kaspersky

Kaspersky Tech.
7+ нюансов карьеры C++ разработчика

Почему C/C++ еще долго будут оставаться актуальными, и что на них можно создавать

Вадим Платонов

Производительность

- C/C++ это языки которые **всегда учитывали производительность** при своем создании и развитии
- C/C++ это **компилируемые языки**, т.е. после компиляции исходный код превращается в машинный код
- Интерпретируемые языки, такие как Python или C# **не смогут сравниться по производительности** с C/C++
- Как пример, **высокопроизводительные** сетевые решения, над которыми работает наш отдел, используют именно языки C/C++
- В C/C++ **полностью ручное** управление ресурсами, т.е. нету никаких Garbage Collector

Популярность

Mar 2023	Mar 2022	Change	Programming Language	Ratings	Change
1	1	▲	Python	14.83%	+0.57%
2	2	▲	C	14.73%	+1.67%
3	3	▲	Java	13.56%	+2.37%
4	4	▲	C++	13.29%	+4.64%
5	5	▲	C#	7.17%	+1.25%
6	6	▼	Visual Basic	4.75%	-1.01%
7	7	▲	JavaScript	2.17%	+0.09%
8	10	▲	SQL	1.95%	+0.11%

https://www.tiobe.com/tiobe-index/

Что можно создавать?

- Базы Данных
- Высокопроизводительные сетевые решения, например, NGFW
- Операционные системы, например, KasperskyOS
- Игры
- Веб-сервера
- Финтех проекты
- Антивирусы и другое системное ПО
- Встроенные решения

Какие специфики нужно знать и учитывать, чтобы успеть за стремительным развитием языка

Алексей Вуколов

Специфики

- Стандартная библиотека стремительно **развивается**
- Некоторые вещи, добавленные в предыдущих стандартах, **объявлены deprecated или удалены**
- **Не все компиляторы поддерживают** самые свежие изменения стандарта
- Переходы на новые подходы написания кода **кажутся неочевидными**

Специфика устаревания языка

- Разные компиляторы имеют **разный набор нестандартных расширений**
- Часть кода в проекте часто **написана до того**, как соответствующие механизмы включили в стандарт, например:
  - **Барьеры памяти:** появился std::atomic_thread_fence
  - **Синхронизация:** в Windows была своя, в Linux – своя (на замену пришел std::mutex)
  - **Потоки были свои:** теперь есть std::thread
  - **Boost**: часть классов доступны в STL. Например, boost::filesystem теперь std::filesystem

Как с этим бороться

- Прежде чем приступить к решению задачи: **узнать, возможно, она уже решена** средствами стандартной библиотеки
- **Читать код коллег**: кто-то находит новые улучшения языка и начинает их применять
- Чем большее число платформ и компиляторов поддерживает ваш проект, тем больше придётся **узнать о тонкостях данных платформ** и компиляторов

Навыки разработчика на C++

Сергей Новоселов

Отличное знание C++

- **Как устроены и чем отличаются стек и куча**, какие проблемы они решают и когда стоит предпочесть одну структуру другой
- **Как устроены стандартные контейнеры внутри** std::vector, std::string, std::list, std::map
- **Как работают исключения**, какие у них есть преимущества и недостатки
- **Современные стандарты C++ - 17 и 20**

Бьерн Страуструп - Язык программирования C++

Работа с кодом - умение писать простой код и упрощать чужой

- Соответствие принципам KISS. **Хороший код — простой код**
- **Умение читать чужой код**, находить ошибки в коде и дизайне
- **Умение упрощать** и декомпозировать сложные задачи
- **Умение разбивать** код на сущности
- **Умение понятно именовать** сущности

Стив Макконнел - "Совершенный код"

