>

#include<set>

#include<map>

#include<fstream>

#include<sstream>

#include"str.h"

using namespace std;

class TextQuery;

class QueryResult;

ostream& print(ostream& out,const QueryResult& ans);

void runQueries(ifstream& infile);

class TextQuery {

friend class QueryResult;

shared\_ptr<vector<string>> textbuf;//文本

map<string, pair<int, shared\_ptr<set<size\_t>>>> word2line;//词,出现次数与行号集合的pair

public:

//建立文本构造函数

TextQuery(ifstream& in):textbuf(new vector<string>){

string s;

size\_t i = 0;

while (getline(in, s)) {//保存一行到s

textbuf->push\_back(s);//保存行

for (auto& w : s)w = tolower(w);//统一小写

istringstream ins(s);

string word;

while (ins >> word) {

word2line[word].first++;//出现次数

auto & spss=word2line[word].second;//行号记录

if (!spss)spss.reset(new set<size\_t>);

spss->insert(i);

}

i++;//行号+1

}

}

//查询并保存结果

QueryResult query(const string& word);

};

class QueryResult {

public:

friend class TextQuery;

shared\_ptr<vector<string>> txtbuf;//文本

string desword;//查找目标词

int times;//出现次数

shared\_ptr<set<size\_t>> linnum;//行号记录

QueryResult(shared\_ptr<vector<string>> tq, string s, int i, shared\_ptr<set<size\_t>> si) :txtbuf(tq), desword(s), times(i), linnum(si) {}

//返回目标文本

shared\_ptr<vector<string>> get\_file() {

return txtbuf;

}

set<size\_t>::iterator begin() {

return (\*linnum).begin();

}

set<size\_t>::iterator end() {

return (\*linnum).end();

}

};

QueryResult TextQuery::query(const string& word) {

static shared\_ptr <set<size\_t>>nodata(new set<size\_t>);

auto loc = word2line.find(word);

if (loc == word2line.end())

return QueryResult(textbuf, word, 0, nodata);

else

return QueryResult(textbuf, word, word2line[word].first, word2line[word].second);

}

//输出

ostream& print(ostream& out,const QueryResult& ans) {

out << ans.desword<< " occurs " << ans.times << ((ans.times > 1 )? " times " : " time " )<< endl;

for (auto w : \*ans.linnum) {

out << "\t( line " << w + 1 << " ) " << \*(ans.txtbuf->begin()+ w )<< endl;

}

return out;

}

void runQueries(ifstream& infile) {

TextQuery tq(infile);

while (true) {

cout << "Enter word to look for,or q to quit:";

string s;

if (!(cin >> s) || s == "q")break;

print(cout, tq.query(s))<< endl;

}

}

int main() {

ifstream inf("text.txt");

runQueries(inf);

return 0;

}