• увеличивается временная нагрузка на преподавателя и по проверке лабораторных работ в удаленном доступе;

• не все студенты достаточно ответственно относятся к работе в онлайн режиме, не могут работать без постоянного контроля со стороны преподавателя, не хватает самодисциплины.

Заключение. Успешность процесса обучения пропорциональна соблюдению основных его принципов, таких как: научность, последовательность и системность, доступность, наглядность, индивидуализация, сознательность и активность в обучении, развитие самостоятельности, прочность полученных знаний и сформированных умений и навыков. С учетом данных принципов разрабатываемые нами ресурсы позволяют эффективно организовать процесс обучения как за счет более полного формата представления учебного материала по сравнению с охваченным во время аудиторных занятий, так и возможности выбора каждым студентом индивидуальной траектории изучения материала, тренинга и самоконтроля, использования прозрачной системы диагностики, коррекции и контроля знаний студентов.

Это говорит о высокой перспективности развития данного педагогического подхода и необходимости его внедрения в учебный процесс по каждой дисциплине.

Литература

1. Прокашева В. А., Лысак В. В. Об опыте обучения в экстремальных ситуациях с использованием интернет технологий в БГУ / Веб-программирование и интернет-технологии WebConf2021 : материалы 5-й Междунар. науч.-практ. конф., Минск, 18–21 мая 2021 г. / Минск : БГУ, 2021. – С. 293-295.

2. Лысак, В.В. Опыт использования виртуальной среды обучения Moodle в преподавании некоторых микробиологических курсов / В.В. Лысак, Г.А. Расолько // Международный конгресс по информатике: информационные системы и технологии (в образовании) [Электронный ресурс] : материалы междунар. науч. конгресса, Респ. Беларусь, Минск, 22–23 окт. 2020 г. / Белорус. гос. ун-т. – Минск : БГУ, 2020. – С. 121–126. https://elib.bsu.by/handle/123456789/249839

НУЖНА ЛИ МАТЕМАТИКА ПРОГРАММИСТАМ В ИТ?

Романчик В. С.

Белорусский государственный университет, г. Минск

В докладе рассматриваются вопросы подготовки студентов по программированию (веб-программированию) и математике в университете и дальнейшее их обучение и трудоустройство в ИТ.

Как готовить программиста в ВУЗе

Что надо изучить сначала в программировании? Программирование – это отдельная отрасль инженерной науки. Начальные знания и умения программиста дают в средней школе: владение компьютером как пользователь, знание английского языка и математики на невысоком уровне, понятие алгоритма, способы описания алгоритмов, работа с документами, мобильные коммуникации и Интернет.

Быстрое дальнейшее вхождение в мир программирования дает веб-программирование. При этом возможен следующий начальный выбор программиста для Фронтэнда: HTML, CSS, JavaScript. Логичным является и выбор JavaScript в качестве

первого языка программирования. Однако здесь традиционно присутствуют C++, а также Java и C# и Python.

Студенты БГУ специальности “Математика и информационные технологии” начиная с первого курса изучают C/C++, который является прародителем большого числа других языков. Параллельно изучаются HTML, CSS, JavaScript и БД.

Таким образом, на базовом уровне (1-й – 2-й курсы) студенты изучают следующие дисциплины, связанные с программированием:

- Английский язык;
- Языки программирования (C++, Java и C#, Python, PHP);
- Технологии программирования;
- HTML, CSS, JavaScript;
- Алгоритмы и структуры данных;
- SQL и базы данных;
- Операционные системы и сети;

Какие компетенции нужны программисту на базовом уровне:

1. Общие понятия: ООП, шаблоны проектирования, тестирование, стэк, поток и пр.
2. Быть продвинутым выше базового уровня хотя бы в одном языке.
3. Операционные среды и компьютерные сети.
4. Уметь читать чужой код.
5. Системы контроля версий GIT.
6. Знать стандартные алгоритмы.
7. Уметь работать в команде.

Нужна ли программисту математика?

Ниже перечислены предметы, входящие в базовый курс математики и программирования:

1. Линейная алгебра. Векторы. Матрицы.
2. Дискретная математика и теория графов.
3. Деревья. Структуры данных.
4. Реляционная алгебра.
5. Построение и анализ алгоритмов.
6. Математическая логика и булева алгебра, множества.

Предметы математики для изучения на углубленном уровне:

7. Теория вероятностей и математическая статистика.
8. Вычислительная геометрия и инфографика.
9. Исследование операций, теория игр, моделирование процессов.
10. Нейросети и глубокое машинное обучение.
11. Теоретико-числовые методы в криптографии.
12. Дифференциальные уравнения, численные методы и моделирование.

Один из основных математических предметов математический анализ здесь не входит в перечисление. Можно сказать в оправдание, что матан подставляет хребет и представляет связку для самой математики, которую без интегралов и дифференциалов представить трудно.

Разделы программирования, где нужны знания математики:

1. Анализ данных и машинное обучение.
2. Предсказание и анализ вероятностей.
3. Оптимизация хранения, проектирование хранилищ, облачные технологии.
4. Цифровая обработка сигналов.
5. Компьютерное зрение.

6. Распределенные вычислительные системы.

7. Анализ производительности распределенных вычислительных систем.

8. Моделирование – описание реальных объектов и процессов в формальных терминах.

