## «Формирование счета на оплату услуг»

По полученным результатам тарификации услуг «Телефония» и «Интернет» влабораторных работах 1, 2 сформировать счет на оплату в формате .pdf. Образец счета на оплату: https://kub-24.ru/schet-na-oplatu/

Все поля печатной формы должны заполняться разработанным программным модулем. Название банка, имена покупателей и прочие формальные поля можно заполнить какими угодно значениями, зависит от фантазии. Стоимость услуг услуг берется из предыдущих двух работ.

В качестве результата работы необходимо представить программный модуль для генерации печатной формы счета на оплату и полученный файл .pdf. Средства реализации выбираются студентом самостоятельно.

Вариант работы:

5. Протарифицировать абонента с IP-адресом 192.168.250.59   
с коэффициентом k:

* 1руб/Мб
* первые 1000Мб бесплатно

Протарифицировать абонента с номером 915642913 с коэффициентом k:

* 1руб/минута исходящие звонки,
* 1руб/минута входящие,
* смс - первые 5шт бесплатно, далее 1руб/шт

Обоснование выбранного средства реализации:

В качестве средства реализации мной был выбран язык программирования Python. Передо мной стоял выбор между Python и С. Не смотря на надежность языка программирования си, он не обладает такой обширной библиотекой готовых инструментов, как Python. Требования к работе и ее объем позволили мне использовать Python.

Ход работы

**Исходный код:**

import csv

import random

from docxtpl import DocxTemplate

doc = DocxTemplate("example.docx")

SPR\_RUB = ""

SPR\_KOP = ""

num2words = {1: 'ОДИН', 2: 'ДВА', 3: 'ТРИ', 4: 'ЧЕТЫРЕ', 5: 'ПЯТЬ', \

6: 'ШЕСТЬ', 7: 'СЕМЬ', 8: 'ВОСЕМЬ', 9: 'ДЕВЯТЬ', 10: 'ДЕСЯТЬ', \

11: 'ОДИННАДЦАТЬ', 12: 'ДВЕНАДЦАТЬ', 13: 'ТРИНАДЦАТЬ', 14: 'ЧЕТЫРНАДЦАТЬ', \

15: 'ПЯТНАДЦАТЬ', 16: 'ШЕСТНАДЦАТЬ', 17: 'СЕМНАДЦАТЬ', 18: 'ВОСЕМНАДЦАТЬ',

19: 'ДЕВЯТНАДЦАТЬ', 20:'ДВАДЦАТЬ', 30:'ТРИДЦАТЬ', 40:'СОРОК', 50:'ПЯТДЕСЯТ',

60: 'ШЕСТЬДЕСЯТ', 70:'СЕМЬДЕСЯТ', 80:'ВОСЕМЬДЕСЯТ', 90:'ДЕВЯНОСТО', 100:'СТО',

200: 'ДВЕСТИ', 300: 'ТРИСТА', 400: 'ЧЕТЫРЕСТА', 500: 'ПЯТЬСОТ', 600: 'ШЕСТЬСОТ',

700: 'СЕМЬСОТ', 800: 'ВОСЕМЬСОТ', 900: 'ДЕВЯТЬСОТ'}

num2words\_k = {1: 'ОДНА', 2: 'ДВЕ', 3: 'ТРИ', 4: 'ЧЕТЫРЕ', 5: 'ПЯТЬ', \

6: 'ШЕСТЬ', 7: 'СЕМЬ', 8: 'ВОСЕМЬ', 9: 'ДЕВЯТЬ', 10: 'ДЕСЯТЬ', \

11: 'ОДИННАДЦАТЬ', 12: 'ДВЕНАДЦАТЬ', 13: 'ТРИНАДЦАТЬ', 14: 'ЧЕТЫРНАДЦАТЬ', \

15: 'ПЯТНАДЦАТЬ', 16: 'ШЕСТНАДЦАТЬ', 17: 'СЕМНАДЦАТЬ', 18: 'ВОСЕМНАДЦАТЬ',

19: 'ДЕВЯТНАДЦАТЬ', 20:'ДВАДЦАТЬ', 30:'ТРИДЦАТЬ', 40:'СОРОК', 50:'ПЯТДЕСЯТ',

60: 'ШЕСТЬДЕСЯТ', 70:'СЕМЬДЕСЯТ', 80:'ВОСЕМЬДЕСЯТ', 90:'ДЕВЯНОСТО', 100:'СТО',

200: 'ДВЕСТИ', 300: 'ТРИСТА', 400: 'ЧЕТЫРЕСТА', 500: 'ПЯТЬСОТ', 600: 'ШЕСТЬСОТ',

700: 'СЕМЬСОТ', 800: 'ВОСЕМЬСОТ', 900: 'ДЕВЯТЬСОТ'}

BANK = "АО \"Банкрот Банк\" Г. НАБЕРЕЖНЫЕ ЧЕЛТЫ"

