9.1答：（a）list（b）deque（c）vector

9.2答：list<deque<int>> ilist;

9.3答：左闭合区间

9.4答：typedef vector<int>::iterator iter;

bool find\_element(iter first, iter last, int i)

{

assert(first<=last);

for (iter it = first; it != last; ++it)

{

if (i == \*it)

return true;

}

return false;

}

9.5答：typedef vector<int>::iterator iter;

iter find\_element(iter first, iter last, int i)

{

assert(first<=last);

for (iter it = first; it != last; ++it)

{

if (i == \*it)

return it;

}

}

9.6答：迭代器不支持关系比较符< ，可以修改为！=

9.7答：vector<int>：：size\_type

9.8答：list<string>::iterator iter

9.9答：cbegin返回const迭代器

9.10答vr、vector<int>::const\_iterator、vector<int>::const\_iterator。

9.11答：

int ia[3] = {1,2,3};

vector<int> ival1(3);

vector<int> ivaector<int>::iterator、vector<int>::iteratol2(3, 1);

vector<int> ival3 = { 1, 2, 3 };

vector<int> ival4{1,2,3};

vector<int> ival5(ia, ia + 3);

vector<int> ival6(ival1.begin(),ival1.end());

vector<int> ival7(ival1);

9.12答：全部元素类型和容器类型必须相同，接受两个迭代器创建拷贝的构造函数不要求容器类型相同，元素类型可以进行转换，所以元素类型也可以不相同。

9.13答：

list<int> ilist1(3, 1);

vector<int> ilist2(4, 1);

vector<double> ival(ilist1.begin(),ilist1.end());

vector<double> ival2(ilist2.begin(),ilist2.end());

9.14答：

list<const char\*> names;

vector<string> oldstyle;

names.assign(oldstyle.begin(), oldstyle.end());

9.15答：判断两个容器是否相同

9.16答：判断两个容器中元素是否相同

9.17答：c1与c2的容器类型必须相同而且必须还支持<关系运算

9.18答：

#include<iostream>

#include<vector>

#include<string>

#include<deque>

using namespace std;

int main()

{

string word;

deque<string> svec;

while (cin >> word)

{

svec.push\_back(word);

}

for (auto iter = svec.begin(); iter != svec.end(); iter++)

{

cout << \*iter << endl;

}

return 0;

}

9.19答：包含头list头文件把deque改成list就可以了。

9.20答：

#include<iostream>

#include<vector>

#include<string>

#include<deque>

#include<list>

using namespace std;

int main()

{

list<int> ilist = {1,2,3,4,5,6,7,8,9};

deque<int> odd\_deque, even\_deque;

for (auto iter = ilist.begin(); iter != ilist.end(); ++iter)

{

if (\*iter % 2 == 0)

even\_deque.push\_back(\*iter);

else

odd\_deque.push\_back(\*iter);

}

for (auto iter = odd\_deque.begin(); iter != odd\_deque.end(); ++iter)

cout << \*iter << ' ';

cout << endl;

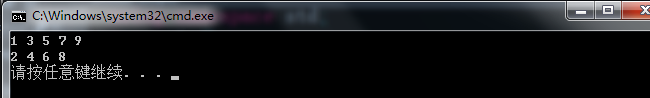
for (auto iter = even\_deque.begin(); iter != even\_deque.end(); ++iter)

cout << \*iter << ' ';

cout << endl;

return 0;

}



9.21答：第一次插入的时候初始化了iter，把读入的string插入到iter所指的元素之前的位置，其insert返回指向这个新插入元素的迭代器，每读入一次string，重复上述工作。

9.22答：

vector<int>::iterator iter = iv.begin(), mid = iv.begin() + iv.size / 2;

while (iter != mid)

{

if (\*iter == some\_val)

{

iv.insert(iter, 2 \* some\_val);

iter += 2;

}

else

++iter;

}

9.23答：肯定是相同的值。

9.24答：

#include<iostream>

#include<vector>

using namespace std;

int main()

{

vector<int> ivec{ 3, 1 };

if (!ivec.empty())

{

cout << ivec.at(0) << endl;

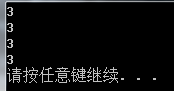
cout << ivec.front() << endl;

cout << \*ivec.cbegin() << endl;

cout << ivec[0] << endl;

}

}



9.25答：相等不会删除删除任何元素即使都是尾后迭代器。

int main()

{

vector<int> ivec{ 3, 1 };

ivec.erase(ivec.begin(),ivec.begin());

for (auto i : ivec)

cout << i << endl;

}

9.26答：

#include<iostream>

#include<vector>

#include<list>

using namespace std;

int main()

{

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

vector<int> ivec(ia, ia + 11);

list<int> ilist(ia, ia + 11);

for (auto iter = ilist.begin(); iter != ilist.end(); ++iter)

{

if (\*iter % 2 == 1)

{

iter = ilist.erase(iter);

iter--;

}

}

for (auto iter = ivec.begin(); iter != ivec.end(); ++iter)

{

if (\*iter % 2 == 0)

{

iter = ivec.erase(iter);

}

}

for (auto i : ilist)

cout << i << ' ';

cout << endl;

for (auto i : ivec)

cout << i << ' ';

cout << endl;

return 0;

}

9.27答：

#include<iostream>

#include<forward\_list>

using namespace std;

int main()

{

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

auto prev = iforward\_list.before\_begin();

auto curr = iforward\_list.begin();

while (curr != iforward\_list.end())

