Правительство Российской Федерации

Государственное образовательное бюджетное учреждение высшего профессионального образования

«Национальный исследовательский университет Высшая школа экономики»

Факультет компьютерных наук
Департамент анализа данных и искусственного интеллекта

Программа дисциплины
Языки разработки программного обеспечения

для направления 01.03.02 – Прикладная математика и информатика
подготовки бакалавра

Автор программы
к.ф.-м.н. Яковлев Виктор Вадимович

Одобрена на заседании
департамента Анализа данных и
искусственного интеллекта

Руководитель департамента

_____С.О.Кузнецов
« ____» _____ 2015 г.

Москва, 2015

1 Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает минимальные требования к знаниям и умениям студента и определяет содержание и виды учебных занятий и отчетности.

Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов направления подготовки 01.03.02 "Прикладная математика и информатика", изучающих дисциплину Языки разработки программного обеспечения.

Программа разработана в соответствии с:

- Образовательным стандартом Федерального государственного образовательного учреждения высшего профессионального образования Национальный исследовательский университет Высшая школа экономики.;
- Образовательной программой по направлению подготовки 01.03.02 "Прикладная математика и информатика".
- Рабочим учебным планом университета по направлению подготовки 01.03.02 "Прикладная математика и информатика", утвержденным в 2015 г.

2 Аннотация

Курс посвящен современным технологиям разработки программного обеспечения и используемым при этом языкам программирования. Целями курса являются:

1. Систематизация знаний в области синтаксиса и семантики современных языков программирования

2. Освоение навыков промышленной разработки прикладного программного обеспечения

3. Получение представления о различных языках программирования и области их применимости к различным задачам.

Настоящий курс содержит четыре раздела. Первый раздел посвящён общим вопросам технологий разработки (контроль версий, нормативные документы, этапы разработки ПО и т. п.). Каждый из трех других разделов посвящен особенностям использования определенного класса языков программирования (компилируемые процедурные языки, интерпретируемые процедурные языки, декларативные языки). В ходе курса будут рассмотрены, в частности, такие языки как Си++, Си, Python, Lua, Erlang, Go, биткод LLVM и др. Курс будет сопровождаться выполнением практических работ. Особое внимание при этом будет уделено технологиям разработки с использованием нескольких языков программирования в рамках одного проекта.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины студент должен:

- Знать основные парадигмы программирования, понимать процесс выполнения программ, написанных на различных языках программирования.
- Уметь проектировать прикладное программное обеспечение, выбирать наиболее подходящие для решения конкретных прикладных задач инструменты и технологии.
- Иметь навыки работы с системами контроля версий и системами непрерывной интеграции программных продуктов.

В результате освоения дисциплины студент осваивает следующие компетенции:

Компетенция	Код по ФГОС/ НИУ	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции
Умение работать на компьютере, навыки использования основных классов программного обеспечения, работы в компьютер-	ИК-2	Студент демонстрирует владение инструментальной средой программирования	Выполнение лабораторных работ и домашних заданий в инструментальной среде программирования

Компетенция	Код по ФГОС/ НИУ	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции
ных сетях			
Готовность к организационно-управленческой работе с малыми коллективами	ИК-3	Студент составляет техническое задание и другие нормативные документы; владеет навыками организации полного цикла разработки и внедрения прикладного программного обеспечения	Лекции по процессам разработки программного обеспечения, домашнее задание, в котором требуется подготовка полного комплекта программной документации и организация процесса тестирования
Способность применять в профессиональной деятельности современные языки программирования и языки баз данных, операционные системы, электронные библиотеки и пакеты программ и т.п.	ПК-9	Студент демонстрирует владение несколькими современными языками программирования; демонстрирует способность проектирования программного обеспечения с обоснованным выбором подходящих к каждой конкретной задаче инструмента	Обзорные лекции по различным языкам программирования и особенностям их применимости; домашнее задание, в котором требуется применять одновременно несколько языков программирования

4 Место дисциплины в структуре образовательной программы

Настоящая учебная дисциплина входит в факультативную часть цикла дисциплин информационных технологий в учебной программе подготовки бакалавра направления 010400.62 «Прикладная математика и информатика».

Изучение курса «Языки разработки программного обеспечения» требует базовых знаний по основам программирования на языке высокого уровня (в объеме курса «Информатика и программирование» первого года обучения указанной бакалаврской программы).

Основные положения дисциплины «Языки разработки программного обеспечения» могут быть использованы в дальнейшем при изучении следующих дисциплин программы бакалавра:

5 Архитектура компьютера и операционные системы.

