ИСПОЛЬЗОВАНИЕ МЕНТАЛЬНЫХ СХЕМ ПРИ ОБУЧЕНИИ СОВРЕМЕННЫМ ЯЗЫКАМ ПРОГРАММИРОВАНИЯ

Ашуров Мухиддин Улджаевич
Старший преподователь, кафедры
«Методика преподования информатики»
ТГПУ им.Низами
Ашурова Муниса Мухиддиновна
Докторант 3-курса ГулГУ

Аннотация: В данной статье обупликовано вопросы ментальных схем и их использования при обучении языкам программирования и совершенствовании профессиональных навыков будущих учителей информатики.

Ключевые слова: интерактивность, когнитивный подход, визуализация, интегрированное обучение, интеллект-карта, ментальная схема.

В процессе глобализации и образовательной интеграции в образовательный процесс применяются креативные технологии развития профессиональной подготовки будущих специалистов. Ведётся планомерная работа по включению профессиональной компетентности квалифицированного специалиста в число важных качеств, изучению состояния подготовки будущих специалистов к профессиональной деятельности, совершенствованию инновационных методов педагогической диагностики и их эффективному и результативному использованию, применить их на практике. Грамотный подход к образованию в мировых образовательных и научно-исследовательских учреждениях, повышение конкурентоспособности выпускников, развитие профессиональной компетентности педагогов, создание современного методического обеспечения проектирования интегрированного образовательного процесса и

развитие научных способностей студентов и проводятся научные исследования по развитию их способностей. При этом особое внимание уделяется научно-исследовательским работам по совершенствованию педагогических механизмов развития профессиональных компетенций у студентов на основе модернизации содержания профессионального образования, формирования инновационной образовательной среды, основанной на компетенциях, широкого внедрения. интерактивных методов и технологий обучения на практике.

Основные задачи когнитивного подхода – подготовить обучающегося воспринимать, запоминать и применять новую информацию в разных условиях, создавая разные стратегии. Педагогу необходимо определить связь между новым потоком информации, подаваемым в процессе обучения программированию, и имеющимися у студента знаниями и попытаться создать новые ментальные схемы. Многие исследователи предполагают, что ментальные схемы используются человеческим мозгом для облегчения обработки и запоминания новой информации.

Ментальная схема (mind map — при переводе с английского языка означает слово карта разума) — техника визуализации мыслей, которая используется для понимания и обработки информации. В классических моделях этих схем основная идея в центре карты, а все дополнения нарисованы вокруг неё.

Важно выбрать идеи и их составляющие и определить их связи для использования при обучении языку программирования ментальных схем. При объяснении новой темы или ряда тем эти схемы помогают творчески подойти к определению интересных способов донесения текстовой информации до читателя. Сегодняшние схемы основаны на базовой концепции когнитивного подхода, под которой когнитивистью понимают одну из психологических структур, которые мозг организует для хранения и использования информации.

По результатам исследований В. В. Калитиной, использование

ментальных схем при обучении программированию оказывает эффективное воздействие на повышение уровня качества, повышение интереса учащихся к алгоритмам и программированию, увеличение объема информации, сохраняемой в памяти, стимулирование творческих процессов [1].

Учёные Дорошенко Е.Г. и Пак Н.И. размышляя о ментальных учебниках, высказали нижеследующие мнения о пользе ментальных схем. «...Визуальная картина лучше передает сознание, формирует причинно-следственные связи, уменьшает время усвоения и понимания информации, увеличивает скорость принятия решений, не интерпретирует кратковременную и непровизорную память» [2].

