ВВЕДЕНИЕ

C# - ЛЕКЦИЯ 1

Павлюк Александр Алексеевич

slack: ссылка на регистрацию
мобильный: +7 (915) 197-31-16

СЕГОДНЯ В ПРОГРАММЕ

- Правила игры
- Лекция: "CLI"
- Практическая часть: быстрый старт C#

УЧЕБНЫЙ ПРОЦЕСС

- Расписание: по субботам, с 10:00 до 13:30
- Лекция: 10:00-11:00 и 11:15-12:15
- Практическая часть: 12:30-13:30
- Домашние задания после каждого занятия
- Тесты перед каждым занятием
  Тест проводится онлайн, но пройти его можно будет с помощью телефона.
- Экзамен в конце курса

ДИСТАНЦИОННЫЙ ФОРМАТ

- Занятия - в zoom
- Вопросы? Поднимайте руку или пишите в чат
- Видеозаписи и презентации - будут

РЕПОЗИТОРИЙ С МАТЕРИАЛАМИ КУРСА

Ссылки на записи занятий и презентации, код с занятий и литература

https://gitlab.com/alexander-pavlyuk-courses/csharp/c-sharp-course-spring-2021

CLI

Common Language Infrastructure

- Конец 90-х гг.
- Экосистема Windows растёт
- Планы на Windows Server
- Как сделать экосистему привлекательной?

ЧТО НУЖНО ПОЛЬЗОВАТЕЛЯМ?

- Большой выбор приложений
- Простая установка
- Программы работают везде
- Программы работают везде одинаково
- Надёжность
- Безопасность

ЧТО НУЖНО РАЗРАБОТЧИКАМ?

- Удобные средства разработки
- Широкий выбор библиотек
- Удобные средства отладки
- Удобный способ развёртывания

ЧТО МЕШАЕТ ВСЕМ?

1. УТЕЧКИ ПАМЯТИ

Процесс неконтролируемого роста объёма оперативной памяти, занимаемой процессом выполнения программы, связанный с ошибками управления памятью.

1. УТЕЧКИ ПАМЯТИ

```c
#include <stdlib.h>

void function_which_allocates(void) {
    /* allocate an array of 45 floats */
    float *a = malloc(sizeof(float) * 45);

    /* additional code making use of 'a' */

    /* return to main, having forgotten to free the memory we malloc'd */
}

int main(void) {
    function_which_allocates();

    /* the pointer 'a' no longer exists, and therefore cannot be freed,
     but the memory is still allocated. a leak has occurred. */
}
```

1. УТЕЧКИ ПАМЯТИ. К ЧЕМУ ПРИВОДЯТ?

- Ресурсы заканчиваются
- Начинает использоваться файл подкачки
- Доступность системы падает до субъективного нуля
- Риск потери данных

1. УТЕЧКИ ПАМЯТИ. ЧТО ДЕЛАТЬ?

- Писать код аккуратно
- "Умные указатели"
- Сборщик мусора
- Система владения памятью (Rust)

2. ОШИБКИ СЕГМЕНТАЦИИ (SEGFAULT)

Возникают при попытке обращения к недоступным для записи участкам памяти.

2. ОШИБКИ СЕГМЕНТАЦИИ (SEGFAULT)

```
//  Попытка изменения константного значения
const char * s = "hello world";
* (char *) s = 'H';

//  Попытка обращения к чужому адресу
int *ptr = NULL;
*ptr = 3;

//  Выход за границы массива
int n = 10;
float* arr = new float[n];
for (int i = 0; i <= n; i++) {
    arr[i] = i * 3.14;
}
```

2. ОШИБКИ СЕГМЕНТАЦИИ. К ЧЕМУ ПРИВОДЯТ?

- Разрушительный сбой процесса
- Гарантированная утеря данных
- Нет способа остановить сбой
- Нет способа предотвратить подобные ситуации

2. ОШИБКИ СЕГМЕНТАЦИИ (SEGFAULT). ЧТО ДЕЛАТЬ?

