Итераторы

Общее описание

В библиотеке STL используются пять основных видов итераторов:

- итераторы чтения,
- итераторы записи,
- однонаправленные итераторы,
- двунаправленные итераторы,
- итераторы произвольного доступа.

Для каждого вида итераторов определен набор доступных операций, причем двумя операциями, доступными для всех видов итераторов, являются *операция инкремента* ++, которая передвигает итератор *p* на следующий элемент последовательности (*++p* и *p++*), и *операция разыменования* *, возвращающая значение текущего элемента (**p* и вариант *p->m* для доступа к члену *m* разыменованного объекта).

Операция разыменования имеет следующие особенности:

- в случае итераторов чтения операция * не может использоваться для изменения элемента;
- в случае итераторов записи операция * не может использоваться для получения значения элемента (выражение **p* можно использовать только в левой части присваивания)
- для прочих итераторов операция * может использоваться как для получения значения элемента, так и для изменения этого значения.

Операции сравнения итераторов на равенство == и != реализованы для всех итераторов, кроме итераторов записи.

Для однонаправленных итераторов не определяются новые операции (по сравнению с итераторами чтения или записи).

Для двунаправленных итераторов в дополнение к операции инкремента ++ вводится *операция декремента* -- (также в двух видах: *--p* и *p--*).

Наконец, для итераторов произвольного доступа добавляются *операция индексирования* [], позволяющая сразу обратиться к элементу последовательности с требуемым индексом (*p*[*i*]), и *операция смещения на указанное количество элементов*, причем в оба направления (*p+i* и *p-i*). Определена также *операция разности двух итераторов*, позволяющая определить расстояние между элементами, с которыми они связаны (*p2-p1*).

Таким образом, набор операций для итераторов произвольного доступа аналогичен набору операций для обычных указателей.

☑ Для итераторов, не являющихся итераторами произвольного доступа, также можно выполнять действия, связанные со смещением и определением расстояния, используя функции из заголовочного файла <*iterator*>:

- *advance*(*p, n*) — передвигает итератор *p* на *n* позиций вперед (*n* >= 0); для двунаправленного итератора можно использовать n < 0 для перемещения назад;
- *distance*(*p1, p2*) — возвращает расстояние между итераторами *p1* и *p2* (в предположении, что расстояние неотрицательно).

В качестве итераторов чтения и итераторов записи можно использовать итераторы всех остальных видов (однонаправленные, двунаправленные, произвольного доступа). В качестве однонаправленных итераторов можно использовать двунаправленные итераторы и итераторы произвольного доступа. В качестве двунаправленных итераторов можно использовать итераторы произвольного доступа.

Два итератора обычно используются для задания *диапазона элементов*, при этом предполагается, что первый итератор (*first*) указывает на начальный элемент диапазона, а второй итератор (*last*) указывает на позицию *за* конечным элементом диапазона (причем эта позиция может не быть связана с существующим элементом). Чтобы подчеркнуть отмеченные особенности для диапазонов, определяемых итераторами, они часто записываются в виде полуинтервала вида [*first, last*) (левая граница диапазона включается, правая — нет). Интервал вида [*first, first*) не содержит ни одного элемента.

Для всех видов итераторов определены их модификации — *константные итераторы*, отличающиеся от обычных тем, что их разыменование дает константное значение.

Особыми итераторами являются итераторы потоков ввода-вывода (рассматриваемые в группе STL1Iter), обратные итераторы (рассматриваемые в группе STL2Seq) и итераторы вставки (рассматриваемые в группе STL3Alg). Особенности обратных итераторов рассматриваются в разделе, посвященном контейнерам, а особенности итераторов вставки — в разделе, посвященном алгоритмам.

Итераторы, определенные в задачнике Programming Taskbook

Итераторы чтения и записи последовательности (*ptin_iterator*<*T*> и *ptout_iterator*<*T*>) обладают теми же свойствами, что и стандартные потоковые итераторы *istream_iterator*<*T*> и *ostream_iterator*<*T*>. Перечислим свойства итератора чтения *ptin_iterator*<*T*>, совпадающие с аналогичными свойствами потоковых итераторов:

- тип *T* определяет тип элементов последовательности;
- чтение элемента последовательности из потока *pt* выполняется в начальный момент работы с итератором, а затем при каждой операции инкремента ++;
- имеются два варианта операции ++: префиксный инкремент (*++p*) и постфиксный инкремент (*p++*);
- операция * возвращает последнее прочитанное значение, причем эту операцию можно использовать неоднократно для получения того же самого значения;
- итератор конца последовательности создается с помощью конструктора без параметров;
- при достижении конца последовательности итератор становится равным итератору конца последовательности, последующие вызовы операции инкремента игнорируются, а в результате вызова операции * всегда возвращается значение последнего прочитанного элемента последовательности.

Свойства итератора записи *ptout_iterator*<*T*>, совпадающие с аналогичными свойствами потоковых итераторов:

- специальный конструктор для создания итератора конца последовательности не предусмотрен;
- операции * и ++ не выполняют никаких действий и просто возвращают сам итератор;
- операция присваивания *p = выражение* (где *p* – имя итератора записи) записывает значение выражения в поток *pt*.

В отличие от стандартного потокового итератора *istream_iterator*, конструктор для итератора *ptin_iterator* (за исключением конструктора итератора конца последовательности) имеет один параметр типа *unsigned int*, определяющий размер считываемой последовательности (в элементах). В случае особого значения параметра, равного 0, размер последовательности считывается из самого потока *pt* непосредственно перед считыванием первого элемента последовательности. Если при считывании размера оказывается, что прочитанный элемент данных не является целым числом или это число не является положительным, то итератор сразу переходит в состояние конца последовательности (т. е. становится равным итератору *ptin_iterator*<*T*>()).

Подобная организация итераторов чтения *ptin_iterator*<*T*> позволяет легко реализовывать с их помощью считывание из потока ввода *pt* нескольких последовательностей, если заранее известен их размер или (как во всех заданиях, входящих в задачник Programming Taskbook for STL) если размер указывается непосредственно перед началом последовательности.