В современную эпоху использование ментальных схем также входит в число элементов информационных технологий, об этом Казачев В.Н. высказывает в своей статье нижеследующие мнения: «Важно научить студентов использовать информационные технологии в профессиональной деятельности и повседневной жизни. Это, в свою очередь, помогает студентам успешно усваивать большие объемы информации». [3]

Будущим учителям уместно предоставить информацию о создании ментальных схем и их использования с учетом этих ситуаций. На первом этапе рекомендуется подготовить информацию о некоторых понятиях, затем рекомендовать их учащимся, а затем приступить к созданию ментальной схемы. Это соответствует рекомендациям некоторых исследователей методу создания ментальных схем «по пунктам». Этот метод часто используется в традиционном обучении и очень прост. В самом общем виде процесс выглядит так:

Это отлично подходит для студентов, которая находяться на лекциях и семинарах, учаться у репетиторов, занимаються с помощью учебников и других источников. (например, для ментальной схемы, которую можно построить, изучая свойства кортежей, списков и множеств в Python)

На первом этапе проводится анализ необходимых данных (сходств и различий), а затем создается схема.

Информации о кортежах:

• Кортежи — это особая структура величин, состоящая из упорядоченных и инвариантных элементов.

• последний элемент кортежа равен -1.

• Кортеж формируется путем разделения элементов запятыми и желательно заключаться в круглые скобки, когда это возможно.

• Создается с помощью круглых скобок ().

• cor1 = (1,2,3,'a') ; Вывод команды print(type(kor1)): <класс 'кортеж'>

• type(parameter) — это внутренняя функция, определяющая тип данного параметра

• Кортежи могут состоять из коллекций разных размеров.

• Состав шествия не может быть изменен после его объявления.

• Кортеж можно объявить без круглых скобок, т. е. cor1 = (1,2,3,4,5) и

cor1 = 1,2,3,4,5 дают одинаковый результат.

• После создания кортежа невозможно изменить его элемент, удалить его или добавить в него элемент.

• Элемент кортежа можно «прочитать», что экономит время выполнения программы по списку.

• Почему при выполнении следующей программы выдается ровно один результат: kor4 = (1, 1, [3,4],1) ; печать (идентификатор (cor4 [0])); печать (идентификатор (cor4 [1])); печать (идентификатор (cor4 [3]))

• В каких случаях можно добавить элемент в кортеж и изменить значение элемента.

• Методы add(), Extend(), Remove() и pop() не используются для кортежей.

• Оператор in может использоваться для определения наличия элемента в кортеже.

• первый индекс элементов кортежа равен -0

• отрицательные значения могут использоваться в качестве индекса.

• count(x) — возвращает номер элемента x в кортеже (функции).

• index(x) — возвращает индекс элемента x в кортеже.

• **Any()-функция**

если элемент кортежа существует, тогда True возвращает элемент

в противном случае (если кортеж пуст) False возвращает элемент.

• Функция max() возвращает максимальный элемент кортежа.

• Функция min() возвращает минимальный элемент кортежа.

• Функция len() возвращает длину кортежа.

• Функция tuple() используется для преобразования других коллекций в кортежи.

• Функция sorted() возвращает новый отсортированный список элементов кортежа.

• Функция sum() возвращает сумму элементов кортежа.

• чтобы проверить, принадлежит ли элемент кортежу,используется ключевого слова in

• Возможно создание частей кортежа с помощью специальных разрезов.

• [первый индекс (входящий): последний индекс (не входящий): шаг] используется в разрезах (он равен 1, если шаг не указан)

• [:] и [::] используются для заполнения исходного кортежа.

• [::-1] меняет местами элементы кортежа.

• Кортежи объединяются операцией «+».

• Пустой кортеж можно создать путем умножения любого кортежа на целое число 0 или меньше.

• Новый кортеж можно создать путем умножения любого кортежа на целое число больше 0.

Список (лист)

• Поскольку каждый элемент в списке упорядочен, мы можем ссылаться на любой элемент по его порядковому номеру (индексу).

• Список создается с использованием квадратных скобок [].

• Оператор del используется для удаления элемента из списка по индексу.

• Оператор удаления используется для удаления элемента из списка по значению.

• Чтобы использовать один из элементов списка, в Python используется метод .pop(index).

• Первый элемент списка начинается с 0.

• Последний элемент списка равен -1.

• Содержимое списка может меняться во время работы программы.

• В список могут быть добавлены новые элементы, некоторые элементы могут быть удалены.

• С помощью метода .append() вы можете добавить новый элемент в конец списка, этот метод также используется для заполнения пустого списка.