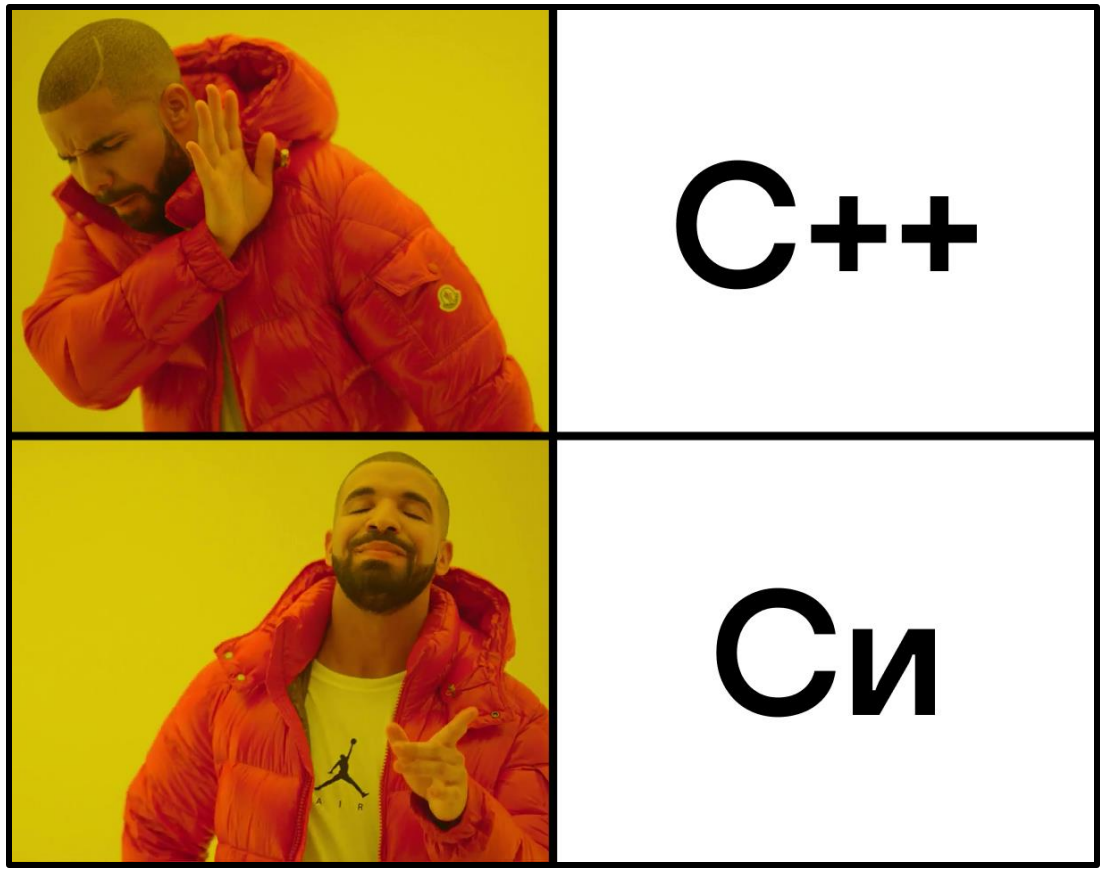
Работа с многопоточным кодом

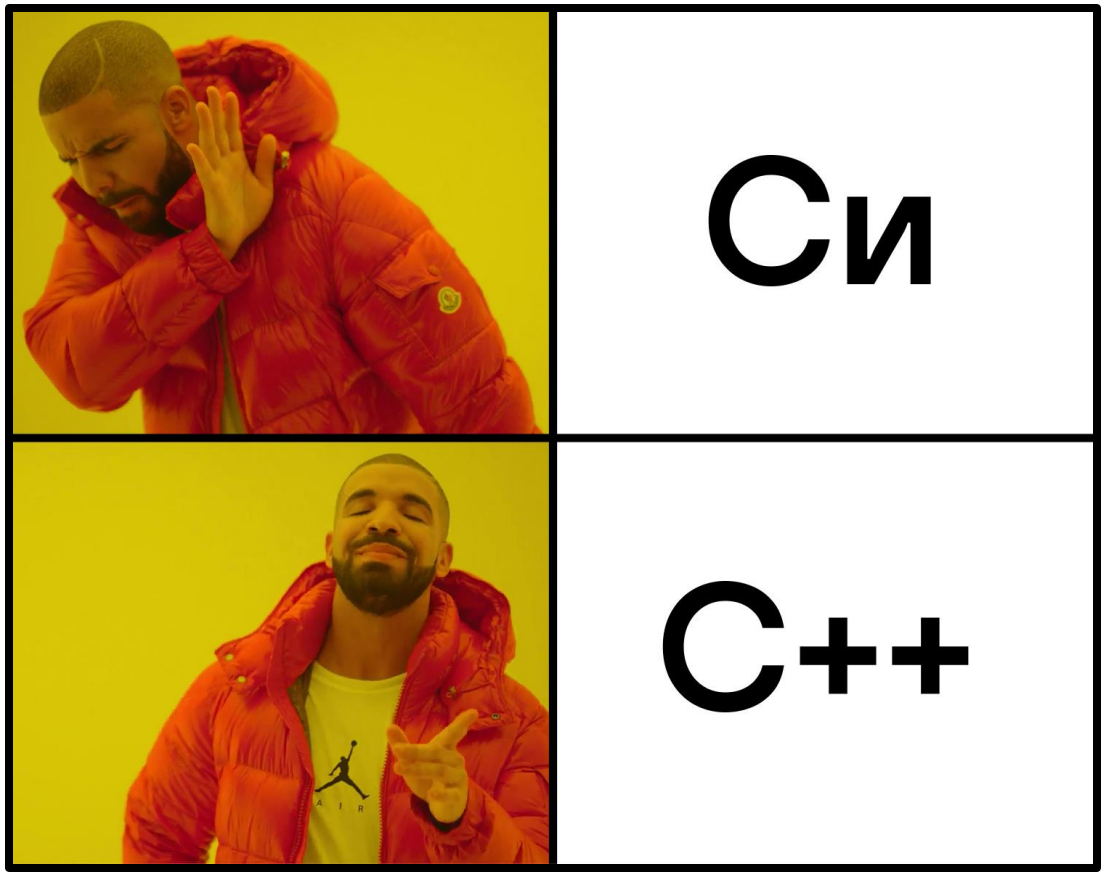
- Умение **смотреть на задачу в целом**, а не на отдельные ее блоки
- Необходимо четко понимать, **какие операции в коде стоит защищать**, а какие — необязательно
- Знание **сложных примитивов синхронизации**

