ПОДГОТОВКА СТУДЕНТОВ ПО ПРОГРАММИРОВАНИЮ И ТРУДОУСТРОЙСТВО В ИТ

Романчик В. С.

Белорусский государственный университет, Минск, Беларусь,
e-mail: romanchikvs@gmail.com

В докладе рассматриваются вопросы подготовки студентов по программированию (веб-программированию) в университете и дальнейшее их обучение и трудоустройство в ИТ. Здесь рассматривается только начальная должность для студента-выпускника junior. Набор для middle и senior происходит в отдельном потоке и здесь не рассматривается.

Процедура трудоустройства

Процедура рекрутинга постоянно изменяется, как и требования к квалификации нанимаемого специалиста. Рассмотрим сначала частично устаревшую процедуру приема на работу, которая применяется и сейчас в ИТ компаниях.

При поступлении на работу в ИТ-компанию каждый студент проходит собеседование / тестирование. Предполагаемая польза от тестирования: убрать субъективность оценки и сократить затраты на рекрутинг. Первый аспект тестирования представляется положительным, второй аспект особого выигрыша по затратам не дает. При этом погрешность при оценке может быть очень большая.

Рассмотрим подробнее проведение тестирования.

1. О технических тестах.

Подготовленные студенты (juniors) тесты проходят неплохо, поскольку у них уже есть опыт подготовки и сдачи тестов. Студенты могут подготовиться к тестированию, как и к любому экзамену. Один из лучших ресурсов для этой цели codility.com. Получает ли при этом работодатель правильные ответы на свои вопросы? Наверно, не всегда.

2. О психологических тестах.

Тестируемый может выдавать либо социально желаемые ответы, либо отвечать по приколу, либо «как придётся». Во всех случаях тестирующий получает ответы на уровне «понравился» «не понравился». Следовательно, тестирование не является лучшей формой рекрутинга.

Предпочтительной является одна из форм собеседования.

Подготовка к собеседованию.

1. Портфолио.

До прохождения собеседования претенденту желательно создать портфолио. Это программа или скрипт для сайта, которая делает работу, похожую на полезную. После правки и улучшения ее можно выставить, как свой код работодателю. При этом студенту могут быть заданы серьезные вопросы по тексту, чтобы оценить его текущую квалификацию. Имеющиеся публикации также включаются в портфолио. Средний балл успеваемости тоже рассматривается

2. Резюме.

По резюме можно многое узнать о человеке и его образовании. Желательно описать проекты, в которых участвовали. Интересные технические решения.

3. Поиск работы.

ИТ компании всегда искали опытных разработчиков и специалистов, но их не хватало. В настоящее время многие компании осознали, что проще и дешевле вырастить своего специалиста, чем взять опытного. Поэтому все поданные резюме рассматриваются. С претендентом связываются и приглашают на собеседование.

4. Собеседование.

Первый этап собеседований может проходить заочно. Работодатель может проверить компетенции и решить, нужно ли очное интервью. В случае положительного решения претендента приглашают на очное собеседование – профессиональное и общее.

Следует сказать, что кроме профессионализма многие компании хотят лояльности претендентов. Если хотите работать будьте готовы объяснить, почему именно хотите работать в данной фирме. Очень интересен отбор в компанию Амазон, в каком-то смысле достигший совершенства. Компания Амазон разработала принципы поведения, личной преданности для сотрудников. Основа системы отбора в Amazon – Amazon Leadership Principles. Эти принципы определяют работу компании практически во всём. При собеседовании просят рассказать истории, которые будут демонстрировать наличие каждого из 14 Leadership Principles у вас. Компания серьезная и это работает.

Разные вопросы и тесты на профессиональных собеседованиях прошлых лет.

Вопросы взяты из различных источников и Интернет.

1. Самый распространенный вопрос среди всех головоломок прошлых лет: Почему канализационные люки круглые?

2. Вы сидите в лодке, плавающей посреди озера. У Вас с собой на борту есть большой кирпич. Если выкинуть его в озеро, уровень воды увеличится? уменьшится? останется неизменным?

3. Дана строчка текста, переставить в ней все слова в противоположном порядке, так чтобы, например, строчка "Здесь был Вася" превратилась в "Вася был Здесь". Дополнительную память выделять не разрешается.

4. Какова сложность алгоритма сортировки “quicksort”? (O(n log n)).

Тест по языку C++.