- Прикладная теория графов

6 Тематический план учебной дисциплины

№	Название темы	Всего часов	Аудиторные часы		Самостоятельная работа
			Лекции	Практические занятия	
1.1	Системы контроля версий	2	1	1	0
1.1	Интерпретируемые языки программирования	22	1	1	20
2.1	Проектирование и разработка динамических библиотек	9	2	2	5
2.2	Внедрение интерпретаторов в произвольные приложения	9	2	2	10

2.3	Расширение возможностей интерпретаторов с помощью кода на Си/Си++	24	2	2	20
3.1	Языки для платформы Java	4	2	2	0
3.2	Низкоуровневое представление программ	4	2	2	0
3.3	Устройство компиляторов и генерация кода	4	2	2	0
3.4	Механизмы освобождения динамической памяти	4	2	2	0
4.1	Введение в функциональное программирование	26	2	4	20
4.2	Теоретические аспекты разработки параллельных программ	6	2	4	0
4.3	Сети и распределенное выполнение программ	36	2	4	30
5.1	Платформа Node.js и языки программирования ECMAScript и TypeScript	16	2	4	10
5.2	Языки программирования Dart и Go. Создание сетевых сервисов с использованием веб-сокетов	19	2	2	15
Итого:		180	30	28	122

7 Формы контроля знаний студентов

Тип контроля	Форма контроля	Параметры
Текущий контроль	Четыре домашних задания	Выполненные домашние задания публикуются студентом в git-репозитории, временем сдачи считается время последней отправки решения (commit). Допускается сдача домашнего задания после нормативного срока, при этом результаты засчитываются с штрафными коэффициентами: • **0,9** — при сдаче не позднее чем через неделю после обозначенного срока; • **0,7** — при сдаче не позднее чем через 2 недели после обозначенного срока; • **0,5** — при сдаче домашнего задания; позднее 2-х недель после объявленного срока.
Итоговый контроль	Экзамен	Экзамен состоит из двух частей.

		1. Выполнение практическое задания (40 минут) Во время выполнения практического задания разрешается использовать ресурсы сети Интернет, за исключением средств коммуникации (электронная почта, форумы, социальные сети и т. д.) 2. Устная защита решения практического задания, и ответ на дополнительные вопросы из программы устного экзамена курса, в том числе не связанные с темой практического задания. Дополнительный вопрос выдается после окончания выполнения студентом практического задания. На подготовку к дополнительному вопросу выделяется 30 минут, использование литературы и сети Интернет в это время запрещено.

7.1 Критерии оценки знаний, навыков

При выполнении домашних заданий оценивается:

1. Умение аккуратно писать тексты программ и сопроводительную документацию.
2. Содержание текстов комментариев к программам и содержание сопроводительной документации.
3. Навыки студента в демонстрации работоспособности программы, быстрого внесения в программу незначительных изменений и фиксирования внесенных изменений в систему контроля версий.

Оценки по всем формам текущего контроля выставляются по вещественнозначной шкале от 0 до 1. Оценка за итоговый контроль выставляется по целочисленной шкале от 0 до 10.

7.2 Порядок формирования оценок по дисциплине

Раздел 1. Темы 1.1 и 1.2

Студент выполняет домашнее задание, связанное с использованием динамических особенностей интерпретируемых языков программирования. Результат своей работы публикует в репозитории системы контроля версий.

За выполненную домашнюю работы ставится вещественнозначная оценка от 0 до 1.

Раздел 2. Темы 2.1, 2.2 и 2.3

Студент выполняет домашнее задание, в котором необходимо реализовать динамическую библиотеку в виде модуля для интерпретатора языка Python. Задача, которую должен решать реализованный модуль, подразумевает использование стороннего интерпретатора.

За выполненную домашнюю работы ставится вещественнозначная оценка от 0 до 1.

Раздел 3. Темы 3.1, 3.2, 3.3 и 3.4

Это теоретические темы, необходимые для понимания устройства систем программирования. Самостоятельная работа и выполнение домашних заданий по темам этого раздела не предусмотрены. Теоретические вопросы из данного раздела включены в программу устного экзамена.

Раздел 4. Темы 4.1, 4.2 и 4.3

Студент выполняет домашнее задание, в котором необходимо реализовать сервер для системы Erlang/OTP, выполняющий несколько независимых процессов и реализующий внешний интерфейс для взаимодействия по протоколу HTTP.

За выполненную домашнюю работы ставится вещественнозначная оценка от 0 до 1.

Раздел 5. Темы 5.1, 5.2

Студент выполняет домашнее задание, в котором он должен реализовать пару клиент-сервер, и продемонстрировать умение реализовать взаимодействие между клиентской частью приложения (выполняемой в браузере) и серверной частью посредством механизма WebSockets.