Энтони Уильямс - "Параллельное программирование на C++ в действии"

Роль абстракции в C++ разработке в крупной компании

Илья Кочетов


C++
Си


Си
C++

Интерфейс

Реализация

```
1  class FileReader
2  {
3  public:
4      FileReader(string filePath);
5      string getLine(size_t line);
6
7  private:
8      bool open(string filePath);
9      bool close();
10     void readLine();
11     bool seek(size_t pos);
12
13 private:
14     File   m_file;
15     size_t m_fileSize;
16     size_t m_lineCount;
17 }
```

```
  SPIFFS_API_CHECK_CFG(fs);
  SPIFFS_API_CHECK_MOUNT(fs);
  SPIFFS_LOCK(fs);

  spiffs_fd *fd;
  s32_t res;

  filePath = SPIFFS_filePath_UNOFFS(fs, filePath);
  res = spiffs_fd_get(fs, filePath, &fd);
  SPIFFS_API_CHECK_RES_UNLOCK(fs, res);

  if ((fd->flags & SPIFFS_O_RDONLY) == 0) {
    res = SPIFFS_ERR_NOT_READABLE;
    SPIFFS_API_CHECK_RES_UNLOCK(fs, res);
  }

  if (fd->size == SPIFFS_UNDEFINED_LEN && len > 0) {
    // special case for zero sized files
    res = SPIFFS_ERR_END_OF_OBJECT;
    SPIFFS_API_CHECK_RES_UNLOCK(fs, res);
  }

#if SPIFFS_CACHE_WR
  spiffs_fflush_cache(fs, filePath);
#endif

  if (fd->fdoffset + len >= fd->size) {
    // reading beyond file size
    s32_t avail = fd->size - fd->fdoffset;
    if (avail <= 0) {
      SPIFFS_API_CHECK_RES_UNLOCK(fs, SPIFFS_ERR_END_OF_OBJECT);
    }
    res = spiffs_object_read(fd, fd->fdoffset, avail, (u8_t*)buf);
    if (res == SPIFFS_ERR_END_OF_OBJECT) {
      fd->fdoffset += avail;
      SPIFFS_UNLOCK(fs);
      return avail;
    } else {
      SPIFFS_API_CHECK_RES_UNLOCK(fs, res);
...
```

```
int fd;
int bytesRead;

fd = open(filePath, O_RDONLY);
if(fd  == -1)
{
  error("Open Failed");
  return 0;
}

buf = malloc(len + 1);

if ((bytesRead = (read(fd, buf, len))) != len)
{
  error("Read Failed");
  return bytesRead;
}

buf[len] = '\0';

if (close(fd) == -1)
{
  error("Close Failed");
}
return len;
```

```
QFile inputFile(filePath);
 if (inputFile.open(QIODevice::ReadOnly))
 {
    buf = inputFile.read(len);
    return buf.size();
 }
 return 0;
```

