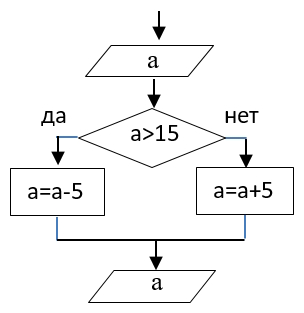
**От абстракции к образу или как я преподаю программирование.**

Актуальность развития алгоритмического мышления и навыков программирования в современном обществе, думаю долго описывать не нужно, достаточно сказать, что современный мир предусматривает глобальное проникновение IT – технологий во все сферы нашей жизни. Это, в свою очередь, формирует запрос рынка труда на IT - специалистов, и как следствие, востребованность и успешность выпускников, выбравших IT – технологии для дальнейшей профессиональной деятельности или хотя бы имеющих навыки в IT – сфере.

Я преподаю информатику в общеобразовательной школе, а также веду занятия по программированию в учреждении дополнительного образования «Детско-юношеский центр «Планетарий» имени космонавта Анны Кикиной. Это дает возможность работать с разными детьми: специально подготовленными, которые прошли конкурсный отбор и обучаются в инженерных классах; с детьми, которые имеют высокую степень мотивации и интереса к дополнительным занятиям программированием; а также с детьми, которые вынуждены заниматься программированием на уроках в школе (без особого интереса и склонностей к данному виду интеллектуальной деятельности).

В этой статье разговор пойдет в большей степени о преподавании программирования для школьников 6-7 классов, которые обучаются не в специализированных классах. У таких ребят, чаще всего, еще не сформировалось абстрактное мышление, они не обладают достаточными математическими знаниями и навыками. А программирование – это абстракция плюс математика, причем, чем глубже изучается программирование, тем более абстрактными и математически емкими становятся конструкции, которыми приходится оперировать. Как решить эту проблему? Нужно перевести абстракцию в образ! Образ может быть любым: от рисунка до игровых технологий, главное, чтобы образ был знаком ребенку, затрагивал его жизненный опыт. Считаю, что огромное преимущество в понимании программирования, дает опыт программирования Черепашки. Черепашка интуитивно понятна, ее действия легко представить и проиграть в живую. Поэтому ученики пятых классов в школе и младшие группы в планетарии начинают именно с нее. Далее работаю на Python или С++. Изложение своего опыта я приведу в примерах для указанных языков.

Итак, **пример 1.** Тема «Структура программы». Основные понятия: ввод, обработка, вывод. Вызываю к доске «смелого, уверенного в себе», сообщаю, что летом планируется менять в моем кабинете линолеум, прошу вычислить площадь кабинета. Варианты развития ситуации: «дайте рулетку – зачем? – измерить размеры», «скажите размеры». Говорю размеры, обычно 8 метров на 6 метров. Ребенок дает ответ. Далее прошу детей проанализировать разыгранную сценку и разбить наши действия на этапы: получил размеры – ввод данных, перемножил – обработка данных, озвучил результат – вывод данных. Далее привожу операторы ввода/вывода и решаю несложные, желательно веселые, задачи бытового характера. Пример задачи: «В саду росли две вишни. Вася залез на первое дерево, Петя на второе. Вася ест вишни со скоростью n вишен в минуту, Петя со скоростью m вишен в минуту. Через t минут сторож увидел мальчиков и выгнал из сада. Сколько всего вишен успели мальчики съесть?»

**Пример 2.** Задача «Поменять местами значения двух переменных». Ставлю два стакана с разными жидкостями или прошу детей это представить (можно заменить стаканы на баночки с гвоздями и саморезами). Можно сделать на них наклейки: a и b. Прошу поменять содержимое местами. Вопрос о третьем стакане (банке) подымается детьми сразу! Проделываем действия в жизни, прописываем в программе.

**Пример 3.** Тема «Условный оператор + сложные условия». Обязательно привожу блок-схему. А дальше ее проигрываем. Ребенок получает карточку с числом, подходит к столу, открывает карточку с условием (числа и условия меняются). Если условие срабатывает, обходит стол слева и выполняет указанные в блок-схеме действия, если нет, то справа и так же выполняет указанные в блок-схеме действия, но уже другую ветку.

