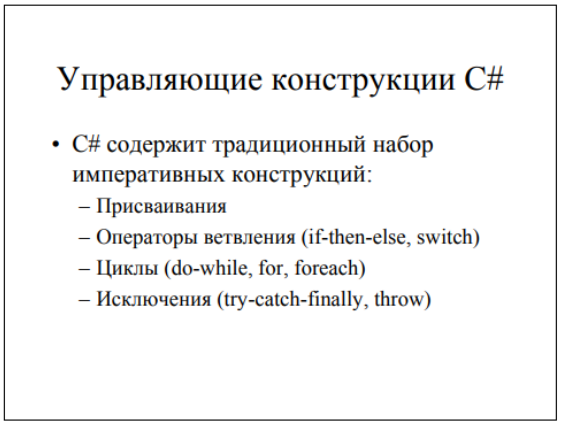
Школа:		
Дата:	ФИО учителя: Бекниязова Замзагуль Канатовна	
Класс:	Участвовали:	Не участволвали:
Тема урока: Основные управлящие конструкции C#		
Цели обучения, которые достигаются на данном уроке	Обучающая: ознакомить обучающихся с правилами оформления операторов управления, научиться использовать их при решении задач; Развивающая: развивать логическое мышление, внимание, память в процессе решения задач;	
Цели урока	**Все учащиеся смогут:** Определять разницу между управляющими конструкции **Большинство учащихся смогут:** Использовать и правильно оформлять операторов управления **Некоторые учащиеся смогут:** Научаться использовать их при решении программирований	
Критерии оценивания	Познакомятся с правилами оформления операторов управления. Научаться использовать их при решении задач	
Воспитание ценностей	Уважение к себе и к другим при постановке задач проекта через работу в парах и группе.	
Предварительные знания	Проектная деятельность	
Межпредметные связи	информатика, математика, программирование	
Запланированные этапы урока	Запланированная деятельность на уроке	Ресурсы
Начало урока **5 мин**	Приветствие, проверка присутствующих. Объявление темы и целей урока. Делим детей на 3 группы при помощи игры Random  Актуализация знаний «Мозговой штурм». (5 минут). Метод «Толстые и тонкие вопросы» (для начала беседы по изучаемой теме).	
Середина урока **25мин**	Начнём с ввода данных. Для того, чтобы дать возможность пользователю ввести данные, используется функция **input** без параметров. Эта функция возвращает значение, которое пользователь ввёл с клавиатуры в строку. Рассмотрим её запись. Все функции в языке Python записываются в составе инструкций. Для вызова функции записывается её имя, после которого в скобках следуют её параметры. Так как функция **input** не имеет параметров, после её имени должны следовать пустые скобки. Так как программа записывает данные в переменную, то результат работы этой функции присваивается некоторой переменной. Таким образом, для считывания значения переменной **a** с клавиатуры нужно записать инструкцию	

присваивания переменной **a** значения функции **input ()**.

Для вывода данных из оперативной памяти компьютера на экран монитора используется инструкция **print**. Мы уже пользовались ей раньше. Вспомним, как она записывается. После служебного слова **print** в скобках следует список выводимых данных. Как и в любой другой операции, в инструкции вывода могут указываться литералы, переменные и выражения.

Итак, мы узнали инструкции, используемые для ввода и вывода данных, теперь давайте попробуем их использовать в программе. Напишем модуль, который принимает на ввод 2 целых числа и выводит на экран их сумму. Начнём написание модуля. Числа мы будем хранить в переменных **a** и **b**. Поэтому в начале модуля запишем инструкцию присваивания переменной **a** значения функции **input ()**. Дальше мы запишем такую же инструкцию для переменной **b**. После этого мы вычислим значение суммы этих переменных и выведем его на экран с помощью функции **print**. Для этого, после служебного слова **print**, в скобках запишем выражение: **a + b**.

```
a = input () b = input () print (a + b)
```



Большинство современных языков программирования содержит приблизительно одинаковый набор конструкций управления и C# не предлагает в этой области ничего радикально нового. В C# используются обычные присваивания для простых переменных; структурные или массовые присваивания не поддерживаются. Операторы ветвления тоже достаточно традиционны (if, switch), но обладают двумя особенностями. Во-первых, условие в операторе if должно вырабатывать именно булевское значение (т.е. целого значения, вырабатываемого при присваивании недостаточно), а во-вторых, каждая ветка case внутри оператора switch должна содержать явное указание о дальнейшем потоке управления (т.е. либо break, либо goto на какую-то переменную, например, на метку другой ветки). Что касается циклов, то C# поддерживает вполне традиционные циклы, такие как dowhile, while-do и циклы с итерацией for, но помимо этого, поддерживает перебор массивов и коллекций с помощью оператора foreach, как в следующем примере: Hashtable ziphash = new Hashtable(); ... foreach (string zip in ziphash.Keys) { Console.WriteLine(zip + " " + ziphash[zip]); } В этом примере надо обратить внимание на то, что перебираемый класс должен поддерживать интерфейс IEnumerator, а также на тот факт, что каждый извлекаемый из коллекции объект явным образом приводится к заявленному типу перебора (в нашем примере это string). Наконец, C# поддерживает структурную обработку исключений с помощью конструкций try, catch и finally. Исключения можно генерировать и явным образом с помощью конструкции

throw.

Сохраним модуль и запустим его на выполнение. Первое число зададим равным 35. После ввода числа необходимо нажать клавишу «Enter». Второе число зададим равным 42. Очевидно, что по нашему замыслу программа должна была вывести на экран число 77, но вместо этого она вывела 3542. Хотя на самом деле это не названное число, а символьная строка, состоящая

	из четырёх цифр. Почему так произошло? Здесь нужно понимать, что пользователь, задавая данные с клавиатуры, вводит их в текстовой форме. То есть функция **input** возвращает данные типа **str**, а нам, по условию задачи, нужны целые числа, то есть данные типа **int**. Для того, чтобы эти данные получить, нам необходимо воспользоваться функцией преобразования типов. Она записывается по названию типа выходных данных. В нашем случае – это целые числа, то есть **int**. В качестве параметра функции задаются данные, которые необходимо преобразовать. Таким образом, чтобы с клавиатуры считать целочисленное значение в переменную **a**, нужно присвоить ей значение функции **int** от значения функции **input ()**. Изменим записанный нами модуль. В инструкциях присваивания возьмём выражения, следующие после знака равенства, в скобки, после чего перед скобками запишем слово **int**. a = int (input ()) b = int (input ()) print (a + b) После этого сохраним модуль и запустим его на выполнение. Введём те же числа, которые мы вводили до этого: 35 и 42. В этот раз на экран выведено предполагаемое значение – 77. Программа работает правильно	
Конец урока **10 мин**	К концу урока учащиеся научатся: Проведите работу по самооцениванию учащихся с помощью Лестницы успеха в рабочей тетради.	

Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными учащимися?	**Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?**	**Охрана здоровья и соблюдение техники безопасности**
1. По уровню поддержки 2.По роли в групповой работе	1.Самооценивание по шаблону 2.Обратная связь по итогам выполнения заданий, по итогам рефлексии	Правила ТБ при работе в кабинете, Психологический комфорт
Рефлексия по уроку Была ли реальной и доступной цель урока или учебные цели? Все ли учащиеся достигли цели обучения? Если ученики еще не достигли цели, как вы думаете, почему? Правильно проводилась дифференциация на уроке? Эффективно ли использовали вы время во время этапов урока? Были ли отклонения от плана урока, и почему?	В планирование урока включены активные формы организации урока:	