COMP = "ООО \"СВЯЗЬ FINGER-STICK\""

BIK = random.randint(100000000, 999999999)

print("generated BIK: ", BIK)

INN = random.randint(1000000000, 9999999999)

print("generated INN: ", INN)

KPP = random.randint(100000000, 999999999)

print("generated KPP: ", KPP)

ACC\_1 = '3010' + str(random.randint(10000, 99999)) + '0000000' + str(random.randint(1000, 9999))

print("generated ACC\_1: ", ACC\_1)

ACC\_2 = '4070' + str(random.randint(10000, 99999)) + '0000000' + str(random.randint(1000, 9999))

print("generated ACC\_2: ", ACC\_2)

ACC\_3 = random.randint(1, 99999)

print("generated ACC\_3: ", ACC\_3)

DAY = random.randint(1, 30)

print("generated DAY: ", DAY)

MONTH = random.randint(1, 12)

print("generated MONTH: ", MONTH)

YEAR = random.randint(0, 20)

if YEAR < 10:

YEAR = '0' + str(YEAR)

print("generated YEAR: ", YEAR)

INDEX = random.randint(100000, 999999)

INDEXE = random.randint(100000, 999999)

INNE = random.randint(1000000000, 9999999999)

print("generated INNE: ", INNE)

KPPE = random.randint(100000000, 999999999)

print("generated KPPE: ", KPPE)

EXECUTOR = "ООО \"СВЯЗЬ FINGER-STICK\", ИНН " + str(INN) + ", КПП " + str(KPP) + ", " + str(INDEX) + ", Сладководск г, Неплохая ул, дом 5"

EMPLOYER = "ООО \"ОТДЫХ\", ИНН " + str(INNE) + ", КПП " + str(KPPE) + ", " + str(INDEXE) + ", Омск г, Такая Себе ул, дом 6"

FOUNDING = str(random.randint(10000000, 99999999)) + " от " + str(random.randint(1, 30)) + "." + str(random.randint(1, 12)) + "." + str(random.randint(1970, 2020))

JOB\_1 = "ИСХОДЯЩИЕ ЗВОНКИ"

JOB\_2 = "ВХОДЯЩИЕ ЗВОНКИ"

JOB\_3 = "СМС"

JOB\_4 = "ИНТЕРНЕТ"

#\_lab1\_#

data\_file\_1 = "D:\DEV\mobile\lab3\data\data1.csv"

sms\_sum = 0

out\_calls\_sum = 0

inc\_calls\_sum = 0

number = '915642913'

sms\_tariff = 1

#first 5 for free

out\_calls\_tariff = 1

inc\_calls\_tariff = 1

# initializing the titles and rows

fields = []

rows = []

# reading csv file

with open(data\_file\_1, 'r') as csvfile:

# creating a csv reader object

csvreader = csv.reader(csvfile)

# extracting field names through first row

fields = next(csvreader)

# extracting each data row one by one

for row in csvreader:

rows.append(row)

# get total number of rows

#print("Total no. of lines in a CDR file: %d"%(csvreader.line\_num))

# calculating SMS tariffication

for row in rows[:csvreader.line\_num]:

if number in row[1]:

sms\_sum += float(row[4])

sms\_price = sms\_sum\*sms\_tariff-5

if sms\_price < 0:

sms\_price = 0

#print(sms\_price, "rubles for SMS")

# calculating outgoing calls tariffication

for row in rows[:10]:

if number in row[1]:

out\_calls\_sum += float(row[3])

out\_calls\_price = out\_calls\_sum\*out\_calls\_tariff

if out\_calls\_price < 0:

out\_calls\_price = 0

#print(out\_calls\_price, "rubles for outgoing calls")

# calculating incoming calls tariffication

for row in rows[:10]:

if number in row[2]:

inc\_calls\_sum += float(row[3])

inc\_calls\_price = inc\_calls\_sum\*inc\_calls\_tariff

if inc\_calls\_price < 0:

inc\_calls\_price = 0

#print(inc\_calls\_price, "rubles for incoming calls")

# total price

total\_price = sms\_price+out\_calls\_price+inc\_calls\_price

#print("total price is:", total\_price)

#\_lab2\_#

total\_traffic = 0

total\_occurences = 0

i = 0

data\_file\_2 = "D:\DEV\mobile\lab3\data\data2.csv"