- Писать код аккуратно
- Управляемый код, отказ от адресной арифметики
- Система владения памятью (Rust)

3. ФРАГМЕНТАЦИЯ ПАМЯТИ

- Нарастает по мере работы программы
- Растёт память процесса
- Медленнее выделяется память

3. ФРАГМЕНТАЦИЯ ПАМЯТИ. ЧТО ДЕЛАТЬ?

- Предусмотреть механизм дефрагментации памяти в умных указателях или сборщике мусора

4. ЗАВИСИМОСТЬ ОТ ПРОГРАММНОГО ОКРУЖЕНИЯ

- ОС
- Версия ОС
- Набор библиотек
- Версии библиотек

4. ЗАВИСИМОСТЬ ОТ ПРОГРАММНОГО ОКРУЖЕНИЯ. ЧТО ДЕЛАТЬ?

- Условная компиляция
- Статическая линковка библиотек

УСЛОВНАЯ КОМПИЛЯЦИЯ

```
#if WIN
    #include <windows.h>
    SYSTEMTIME st;
    GetSystemTime(&st);
    unsigned short hours = st.wHour;
    unsigned short mins = st.wMinute;
#else
    #include <time.h>

    struct tm time = localtime(time(NULL));
    unsigned short hours = time->tm_hour;
    unsigned short hours = time->tm_min;
#endif
```

СТАТИЧЕСКАЯ ЛИНКОВКА

- + нет зависимостей
- - повышенное потребление памяти процессом
- - для обновления зависимостей требуется пересборка приложения (каждого)

5. ЗАВИСИМОСТЬ ОТ АППАРАТНОГО ОКРУЖЕНИЯ

- Различные оптимизации под разные процессоры

5. ЗАВИСИМОСТЬ ОТ АППАРАТНОГО ОКРУЖЕНИЯ. ЧТО ДЕЛАТЬ?

- Условная компиляция
- Оптимизация каждой программы под каждую платформу

6. ПРОГРАММЫ НАПИСАНЫ НА РАЗНЫХ ЯЗЫКАХ

- Одни и те же задачи решаются многократно на разных языках
- Нужно изучать разные языки и сопутствующие технологии
- Иногда приходится переписывать на другой язык

7. БЕЗОПАСНОСТЬ

- Как защитить пользователя от подмены библиотек?

8. УДОБСТВО РАЗВЁРТЫВАНИЯ

Precompiled Binaries for Android

sqlite-android-3270100.aar (3.05 MiB)	A precompiled Android library containing the core SQLite together with appropriate Java bindings, ready to drop into any Android Studio project. (sha1: b5afcdf9f27f2a6ad6c07105953652654c893c5d)

Precompiled Binaries for Linux

sqlite-tools-linux-x86-3270100.zip (1.89 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program. (sha1: 30851e09a7a02cb33928aa283d4c51942bafe786)

Precompiled Binaries for Mac OS X (x86)

sqlite-tools-osx-x86-3270100.zip (1.26 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program. (sha1: 381554371289cef16161a8bb7334415aa8272b17)

Precompiled Binaries for Windows

sqlite-dll-win32-x86-3270100.zip (469.06 KiB)	32-bit DLL (x86) for SQLite version 3.27.1. (sha1: eb4c7262e32a7225f010427b25ba681f540ddd6a)
sqlite-dll-win64-x64-3270100.zip (781.40 KiB)	64-bit DLL (x64) for SQLite version 3.27.1. (sha1: 213a5868f3e876983242a1ac351893af80c76a81)
sqlite-tools-win32-x86-3270100.zip (1.69 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3_analyzer.exe program. (sha1: 6eaed19d1734c15ff7b2b4bcde5a8b71a856e098)

