Последовательные контейнеры

Во всех заданиях данной группы, кроме пяти начальных, заготовки решения уже содержат операторы, обеспечивающие заполнение исходных контейнеров и отладочную печать преобразованных контейнеров. Кроме того, заготовки содержат закомментированные операторы вывода полученных контейнеров в раздел результатов. Например, если в задании требуется преобразовать вектор целых чисел, то заготовка решения имеет следующий вид:

```
typedef ptin_iterator<int> ptin;
typedef ptout_iterator<int> ptout;
vector<int> V(ptin(0), ptin());

ShowLine(V.begin(), V.end(), "V: ");
//copy(V.begin(), V.end(), ptout());
```

Решение следует ввести перед оператором отладочной печати ShowLine, после чего раскомментировать вызов copy для вывода результатов и их автоматической проверки.

Во всех заданиях элементами контейнеров являются целые числа.

Последовательные контейнеры: заполнение и доступ к элементам. Обратные итераторы

STL2Seq1. Дан текстовый файл с именем *name*, содержащий строковые представления целых чисел. Заполнить вектор *V* числами из исходного файла и вывести элементы вектора в исходном порядке. Для заполнения вектора использовать итератор istream_iterator и конструктор вектора, для вывода элементов вектора использовать алгоритм copy, применяя его к итераторам полученного вектора и итератору ptout_iterator.

Указание. Для инициализации вектора `V` данными из файла `f` с применением итераторов ввода наиболее естественной представляется следующая конструкция:

```
vector<int> V(istream_iterator<int>(f),
  istream_iterator<int>());
```

Однако в силу особенностей лексического разбора языка C++ данная конструкция будет интерпретироваться как объявление прототипа функции `V` с двумя параметрами-указателями на функции. Простейшим вариантом исправления является заключение одного из параметров конструктора вектора в дополнительные круглые скобки. Можно также ввести вспомогательную переменную для одного или обоих итераторов:

```
istream_iterator<int> eof;
vector<int> data(istream_iterator<int>(f), eof);
```

или

```
istream_iterator<int> fi(f), eof;
vector<int> data(fi, eof);
```

STL2Seq2. Дан набор целых чисел. Заполнить список *L* исходными числами и вывести элементы списка *L* вначале в исходном, а затем в обратном порядке. Для заполнения списка использовать итератор ptin_iterator и конструктор списка, для вывода элементов списка дважды применить алгоритм copy к итераторам полученного списка и итератору ptout_iterator, причем для вывода элементов в обратном порядке использовать обратные итераторы, возвращаемые функциями-членами rbegin и rend.

Указание. Если при использовании конструктора итератора ptin_iterator указывать его параметр в виде константы, то проблемы, описанные в указании к задаче STL2Seq1, не возникают (если же используется переменная, то возникает аналогичная проблема, решить которую можно так, как описано в указании к задаче STL2Seq1, или превратив переменную в выражение: вместо `x` указать, например, `x + 0`).

STL2Seq3. Дан набор целых чисел с четным количеством элементов. Заполнить вектор *V* исходными числами и вывести вначале вторую половину элементов вектора *V*, а затем первую половину (в каждой половине порядок элементов не изменять). Для заполнения вектора использовать итератор ptin_iterator и конструктор вектора, для вывода элементов дважды применить алгоритм copy к итераторам полученного вектора и итератору ptout_iterator. При задании некоторых итераторов вектора использовать выражение `V.size()/2` и операцию «+».

STL2Seq4. Дан набор целых чисел с четным количеством элементов. Заполнить дек *D* исходными числами и вывести первую половину элементов дека *D* в обратном порядке, а затем — вторую половину (также в обратном порядке). Для заполнения дека использовать итератор ptin_iterator и конструктор дека, для вывода элементов дважды применить алгоритм copy к итераторам полученного дека и итератору ptout_iterator. Использовать обратные итераторы дека; при определении некоторых из них использовать выражение `D.size()/2` и операцию «+».