За выполненную домашнюю работы ставится вещественнозначная оценка от 0 до 1.

Итоговые формулы для оценивания

Накопленная оценка текущего контроля рассчитывается по формуле:

$$О_{накопленная} = (О_{ДЗ\_1} + О_{ДЗ\_2} + О_{ДЗ\_4} + О_{ДЗ\_5}) / 4$$

Где $О_{ДЗ\_n}$ – оценка за выполнение соответствующего домашнего задания, с учетом срока его выполнения (см. штрафные коэффициенты в параметрах текущего контроля).

В случае, если оценка за текущий контроль составляет менее чем 0,8, то в диплом выставляет результирующая оценка по учебной дисциплине, которая формируется по следующей формуле:

$$О_{результ} = 10 \cdot \mathbf{0,6} \cdot О_{накопл} + \mathbf{0,4} \cdot О_{итоговый}$$

Результирующая оценка выставляется по целочисленной шкале от 0 до 10, для этого накопленная оценка из шкалы [0..1] учитывается в результирующей с масштабом, равным 10, что отражено в формуле. Способ округления результирующей оценки по учебной дисциплине – арифметический, в пользу студента.

В случае, если оценка за текущий контроль составляет не менее, чем **0,8**, то студент освобождается от итогового контроля, а результирующая оценка определяется следующим образом:

Оценка за текущий контроль	**Результирующая оценка (по 10-бальной шкале)**
$0,8 \le О_{накопл} < 0,9$	8
$0,9 \le О_{накопл} < 0,95$	9
$0,95 \le О_{накопл}$	10

8 Содержание дисциплины

Раздел 1. Технологии разработки программного обеспечения и интерпретируемые языки программирования

Тема 1.1. Системы контроля версий

1. Обзор систем контроля версий CVS, Subversion, Mercurial, GIT.
2. Алгоритм сравнения текстовых файлов. Его реализация в утилите diff.
3. Основные понятия систем контроля версий: хранилище (origin), правка (commit), ветка (branch).
4. Принцип работы с системой контроля версий GIT. Организация рабочего процесса. Основы ветвления и слияния.

Основная литература

1. Scott Chacon. ProGit. 2012.
   Перевод на русский язык доступен в электронном виде для некоммерческого использования по адресу: https://github.com/downloads/GArik/progit/progit.ru.pdf

Дополнительная литература

1. L. Bergroth and H. Hakonen and T. Raita (2000). "A Survey of Longest Common Subsequence Algorithms". SPIRE (IEEE Computer Society) 00: 39–48. doi:10.1109/SPIRE.2000.878178
2. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. Version Control with Subversion. O'Reilly Media, 2004. 320 p.

Тема 1.2. Интерпретируемые языки программирования

1. Классификация языков программирования
2. Типизация языков программирования и способы ее классификации
3. Формальные грамматики и их использование в реальных системах программирования
4. Особенности интерпретируемых языков программирования на примере Python.

Основная литература

1. Бизли, Дэвид М. Язык программирования Python. Справочник. — К.: ДиаСофт, 2000. — 336 с.

Дополнительная литература

1. А. Ахо., Моника С. Лам, Р. Сети, Дж. Ульман. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2015. 1184 С.

Раздел 2. Взаимодействие интерпретаторов с программным кодом, реализованном на компилируемом языке программирования

Тема 2.1. Проектирование и разработка динамических библиотек

1. Стадии компиляции исходных текстов программ на языках Си и Си++. Промежуточные файлы, получаемые в ходе компиляции
2. Системные вызовы ядра и библиотечные функции. Различия в реализации
3. Размещение кода программы и библиотек в адресном пространстве процесса
4. Проблема адресации глобальных переменных и функций, перемещаемый код.

Основная литература

1. Александреску А. Современное проектирование на C++: Обобщенное программирование и прикладные шаблоны проектирования. — С. П.: Вильямс, 2008. — 336 с.
2. Александреску А., Саттер Г. Стандарты программирования на C++. — С. П.: Вильямс, 2008. — 224 с.
3. А. Робачевский, С. Немнюгин. Операционная система UNIX. Спб: БХВ, 2010. 656 С.

Дополнительная литература

1. Саттер Г. Решение сложных задач на C++. — М.: Вильямс, 2008. — 400 с.
2. Э. Таненбаум, Х. Босс. Современные операционные системы. Спб.: Питер, 2016. 1120 С.