Universal Windows Platform

8. УДОБСТВО РАЗВЁРТЫВАНИЯ

Universal Windows Platform

sqlite-uwp-3270100.vsix (6.87 MiB)	VSIX package for Universal Windows Platform development using Visual Studio 2015. (sha1: cf22e6b3e3f750cc69b83c688d2fb10346c35f5b)

Precompiled Binaries for Windows Phone 8

sqlite-wp80-winrt-3270100.vsix (4.45 MiB)	A complete VSIX package with an extension SDK and all other components needed to use SQLite for application development with Visual Studio 2012 targeting Windows Phone 8.0. (sha1: 32fa6219796e1703fd48f1b9232699ce59573b99)
sqlite-wp81-winrt-3270100.vsix (4.50 MiB)	A complete VSIX package with an extension SDK and all other components needed to use SQLite for application development with Visual Studio 2013 targeting Windows Phone 8.1. (sha1: 92818b8c9e0f84a29c2cd05d1bd5573191e932f8)

Precompiled Binaries for Windows Runtime

sqlite-winrt-3270100.vsix (6.77 MiB)	A complete VSIX package with an extension SDK and all other components needed to use SQLite for WinRT application development with Visual Studio 2012. (sha1: d16e7750f70bceefd30febb0e4d6837c685ba634)
sqlite-winrt81-3270100.vsix (6.78 MiB)	A complete VSIX package with an extension SDK and all other components needed to use SQLite for WinRT 8.1 application development with Visual Studio 2013. (sha1: dc7b6a62cee901dcac21b1b3df3867d522d30aca)

А ТЕПЕРЬ ДАВАЙТЕ РЕШИМ ВСЕ ПРОБЛЕМЫ РАЗОМ

- Утечки памяти
- Ошибки сегментации
- Фрагментация памяти
- Зависимость от программного окружения
- Зависимость от аппаратного окружения
- Программы написаны на разных языках
- Безопасность
- Удобство развёртывания

КАК?

22 июня 2000 г. Microsoft провозглашает .NET Strategy —

долгосрочный план разработки и вывода на рынок программных продуктов.

В рамках .NET Strategy был разработан стандарт Common Language Infrastructure (CLI).

COMMON LANGUAGE INFRASTRUCTURE

(CLI)

COMMON LANGUAGE INFRASTRUCTURE

- Технический стандарт
- Разработан Microsoft
- Стандартизован ISO (ISO/IEC 2327:2012) и ECMA (ECMA-335 6th Edition)

