

Контрольные вопросы. Задание 14.

17 февраля 2023 г.

1 В каких ситуациях используются типы `std::pair` и `std::tuple`?

Тип `std::pair` удобен для хранения пар ключ-значение. При этом элементы могут быть разных типов. Хранение таких пар необходимо, например, при использовании ассоциативных массивов). `std::pair` используется 1) классами `map`, `multimap` и т.д.; 2) функциями, возвращающими два значения). Т.о., `std::pair` используется всюду, где два значения необходимо интерпретировать как одно целое.

Тип `std::tuple` (кортеж) расширяет концепцию пары на произвольное количество элементов (с помощью вариативных шаблонов), т.е. кортежи – неоднородные списки элементов типы которых задаются или выводятся во время компиляции. При этом кортеж не является обычным контейнерным классом, в котором можно осуществлять обход элементов.

2 Когда следует использовать контейнер `std::array`?

Контейнер `std::array` следует использовать для хранения массива однотипных данных фиксированного размера (т.е. размер массива должен быть известен на этапе компиляции) с произвольным доступом при необходимости относительно быстрой работы с данными (память для данных выделяется в "быстром" стеке). В целом, `std::array`, формально говоря, является более безопасной заменой встроенного массива `[]-array` (например, метод `array.at(index)` является более безопасным, чем операция `array[index]`).

3 Когда следует использовать контейнер `std::vector`?

Контейнер `std::vector` управляет однотипными элементами, хранящимися в динамическом массиве (аналог `new[]`-массива), обеспечивая произвольный доступ к элементам. Добавление и удаление элементов происходит в конце массива и выполняется очень быстро. Но вставка в начало/в середину производится достаточно долго ($O(n)$). Вектор "следует использовать по умолчанию когда заранее неизвестно количество данных и не определено время работы структуры. Он соблюдает требования идиомы RAII и основы ООП. Кроме того, в векторе данные хранятся непрерывно (благодаря чему можно использовать арифметику указателей, что уже не справедливо, например, для дека).

4 Когда следует использовать контейнер `std::deque`?

Контейнер `std::deque` (двусторонняя очередь, дек) (является "двунаправленным" динамическим массивом), следует использовать, когда необходимо большое количество добавлений элементов (однотипных) в начало или конец контейнера (они будут выполняться за $O(1)$ благодаря "страничной" структуре памяти). При этом вставка элемента в середину очереди может потребовать значительно больше времени.

5 Когда следует использовать контейнер `std::list`?

Контейнер `std::list` – двусвязный список (возможно итерирование в двух направлениях), который следует использовать, когда есть необходимость в быстром осуществлении вставки или удаления элементов (однотипных) с любой позиции (за $O(1)$). В то же время `std::list` не поддерживает произвольный доступ (лишь при итерировании по контейнеру – за линейное время).

6 Когда следует использовать контейнер `std::forward_list`?

Контейнер `std::forward_list` – односвязный список (в отличие от двусвязного каждый элемент ссылается только на следующий, а последний – на `nullptr`), являющийся сильно ограниченным (не реализованы даже `push_back()` и `size()`) и в то же время очень экономным по памяти контейнером, чем и может быть обусловлена области использования односвязных списков. Работа с данным контейнером осуществляется с помощью специальных функций-членов.

7 Какие адаптеры контейнеров есть в стандартной библиотеке?

Адаптеры контейнеров STL приспособливают стандартные контейнеры для особых целей. В соответствии с этим существуют такие адаптеры как:

- Стек (LIFO).
- Очередь (FIFO).
- Очередь с приоритетами (FIFO + sort).

8 Когда следует использовать контейнер `circular buffer` из Boost?

Контейнер `circular buffer` из Boost следует использовать для хранения истории фиксированной длины, когда непрерывно поступают новые данные: после заполнения всех ячеек контейнера, новые данные начинают перезаписываться с начала (что делает функционирование циклического буфера довольно быстрым). Часто `circular buffer` справляется с задачами быстрее, чем `std::list` или `std::deque`. В остальном кольцевой список предлагает возможности, аналогичные `std::queue`.

9 Почему контейнер `circular buffer` из Boost не может войти в стандарт?

Контейнер `circular buffer` из Boost не может войти в стандарт, т.к последний включает часто используемые алгоритмы, структуры данных, механизмы ввода и вывода и т.д., а `circular buffer` не является одним из часто используемых контейнеров.

Тем не менее существует предложение добавить кольцевой список в стандартную библиотеку (ссылка [3] в списке литературы). Отмечается, что авторы предложения ищут отзывы о дизайне кольца, прежде чем представить формулировку стандарта. В пункте 7 предложения есть абзац:

The popper class templates are defined at an overly broad scope, rather than in the scope of the `ring_span`. However, no way of doing this is immediately apparent, beyond the obvious solution of creating a ring namespace and defining the poppers and the span inside it. **Since this is somewhat counterintuitive in the context of the remainder of the standard library**, the authors remain open to suggestions. If the popper class templates might have use in other container spans, then they could remain in the broader scope.

Этот также фрагмент свидетельствует необходимости дальнейшей проработки (выделил полужирным шрифтом) дизайна кольцевого списка.

10 Какие типы данных для работы с многомерными массивами вы можете назвать?

Данные в многомерных массивах можно хранить, используя:

- `boost::multi_array`;
- `std::valarray`;
- Кроме того можно составить контейнер из контейнеров (или массив из массивов) с помощью?
 - `std::vector`;
 - `std::array`;
 - `std::deque`;
 - встроены `[][]`-массивы.

Кроме того, при работе с многомерными массивами (и одномерными) оказываются полезными итераторы и указатели.

Литература.

- [1] Конспект семинара. Макаров И.С.
- [2] Standard library. Dzhosattis N.
- [3] <chrome-extension://efaidnbmnnnibpcajpcgclefindmkaj/https://openstd.org/JTC1/SC22/WG21/docs/papers/2017/p0059r3.pdf>
- [4] <https://www.codeproject.com/Articles/1185449/Performance-of-a-Circular-Buffer-vs-Vector-Deque-a>