

**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»  
Отчет по лабораторной работе №4  
«Разработка бота на основе конечного автомата для Telegram с  
использованием языка Python»

Выполнил:  
студент группы ИУ5-33Б  
Смирнов Артём

Проверил:  
преподаватель каф. ИУ5  
Гапанюк Ю. Е.

Москва, 2023 г.

## **Задание:**

Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

## Текст Программы:

```
from typing import Final
import requests

from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup,
CallbackQuery, ReplyKeyboardMarkup

from telegram.ext import Application, CommandHandler, MessageHandler,
filters, ContextTypes, CallbackQueryHandler, CallbackContext,
ConversationHandler

bot_token: Final = ""
bot_name: Final = ""
currency_api_key: Final = ""

STATE_START = 1
STATE_MENU = 2
STATE_RATE = 3
STATE_CURRENCY = 4
STATE_HOW_MUCH = 5
STATE_TO_WHAT_CURR = 6

#States
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'start'
    response = '''Привет! Я бот-конвертер - могу подсказать актуальный курс
и перевести деньги в нужную валюту'''
    await update.message.reply_text(response)
    await menu(update, context)

async def menu(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'menu'
    response = '''Что вы хотите сделать ? '''
    keyboard = [
        InlineKeyboardButton("Узнать курс", callback_data="rate"),
        InlineKeyboardButton("Перевести", callback_data="curr"),
        InlineKeyboardButton("Далее", callback_data="next")
    ]
    await update.message.reply_text(response, reply_markup=InlineKeyboardMarkup(keyboard))
```

```

    ]
    reply_markup = InlineKeyboardMarkup([keyboard])
    if update.message: # Check if update.message is not None
        await update.message.reply_text(response,
reply_markup=reply_markup)
    elif update.callback_query and update.callback_query.message: # Check
if there is a message associated with the callback query
        await update.callback_query.message.reply_text(response,
reply_markup=reply_markup)

```

```

async def rate(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'rate'
    keyboard_1 = [
        InlineKeyboardButton("💵 Доллар 💵",
callback_data="USD"),
        InlineKeyboardButton("🇬🇧 Фунт 🇬🇧", callback_data="GBP"),
        InlineKeyboardButton("🇪🇺 Евро 🇪🇺", callback_data="EUR")
    ]
    keyboard_2 = [InlineKeyboardButton("Назад", callback_data="back")]
    reply_markup = InlineKeyboardMarkup([keyboard_1, keyboard_2])
    if update.callback_query:
        query = update.callback_query
        await query.message.reply_text("Курс какой валюты вы хотите узнать
?", reply_markup=reply_markup)

```

```

async def curr(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'curr'
    keyboard_1 = [
        InlineKeyboardButton("₽ Рубли ₽", callback_data="RUB"),
        InlineKeyboardButton("💵 Доллар 💵",
callback_data="USD"),
        InlineKeyboardButton("🇬🇧 Фунт 🇬🇧", callback_data="GBP"),
        InlineKeyboardButton("🇪🇺 Евро 🇪🇺", callback_data="EUR")
    ]
    keyboard_2 = [InlineKeyboardButton("Назад", callback_data="back")]
    reply_markup = InlineKeyboardMarkup([keyboard_1, keyboard_2])