1. Результат выполнения программы (AC~B~A)/

```
#include "iostream.h"
class A
{
public:
A() { cout<<"A"; }
A(char c) { cout<<c; }
~A() { cout<<"~A"; }
};
class B:public A
{
public:
B() { cout<<"B"; }
B(char c) { cout<<c; }
```

```
~B() { cout<<"~B"; }
};
int main(int argc, char* argv[])
{
B c('C');
return 0;
}
```

2. Выбрать результаты выполнения программы.(AC~A).

```
class A{
int *a;
public:
A() { a=new int[10]; cout<<"A"; }
A(char c) { a=new int[10]; cout<<c; }
~A() { delete a; cout<<"~A"; }
};
class B:public A
{
int *b;
public:
B() { b=new int[10]; cout<<"B"; }
B(char c) { b=new int[10]; cout<<c;}
~B() { delete b; cout<<"~B"; }
};
int main(int argc, char* argv[])
{
B *b=new B('C');
A *a=b;
delete a;
return 0;
}
```

3. Что делает приведённый ниже код (^ означает операцию XOR)?

```
int main(int argc, char* argv[])
{ int a=1, b=2;
a = b^a; b = b^a; a = b^a;
cout<<a<<b; //21
return 0;
}
```

Java

4. Что будет выведено на Java: (1 default 2 2 default 2)

```
class One
{ public static void testSwitch(int arg) {
switch (arg) {
case 1: System.out.print("1 ");
```

```
default: System.out.print("default ");
case 2: System.out.print("2 ");
}
System.out.println();
}
public static void main(String[] args) {
testSwitch(1);
testSwitch(2);
testSwitch(3);
}
}
```

Язык PHP

1. Что выведет следующая программа: (aaa bbb)

```
<?php
$a = "aaa";
$$a = "bbb";
echo "$a ${$a}";
?>
```

2. Что выведет следующая программа (012)

```
    <?php
function foo()
{
    static $counter = 0;
    echo $counter++;
}
foo();
foo();
foo();
?>
```

3. Что выведет следующая программа (3)

```
<?php
$str = "a\nb";
$fh = fopen('test.dat', 'wt');
fwrite($fh, $str);
fclose($fh);
echo strlen($str);
echo filesize('test.dat');
unlink('test.dat');
?>
```

4. Какой номер HTTP ошибки надо вернуть, для вывода стандартного диалога для запроса имени и пароля пользователя:

(401 – не авторизован)

Как готовить программиста в ВУЗе

Что надо изучить сначала в программировании?

Программирование – это отдельная отрасль инженерной науки. Начальные знания и умения программиста дают в школе: владение компьютером как пользователь, знание английского языка и математики на невысоком уровне, понятие алгоритма, способы описания алгоритмов, работа с документами.

Быстрое дальнейшее вхождение в мир программирования дает веб-программирование. При этом возможен следующий начальный выбор программиста: HTML, CSS, JavaScript. Логичным является и выбор JavaScript в качестве первого языка программирования. Однако здесь присутствуют 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.

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

- 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;

Системы контроля версий GIT.

Какие компетенции нужны программисту на базовом уровне:

1. Общие понятия: ООП, шаблоны проектирования, тестирование, стек, 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.
6. Знать стандартные алгоритмы
7. Уметь работать в команде.

Нужна ли программисту математика?

Здесь 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.

Разделы программирования, где нужны эти знания:

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 тестирования и собеседования 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. Например:

1. Computer Science

Структуры данных/Алгоритмы/Программирование систем

2. Software Engineering

Контроль версий исходного кода/Автоматизация сборки/Автоматическое тестирование

3. Программирование

Декомпозиция проблем/Декомпозиция систем/Организация дерева исходников /Читабельность кода/Защитное программирование/Обработка ошибок /IDE /API /Фреймворки/Требования/Скрипты/Базы данных

Матрицы компетенций представляют дальнейшее продвижение в направлении улучшения процесса тестирования и собеседования. При этом сначала определяются необходимые компетенции специалиста – это самостоятельная задача. Затем в процессе собеседования составляется матрица компетенций.

Примеры компетенций для разработчиков.

Что должен знать и уметь junior на Фронтэнд?

На первый взгляд HTML – CSS – JavaScript – React or Angular or Vue – Пишет unit-тесты – Владеет базовыми инструментами – Решает локальные задачи на фронтэнде.

В Интернет существует огромное количество компетенций для Junior, часть которых можно бы сюда добавить. 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 из Интернет удовлетворить невозможно. К тому же университетское образование предполагает широкую подготовку студентов, которая не позволяет узко готовить студентов. 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.

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; Front-end

Эти направления становятся “колонками” матрицы компетенций. Подготовка здесь подобна предыдущей: углубленно и самостоятельно готовить одну две компетенции, при этом везде 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. За толковыми студентами идет охота различных компаний.

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.

Примеры тренинговых курсов: Software Testing, Java, .NET, Business Analysis, Automated Testing, Android, BI, PHP, Data Science, Frontend. 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 научится с возрастом решать, например, задачи моделирования достаточно сложно.

Таким образом, существуют проблемы нестыковки широкого университетского образования с конкретным ИТ обучением и работой.

В современном информационном обществе широкое образование необходимо еще и потому, что оно дает возможность быстро сменить направление работы или работу на актуальную.

Литература

1. https://www.intervolga.ru/blog/life/web-developer-competence-matrix/
2. https://devpro.blob.core.windows.net/presentations/