MyFile(fs, filePath).read(buf, len);

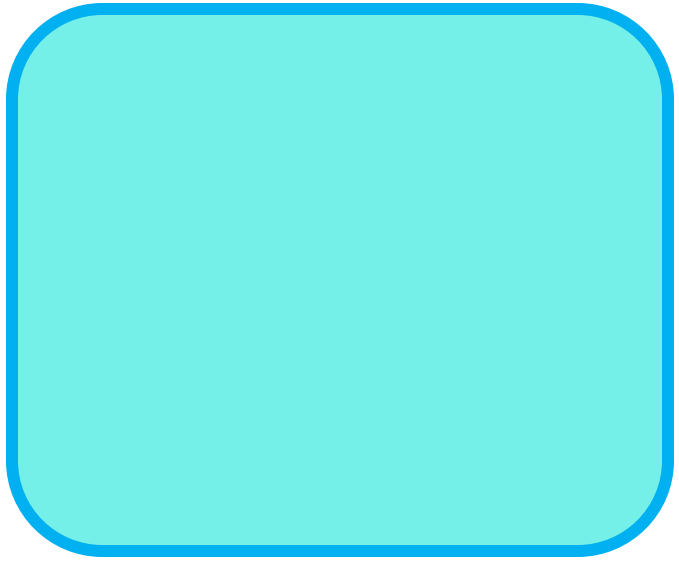
MyFile(filePath).read(buf, len);

```
int fd;
int bytesRead;

fd = open(filePath, O_RDONLY);
if(fd  == -1)
{
  error("Open failed");
  return 0;
}

buf = malloc(len + 1);

if ((bytesRead = (read(fd, buf, len))) != len)
{
  error("Read failed");
  return bytesRead;
}

buf[len] = '\0';

if (close(fd) == -1)
{
  error("Close failed");
}
return len;
```

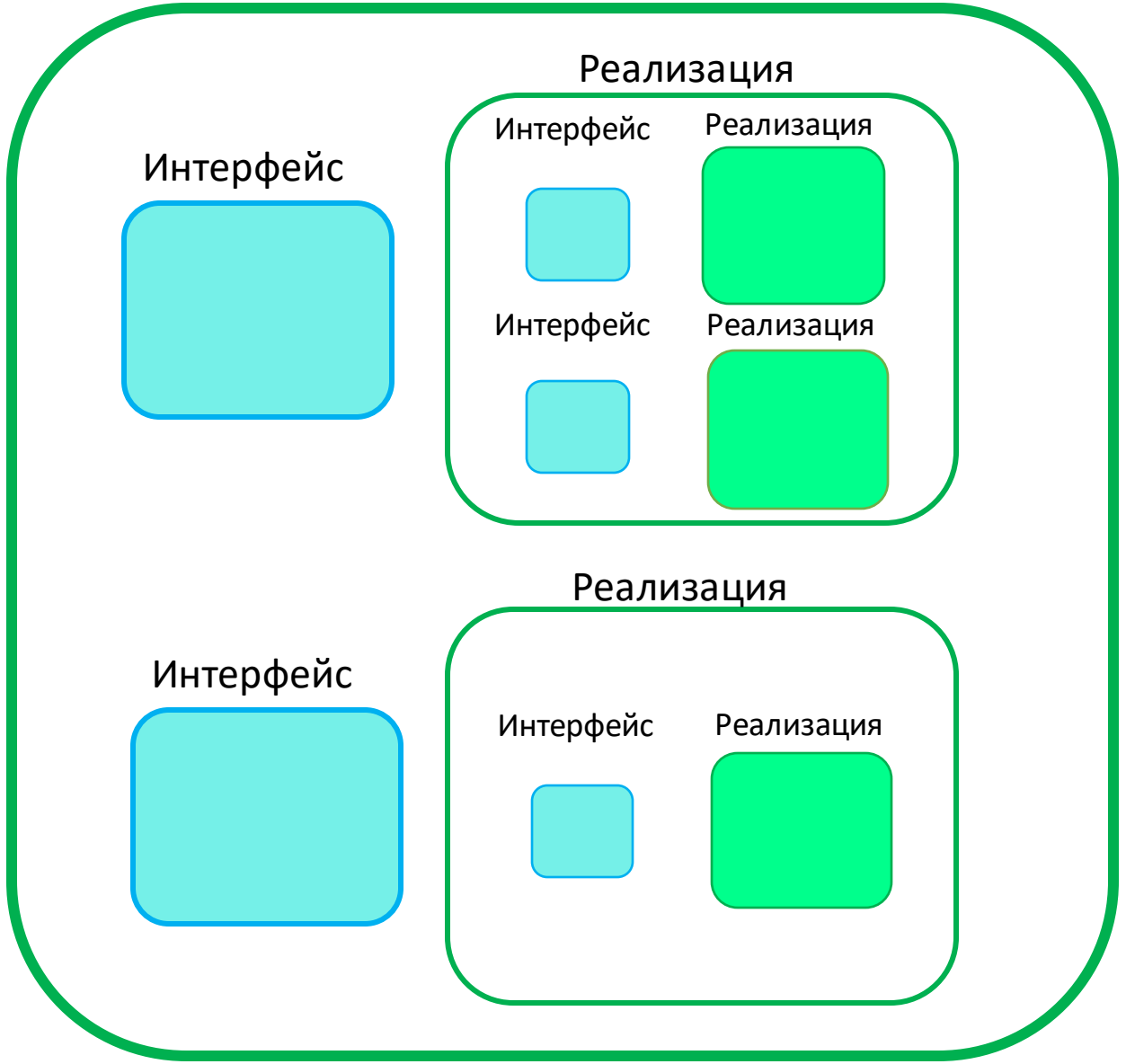
MyFile(filePath).read(buf, len);

```
QFile inputFile(filePath);
 if (inputFile.open(QIODevice::ReadOnly))
 {
    buf = inputFile.read(len);
    return buf.size();
 }
 return 0;
```