{

if (\*curr % 2)

{

curr = iforward\_list.erase\_after(prev);

}

else

{

prev = curr;

++curr;

}

}

for (auto i : iforward\_list)

cout << i << ' ';

cout << endl;

return 0;

}



9.28答：

#include<iostream>

#include<vector>

#include<string>

#include<deque>

#include<list>

#include<forward\_list>

using namespace std;

void my\_insert(forward\_list<string> &f, const string &s1,const string &s2)

{

auto prev = f.before\_begin();

auto curr = f.begin();

while (curr != f.end())

{

if (\*curr == s1)

{ f.insert\_after(curr, s2);

return;

}

else

{

prev = curr;

++curr;

}

}

f.insert\_after(prev,s2);

return;

}//这里必须有return，Google很好用！

int main()

{

forward\_list<string> my\_vector{ "chen", "xun", "is", "an","student" };

my\_insert(my\_vector, "anqq", "excellent");

my\_insert(my\_vector, "student", ",oh yeah!");

for (auto &a : my\_vector)

cout << a << ' ';

cout << endl;

return 0;

}

9.29答：将vec的大小改成100，然后初始化新添加的元素，vec.resize（10）将删除后面90个元素。

9.30答：容器类型必须支持默认初始化。

9.31答：

#include<iostream>

#include<vector>

#include<forward\_list>

using namespace std;

int main()

{

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

auto iter = vi.begin();

while (iter != vi.end())

{

if (\*iter % 2==1)

{

iter = vi.insert(iter, \*iter);

iter += 2;

}

else

iter = vi.erase(iter);

}

for (auto i : vi)

cout << i << ' ';

cout << endl;

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

auto prev = ilist.cbefore\_begin();

auto curr = ilist.begin();

while (curr != ilist.end())

{

if (\*curr % 2 == 0 )

{

curr = ilist.erase\_after(prev);

}

else

{

curr = ilist.insert\_after(curr, \*curr);

prev = curr;

++curr;

}

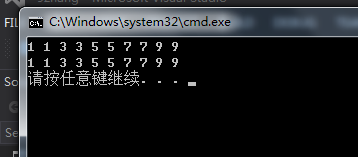
}

for (auto i : ilist)

cout << i << ' ';

cout << endl;

}



9.32答：不合法，当iter指向尾后迭代器的时候是非法的（未定义的行为）。

9.33答：未定义的行为。运行错误。

9.34答：当碰到第一个奇数的时候就发生无限循环。

9.35答：capacity表示容易在重新分配空间之前的可以容纳元素的个数，而seize表示容器当前所存储的元素的个数。

9.36答：capacity只能大于等于size。

9.37答：list不需要对内存重新分配，在插入元素的对对数据分配空间就可以，然后把数据链接到list中就可以了。

9.38答：push\_back、resize、capacity。

9.39答：将svec的capacity至少为1024，然后从输入设备添加元素，然后将其大小改成其读入的word个数的1.5倍。

9.40答：当读入256或512个数的时候capacity是1024.读入1000个的时候变成1536.读入1048的时候变成2034.

9.41答：

int main()

{

char c;

vector<char> cvec;

while (cin >> c)

cvec.push\_back(c);

string s(cvec.begin(), cvec.end());

cout << s << endl;

return 0;

}

9.42答：使用reserve分配至少100的空间内存。

9.43答：

9.47答：#include<iostream>

#include<string>

using namespace std;

int main()

{

string::size\_type pos = 0;

string numbers("0123456789");

string s1 = "ab2c3d7R4E6";

while ((pos = s1.find\_first\_of(numbers, pos)) != string::npos)

{

cout << pos << ' ' << s1[pos] << endl;

pos++;

}

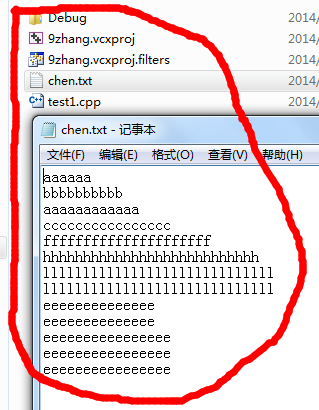
return 0;

}

9.48答：等于string最大的可能大小；在32为系统中4294967295

9.49答：

例子中所用chen.txt文件是



#include<iostream>

#include<string>

#include<fstream>

#include<vector>

using namespace std;

int main()

{

string::size\_type pos=0;

string word;

vector<string> svec;

string s1 = "bfghjlklpqty";

fstream myfile("chen.txt");

while (myfile >> word)

{

pos = word.find\_first\_of(s1);

if (pos == -1)

svec.push\_back(word);

}

for (auto i : svec)

cout << i << endl;

string::size\_type maxlen=0;

auto iter = svec.begin();

while (iter != svec.end())

{

auto theVal = (\*iter).size();

if (theVal > maxlen)

{

maxlen = theVal;

}

++iter;

}

cout << maxlen << endl;;

for (auto iter1 = svec.begin(); iter1 != svec.end();++iter1)

{

if ((\*iter1).size() == maxlen)

{

cout << \*iter1 << endl;

}

}

return 0;

}

9.51答：vector中存储是表示int型的字符串”123”、”123456”,那么就用stoi，如果存储的表示是浮点型就stod或者stof。

9.52答：合理运用字符串转数值的函数把表示日期的string转换成数值就可以了。设计其构造函数可以接受各种格式的string然后转换成数值。

9.53答：