• Чтобы добавить новый элемент в список, мы используем метод .insert(), индекс и значение нового элемента задаются в методе Insert().

• В большинстве случаев может потребоваться отсортировать элементы списка в алфавитном порядке. Для этого мы используем специальный метод

.sort() для списков.

• Чтобы отсортировать список в обратном порядке, введите аргументverse=True в методе .sort().

• список можно отсортировать с помощью функции sorted(list name).

• список можно отсортировать с помощью функции sorted(list name,verse=True).

• Используя методverse(), вы можете перевернуть элементы списка.

• Чтобы определить количество элементов списка, мы используем функцию len().

• С помощью функции range() мы можем создать последовательность чисел в определенном диапазоне, а с помощью функции list() сохранить этот диапазон в виде списка.

• Мы также можем указать шаг в функции range().

• В Python вы можете использовать функцию min() для поиска наименьшего числа в списке, функцию max() для поиска наибольшего числа и функцию sum() для поиска суммы чисел.

• Иногда может потребоваться извлечь определенную часть списка, например, мы делаем новый список из 4 элементов из списка чисел. Для этого показываем индекс диапазона начального и последнего элементов.

• Вы можете создавать части списка, используя специальные разрезы.

• [первый индекс (входящий): последний индекс (не входящий): шаг] используется в разрезах (он равен 1, если шаг не указан)

• [:] и [::] используются для заполнения исходного списка.

• [::-1] меняет местами элементы списка.

Эта же информация подготовлена и для сборника. Ниже приведен пример одной из созданных ментальных схем.

Коллекция-SET {.....}
1. Elementlarni takrorlanmaydigan, tariblanmagan va indekslanmagan ro'yxat.
2. set() funksiyasi kortej yoki ro'yxatni to'plam ko'rinishiga aylantiradi.
3.nomi.add () -yangi element qo'shadi.(bo'sh to'plamga element qo'shib bo'lmaydi).
3.nomi.discard()- ko'rsatilgan elementni o'chiradi.
4.nomi.clear()-ro'yxat yoki to'plamni tozalaydi, [] yoki set() ko'rinishida natijani chop etadi.

Общие свойства
1Функция .len() возвращает длину;
2. max()- самый большой элемент; min() — находит наименьший элемент.
3. sorted() — переходит в отсортированный список.
4.sum() — определяет сумму.
5. Ключевое слово in используется для проверки актуальности элементаэлементfoydalaniladi.

В заключение отметим, что ментальные схемы оказывают эффективное воздействие на углубленное изучение предмета студентами в процессе преподавания и обучения. Благодаря объему информации и потоку информации в современной системе образования использование интеллект-карт в процессе обучения показывает положительные результаты, поскольку помогает учащимся выделить, систематизировать и запомнить основную информацию и использовать ее в процессе обучения в будущем. Кроме того, использование ментальных карт в учебном процессе способствует развитию творческого, критического мышления учащихся.

Список использованной литературы

1. В. В. Калитина. Методика ментального обучения программированию студентов информационных направлений.Педагогические науки. Теория и методика обучения и воспитания.

2. Е.Г. Дорошенко, Н.И.Пак, Н.В.Рукосева, Л.Б.Хегай. О технологии разработки ментальных учебников. //Вестник ТГПУ, 2013, 12 (140) стр.148

3. Казагачев, В. Н. Ментальные карты как средство повышения творческого мышления / В. Н. Казагачев, Л. Г. Горбань, Я. И. Толочко. // Молодой ученый. — 2015. — № 7 (87). — С. 766-769.

4. Ashurova M.M. (2023). O’yinlar orqali talabalarning dasturlashga bo’lgan motivasiyasini oshirish. ПЕРСПЕКТИВЫ СОВРЕМЕННОЙ НАУКИ И ОБРАЗОВАНИЯ, 1(1), 204-209.

5. M.U. Ashurov, M.M.Ashurova (2023).The Role and Significance of the Concepts of Hard Skill and Soft Skill in Teaching It and Programming Languages. Journal of Pedagogical Inventions and Practices, 18 (1), 8-70.