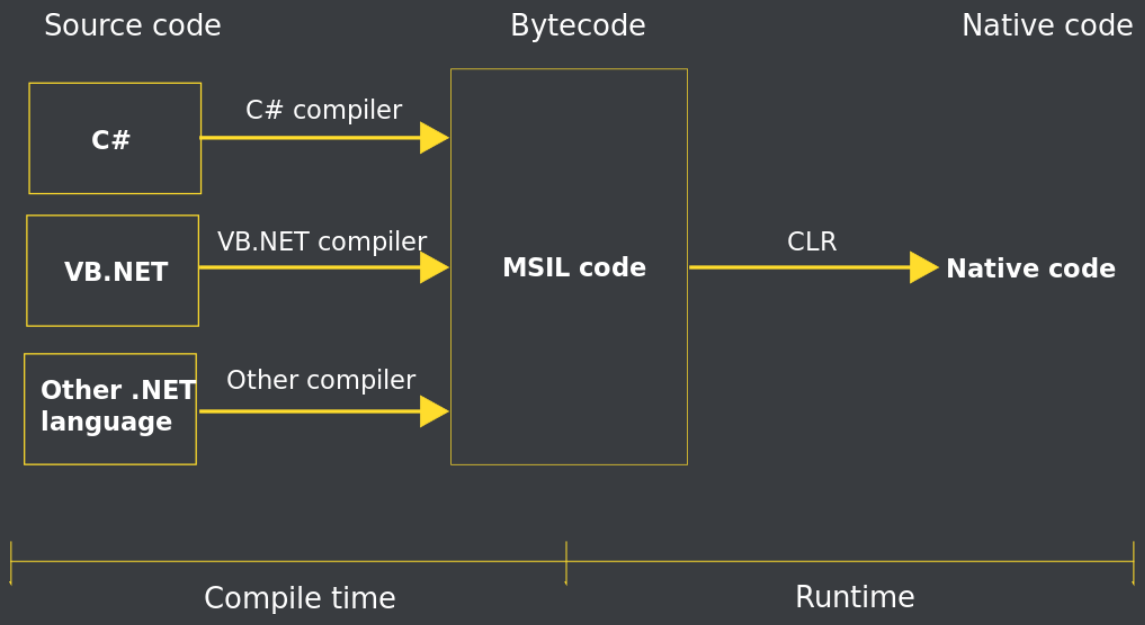
COMMON LANGUAGE INFRASTRUCTURE

Описывает исполняемый код и среду исполнения, позволяющую использовать различные языки высокого уровня на различных компьютерных платформах без переписывания под конкретную архитектуру (независимость от выбора языка и платформы)

АСПЕКТЫ CLI

1. Общая система типов - Common Type System (CTS)
2. Метаданные - Metadata
3. Общая спецификация языков - Common Language Specification (CLS)
4. Виртуальная система исполнения - Virtual Execution System (VES)

ПРИНЦИП РАБОТЫ CLI



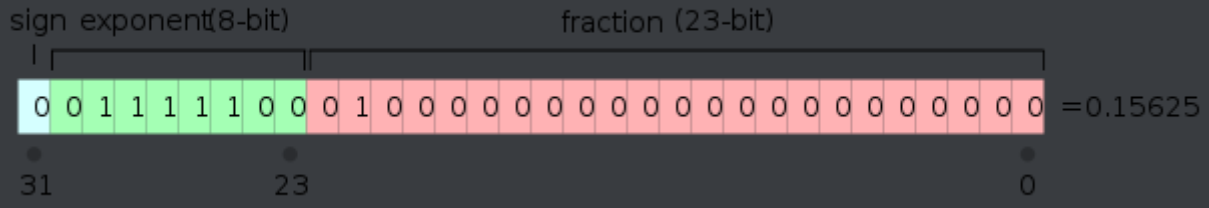
COMMON TYPE SYSTEM

Часть стандарта CLI, описывающая типы данных.

ТИП ДАННЫХ

1. Объем занимаемой памяти
2. Способ представления в памяти
3. Диапазон значений
4. Набор операций

IEEE 754, 32-BIT FLOAT



МЕТАДАННЫЕ В CLI

Часть стандарта CLI, описывающая независимый от языка способ представления информации о структуре программы.

МЕТАДАННЫЕ В CLI

CLI использует метаданные для описания типов, объявленных согласно CTS. Это делает метаданные общим форматом и механизмом обмена информацией:

- Между инструментами: компиляторами, отладчиками
- Между инструментами и VES

COMMON LANGUAGE SPECIFICATION

Часть стандарта CLI, описывающая соглашение между разработчиками языка и разработчиками библиотек и фреймворков. Содержит указание подмножества CTS и набора конвенций использования.

COMMON LANGUAGE SPECIFICATION

Языки позволяют использовать библиотеки и фреймворки, если реализуют по крайней мере те части CTS, которые указаны в CLS.

COMMON LANGUAGE SPECIFICATION

Библиотеки и фреймворки, в свою очередь, будут максимально совместимы, если их открытые для взаимодействия типы являются теми, что указаны в CLS или удовлетворяют конвенциям CLS.

VIRTUAL EXECUTION SYSTEM

Часть стандарта CLI, описывающая среду исполнения программ на языках, совместимых с CLI.

