**Рекурсия**

**1 часть**

**Рекурсия** – это термин в программировании, означающий вызов функцией самой себя.

Примерно так выглядит рекурсия:



К примеру, создадим метод Make, который принимает какое-то целое число n и выводит на консоль число n – 1 (то есть i), а дальше вызывает себя же, только с новой переменной:



Дальше в методе Main вызовем метод Make и передадим ему число 4:



То есть получится, что метод Make будет печатать на каждой итерации какое-то число. Рассмотрим целую программу и попытаемся подумать, что выведется на консоль?



Варианты ответов:

- 2 1 0 0 1 0 0

- 3 2 1 0

- 2 1 0

1)«Всё верно! +3 балла»

2)«К сожалению +2 балла…»
3)«К сожалению +1 балл… Неожиданно?»

**2 часть**

Здесь вставим видео с пояснением «Рекурсия 1 задание».

Важно заметить, что рекурсия должна иметь конец. Иначе это уже будет бесконечная рекурсия… В нашем примере конец – это там, где i принимает значение меньшее нуля.



**1 практическое задание с паролем для Wi-Fi**

Я знал, что пароль может содержать буквы a, b и c в разных комбинациях… Не знал как именно…

namespace UlearnRecursion

{

 public class Program

 {

 static void MakeSubsets(char[] symbols, int position)

 {

 if (position == ...)

 {

 Console.WriteLine(new string(symbols));

 return;

 }

 for (char symbol = 'a'; symbol <= 'c'; symbol++)

 {

 symbols[position] = symbol;

 MakeSubsets(symbols, ...);

 }

 }

 public static void Main(string[] args)

 {

 var symbols = new char[] { 'a', 'b', 'c' };

 MakeSubsets(symbols, 0);

 }

 }

}

Нужно изучить код, пояснение к написанному коду:

MakeSubsets принимает массив с буквами и позицию.



Дальше заходим в метод и проверяем какое-то условие… И если это условие выполняется, то печатаем полученную строчку и выходим из метода, возвращаясь туда, где цикл ещё не закончен:



В цикле перебираем значения от a до c:



Что в итоге должно быть на консоли:



(На всякий случай, как это выглядит в Visual Studio)



Здесь нужно заполнить пропуски «…».

1. Пользователь вводит оба ответа правильно: «Всё верно! +3 балла»
2. Пользователь вводит правильно один из двух: «Хорошо, но где-то ошибка… +2 балла»
3. Пользователь вводит оба ответа неправильно: «К сожалению, неправильно… +1 балл»

Ответ:



**3 часть**

Здесь видео с кратким пояснением «Рекурсия задание 2».

**Всплывающее окно**

Нам уже известно, что рекурсия должна где-то заканчиваться, иначе она будет бесконечной… Почему это плохо?

Когда рекурсивно вызывается метод, каждый вызов лежит в стеке. Если у рекурсии есть условие, при котором она заканчивается, то, когда мы доходим до этого условия, вызов снимается со стека и стек, таким образом, не переполняется.

Когда у рекурсии нет конца, то стек переполняется вызовами и возникает ошибка «StackOverflowException»:

**4 часть**

Здесь будет маленький опрос на понимание рекурсии:



Что делает функция F?

* Вычисляет НОД x и у по алгоритму Евклида;
* Вычисляет значение xy;
* Вычисляет остаток от деления х на у;
* Вычисляет минимальный разряд десятичного числа х в котором стоит 0.

Если пользователь выбирает 4 вариант ответа: «Молодец! Всё верно, +1 балл!»

Иначе: «Нет, 0 баллов…»



Что делает функция G?

* Вычисляет НОД x и у по алгоритму Евклида;
* Вычисляет значение xy;
* Вычисляет остаток от деления х на у;
* Вычисляет минимальный разряд десятичного числа х в котором стоит 0.

Если пользователь выбирает 1 вариант ответа: «Молодец! Всё верно, +1 балл!»

Иначе: «Нет, 0 баллов…»

Постройте самостоятельно схему рекурсивных вызовов задачи с перебором паролей. Мы показали вам, как построить схему, если пароль может состоять из 2 букв, теперь нужно построить для трёх букв «abc».

Вариантов ответа и проверки не будет.

Составьте схему и проверьте результат:

(Здесь будет кнопка, которая переключит на результат)

Ответ в видео «Рекурсия схема»

Оцените себя сами, как вы справились с поставленной задачей?

- Моё решение полностью соответствует! Всё верно!

- Не совсем правильно, но я был очень близок…

- Не получилось, но я постараюсь…