```

```

    if update.callback_query:
        query = update.callback_query
        await query.message.reply_text("Введите изначальную валюту",
reply_markup=reply_markup)

async def how_much(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'how_much'
    if update.callback_query:
        query = update.callback_query
        await query.message.reply_text("Введите сумму")

async def to_what_curr(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['state'] = 'to_what_curr'
    keyboard = [
        InlineKeyboardButton("₽ Рубли ₽", callback_data="RUB"),
        InlineKeyboardButton("🇺🇸 Доллар 🇺🇸",
callback_data="USD"),
        InlineKeyboardButton("🇬🇧 Фунт 🇬🇧", callback_data="GBP"),
        InlineKeyboardButton("🇪🇺 Евро 🇪🇺", callback_data="EUR")
    ]
    reply_markup = InlineKeyboardMarkup([keyboard])
    await update.message.reply_text("В какую валюту вы хотите перевести ?",
reply_markup=reply_markup)

async def handle_callback(update: Update, context):
    global summ
    currencies = ['RUB', 'USD', 'GBP', 'EUR']
    state = context.user_data.get('state', 'start')
    query = update.callback_query
    global old_currency, new_currency
    await update.callback_query.answer()
    button_pressed = query.data
    if button_pressed == 'rate' and state == 'menu' :
        await rate(update, context)
    elif button_pressed == 'curr' and state == 'menu':
        await curr(update, context)

```

```

elif button_pressed == 'back' :
    await menu(update,context)
elif button_pressed in currencies and state == 'rate':
    api_url = f'https://v6.exchangerate-
api.com/v6/{currency_api_key}/latest/{button_pressed}'
    response = requests.get(api_url).json()
    await query.message.reply_text(f' {button_pressed} =
{round(response["conversion_rates"]["RUB"],2)} RUB')
    await menu(update,context)
elif button_pressed in currencies and state == 'curr':
    old_currency = button_pressed
    await how_much(update,context)
elif button_pressed in currencies and state == 'to_what_curr':
    new_currency = button_pressed
    api_url = f'https://v6.exchangerate-
api.com/v6/{currency_api_key}/latest/{old_currency}'
    response = requests.get(api_url).json()
    await query.message.reply_text(f'{summ} {old_currency} =
{round(float(response["conversion_rates"][new_currency])*summ,2)}
{new_currency}')
    await menu(update,context)

```

#messages

```

async def handle_message(update: Update, context):
    state = context.user_data.get('state','start')
    global summ
    text: str = update.message.text
    if text.isdigit() and state == "how_much":
        summ = int(text)
        await to_what_curr(update,context)
    elif not(text.isdigit()) and state == "how_much":
        await update.message.reply_text("Неправильное значение -
попробуйте еще раз")
        await how_much(update,context)
    else:
        await update.message.reply_text("Сейчас не требуется ничего
вводить")