Что бы запомнить как работают логические связки and и or, проговариваю:   
- если будет хорошая погода и будет выходной, мы идем в поход – чтобы пойти в поход должны выполняться оба условия;

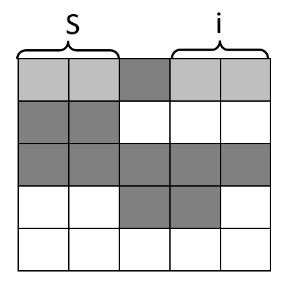
- если дома кончился хлеб или молоко, идем в магазин – чтобы пойти в магазин должно выполниться хотя бы одно условие.

Далее предлагаю каждому ребенку придумать свои ключевые фразы, которые ему «близки». Сложные условия обязательно привязываю к заданию №6 КИМ ОГЭ. Оговариваем с детьми, что связка and означает «вместе», связка or – «достаточно одного». Так же оговариваем, где лежит карточка «Да», где карточка «Нет». Далее проигрываем по очереди все наборы от конкретной задачи. Только теперь два участника играют роль двух условий, распределяем кто, за какое условие играет. Если условие, за которое играет ребенок, выполняется, то он подымает руку. При связке and должны быть подняты руки у обоих детей, тогда они берутся за руки и идут к карточке «Да», показывают ее. Иначе, ребята, также держась за руки, идут к карточке «Нет» и демонстрируют ее классу. Аналогично проигрываем задачу со связкой or, но теперь оговариваем, что достаточно одной поднятой руки. Сначала разбираем задачи на и/или, потом усложняем их (уже без проигрывания) связкой not в различных вариациях.

**Пример 4.** Тема: «Обработка данных в цикле. Инициализация переменных». Рассматриваю простейшую задачу на цикл: «Найти сумму чисел от 1 до N». Сначала обязательно демонстрирую сумму ряда на конкретном примере. Пусть N = 5, тогда надо найти: 1 + 2 + 3 + 4 + 5. Пишу программу, сначала не акцентируя внимания на инициализации начального значения суммы, обсуждаю с какого значения цикл начинается, каким заканчивается и почему.

s=0

for i in range(1,5+1):

 s=s+i

print(s)

А далее подчеркиваю строчку s=s+i и задаю провокационный вопрос: сколько в ней действий? Обычно следует ответ: «одно». На мое веселое: «А вот и нет», начинается такое же веселое гадание: 2, 3 и т.д. Когда я озвучиваю число 4, у детей внимание уже максимально сконцентрировано на вопросе: «Почему!?» и мотивации понять, то, что идет в разрез с их предыдущим опытом.

Далее я рисую клетчатое поле и говорю, что оперативная память имеет ячеистую структуру и каждой переменной отводится в этой структуре свое место. Есть ячейки уже занятые (закрашенные), есть свободные (не закрашенные). Закрашиваю две ячейки (важно не одну, это далее позволит детям прийти к пониманию разных типов данных, особенно если в дальнейшем они перейдут на строго типизированный язык), пишу над ними s. Еще две ячейки именую символом i.

Далее проговариваю, что запись s=s+i сначала подразумевает получение значений из ячеек памяти с именами s и i (это 1 и 2 действие), затем вычисление их суммы (3 действие), а далее результат суммирования перезаписывается в ячейки памяти с именем s (4 действие). Внимание! Ключевое слово перезапись. Оно дает четкое понимание, что прежнее значение стерто, а записано новое.

Далее я закрываю рукой строчку s=0 и задаю второй каверзный вопрос: «А что лежит в ячейке, до того как, мы туда первый раз что-то перезаписали, ведь первое действие – взять, а записать только четвертое?». Даю время детям пофантазировать, и если не слышу правильный ответ, то к радости детей сообщаю, что там лежит цифровой мусор, который остался от работы какой-то другой программы. А если взять мусор, к нему прибавить что-то, то получится мусор, а не правильный ответ. И вот для этого мы должны в начале программы положить в будущую сумму нейтральное значение 0. Уточняю, почему я его назвала нейтральным.