Тема 2.2. Внедрение интерпретаторов в произвольные приложения

1. Классы решаемых интерпретаторами задач и необходимость их внедрения в существующие приложения
2. Способы использования существующих интерпретаторов на уровне программного интерфейса
3. Реализация механизма выполнения кода на интерпретируемом языке программирования с доступом к состоянию и функциональности хост-системы

Основная литература

1. Бизли, Дэвид М. Язык программирования Python. Справочник. — К.: ДиаСофт, 2000. — 336 с.
2. Справочное руководство по языку Lua: http://www.lua.ru
3. Справочное руководство по Python/C API: https://docs.python.org/3/c-api/index.html

Тема 2.3. Расширение функциональности интерпретаторов

1. Проблема скорости выполнения программ на интерпретируемых языках программирования
2. Внутреннее устройство интерпретатора Python
3. Реализация внешних модулей для интерпретатора на примере Python

Основная литература

1. Бизли, Дэвид М. Язык программирования Python. Справочник. — К.: ДиаСофт, 2000. — 336 с.
2. Справочное руководтсво по Python/C API: https://docs.python.org/3/c-api/index.html

Раздел 3. Устройство и реализация компиляторов различных языков программирования

Тема 3.1. Языки для платформы Java

1. Язык программирования Java и реализации Java ME/SE/EE
2. Базовые и объектные типы Java
3. Виртуальная машина JVM
4. Байткод виртуальной машины JVM, его текстовое представление
5. Языки программирования, предназначенные для выполнения виртуальной машиной JVM

Основная литература

1. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала, том 1. Основы. 9-е издание. — М.: «Вильямс», 2013. — 864 с.
2. Oracle Java Virtual Machine Specification. Chapter 6. The Java Virtual Machine Instruction Set: http://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html

Дополнительная литература

1. The Java Virtual Machine Specification, Second Edition, Tim Lindholm and Frank Yellin, Addison-Wesley, 1999, ISBN 0-201-43294-3.

Тема 3.2. Низкоуровневое представление программ

1. Принцип работы центрального процессора. Регистры, флаги, обращение к внешней памяти.
2. Язык ассемблера AVR
3. Языки ассемблера x86 и x86_64
4. Стек вызова функций, регистры SP и BP
5. Соглашения об использовании регистров при вызове функций для архитектур x86 и x86_64

Основная литература

1. Р. Э. Брайант, Д. Р. О'Халларон. Компьютерные системы: архитектура и программирование. Спб.: БХВ, 2005.
2. Руководство (на русском языке) по GNU Assembler: https://www.opennet.ru/docs/RUS/gas/
3. Справочник по командам x86: http://ref.x86asm.net/geek.html

Дополнительная литература

1. Atmel AVR 8-bit Instruction Set. Instruction Set Manual. Atmel Corporation, 2014.

Тема 3.3. Устройство компиляторов и генерация кода

1. Формальные грамматики
2. Стадии компиляции
3. Машинный код
4. Платформо-независимый код LLVM
5. Бинарное и текстовое представление кода LLVM, использование LLVM в качестве языка программирования

Основная литература

1. А. Ахо., Моника С. Лам, Р. Сети, Дж. Ульман. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2015. 1184 С.
2. LLVM Language Reference Manual: http://llvm.org/docs/LangRef.html

Дополнительная литература

1. Chris Lattner. "Introduction to the LLVM Compiler System". Plenary Talk, ACAT 2008: Advanced Computing and Analysis Techniques in Physics Research, Erice, Sicily, Italy, Nov. 2008.
2. Chris Lattner. "LLVM and Clang: Advancing Compiler Technology". Keynote Talk, FOSDEM 2011: Free and Open Source Developers European Meeting, Brussels, Belgium, Feb. 2011.

Тема 3.4. Механизмы контроля выделения и освобождения памяти и язык Rust

1. Уборщик мусора и подсчет ссылок во время выполнения программы
2. Выделение памяти в стеке и в куче
3. «Умные» указатели: уникальный указатель, разделяемый указатель, слабый указатель
4. Язык программирования Rust
5. Механизм владения объектами в языке Rust. Правила заимствования переменных
6. Обработка исключительных ситуаций
7. Представление Rust-программ на промежуточном языке LLVM
8. Связывание кода Rust и Си/Си++

Основная литература

1. Rust Documentation: https://doc.rust-lang.org
2. David Kieras. Using C++11's Smart Pointers. EECS Department, University of Michigan, May 2015
3. Александреску А. Современное проектирование на C++: Обобщенное программирование и прикладные шаблоны проектирования. — С. П.: Вильямс, 2008. — 336 с.

