STL: стандартная библиотека шаблонов C++

Коллекции

Для начала рассмотрим самые популярные коллекции из библиотеки. Для использования коллекции в своем коде используйте следующую директиву: `#include <T>`, где T — название коллекции. Итак, наиболее часто используются (* отмечены те, что мы будем реализовывать в этом семестре):

- `std::vector*` — коллекция элементов, сохраненных в массиве, изменяющегося по мере необходимости размера (обычно, увеличивающегося);
- `std::list*` — коллекция, хранящая элементы в виде двунаправленного связанного списка;
- `std::map` — коллекция, сохраняющая пары вида <const Key, T>, т.е. каждый элемент — это пара вида <ключ, значение>, при этом однозначная (каждому ключу соответствует единственное значение), где ключ — некоторая характеризующая значение величина, для которой применима операция сравнения; пары хранятся в отсортированном виде, что позволяет осуществлять быстрый поиск по ключу, но за это, естественно, придется заплатить: придется так реализовывать вставку, чтобы условие отсортированности не нарушилось;
- `std::set*` — это отсортированная коллекция одних только ключей, т.е. значений, для которых применима операция сравнения, при этом уникальных — каждый ключ может встретиться во множестве только один раз;
- `std::stack*` — контейнер, в котором добавление и удаление элементов осуществляется с одного конца;
- `std::queue*` — контейнер, с одного конца которого можно добавлять элементы, а с другого — вынимать;
- `std::priority_queue*` — очередь с приоритетом, организованная так, что самый большой элемент всегда стоит на первом месте;
- `std::string` — коллекция однобайтных символов в формате ASCII;
- `std::wstring` — коллекция двухбайтных символов в формате Unicode; включается командой `#include <xstring>`.

Методы коллекций

Основными методами, присутствующими почти во всех коллекциях являются следующие: empty — определяет, пуста ли коллекция; size — возвращает размер коллекции; begin — возвращает прямой итератор, указывающий на начало коллекции; end — возвращает прямой итератор, указывающий на конец коллекции, т.е. на несуществующий элемент, идущий после последнего; rbegin — возвращает обратный итератор на начало коллекции; rend — возвращает обратный итератор на конец коллекции; clear — очищает коллекцию, т.е. удаляет все ее элементы; erase — удаляет определенные элементы из коллекции.

Вектор

Самая часто используемая коллекция — это вектор. Очень удобно, что у этой коллекции есть такой же оператор operator [], что и у обычного массива. Такой же оператор есть и у коллекций map, deque, string и wstring.

Важно понимать, что вместимость vector'a изменяется динамически. Обычно для увеличения размера используется мультипликативный подход: выделенная под vector память увеличивается при необходимости в константное число раз, т.е. если добавление нового элемента приведет к тому, что размер массива превысит вместимость, то операционной системой для программы будет выделен новый участок памяти, например, в два раза больший, в который будут скопированы все значения из старого участка памяти и к которому будет дописано новое значение.

Подробнее о методах: https://ru.wikipedia.org/wiki/Vector_(C%2B%2B)

Задача 1. Применение шаблонного вектора.

1. Объявить вектор целых чисел на 20 элементов, заполненный сразу нулями (погуглить какие конструкторы есть у вектора, гугл при работе с шаблонами лучший ваш друг, можно обратиться к ссылке выше).
2. Заполнить вектор случайно (от -100 до 100), параллельно выводя его на экран (использовать итератор).
3. Вывести максимальный и минимальный элементы.
4. Отсортировать вектор.
5. Добавить 4 любых элемента в конец вектора.
6. Заменить элементы меньшие 10 на ноль.
7. Найти и вывести элементы больше 20.
8. Все чётные элементы домножить на 3.
9. Произвольно перемешать элементы.
10. Удалить элементы большие 50.
11. Вывести количество элементов в векторе. Если количество нечётное – удалить последний элемент, иначе – напечатать элементы в обратном порядке (реверсивный итератор).
12. Очистить вектор.

Проверить, очистился ли он, вывести сообщение о пустоте, если да. Иначе очистить.

Список

Контейнер list представляет двухсвязный список. Для его использования необходимо подключить заголовочный файл list. Для контейнера list можно использовать функции front() и back(), которые возвращают соответственно первый и последний элементы.

