# Численные методы решения дифференциальных уравнений первого порядка. Метод Эйлера.

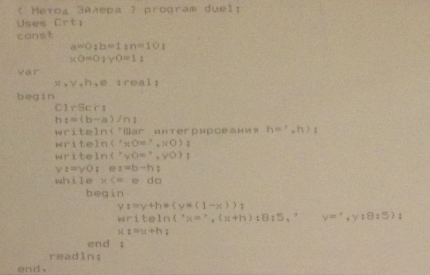
Постановка задачи. Решить обыкновенное д. у. на отрезке х [0,1] при начальных условиях y(0) = 1

Решение получается в виде интегральной кривой или таблицы.

**Решение:**

1. Вычисления ведем с числом разбиений n=10
2. Используем **рекурсивную формулу Эйлера**
3. Шаг вычисления h = (b - a) / n
4. Первый раз (начальные условия):
5. В программе используем идентификаторы:

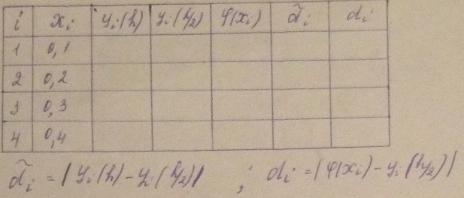
* a, b – начало и конец отрезка, на который ищем решение д.у.
* x0, y0 – начальные заначения.
* x, y, - текущие (вычисляемые) значения (это и есть решения д.у.)
* h – шаг вычисления
* n – число разбиений [a; b]

1. Программа для решения д.у. имеет вид:
2. Решить это д.у., изменив шаг интегрирования, уменьшив его до величины h = h/2
3. Используя отлаженную программу, решить д.у. на интервале [0; 0.4] с начальными условиями y(0)=1. Решение найти с шагом h = 0.1
4. Вычислить погрешности приближений (при решение уравнения ) при расчете с шагом h = 0.05:
   1. С помощью формулы:

(P=1) – порядок точности численного метода, Аналитически решение д.у. имеет вид:

* 1. Сравнив с точными значениями

1. Результаты вычислений удачно представить в таблице:



1. Получить решения д.у. в виде интегральной кривой:
   1. Сначала для контрольного примера
   2. Для д.у.

Для этого:

1. Используем ту же программу с некоторыми дополнениями и изменениями.
2. Дополнение:
   1. Инициализируем графику и для этого объявляем переменные GtaphDriver и GtaphMode типа Integer, а так же в начале основного блока программы пишем:

GraphDriver := Detect;

InitGtaph ( GtaphDriver, GtaphMode, ‘d:\TP7\BGI’);

* 1. В цикле на зкран выводим не x, y, а соответствующие этим координатам точки. Это делаем, используя функцию putpixel(200 + trune(x), 300 – trune(y),15)

**Пример программы Код на Python:**

def func(x, y):

return y \* (1 - x)

print('Число разбиений = ', end = '')

n = float(input())

x = 0.

y = 1.

a = 0

b = 1

h = (b - a) / n

print("Шаг(h) = ", h)

while(x <= b - h):

y += h \* func(x, y)

print("x = {:.3f}".format(x), " y = {:.3f}". format(y))

x += h