Раздел 4. Практическое применение функционального программирования

Тема 4.1. Введение в функциональное программирование

1. Классификация современных языков программирования по парадигмам
2. Рекурсивные функции и рекурсивные типы данных
3. Чистые и не чистые рекурсивные функции. Чистые функциональные языки программирования
4. Мемоизация и хвостовая рекурсия
5. Представление рекурсивных функций в виде императивных программ с использованием циклов и наоборот
6. Функции высшего порядка
7. Функции map, fold (reduce) и filter. Представление map и filter через fold
8. Язык программирования Erlang
9. Связываение и присваивание переменных.

Основная литература

1. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: Наука, 1986. – 368 с.
2. Joe Armstrong. Programming Erlang: Software for a Concurrent World. — Pragmatic Bookshelf, 2007. — 536 p.
3. Зыков С. В. Введение в теорию программирования. Функциональный подход. – М.: ДМП Пресс, 2008.

Дополнительная литература

1. Цикл статей Erlang для самых маленьких. Хабрахабр: http://habrahabr.ru/post/195542/
2. Душкин Р. В. Функциональное программирование на языке Haskell / Гл. ред. Д. А. Мовчан;. — М.: ДМК Пресс, 2008. — 544 с.
3. Чезарини Ф., Томпсон С. Программирование в Erlang. — М.: ДМК Пресс, 2012. — 488 с.

Тема 4.2. Теоретические аспекты разработки параллельных программ

1. Графовые модели программ
2. Ярусно-параллельная форма программы. Анализ возможности параллельного выполнения инструкций
3. Способы устранения зависимостей при распараллеливании. Функции с побочными эффектами и чистые функции
4. Планирование задач в системах с общей памятью
5. Кооперативная многозадачность. «Зеленые» нити, и их реализация в языках программирования
6. Вытесняющая многозадачность. Алгоритмы Round Robin и Multilevel Queue
7. Синхронизация нитей с использованием блокировок
8. Функция Compare-And-Swap. Неблокирующие структуры данных
9. Модель параллельного выполнения программ в Erlang

Основная литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Спб.:БХВ, 2002. 602 С.
2. Joe Armstrong. Programming Erlang: Software for a Concurrent World. — Pragmatic Bookshelf, 2007. — 536 p.
3. C.Evequoz Non-Blocking Concurrent FIFO Queues With Single Word Synchronization Primitives. 13th Ada-Europe International Conference on Reliable Software Technologies, Venice, Italy, June 16-20, 2008. pp. 59-72

Дополнительная литература

1. Энтони Уильямс. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. М.:ДМК-Пресс, 2012. 672 С.

Тема 4.3. Сети и распределенное выполнение программ

1. Устройство стека TCP/IP
2. Работа с сокетами с использованием POSIX API
3. Механизмы очереди ядра в системах Linux и BSD. Прокси-сервер nginx
4. Реализация HTTP сервера. Команды протокола HTTP. Протокол REST
5. Приложения Erlang/OTP
6. Приложение inets из стандартной библиотеки Erlang/OTP
7. Веб-сервер cowboy

Основная литература

1. Э. Таненбаум, Д. Уэзеролл. Компьютерные сети. Спб.: Питер, 2012. 960 С.
2. Joe Armstrong. Programming Erlang: Software for a Concurrent World. — Pragmatic Bookshelf, 2007. — 536 p.
3. Erlang/OTP Documentation: http://erlang.org/erldoc
4. Cowboy 1.0 User Guide: http://ninenines.eu/docs/en/cowboy/1.0/guide/

Дополнительная литература

1. Maxim Sokhatsky. Erlang Web Stack. Erlang Factory San Francisco, 2014. http://slides.com/maximsokhatsky/n2o/

Раздел 5. Современные языки программирования для Web

Тема 5.1. Платформа Node.js и языки программирования семейства JavaScript

1. Языки ECMAScript и JavaScript
2. Использование JavaScript в браузерах. Библиотека jQuery
3. Модель ООП с использованием прототипов
4. Интерпретатор Node.js и пакетный менеджер npm
5. Язык программирования TypeScript
6. Язык программирования Dart
7. Компиляция программ на C++ в программы на ECMAScript

Основная литература

1. Дэвид Флэнаган. JavaScript. Подробное руководство, 6-е издание. – М.: O'Reilly, «Символ-Плюс», 2012. – 1080 с
2. Nathan Rozentals Mastering TypeScript. Packt Publishing. 2015. 364 p.
3. Kathy Walrath, Seth Ladd Dart: Up and Running. A New, Tool-Friendly Language for Structured Web Apps. O'Reilly Media. 2012. 158 p.