#file = input("Enter path to CDR file: ")

#number = input("Enter IP address: ")

ip\_number = "192.168.250.59"

#first 1000#b for free

tariff = 1

# initializing the titles and rows

fields\_2 = []

rows\_2 = []

rowlist = []

traflist = []

# reading csv file

with open(data\_file\_2, 'r') as csvfile\_2:

# creating a csv reader object

csvreader\_2 = csv.reader(csvfile\_2)

# extracting field names through first row

fields\_2 = next(csvreader\_2)

# extracting each data row one by one

for row in csvreader\_2:

rows\_2.append(row)

# get total number of rows

#print("Total no. of lines in a CDR file: %d"%(csvreader.line\_num))

#

# calculating incoming calls tariffication

for row in rows\_2[1:17450]:

if row[3] == ip\_number:

total\_traffic = total\_traffic + int(row[12])

total\_occurences = total\_occurences + 1

#print(int(row[12]))

traflist += {(row[1], row[12])}

if row[4] == ip\_number:

total\_traffic = total\_traffic + int(row[12])

total\_occurences = total\_occurences + 1

traflist += {(row[1], row[12])}

c = ""

# total price

if total\_traffic < 1000:

c = "байт"

total\_cost = (total\_traffic\*1000-1000)\*1/1000000000

if total\_cost < 0:

total\_cost = 0

if 1000000 > total\_traffic > 1000:

total\_traffic = total\_traffic/1000

c = "КБ"

total\_cost = (total\_traffic\*1000-1000)\*1/1000000

if total\_cost < 0:

total\_cost = 0

if 1000000000 > total\_traffic > 1000000:

total\_traffic = total\_traffic/1000000

c = "МБ"

total\_cost = (total\_traffic-1)\*1

if total\_cost < 0:

total\_cost = 0

#print("total traffic:", total\_traffic, c)

#print("total occurences:", total\_occurences)

#print("total cost: ", round(total\_cost, 2), "rubles")

def cringe(N):

try:

print(num2words[N])

except KeyError:

try:

sotkas = num2words[N-N%100]

tens = num2words[N%100-N%10]

odins = num2words[N%10]

if 9 < (N%100) < 20:

odins = ''

tens = num2words[N%100]

#print(sotkas + " " + tens + " " + odins)

return(sotkas + " " + tens + " " + odins)

except KeyError:

print('Number out of range')

def kringe(N):

try:

print(num2words\_k[N])

except KeyError:

try:

tens = num2words\_k[N%100-N%10]

odins = num2words\_k[N%10]

if 9 < (N%100) < 20:

odins = ''

tens = num2words\_k[N%100]

#print(tens + " " + odins)

return(tens + " " + odins)

except KeyError:

print('Number out of range')

TOTAL = (out\_calls\_price+inc\_calls\_price+sms\_price+round(total\_cost, 2))

pogs = round(TOTAL%1, 2)\*100

if 4 < pogs%10 < 10:

SPR\_KOP = "КОПЕЕК"

elif pogs%10 == 1:

SPR\_KOP = "КОПЕЙКА"

elif pogs%10 == 0:

SPR\_KOP = "КОПЕЕК"

elif 1 < pogs%10 < 5:

SPR\_KOP = "КОПЕЙКИ"

if pogs-pogs%10 == 10:

SPR\_KOP = "КОПЕЕК"

#

rpogs = int(TOTAL)%10

if 4 < rpogs%10 < 10:

SPR\_RUB = "РУБЛЕЙ"

elif rpogs%10 == 1:

SPR\_RUB = "РУБЛЬ"

elif rpogs%10 == 0:

SPR\_RUB = "РУБЛЕЙ"

elif 1 < rpogs%10 < 5:

SPR\_RUB = "РУБЛЯ"

if int(TOTAL)%100-rpogs == 10:

SPR\_RUB = "РУБЛЕЙ"

TOTAL\_K = round(TOTAL%1, 2)

TOTAL\_K = TOTAL\_K\*100

TOTAL\_K = kringe(int(TOTAL\_K))

context = { 'BANK' : BANK, 'BIK' : BIK, 'COMP' : COMP, 'INN' : INN, 'KPP' : KPP, 'ACC\_1' : ACC\_1, 'ACC\_2' : ACC\_2,

'ACC\_3' : ACC\_3, 'DAY' : DAY, 'MONTH' : MONTH, 'YEAR' : YEAR, 'EXECUTOR' : EXECUTOR, 'EMPLOYER' : EMPLOYER,

'FOUNDING' : FOUNDING, 'JOB\_1' : JOB\_1, 'JOB\_2' : JOB\_2, 'JOB\_3' : JOB\_3, 'JOB\_4' : JOB\_4, 'AM\_1' : out\_calls\_sum,