9. Криптография. Здесь речь идет о фундаментальных принципах сохранения информации приватной.

На этом этапе студент должен выбрать направление и язык программирования, в которых будет дальше углубляться. Возможно, хорошим выбором будет Java. Замечательный язык Python будет очень полезен математикам, статистикам, т.к. открывает им дверь в мир Data Science.

Знание SQL и английского языка дают шанс попасть на работу тестировщиком.

Современные подходы к компьютерной подготовке в ИТ

В последнее время системы подготовки и продвижения кадров получили развитие и применение в виде матриц компетенций и карт развития. Матрица представляет таблицу, столбцами которой являются компетенции или категории компетенций. В строке стоит оценка степени продвинутости специалиста.

Примеры компетенций для junior на Фронтэнд: HTML – CSS – JavaScript – React or Angular or Vue. В Интернет существует также огромное количество других компетенций для Junior, Это JSON, Suss, Bootstrap, Visual Studio Code, Git, Debugging, Regular Expression, Canvas, Design Principles, Refactoring. Еще язык разметки типа MarkDown и графику типа Figma(кроме Fotoshop).

Все компетенции удовлетворить невозможно Что же делать? Ответ: углубленно готовить одну две компетенции, например React. Достичь этих компетенций студент должен сам. В остальном базовое обучение ему дают в университете или на курсах в ИТ.

Что делает backend-веб-разработчик? Как оценить уровень?

Знания PHP/Java/C#/NodeJS/Python; Знания SQL и DB; работа с технологиями и фреймворками; взаимодействия с сервером Unix.

Подготовка здесь подобна предыдущей: углубленно и самостоятельно готовить одну две компетенции, при этом ценятся не знания, а умения.

Как реально попадают студенты в большие ИТ компании?

Наиболее простой путь через ИТ курсы и внешний тренинг в самой ИТ компании. Внешний тренинг проводит EPAM, бесплатные курсы – ПВТ, IBA и др. Реально студенты проходят достаточно легкое собеседование. После этого: 1) они зачисляются на трехмесячные курсы в определенном направлении; 2) проходят фильтрацию после курсов; 3) зачисляются на три месяца в менторинг – лаборатории, в которых выполняют учебные и тестовые проекты; 4) оставшиеся после менторства студенты зачисляются на предпродакшен и выполняют какую-то работу; 5) из этого резерва зачисляются на продакшен и конкретные проекты.

Таким образом ИТ – компании “затачивают” студентов под определенные технологии и проекты. Это необходимо для существования и роста самой компании. Широкое образование и математика на этом этапе не нужна по характеру выполняемых задач. Математика понадобится, когда junior станет сеньором, но к этому времени математика забудется. В принципе существует система переподготовки, но научиться с возрастом решать, например, задачи моделирования достаточно сложно.

Таким образом, существуют проблема нестыковки широкого университетского математического образования с конкретным ИТ обучением и работой. Полученные математические знания для junior не востребованы.

Ресурсы по математике для программистов в Интернет

Чтобы развиваться в программировании, необходимо знать хотя бы основы дискретной математики, линейной алгебры, теории вероятностей, криптографии, геометрии и статистики.

Онлайн-курсы по математике

1. Основы линейной алгебры, Техасский университет в Остине
2. Введение в математическое мышление, Стэнфорд
3. Введение в дискретную математику, Калифорнийский университет Диего
4. Математика для программистов, Pluralsight
5. Криптография 1, Стэнфорд
6. Теория игр, Стэнфорд и Университет Британской Колумбии
7. Наука о данных и математика, Университет Дьюка
8. Многомерный математический анализ, MIT
9. Введение в теорию вероятностей, Гарвард
10. Введение в теорию вероятностей – наука о неопределенности, MIT
11. Математика для машинного обучения, Имперский Колледж Лондона

Русскоязычные курсы по математике

1. Онлайн-курс по математике в Data Science Lite, Библиотека программиста
2. Онлайн-курс по математике в Data Science Pro, Библиотека программиста
3. Линейная алгебра
4. Основы перечислительной комбинаторики
5. Теоретическая информатика: сложность вычислений
6. Основы дискретной математики
7. Алгоритмы и структуры данных
8. Теория вероятностей
9. Основы статистики

Литература

1. https://www.coursera.org/learn/datasciencemathskills
2. https://www.coursera.org/specializations/mathematics-machine-learning?newQueryParams=%5Bobject+Object%5D#faq-list
3. https://stepik.org/course/2461/promo

УСТОЙЧИВОСТЬ, КАК ЭКОНОМИКО-МАТЕМАТИЧЕСКАЯ КАТЕГОРИЯ

[1]Самаль С.А., [2]Катан П.И.

[1]Белорусский государственный университет, Минск
[2]Европейский университет Молдовы, Кишинев

Терминология и практика ее применения для научных и образовательных целей зачастую использует понятия, которые в различных предметных областях подразумевают отличающиеся, хотя зачастую, и достаточно близкие значения. Одним из таких терминов в математике и экономике является понятие *устойчивость.*

Являясь фундаментальным понятием кибернетики и общей теории систем там под *устойчивостью* подразумевают способность системы с достаточно сложным поведением оставаться стабильной, сохранять ключевые свойства и характеристики неизменными. Если система обладает рассматриваемым свойством, зачастую