Дополнительная литература

1. Alon Zakai. Emscripten & Asm.js: C++'s role in the modern web. CppCon, 2014. http://kripken.github.io/mloc_emscripten_talk/cppcon.html

Тема 5.1. Языки программирования Dart и Go. Создание сетевых сервисов с использованием Веб-сокетов

1. Язык программирования Dart. Общая характеристика и использование
2. Язык программирования Go. Общая характеристика и использование
3. Параллельное выполнение функций в языке Go. Взаимодействие через общую память и обмен сообщениями
4. Функциональность стандартной библиотеки языков Go и Dart для создания веб-сервисов
5. Взаимодействие браузера с сервером. Механизмы AJAX и взаимодействие через веб-сокеты.

Основная литература

1. Ivo Balbaert, Dzenan Ridjanovic. Learning Dart. Second Edition.Packt Publishing, 2015. 368 p.
2. Dart SDK Reference: https://api.dartlang.org/
3. Nathan Kozyra. Mastering Concurrency in Go.Packt Publishing, 2014. 328 p.
4. A Tour of Go: https://tour.golang.org/
5. RFC 6455 — The WebSocket Protocol. Internet Engineering Task Force (IETF), 2011.

Дополнительная литература

1. BeeGo Getting Started: http://beego.me/quickstart

9 Образовательные технологии

9.1 Методические рекомендации преподавателю

Оформлено в виде приложения.

9.2 Методические указания студентам

Рекомендуется использовать комплект программного обеспечения, входящий в один из дистрибутивов операционной системы Linux. Возможно использование операционной системы Linux внутри виртуальной машины.

10 Оценочные средства для текущего контроля и аттестации студента

10.1 Тематика заданий текущего контроля

Примерное домашнее задание по разделу 1. Технологии разработки программного обеспечения и интерпретируемые языки программирования

Реализуйте на языке Python вычисление значений формул в ячейках CSV-таблицы. Формулы могут содержать арифметические операции, подвыражения в скобках, а также использовать математические, либо написанные пользователем функции на языке Python.

Примерное домашнее задание по разделу 2. Взаимодействие интерпретаторов с программным кодом, реализованном на компилируемом языке программирования

Реализуйте Python-модуль, который использует в своей работе внешнюю Си/Си++ библиотеку. Модуль должен реализовать выполнение текстов программ на языке Lua, принимая текстовую строку программы в качестве аргумента, и возвращать результат работы программы.

Примерное домашнее задание по разделу 4. Практическое применение функционального программирования

Реализуйте на Erlang модель выдачи денег банкоматом: по запросу числа — необходимой суммы, выдается список чисел — номиналов купюр, таким образом, чтобы число купюр было минимальным. Модель подразумевает, что у банкомата ограниченное количество купюр, и ситуация, когда все купюры заканчиваются являются аварийной, для которой должна быть предусмотрена операция перезапуска банкомата (имитация «ATM out of service» и его устранения). Интерфейс взаимодействия с моделью: HTTP/REST.

Примерное домашнее задание по разделу 5. Современные языки программирования для Web

Реализуйте веб-приложение: игру «Крестики-нолики на бесконечном поле». Серверная часть может быть реализована на одном из языков программирования: Erlang, JavaScript, Dart или Go; клиентская часть — на одном из языков JavaScript, Dart или TypeScript. Взаимодействие между сервером и клиентом должно осуществляться в реальном времени, с использованием механизма Веб-сокетов.

10.2 Вопросы для оценки качества освоения дисциплины

Примерный перечень вопросов к экзамену (итоговый контроль) для самопроверки студентов.

1. Хранение файлов в системе контроля версий. Атомарные изменения, способ их получения, идентификации и применения к дереву исходных текстов
2. Принцип работы с системой контроля версий GIT. Организация рабочего процесса. Основы ветвления и слияния
3. Типизация языков программирования и способы ее классификации. Примеры типизации в различных современных языках программирования
4. Формальные грамматики и их использование в реальных системах программирования. Пример использования контекстно-зависимой грамматики
5. Стадии компиляции исходных текстов программ на языках Си и Си++. Промежуточные файлы, получаемые в ходе компиляции. Статические и динамические библиотеки
6. Размещение кода программы и библиотек в адресном пространстве процесса. Проблема адресации глобальных переменных и функций, перемещаемый код
7. Классификация языков программирования по способу выполнения программного кода
8. Скорость выполнения программ на различных языках программирования
9. Механизм вызова функций (процедур). Его реализация на компьютерах с архитектурой процессора x86/x86_64
10. Фон-Неймановская и Гарвардская архитектуры вычислительных систем. Аппаратные и программные прерывания
11. Механизмы выделения и освобождения памяти. Уборка мусора и подсчет ссылок
12. «Умные» указатели. Их виды, назначение, внутреннее устройство и области применимости
13. Рекурсивные функции. Чистые рекурсивные функции, и функции с побочными эффектами
14. Хвостовая рекурсия. Мемоизация
15. Функции высшего порядка MAP, FILTER и REDUCE (FOLD). Представление MAP и FILTER через FOLD
16. Эквивалентность частично-рекурсивных функций, и процедур с использованием циклов