```

```

#errors
async def error(update: Update, context: ContextTypes.DEFAULT_TYPE):
    print(f'Update {update} caused error {context.error}')

if __name__ == '__main__':
    print("Starting bot ...")
    app = Application.builder().token(bot_token).build()

    #Commands
    app.add_handler(CommandHandler('start', start))
    app.add_handler(CallbackQueryHandler(handle_callback))

    #Messages
    app.add_handler(MessageHandler(filters.TEXT, handle_message))

    #States

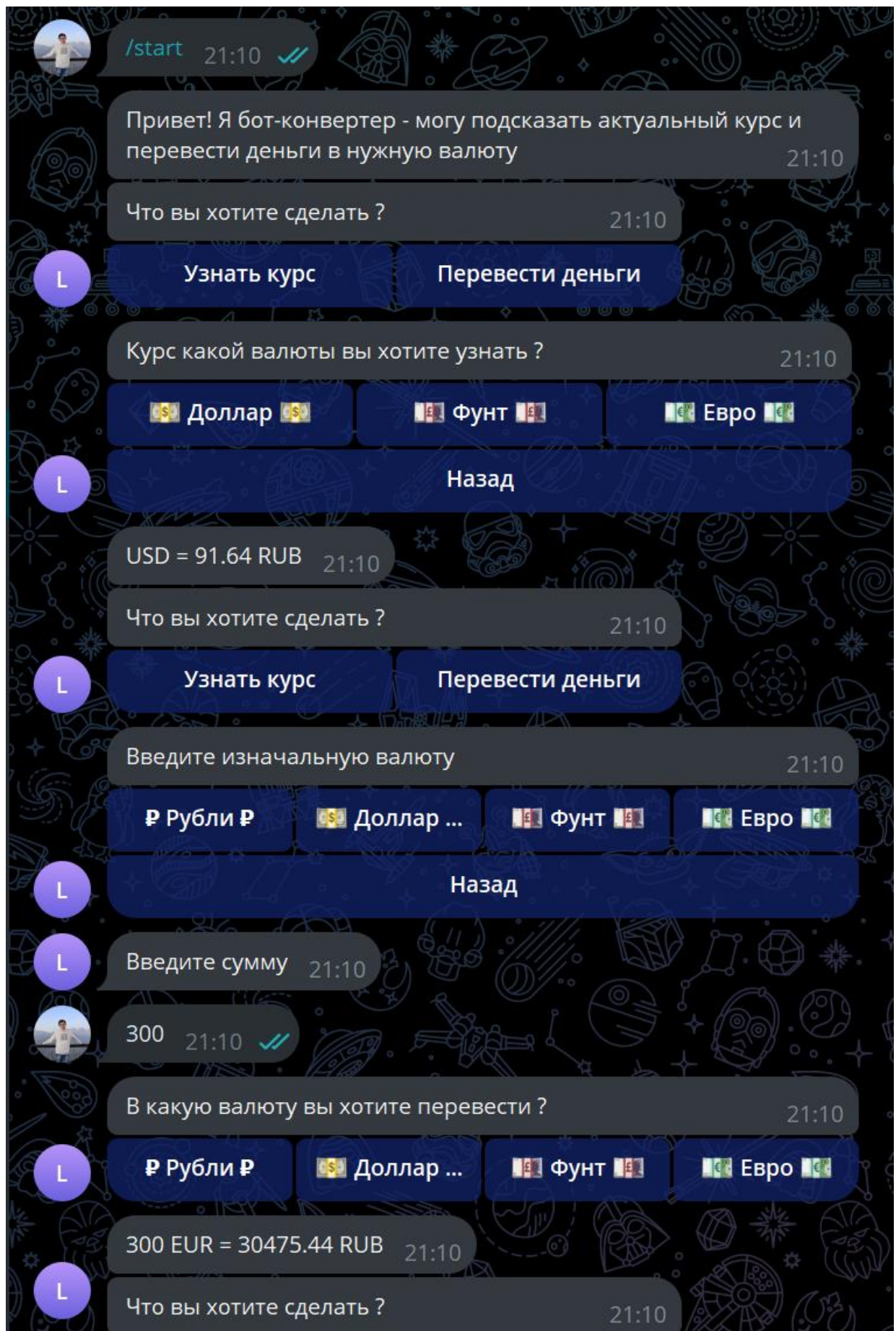
app.add_handler(ConversationHandler(entry_points=[CommandHandler('start',
start)], states= {
    STATE_START: [CommandHandler('start', start)],
    STATE_MENU: [CommandHandler('menu', menu)],
    STATE_RATE: [CallbackQueryHandler(handle_callback)],
    STATE_CURRENCY: [CallbackQueryHandler(handle_callback)],
    STATE_HOW_MUCH: [CallbackQueryHandler(handle_callback)],
    STATE_TO_WHAT_CURR: [MessageHandler(filters.TEXT &
~filters.COMMAND, handle_message)]
}, fallbacks = [CommandHandler('start', start)]))

#Error
app.add_error_handler(error)

#Polls the bot
print("Poling ...")
app.run_polling(poll_interval=0.1)

```

## Результат выполнения:





Что вы хотите сделать ?

21:10



L

Узнать курс



Перевести деньги

Курс какой валюты вы хотите узнать ?

21:11

 Доллар 

 Фунт 

 Евро 

L

Назад

Что вы хотите сделать ?

21:11

L

Узнать курс

Перевести деньги



5000

21:11



L

Сейчас не требуется ничего вводить

21:11