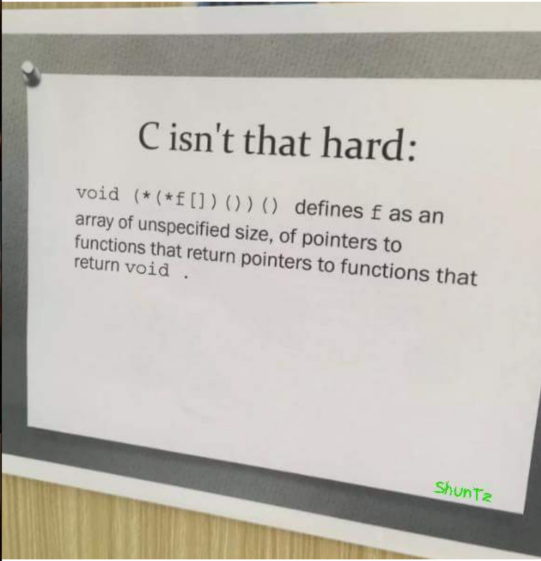


Сложности при наследовании

ООП/C++: Лекция 5

Память, модификатор const, потоки ввода-вывода

ака «Повседневные нюансы: почти как в C, но по-новому»

 CC-BY-SA 4.0

О чём лекция сегодня

1. Как жить без malloc / free
2. Pointer vs Reference
3. Многоликий const
4. Потоки ввода-вывода

Код на сегодня

https://github.com/avasyukov/oop-2nd-term/tree/master/2019/lection05

Контест к лекции

http://judge2.vdi.mipt.ru/cgi-bin/new-client?contest_id=911141

Как жить без malloc / free

Выделение памяти

- В коде на C++ не принято использовать malloc и free
- Их заменяют new и delete (не только для классов, для базовых типов тоже)
- Прямого базового аналога realloc нет, об этом ещё будет дальше

new / delete: примеры

Разбираем пример

- 01_no_malloc_anymore.cpp

Pointer vs Reference

Что такое ссылки

Ссылка (int& reference) – почти как указатель (int* pointer), но:

- Имеет немного другой синтаксис (меньше -> и *, чуть более читаемый код в итоге)
- Ссылка присваивается один раз и не меняется
- Ссылка не может быть NULL
- Нет аналога указателя на указатель
- Нет аналога арифметики указателей

Ссылки и указатели: примеры

Разбираем примеры

- 02_refs_intro.cpp
- 03_refs_and_functions.cpp

Что когда использовать

Общее правило:

- В функциях / методах использовать ссылки (references) для параметров и возвращаемых значений – так код чище и читаемее (и немного безопаснее)
- Для алгоритмов и структур данных использовать указатели (pointers) – их фичи там важны (как минимум, без повторного присваивания и NULL-ов не обойтись)

Многоликий const

Что такое const

Используется для указания на что-то неизменное:

- Значение созданной переменной
- Параметр, переданный в функцию
- Метод класса (в этом случае смысл «при работе метода не меняется экземпляр класса»)
- ... (больше по ссылкам, но это за рамками обязательного)

http://alenacpp.blogspot.com/2005/09/const-1.html
http://alenacpp.blogspot.com/2005/09/const-2.html

Многоликий const: примеры

Разбираем примеры

- 04_const_functions.cpp
- 05_const_methods.cpp

Зачем реально нужен const

Использование const не даёт менять объекты, которые логически ожидаются неизменными. При взгляде на чужой код (с аккуратно расставленными const-ами) сразу понятно, где нужно быть готовым к тому, что при вызове состояние объекта изменится. И это очень хорошо.

Подробнее про концепцию:
http://www.parashift.com/c++-faq-lite/const-correctness.html

Особенность const

Имеет свойство распространяться между вызовами, поэтому const нужно расставлять «с самого низу» всей иерархии методов и классов.

Потоки ввода-вывода

Ввод-вывод в C++

Для ввода-вывода в C++ используются потоки:

- Клавиатура/экран – это потоки (по умолчанию cin/cout)
- Файл – это поток
- Что-нибудь более творческое – тоже можно сделать потоком (но это отдельная история)