VIRTUAL EXECUTION SYSTEM

VES использует метаданные сборок для связывания их в единую программу.

VIRTUAL EXECUTION SYSTEM

Машинный код генерируется just-in-time компилятором (JIT)

VIRTUAL EXECUTION SYSTEM. JIT

- Компилирует на этапе исполнения
- Компилирует "лениво"
- Доступны метаданные единиц компиляции
- Возможна "горячая" замена

VIRTUAL EXECUTION SYSTEM. JIT

Выполняет управление памятью процесса. За это отвечает сбощик мусора (garbage collector, GC).

VIRTUAL EXECUTION SYSTEM. GC

- Выполняется в одном процессе с программой, но в отдельном потоке
- Регистрирует все выделяемые программой участки памяти
- Отслеживает неиспользуемые участки памяти
- Очищает и дефрагментирует память программы

VIRTUAL EXECUTION SYSTEM. GC

Подробнее про сборку мусора - в книге

"Under the hood of .NET memory management" от Chris Farrell и Nick Harrison

VIRTUAL EXECUTION SYSTEM

При генерации машинного кода VES выполняет платформоспецифичные оптимизации.

СБОРКА (ASSEMBLY)

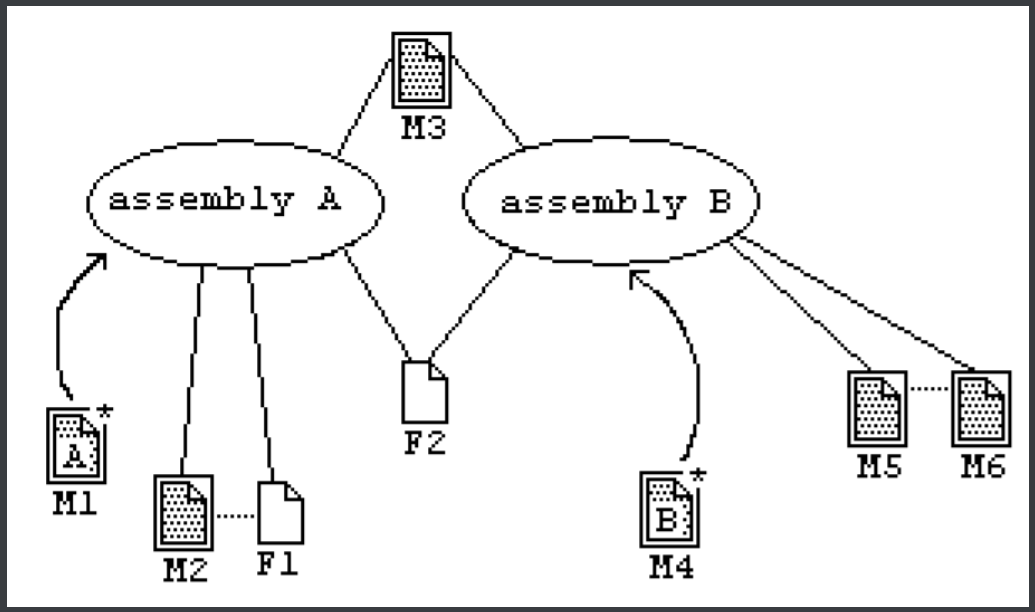
Набор из одного или нескольких файлов, поставляемых как единое целое. Обязательно включает в себя манифест.

ФАЙЛЫ, ВХОДЯЩИЕ В СБОРКУ

- Модули: код на CIL + метаданные
- Ресурсы: строки, изображения и т.д.