Интерфейс



Реализация



В каких случаях справедливы фразы вида «выучи это и это – станешь сеньором»

Вадим Платонов

Процесс Senior Promotion в ЛК

- Руководитель рекомендует разработчика к Senior Promotion
- Разработчик подготавливает артефакты:
  - Примеры решенных задач
  - Примеры разработанной документации
  - Лучшие исправленные ошибки
  - Лучшие коммиты
  - Проведенные Ревью
- Эксперты ЛК оценивают артефакты и выносят свой вердикт

Какие именно софт-скилы нужны C++ разработчику и где их прокачать

Евгений Никитин

О себе

Евгений Никитин

- в ИБ-разработке больше 20 лет (Информзащита, Код Безопасности, ИнфоТеКС, Лаборатория Касперского)
- работал разработчиком C/C++, руководителем группы/отдела, менеджером проектов
- опыт создания проектов и команд с нуля
- работа с **outsource**
- проекты, в которых участвовал, до сих пор продаются в РФ (десятки тысяч в год)
- играю в футбол и организую тренировки

О проекте

Делаем Next Generation Firewall

1. Сетевое устройство с большим количеством функций – подробнее по ссылке
2. Исполнения: ПАК, VM, FWaaS
3. Большое количество конкурентов (только в РФ не менее 20)
4. Проекту более 3х лет
5. Публичная демо-версия осенью

Команда

1. 20+ человек
2. Баланс – тимлиды, сеньоры, мидлы, джуны и даже стажёры
3. 3х недельные итерации – планирование, стендапы, демо, ретро
4. C++ 20

Софт-скилы

Тенденции

- Ушли времена крутых одиночек
- Сложные проекты реализуются командами, в которых важная «химия» (почти как в футболе)

Как сформировать идеальную команду

- Интервью
- Испытательный срок
- В процессе работы
- Performance review

Как попасть в хорошую команду

- Интервью
- «Тренировки» в работе и в обычной жизни

Как прокачать софт-скилы

Умение формулировать свои мысли и слушать других

- Общаться (оффлайн и онлайн)
- Участвовать в технических обсуждениях, принятии решений
- Делать доклады и презентации, хотя бы внутренние

Работа с эмоциями, развитие эмпатии

- Технические дискуссии
- Code review
- Спорт
- Больше интересоваться другими людьми
- Медитации

Взаимодействие в команде, общие цели

- Тимбилдинги, квесты
- Командные виды спорта

Спасибо за внимание!
Вопросы?