Потоки в C++: примеры

Разбираем примеры

- 06_streams_intro.cpp
- 07_file_streams_intro.cpp

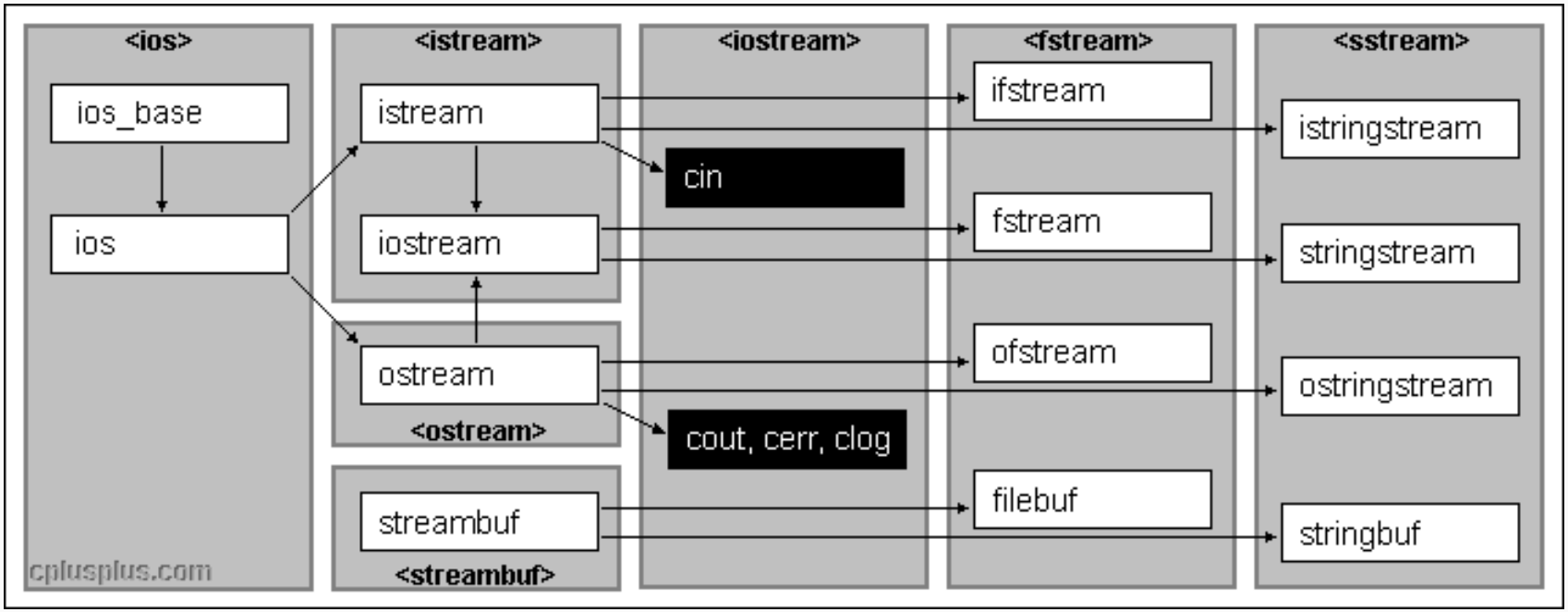
Реализация потоков

Любой поток – это экземпляр некоторого класса:

- У него есть своё внутреннее состояние.
- С ним можно общаться через операторы « и » или (при желании) через явный вызов методов

Иерархия классов ввода-вывода

У потоков есть своя иерархия классов:



Есть базовые интерфейсы потоков, что очень удобно. Можно (и нужно) общаться именно с ними. В этом случае автоматически обеспечена поддержка любых конкретных реализаций потоков.

Иерархия классов потоков: пример

Разбираем пример

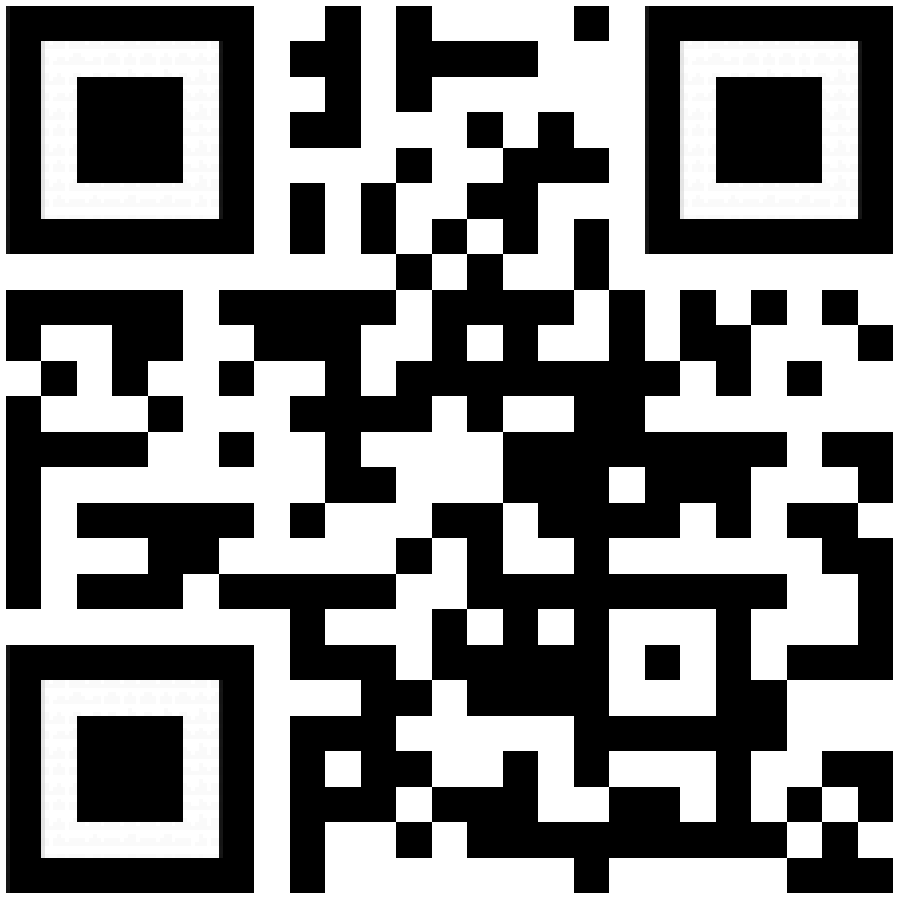
- 08_streams_inheritance.cpp

Что стоит запомнить из лекции

- В C++ malloc / free принято заменять на new / delete
- Кроме int* pointer в C++ появились int& reference, они в целом о том же, но с другими целями
- Есть const, который крайне полезен, если о нём не забывать
- Для ввода-вывода в C++ используются потоки, у них есть базовые интерфейсы, и это удобно

Мнение о лекции

https://tinyurl.com/yygaehwn




MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.