**Пример 5.** Задача «Поиск максимума из n чисел». Пояснение принципа решения данной задачи можно провести как на карандашах или других подручных материалах, так и на обучающихся. С участием ребят получается веселее, а главное - продуктивнее. Вызываю ребят, строю их друг за другом. Оговариваем, что широкий шаг вперед означает ввод числа. Дети быстро приходят к выводу, что нужно взять первого (ввести первое число до цикла) и сделать его эталоном сравнения, далее взять второго и сравнить его эталоном, выбрать большего и сделать его эталоном. И это проделать со всеми остальными. В итоге эталоном (максимумом) будет самый высокий.

**Пример 6.** Тема «Массивы». Сами по себе массивы сложны в понимании. Я их объясняю на примере паровозика: индекс элемента массива – это номер вагона, а сам элемент – это количество пассажиров в вагоне. Иногда демонстрирую набор пронумерованных с нуля спичечных коробков, внутри которых лежат скрепки или кнопки. Если дети не очень сильные играю в игру: каждый ребенок получает номер. Я объявляю, что наш массив имеет имя, например, 9b и хранит имена. Далее называю 9b[5], 9b[7]. Ребята говорят свои имена. Меняю тип массива, например, полное количество лет. Снова называю элементы, ребята озвучивают свой возраст. Еще один важный момент, пусть в группе 12 человек, я называю 9b[13]. Обычно, после небольшого замешательства, обучающиеся говорят: «А у нас такого нет!». Акцентирую их внимание, что у нас возникла ошибка обращения к несуществующему элементу. Для объяснения поиска по массиву обычно рисую таблицу в две строки, верхняя строка – индексы, нижняя – сами элементы. И обязательно рисую каретку, которая указывает на исследуемый элемент, постепенно сдвигаясь в процессе обработки массива.

**Пример 7.** Сортировки массива методами выбора и пузырька. В школе в обычных классах, чаще всего, я не даю сортировки, в инженерных же, мы изучаем их уже в 8 классе. В Планетарии к сортировкам подходим на второй год изучения языка Python в средних возрастных группах (7-9 классы) и в первый же год в старших (работают на С++). Принцип объяснения такой же, как и при поиске по массиву: таблица + каретка (↓). Только теперь прописываю несколько строк таблицы, чтобы обучающиеся поняли какую роль играет внешний цикл (положение каретки и начало диапазона поиска), а какую внутренний цикл (проход по диапазону поиска).

Сортировку методом пузырька стандартно проигрываем в лицах: есть обучающийся, играющий роль переменной – флага (true - рука поднята, false – рука опущена), 4-5 человек разного роста - элементы массива. Я сама играю роль каретки.

Даю через внешний цикл while. Проигрываем сортировку по возрастанию, а далее все, что проигрывали переводим в код.

С ростом опыта необходимость «ситуационного проигрывания» снижается и хватает простой таблицы трассировки, которая по сути, тоже выступает образом.

На протяжении многих лет собираю изюминки: как лучше ту или иную абстракцию перевести в образ. Иногда идеи подкидывают сами дети, например, история про банки с гвоздями и саморезами: ребенок не мог понять алгоритм, так как не видел причины менять содержимое местами. Понял, когда кто-то из друзей догадался привести пример полной маленькой баночки с гвоздями и почти пустой большой банки с саморезами: «А у тебя еще гвозди есть, а класть их некуда». Я всегда поощряю ситуации, когда понявший, объясняет тому, кто еще не понял. Чем доступнее для понимания образ ты или сам ребенок подберет, тем глубже и качественнее усвоение материала.

Итак, перевод абстракции в образ, с целью качественного понимания процесса, это необходимая и неотъемлемая черта преподавания программирования в 6-7 классах и более старших ребят с не сформированным абстрактным мышлением и не достаточными математическими знаниями и навыками. Продуманно подобранными к абстракции образами, мы не только добиваемся качественного усвоения материала, но и повышаем мотивацию детей к освоению перспективного и востребованного в современном мире направления профессиональной деятельности.