STL2Seq5. Дан набор целых чисел, количество которых делится на 3. Заполнить список *L* исходными числами и вывести вначале первую треть элементов списка *L* в исходном порядке, затем вторую треть элементов в обратном порядке, а затем последнюю треть (также в обратном порядке). Для заполнения списка использовать итератор ptin_iterator и конструктор списка, для вывода элементов трижды применить алгоритм copy к итераторам полученного списка и итератору ptout_iterator. Использовать как прямые, так и обратные итераторы; при определении некоторых из них использовать выражение `L.size()/3` и функцию advance.

STL2Seq6. Даны вектор *V*, дек *D* и список *L*. Каждый исходный контейнер содержит не менее трех элементов, количество элементов является нечетным. Удвоить значения первого, среднего и последнего элемента каждого из исходных контейнеров. Для доступа к первому и последнему элементу любого контейнера использовать функции-члены front и back. Для доступа к среднему элементу вектора и дека использовать операцию индексирования []. Для доступа к среднему элементу списка использовать функцию advance для итераторов и операцию разыменования итератора *.

STL2Seq7. Даны вектор *V*, дек *D* и список *L*. Каждый исходный контейнер содержит не менее двух элементов, количество элементов является четным. Поменять значения двух средних элементов каждого из исходных контейнеров. Использовать алгоритм swap (не путать его с одноименной функцией-членом контейнера).

Последовательные контейнеры: вставка элементов

STL2Seq8. Дан вектор *V* с четным количеством элементов. Добавить в середину вектора 5 нулевых элементов. Использовать один вызов функции-члена insert.

STL2Seq9. Дан дек *D* с нечетным количеством элементов *N* (≥ 5). Добавить в начало дека пять его средних элементов в исходном порядке. Использовать один вызов функции-члена insert.

STL2Seq10. Дан список *L*, количество элементов которого делится на 3. Добавить в конец списка первую треть его исходных элементов в обратном порядке. Использовать один вызов функции-члена insert.

STL2Seq11. Даны вектор *V* и список *L*. Каждый исходный контейнер содержит не менее 5 элементов. Вставить после элемента списка с порядковым номером 5 первые 5 элементов вектора в обратном порядке. Использовать один вызов функции-члена insert.

STL2Seq12. Даны дек *D* и список *L*. Каждый исходный контейнер содержит не менее 5 элементов. Вставить перед пятым с конца элементом списка последние 5 элементов дека в обратном порядке. Использовать один вызов функции-члена insert.

STL2Seq13. Даны вектор *V* и дек *D*, имеющие четное количество элементов. Добавить в конец вектора первую половину элементов дека (в исходном порядке), а в начало дека — вторую половину исходных элементов вектора (в обратном порядке). Использовать два вызова функции-члена insert.

STL2Seq14. Дан вектор *V*. Вставить после каждого элемента исходного вектора число −1. Использовать функцию-член insert в цикле с параметром-итератором.

Указание. Организуйте перебор элементов вектора в цикле с параметром-итератором `i`. Вставку выполняйте в позицию `++i`, обязательно присваивая параметру `i` значение, возвращаемое функцией-членом insert.

STL2Seq15. Дан дек *D* с четным количеством элементов. Вставить перед каждым элементом из первой половины исходного дека число −1. Использовать функцию-член insert в цикле с числовым параметром.

Указание. Используйте цикл с числовым параметром, повторяющийся *N*/2 раз (где *N* — исходный размер дека). Свяжите вспомогательный итератор `i` с началом второй половины элементов дека. В цикле выполняйте функцию-член insert для первого параметра, равного `--i`, обязательно присваивая возвращаемое значение итератору `i`.

STL2Seq16. Решить задачу STL2Seq15, используя функцию-член insert в цикле по обратному итератору.

Указание. Организуйте цикл for с параметром `r` — обратным итератором для перебора элементов первой половины исходного дека в обратном порядке. После вставки нового элемента функцией insert (с первым параметром `--r.base()`) следует использовать возвращаемое значение функции insert для обновления значения итератора `r`. Поскольку функция insert возвращает обычный итератор, необходимо выполнить явное приведение этого итератора к типу `vector<int>::reverse_iterator` (если компилятор поддерживает стандарт C++11, то в качестве имени типа можно указать `decltype(r)`). При такой организации цикла в его заголовке не следует указывать операцию инкремента `++r`.