МАНИФЕСТ СБОРКИ

- Версия сборки
- Имя сборки
- Языковая культура сборки
- Требования безопасности сборки
- Какие ещё файлы входят в сборку
- Цифровая подпись и публичный ключ


M3
assembly A
assembly B
A
M1
M2
F1
F2
B
M4
M5
M6

ВЕРСИЯ СБОРКИ

4 32-битных целых числа

- Major - крупные изменения, потеря обратной совместимости
- Minor - важные улучшения, сохранение совместимости
- Build - различные компиляции одного исходного кода
- Revision - полная взаимозаменяемость в пределах ревизии

ИМЯ СБОРКИ

Может включать точки:

System.Configuration.dll

ЯЗЫКОВАЯ КУЛЬТУРА СБОРКИ

Опциональна. Указывает, что сборка относится к определённой языковой культуре.

Названия культур удовлетворяют IETF RFC1766

ЯЗЫКОВАЯ КУЛЬТУРА СБОРКИ

"<язык>-<страна/регион>"

- <язык> - 2-буквенный код в нижнем регистре (ISO 639-1)
- <страна/регион> - 2-буквенный код в верхнем регистре (ISO 3166)

ЯЗЫКОВАЯ КУЛЬТУРА СБОРКИ

"en-US"

"ru-RU"

ТРЕБОВАНИЯ БЕЗОПАСНОСТИ СБОРКИ

- Требуемые разрешения
- Желательные разрешения
- Запрещаемые разрешения

ЦИФРОВАЯ ПОДПИСЬ (СИЛЬНОЕ ИМЯ СБОРКИ)

- Гарантирует аутентичность сборки
- Проверяется на уровне VES

РЕАЛИЗАЦИИ CLI

РЕАЛИЗАЦИИ CLI

- .NET Framework
- Mono
- .NET Core
- .NET

.NET FRAMEWORK

- Первая и официальная реализация от Microsoft
- Наиболее широкий набор библиотек.
- Только под Windows.
- Большая установка, GAC (global assembly cache)

.NET FRAMEWORK

- Open Source (MIT, 12.11.2014)
- https://referencesource.microsoft.com
- https://github.com/microsoft/referencesource

.NET FRAMEWORK

Версия	Дата выхода
1.0	13.02.2002
...	...
2.0	7.11.2005

3.0	5.11.2006
3.5	19.11.2007
4.0	12.04.2010
4.5	15.08.2012
...	...
4.8	18.04.2019

MONO

https://www.mono-project.com/

- Open Source реализация
- Стартовала 30.06.2004
- Кроссплатформенная: Windows, Linux, macOS, Android, BSD, Solaris, PS 3, Wii, Xbox 360
- Не такой широкий набор библиотек (в прошлом?)

.NET CORE

- Open Source и cross-platform реализация от .NET Foundation
- Первая версия - 27.06.2016
- Только Web, Console и Dll.
- Windows, Linux, macOS
- Лёгкая, модульная установка, частые обновления модулей

.NET CORE

Версия	Дата выхода
.NET Core 1.0	27.06.2016
.NET Core 2.0	14.08.2017
.NET Core 3.0	23.09.2019

.NET Core 3.1	15.01.2020
.NET 5 (без "Core")	10.11.2020
.NET 6 Preview	17.02.2021

Выбирайте .NET

https://github.com/dotnet
https://dotnet.microsoft.com

ЯЗЫК C#

ЯЗЫК C#

- Флагманский язык .NET
- Объектно-ориентированный
- C-подобный
- Разработан в Microsoft в 1998-2001 гг. под руководством Андерса Хейлсберга и Скотта Вильтаумота
- ECMA-334 и ISO/IEC 23270

ЯЗЫК C#

- Развитие языка - открытое: https://github.com/dotnet/csharplang
- Компилятор - Roslyn: https://github.com/dotnet/roslyn

ЯЗЫК C#

Почему "шарп"?

Может, хештег?

Может, это (C++)++?

ЯЗЫК C#

Название «Си шарп» (от англ. sharp — диез) происходит от буквенной музыкальной нотации, где латинской букве C соответствует нота До, а знак диез (англ. sharp) означает повышение соответствующего ноте звука на полутон

ВОПРОСЫ