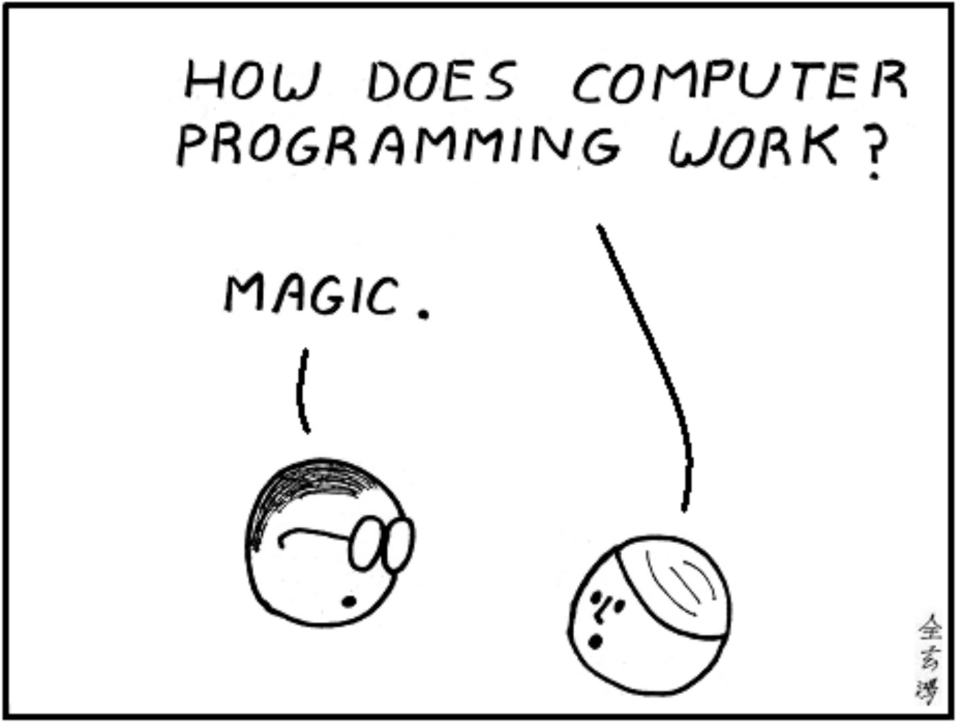
HOW DOES COMPUTER PROGRAMMING WORK?
MAGIC.

ООП/C++: Лекция 7

Введение в STL (vector, iterator, algorithm)

ака «Полезные кубики для изготовления задания или проекта»

О чём лекция сегодня

1. Что такое STL (на пальцах)
2. Как жить без realloc (vector)
3. Немного страшного синтаксиса (iterator)
4. Немного повседневного STL (algorithm и vector)

Код на сегодня

https://github.com/avasyukov/oop-2nd-term/tree/master/2019/lection07

Контест к лекции

http://judge2.vdi.mipt.ru/cgi-bin/new-client?contest_id=911143

Что такое STL (на пальцах)

Пара слов об STL

- Возник как отдельная библиотека с типовыми примитивами для программ на C++
- Постепенно стал частью стандарта C++
- Позволяет меньше изобретать велосипеды
- Очень обширная и глубокая тема, которую можно отдельно изучать пару семестров
- Реализация опирается на шаблоны (templates), но использовать STL для простых задач можно и без их особого понимания

Пара слов об STL

Disclaimer: в рамках этой лекции STL обсуждается на уровне набора полезных кубиков (исходно он так и возник) и аргументации «интуитивно очевидно что»

Что содержит STL

- Контейнеры — примитивы для хранения наборов объектов
- Итераторы — средство доступа к данным в контейнерах произвольного вида
- Алгоритмы — наиболее типовые операции над данными в контейнерах
- Адаптеры — обёртки для обеспечения наиболее употребляемых интерфейсов (например, стек и очередь)
- Функторы — конструкции, позволяющие использовать объект как функцию (да, звучит непонятно)

Как жить без realloc (vector)

Как жить без realloc

Типовой контейнер из состава STL по имени vector:

- Логически примерно равен динамическому массиву на базе malloc/realloc
- Хранит данные произвольного типа (как именно — будет в лекции про шаблоны)
- Обеспечивает изменение размера выделенной памяти по мере необходимости

vector: примеры

Разбираем примеры

- 01_vector_basics.cpp

vector: реализация под капотом

- Вектор должен фоново и «как-нибудь сам» обеспечить изменение объёма памяти
- Это вызывает необходимость делать новый malloc или realloc
  - Что будет с асимптотической сложностью?
  - Что будет с адресами элементов?