Чтобы обратиться к элементам, которые находятся в середине (после первого и до последнего элементов), придется выполнять перебор элементов с помощью циклов или итераторов:

```
#include <iostream>
#include <list>

int main() {
    std::list<int> numbers = { 1, 2, 3, 4, 5 };

    int first = numbers.front();   // 1
    int last = numbers.back();     // 5

    // перебор в цикле
    for (int n : numbers)
        std::cout << n << " ";
    std::cout << std::endl;

    // перебор с помощью итераторов
    for (auto iter = numbers.begin(); iter != numbers.end(); iter++) {
        std::cout << *iter << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

Для добавления элементов в контейнер list применяется ряд функций.

- push_back(val): добавляет значение val в конец списка
- push_front(val): добавляет значение val в начало списка
- emplace_back(val): добавляет значение val в конец списка
- emplace_front(val): добавляет значение val в начало списка

- emplace(pos, val): вставляет элемент val на позицию, на которую указывает итератор pos. Возвращает итератор на добавленный элемент
- insert(pos, val): вставляет элемент val на позицию, на которую указывает итератор pos, аналогично функции emplace. Возвращает итератор на добавленный элемент
- insert(pos, n, val): вставляет n элементов val начиная с позиции, на которую указывает итератор pos. Возвращает итератор на первый добавленный элемент. Если n = 0, то возвращается итератор pos.
- insert(pos, begin, end): вставляет начиная с позиции, на которую указывает итератор pos, элементы из другого контейнера из диапазона между итераторами begin и end. Возвращает итератор на первый добавленный элемент. Если между итераторами begin и end нет элементов, то возвращается итератор pos.
- insert(pos, values): вставляет список значений values начиная с позиции, на которую указывает итератор pos. Возвращает итератор на первый добавленный элемент. Если values не содержит элементов, то возвращается итератор pos.

Про разницу в «одинаковых» методах: http://candcplusplus.com/c-difference-between-emplace_back-and-push_back-function

Для удаления элементов из контейнера list могут применяться следующие функции:

- clear(p): удаляет все элементы.
- pop_back(): удаляет последний элемент.
- pop_front(): удаляет первый элемент.
- erase(p): удаляет элемент, на который указывает итератор p. Возвращает итератор на элемент, следующий после удаленного, или на конец контейнера, если удален последний элемент.
- erase(begin, end): удаляет элементы из диапазона, на начало и конец которого указывают итераторы begin и end. Возвращает итератор на элемент, следующий после последнего удаленного, или на конец контейнера, если удален последний элемент.

Другие методы: https://learn.microsoft.com/ru-ru/cpp/standard-library/list-class?view=msvc-170

Задача 2. Применение шаблонного списка.

1. Объявить список. Изначально он будет пустой.
2. Добавить 5 элементов в конец списка.
3. Вывести на экран элемент из начала списка.
4. Добавить в начало списка 2 элемента.
5. Удалить 4-й элемент.
6. Вставить на 3 случайные позиции случайный элемент (одинаковый).
7. Удалить последний элемент списка.
8. Удалить элемент из начала списка.
9. Вставить 2 элемента в середину списка (2 сразу).
10. Удалить повторяющиеся элементы списка.
11. Очистить список.
12. Проверить, что список пуст. Вывести сообщение, если он пуст.

Стек

Стек — это структура данных, которая работает по принципу FILO (first in — last out; первый пришел — последний ушел). В C++ уже есть готовый шаблон — stack. В стеке элемент, который вошел самый первый — выйдет самым последним. Получается, если вы добавили три элемента в стек первым будет удален последний добавленный элемент.

Стеки имеют некоторые ассоциируемые методы: push — добавить элемент в стек; pop — удалить элемент из стека; peek — просмотреть элементы стека.

Задача 3. Написать программку, которая предлагает набор действий (вспоминаем меню) со стеком:
- положить элемент (пользователь задаёт элемент)
- забрать элемент (пишется сообщение «Элемент % был удалён из стека.»)
- напечатать содержимое стека в виде

```
|     |
|  6  |
|  5  |
|  8  |
|  3  |
|  1  |
---------
```

- очистить стек
- выход

Множество

set — это контейнер, который автоматически сортирует добавляемые элементы в порядке возрастания. Но при добавлении одинаковых значений, set будет хранить только один его экземпляр. По другому его еще называют множеством.

Задача 3. Написать программку, которая предлагает набор действий (вспоминаем меню) со стеком:
- задать множество A
- задать множество B
- найти пересечение заданных множеств
- найти объединение заданных множеств
- найти разность множеств A - B
- найти дополнение множества (далее уточняется для какого множества)
- выход