ЧЕК-ЛИСТ

ПУТЬ В IT: УЧИМСЯ СВОИМИ СИЛАМИ

Практичный план с 10 до 15 лет
— без школы, но с системой

1 ЭТАП 1: 10–12 ЛЕТ — ПРОБУЕМ, ИГРАЕМ, ДУМАЕМ

- **Цель: понять, что такое логика и алгоритмы.**
- **Не кодим — мыслим.**
- **Что учить:**
  - **Основы логики**: условия, циклы, переменные.
  - **Визуальное программирование**: Scratch, MakeCode.
- **Ресурсы:**
  - **scratch.mit.edu** — бесплатно, интерактивно.
  - Курс **"Программирование для детей"** на Stepik.
  - **Любой материал** связанный с алгоритмикой в открытом доступе.
- **На что обращать внимание:**
  - **Не торопите.** Главное — чтобы ребенок сам захотел что-то улучшить или добавить новые элементы.
  - **Хвалите** не за результат, а **за попытку**: «Ты сам это придумал? Круто!»
  - Следите, чтобы он **не просто копировал** проекты, а **экспериментировал**.

2 ЭТАП 2: 12–13 ЛЕТ — ПЕРВЫЙ НАСТОЯЩИЙ КОД

Цель: перейти от блоков к тексту, увидеть, как работает веб.

Что учить:

- **HTML + CSS** — как устроен сайт.
- **JavaScript** (база): переменные, события, кнопки.
- **Git и GitHub** (начальный уровень).

Ресурсы:

- **metanit.com** — Основы веб разработки
- **learn.javascript.ru** — Основы программирования на JavaScript
- **freeCodeCamp.org** — модуль Responsive Web Design.
- **Scrimba.com** — интерактивные уроки по JS.
- **CodePen.io** — чтобы сразу видеть результат.

На что обращать внимание:

- **Не заставляйте** ребёнка запоминать **все правила** (например, какой знак ставить после команды). В реальной жизни даже **профессионалы не держат всё в голове**.
- Давайте **задачи из жизни**: «Сделай страницу про свою любимую игру».
- Объясните, что **ошибки** — это не провал, а **подсказка**.

3 ЭТАП 3: 14–15 ЛЕТ — СВОЙ ПРОЕКТ

Цель: создать что-то своё, а не копию.

Что учить:

- **Один язык глубоко**: JS (фронтенд) или Python (боты, анализ).
- Простые **бэкенд-концепции**: API, fetch, JSON.
- Как развернуть сайт (**GitHub Pages**).

Проекты:

- Личный **сайт-визитка**.
- **Бот** для Telegram (на Python + aiogram).
- **Мини-игра** на JS.
- **Таблица дел** с сохранением в localStorage.

Ресурсы:

- **The Odin Project** (для веба).
- **Real Python** (статьи и туториалы).
- **SoloLearn** — для практики.

На что обращать внимание:

- **Не требуйте** идеального кода. Главное — чтобы **работало** и он мог объяснить, **как**.
- **Поощряйте** ребенка **делиться** своими работами в GitHub — это как **онлайн-портфолио** или **облачное хранилище** для кода, где профессионалы выкладывают свои проекты.
- **Знакомьте** его с **профессиональными сообществами**: Discord, Telegram-чаты по IT для подростков.

ЧЕГО СТОИТ ИЗБЕГАТЬ

Гнаться за количеством

- **Помогите** ребёнку **сконцентрироваться** на одном проекте и **довести** его до работающего состояния.
- Лучше сделать **одну законченную игру/сайт**, чем начать десять и ни один не закончить.
- Успешное завершение даёт **чувство гордости и уверенность.**

Учить всё сразу

- **Не стоит** сразу браться за **сложные технологии** (как React), если ещё не освоены **базовые знания** (как основной JS).
- Изучайте материал **постепенно**. Сначала пусть ребёнок поймёт основные принципы языка программирования.
- Только после того, как он **уверенно пользуется базой**, можно переходить **к более сложным и специализированным** инструментам.

Игнорировать ошибки

- Когда программа **ломается** и выдаёт сообщение об ошибке, **нельзя это пропускать**.
- Научите ребёнка **читать и понимать эти сообщения**, а не просто их игнорировать. Компьютер сам говорит, где именно он наткнулся на проблему.
- Умение находить **причину ошибки** (это называется "отладка" или "дебаг") — **самый важный навык** программиста, намного более важный, чем простое запоминание команд.

ПОЛЕЗНЫЕ СОВЕТЫ

Делайте вместе мини-проект раз в месяц

- **Раз в месяц** (например, в одну из суббот) выделяйте 2-4 часа для **совместной работы** над одной маленькой задачей.
- Формирует **дисциплину** и привычку регулярно **применять знания**.

Смотрите короткие видео о реальных IT-профессиях

- Ищите видео не только о **программистах,** но и о **тестировщиках** (QA), **дизайнерах** (UI/UX), **аналитиках данных**, **менеджерах** проектов.
- Выбирайте контент, который показывает **реальные рабочие будни**, а не только успех и большие зарплаты.
- Это даёт **реалистичное представление** о профессии, снижает иллюзии и готовит к будущему выбору.

Раз в полгода делайте «выпускную работу»: презентация проекта перед семьёй.

- Это должна быть **не просто демонстрация**, а короткая **публичная презентация**.
- Помогите ему продумать **структуру**: "В чём была проблема?", "Как я её решил?", "Что я сделал впервые?"
- Это учит **структурировать мысль** и чётко объяснять сложные идеи **простыми словами**.

+7 (495) 065-71-71

@adel_managers