vector: примеры

Разбираем примеры

- 02_vector_internals_capacity.cpp
- 03_vector_internals_reallocation.cpp

vector: нюансы

Где можно поймать проблем:

- При использовании «в лоб» памяти расходуется больше, чем необходимо (до 2-х раз)
- При динамическом изменении может неожиданно перемещаться по памяти (получение pointer-ов и reference-ов на элементы vector-а опасно)

Проблемы лечатся resize-ом, его использование правда важно, если vector используется для нагруженных задач.

Немного страшного синтаксиса (iterator)

Итератор: откуда он взялся

- Когда у нас есть какие-нибудь данные, часто хочется их все перебрать
- Для «плоской» структуры интуитивно хочется использовать указатель

А для неплоской? (дерево, хэш-таблица и т.д.)

Итератор: что это

Итератор:

- Логически — сущность вида «служебный объект для обхода контейнера с данными»
- В простейшем случае технически равен указателю
- Для сложных контейнеров может быть реализован очень небанально внутри

Позволяет отделить алгоритм от структуры, в которой хранятся данные (затем и придуман)

Итератор: примеры

Разбираем примеры (на примере итераторов для vector и map):

- 04_hello_iterator.cpp
- 05_iterator_over_map.cpp
- 06_const_iterator.cpp

Немного повседневного STL (algorithm и vector)

STL: алгоритмы

- В STL реализация алгоритмов отделена от реализации контейнеров данных (и это прекрасно) – в том числе для этого нужны итераторы
- Готовых типовых алгоритмов довольно много – мы сейчас смотрим только примеры, не «все» и даже не «самые главные»
- Если возникла конкретная «низкоуровневая» задача, которая кажется типовой – загляните в документацию, возможно, есть готовый «кубик»

algorithm и vector: примеры

Разбираем примеры:

- 07_stl_algorithms_examples.cpp
- 08_more_stl_examples.cpp
- 09_even_more_stl_examples.cpp

Что стоит запомнить из лекции

- Есть STL – набор типовых примитивов, которым можно и нужно пользоваться
- Свои классы стоит писать так, чтобы они вписались в контейнеры и алгоритмы STL
- Для массива переменного размера можно использовать vector, который делает работу realloc-а под капотом и незаметно для вас (но есть нюансы)
- Есть такая штука iterator, которая немного похожа на указатель, но нужна для единообразного обхода произвольных структур, в том числе не плоских (дерева, хэш-таблицы и т.д.)

Мнение о лекции

https://tinyurl.com/y4m8qqo3


Province of Value Queries
count
all_of
any_of
none_of
Province of Property Queries
transform_inclusive_scan
Valley of Partial Sums
transform_exclusive_scan
inclusive_scan
exclusive_scan
partial_sum
Province of 2-ranges properties
set_difference
set_symmetric_difference
equal
mismatch
lexicographical compare
is_permutation
Glorious County of Algos on Sets
set_intersection
set_union
includes
merge
for_each
transform
Lonely Islands
uninitialized_value_construct
uninitialized_default_construct
uninitialized_copy
uninitialized_fill
uninitialized_move
destroy
copy_backward
Lands of Queries
find
adjacent_find
transform_reduce
Value Searchers
Lagoon of Folds
reduce
equal_range
upper_bound
inner_product
accumulate
binary_search
lower_bound
sample
Peninsula of Raw Memory
Territory of Movers
copy
move
swap_ranges
move_backward
adjacent_difference
Province of Research
max_element
min_element
Bay of Range Searches
find_end
Relative Value Searchers
search
minmax_element
find_first_of
remove
unique
Island of Structure Changers
iota
fill
generate
Land of Value Modifiers
replace
Secret Runes
partition_point
sort
Shore of Sorting
partial_sort
partition
reverse
sort_heap
nth_element
Lands of Permutations
rotate
make_heap
inplace_merge
pop_heap
next_permutation
shuffle
push_heap
prev_permutation
Province of Heaps
random_shuffle (wreck)
stable_*
*_copy
*_n
*_if
is_*_until
is_*
The World of C++ STL Algorithms