17. Ярусно-параллельная форма программы. Анализ возможности параллельного выполнения инструкций
18. Кооперативная многозадачность и «зеленые» нити
19. Вытесняющая многозадачность. Алгоритмы планирования Round Robin и Multilevel Queue
20. Взаимодействие между параллельно выполняемыми задачами. Механизмы синхронизации в системах с общей памятью. Неблокирующие структуры данных
21. Сетевые уровни стека TCP/IP. Механизм передачи данных на примере обработки запроса от браузера к веб-серверу
22. Механизм взаимодействия по протоколу HTTP. Команды HTTP и протокол REST
23. Модель объектно-ориентированного программирования с использованием прототипов

10.3 Примеры практических заданий итогового контроля

1. Реализовать Python-модуль с функцией, выполняющей конкатенацию двух строк
2. Реализовать интерпретатор Python с использованием библиотеки libpython
3. Реализовать интерпретатор Lua с использованием библиотеки liblua
4. Реализовать рекурсивную функцию вычисления числа Фибоначчи, которая требует для своей работы фиксированный объем памяти
5. Реализовать рекурсивную функцию вычисления Факториала, которая требует для своей работы фиксированный объем памяти

11 Учебно-методическое и информационное обеспечение дисциплины

11.1 Базовый учебник

Ридер, составленный по источникам.

11.2 Основная литература

1. Scott Chacon. ProGit. 2012.
   Перевод на русский язык доступен в электронном виде для некоммерческого использования по адресу: https://github.com/downloads/GArik/progit/progit.ru.pdf
2. Бизли, Дэвид М. Язык программирования Python. Справочник. — К.: ДиаСофт, 2000. — 336 с.
3. Александреску А. Современное проектирование на C++: Обобщенное программирование и прикладные шаблоны проектирования. — С. П.: Вильямс, 2008. — 336 с.
4. Александреску А., Саттер Г. Стандарты программирования на C++. — С. П.: Вильямс, 2008. — 224 с.
5. А. Робачевский, С. Немнюгин. Операционная система UNIX. Спб: БХВ, 2010. 656 С.
6. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала, том 1. Основы. 9-е издание. — М.: «Вильямс», 2013. — 864 с.
7. Р. Э. Брайант, Д. Р. О'Халларон. Компьютерные системы: архитектура и программирование. Спб.: БХВ, 2005.
8. А. Ахо., Моника С. Лам, Р. Сети, Дж. Ульман. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2015. 1184 С.
9. David Kieras. Using C++11’s Smart Pointers. EECS Department, University of Michigan, May 2015
10. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: Наука, 1986. – 368 с.
11. Joe Armstrong. Programming Erlang: Software for a Concurrent World. — Pragmatic Bookshelf, 2007. — 536 p.
12. Зыков С. В. Введение в теорию программирования. Функциональный подход. – М.: ДМП Пресс, 2008.
13. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Спб.:БХВ, 2002. 602 С.

14. C.Evequoz Non-Blocking Concurrent FIFO Queues With Single Word Synchronization Primitives. 13th Ada-Europe International Conference on Reliable Software Technologies, Venice, Italy, June 16-20, 2008. pp. 59-72
15. Э. Таненбаум, Д. Уэзеролл. Компьютерные сети. Спб.: Питер, 2012. 960 С.
16. Дэвид Флэнаган. JavaScript. Подробное руководство, 6-е издание. – М.: O'Reilly, «Символ-Плюс», 2012. – 1080 с
17. Nathan Rozentals Mastering TypeScript. Packt Publishing. 2015. 364 p.
18. Kathy Walrath, Seth Ladd Dart: Up and Running. A New, Tool-Friendly Language for Structured Web Apps. O'Reilly Media. 2012. 158 p.
19. Ivo Balbaert, Dzenan Ridjanovic. Learning Dart. Second Edition.Packt Publishing, 2015. 368 p.
20. Nathan Kozyra. Mastering Concurrency in Go. Packt Publishing, 2014. 328 p.
21. RFC 6455 — The WebSocket Protocol. Internet Engineering Task Force (IETF), 2011.