STL2Seq17. Дан список *L*. Вставить перед каждым элементом исходного списка число −1. Использовать функцию-член insert в цикле с параметром-итератором.

Указание. См. указание к задаче STL2Seq14. Вставку следует выполнять в позицию `i`, причем не следует присваивать параметру `i` значение, возвращаемое функцией insert (иначе произойдет зацикливание). Вставка новых элементов в список (в отличие от вставки новых элементов в вектор или дек) не делает недействительными итераторы, указывающие на последующие элементы списка.

STL2Seq18. Дан список *L* с четным количеством элементов. Вставить после каждого элемента из первой половины исходного списка число −1. Использовать функцию-член insert в цикле с параметром-итератором.

Указание. См. указание к задаче STL2Seq15. Для задания начального значения итератора `i` используйте функцию advance. В данном случае первым параметром функции-члена insert должно быть выражение `i--`, причем сохранять значение, возвращенное функцией insert, не требуется.

STL2Seq19. Решить задачу STL2Seq18, используя функцию-член insert в цикле по обратному итератору.

Указание. См. указание к задаче STL2Seq16. Для определения диапазона итераторов используйте функцию advance. В данном случае первым параметром функции-члена insert должно быть выражение `r.base()`. В случае списков в результате подобной вставки обратный итератор `r` будет возвращать значение вставленного элемента. Если, как в задаче STL2Seq16, сохранять возвращаемое значение функции insert в итераторе `r` (с применением явного приведения типа), то в результате этого итератор `r` будет возвращать значение элемента, предшествующего вставленному (т. е. расположенного слева от вставленного). Поэтому в конце итерации цикла необходимо дополнительно выполнить операцию `++r` (в отличие от решения задачи STL2Seq16). В данной программе, в отличие от решения задачи STL2Seq16, можно не сохранять значение, возвращаемое функцией insert, но в этом случае в конце цикла требуется дважды инкрементировать итератор `r` (например, указывая в заголовке цикла выражение `++r`, `++r`).

Последовательные контейнеры: удаление элементов

STL2Seq20. Дан дек *D* с нечетным количеством элементов *N* (≥ 3). Удалить средний элемент дека. Использовать функцию-член erase.

STL2Seq21. Дан вектор *V* с нечетным количеством элементов *N* (≥ 5). Удалить три средних элемента вектора. Использовать один вызов функции-члена erase.

STL2Seq22. Даны список *L* и вектор *V*; список *L* имеет нечетное количество элементов. Переместить средний элемент списка *L* в конец вектора *V*. Использовать функции-члены push_back и erase.

STL2Seq23. Даны список *L* и дек *D*; дек *D* имеет четное количество элементов. Переместить первую половину элементов дека *D* в начало списка *L*. Использовать один вызов функций-членов insert и erase.

STL2Seq24. Даны списки L_1 и L_2; список L_1 имеет нечетное количество элементов. Переместить средний элемент списка L_1 в конец списка L_2. Использовать один вызов функции-члена splice.

STL2Seq25. Даны списки L_1 и L_2; список L_2 имеет четное количество элементов. Переместить первую половину элементов списка L_2 в начало списка L_1. Использовать один вызов функции-члена splice.

STL2Seq26. Даны списки L_1 и L_2, имеющие четное количество элементов. Поменять местами первую половину исходного списка L_1 и вторую половину исходного списка L_2. Использовать два вызова функции-члена splice.

STL2Seq27. Дан вектор *V*. Удалить все элементы исходного вектора с нечетными порядковыми номерами (считая, что начальный элемент вектора имеет порядковый номер 1). Использовать функцию-член erase в цикле с параметром-итератором.

Указание. Организуйте перебор элементов вектора в цикле с параметром-итератором `i`, увеличивая параметр в заголовке цикла. Удаление выполняйте в позиции `i`, обязательно присваивая параметру `i` значение, возвращаемое функцией-членом erase. В конце каждой итерации цикла дополнительно проверяйте, достигнут ли конец вектора.