'C\_1' : "мин", 'P\_1' : out\_calls\_tariff, 'S\_1' : out\_calls\_price, 'AM\_2' : inc\_calls\_sum, 'C\_2' : "мин",

'P\_2' : inc\_calls\_tariff, 'S\_2' : inc\_calls\_price, 'AM\_3' : sms\_sum,'C\_3' : "шт", 'P\_3' : sms\_tariff,

'S\_3' : sms\_price, 'AM\_4' : round(total\_traffic, 2), 'C\_4' : c, 'P\_4' : '1', 'S\_4' : round(total\_cost, 2),

'TOTAL' : TOTAL, 'NDS' : round(TOTAL/5, 2), 'TOTAL\_TEXT\_RUB' : cringe(int(TOTAL)), 'TOTAL\_TEXT\_KOP' : TOTAL\_K,

'SPR\_KOP' : SPR\_KOP, 'SPR\_RUB' : SPR\_RUB, 'BOSS' : "Важный Х.З.", 'BUHGALTER' : "Маркелова Е.А."}

doc.render(context)

doc.save("example\_final.docx")

print("Starting conversion.... ")

from docx2pdf import convert

#### Edit this to edit output file location

convert("D:\DEV\mobile\lab3\example\_final.docx")

print("pdf ready!")

Результаты работы:

На рисунке 1 представлен результат выполнения программного модуля:

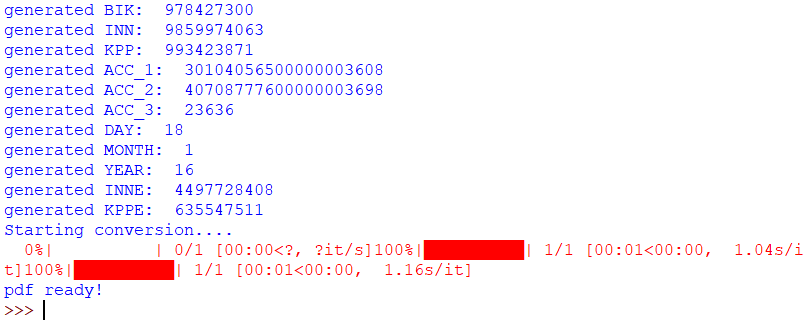


Рисунок 1 – результат выполнения программного модуля

На рисунке 2 представлен фрагмент составленного счета:

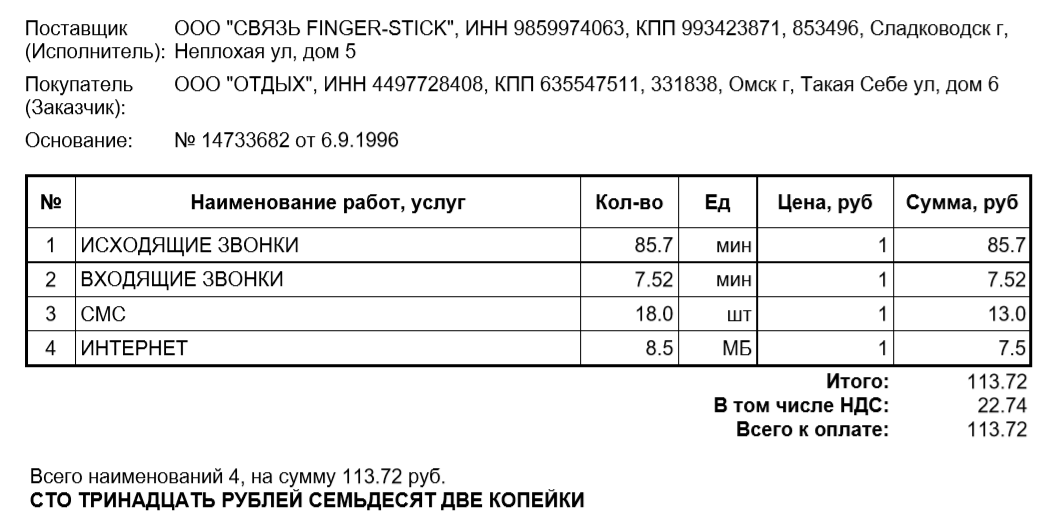


Рисунок 2 – фрагмент составленного счета

Выводы:

Цель работы достигнута, получены знания о тарификации услуг типа «Интернет» и «Телефония». Создан программный модуль, формирующий счет на оплату.