11.3 Дополнительная литература

1. L. Bergroth and H. Hakonen and T. Raita (2000). "A Survey of Longest Common Subsequence Algorithms". SPIRE (IEEE Computer Society) 00: 39–48. doi:10.1109/SPIRE.2000.878178
2. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. Version Control with Subversion. O'Reilly Media, 2004. 320 p.
3. А. Ахо., Моника С. Лам, Р. Сети, Дж. Ульман. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2015. 1184 С.
4. Саттер Г. Решение сложных задач на C++. — М.: Вильямс, 2008. — 400 с.
5. Э. Таненбаум, Х. Босс. Современные операционные системы. Спб.: Питер, 2016. 1120 С.
6. The Java Virtual Machine Specification, Second Edition, Tim Lindholm and Frank Yellin, Addison-Wesley, 1999, ISBN 0-201-43294-3.
7. Atmel AVR 8-bit Instruction Set. Instruction Set Manual. Atmel Corporation, 2014.
8. Chris Lattner. "Introduction to the LLVM Compiler System". Plenary Talk, ACAT 2008: Advanced Computing and Analysis Techniques in Physics Research, Erice, Sicily, Italy, Nov. 2008.
9. Chris Lattner. "LLVM and Clang: Advancing Compiler Technology". Keynote Talk, FOSDEM 2011: Free and Open Source Developers European Meeting, Brussels, Belgium, Feb. 2011.
10. Nicolas Geoffray, Gael Thomas, Julia Lawall, Gilles Muller, and Bertil Folliot "VMKit: a Substrate for Managed Runtime Environments". Proc. of the Virtual Execution Environments Conference (VEE '10), Pittsburgh, PA, Mar. 2010. (доступно в электронном виде http://llvm.org/pubs/)
11. Цикл статей Erlang для самых маленьких. Хабрахабр: http://habrahabr.ru/post/195542/
12. Душкин Р. В. Функциональное программирование на языке Haskell / Гл. ред. Д. А. Мовчан;. — М.: ДМК Пресс, 2008. — 544 с.
13. Чезарини Ф., Томпсон С. Программирование в Erlang. — М.: ДМК Пресс, 2012. — 488 с.
14. Энтони Уильямс. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. М.:ДМК-Пресс, 2012. 672 С.
15. Maxim Sokhatsky. Erlang Web Stack. Erlang Factory San Francisco, 2014. http://slides.com/maximsokhatsky/n2o/
16. Alon Zakai. Emscripten & Asm.js: C++'s role in the modern web. CppCon, 2014. http://kripken.github.io/mloc_emscripten_talk/cppcon.html

11.4 Справочники, словари, энциклопедии

1. Справочное руководство по языку Lua: http://www.lua.ru
2. Справочное руководтсво по Python/C API: https://docs.python.org/3/c-api/index.html
3. Oracle Java Virtual Machine Specification. Chapter 6. The Java Virtual Machine Instruction Set: http://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html
4. Руководство (на русском языке) по GNU Assembler: https://www.opennet.ru/docs/RUS/gas/
5. Справочник по командам x86: http://ref.x86asm.net/geek.html

6. LLVM Language Reference Manual: http://llvm.org/docs/LangRef.html
7. Rust Documentation: https://doc.rust-lang.org
8. Erlang/OTP Documentation: http://erlang.org/erldoc
9. Cowboy 1.0 User Guide: http://ninenines.eu/docs/en/cowboy/1.0/guide/
10. Dart SDK Reference: https://api.dartlang.org/
11. A Tour of Go: https://tour.golang.org/

11.5 Программные средства

Для успешного освоения дисциплины, студент использует следующие программные средства:

- Операционная система Linux
- Набор компиляторов C/C++ GCC и Clang. Рекомендуется использовать Linux-версии этих программ.
- Система управления версиями Git.
- Интерпретаторы Python 3.x и Lua, в полной поставке, включая компоненты для разработчика
- Система управления сборкой CMake.
- Текстовый редактор с подсветкой синтаксиса. Рекомендуется любая среда разработки на языке C++.
- Система Erlang/OTP.
- Компиляторы LLVM, Rust и Go.
- Среда выполнения Dart.
- Среда Node.js и пакетный менеджер npm.
- Современный веб-браузер, желательно на базе WebKit или Chromium.

12 Материально-техническое обеспечение дисциплины

Необходимы современные персональные компьютеры по количеству студентов в подгруппе. На компьютерах необходимо наличие установленной операционной системы семейства Linux, либо технические характеристики компьютеров должны позволять работу с виртуальной машиной VirtualBox, и выделением виртуальной машине не менее 1Гб оперативной памяти.