STL2Seq28. Дан дек *D* с количеством элементов, кратным 4. Удалить в первой половине исходного дека все элементы с четными порядковыми номерами (считая, что начальный элемент дека имеет порядковый номер 1). Использовать функцию-член erase в цикле с числовым параметром.

Указание. Используйте цикл с числовым параметром, повторяющийся *N*/4 раз (где *N* — исходный размер дека). Свяжите вспомогательный итератор `i` с началом дека. В цикле

выполняйте функцию-член erase с параметром, равным `++i`, обязательно присваивая возвращаемое значение итератору `i`.

STL2Seq29. Решить задачу STL2Seq28, используя функцию-член erase в цикле по обратному итератору.

Указание. Организуйте цикл с параметром `r` — обратным итератором для перебора элементов первой половины исходного дека в обратном порядке (параметр `r` должен увеличиваться в заголовке цикла). После удаления элемента функцией-членом erase (с параметром `--r.base()`) следует обязательно использовать возвращаемое значение функции erase для обновления значения итератора `r`. Поскольку функция erase возвращает обычный итератор, необходимо выполнить явное приведение этого итератора к типу `deque<int>::reverse_iterator` (если компилятор поддерживает стандарт C++11, то в качестве имени типа можно указать `decltype(r)`).

STL2Seq30. Дан список L. Удалить все элементы исходного списка с четными порядковыми номерами (считая, что начальный элемент списка имеет порядковый номер 1). Использовать функцию-член erase в цикле с параметром-итератором.

Указание. См. указание к задаче STL2Seq27. В данном случае в заголовке цикла по итератору `i` не нужно использовать операцию инкремента, а удаление следует выполнять в позиции `++i`, присваивая параметру `i` значение, возвращаемое функцией-членом erase.

STL2Seq31. Дан список L с количеством элементов, кратным 4. Удалить в первой половине исходного списка все элементы с нечетными порядковыми номерами (считая, что начальный элемент списка имеет порядковый номер 1). Использовать функцию-член erase в цикле с параметром-итератором.

Указание. См. указание к задаче STL2Seq28. В данном случае параметром функции-члена erase должно быть выражение `i++`, причем сохранять значение, возвращенное функцией erase, не следует; вместо этого в конце каждой итерации требуется выполнять дополнительный инкремент: `i++`.

STL2Seq32. Решить задачу STL2Seq31, используя функцию-член erase в цикле по обратному итератору.

Указание. См. указание к задаче STL2Seq29. Для определения диапазона итераторов используйте функцию advance. В данном случае в качестве параметра функции-члена erase можно использовать выражение `--(++r).base()`, причем сохранять значение, возвращаемое функцией erase, не требуется (не требуется также выполнять инкремент итератора `r` в заголовке цикла).

STL2Seq33. Дан список L с элементами A_1, A_2, A_3, …, A_{N-1}, A_N (N — четное). Изменить порядок элементов в списке на следующий: A_N, A_1, A_{N-1}, A_2, A_{N-2}, …, $A_{N/2}$, $A_{N/2-1}$. Для этого использовать два итератора `i` и `r`, связав их с первым и последним элементом списка. В цикле, который должен повторяться $N/2$ раз, вызывать функцию-член splice с первым параметром `i++` и третьим параметром `r--`.

STL2Seq34. Даны два списка L_1 и L_2 с одинаковым количеством элементов N. Получить в списке L_2 комбинированный набор элементов исходных списков вида A_1, B_1, A_2, B_2, A_3, B_3, …, A_N, B_N, где A_I обозначают элементы исходного списка L_1, а B_I — элементы исходного списка L_2. Для этого использовать итераторы `i1` (для списка L_1) и `i2` (для списка L_2), связав их с первым элементом соответствующего списка. В цикле, который должен повторяться N раз, вызывать функцию-член splice для списка L_2 с первым параметром `++i2` и третьим параметром `i1++`.