МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторной работе №9
“ Разработка бота на основе конечного автомата для Telegram с использованием языка Python.”

Выполнил:
Студент группы ИУ5Ц-41Б
Цурин А.П.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

1. Задания для выполнения

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

2. Листинг программы

State_machine_bot.py

```
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import (
    ApplicationBuilder,
    CommandHandler,
    MessageHandler,
    filters,
    CallbackQueryHandler,
    ConversationHandler,
    ContextTypes,
)

# Определим константы состояний нашего конечного автомата
STATE_START, STATE_INTERMEDIATE, STATE_FINISH = range(3)

# Токен бота, полученный от BotFather
TOKEN = ''

# Функции состояний

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Начало разговора."""
    user_id = update.effective_user.id
    # Инициализируем состояние пользователя
    if user_id not in context.user_data:
        context.user_data[user_id] = {'state': STATE_START}

    keyboard = [
        [InlineKeyboardButton("Начать", callback_data='start')],
        [InlineKeyboardButton("Среднее", callback_data='middle')],
        [InlineKeyboardButton("Закончить", callback_data='finish')]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text(
        text="Выберите состояние:",
        reply_markup=reply_markup
    )
    return STATE_START

async def state_start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Начальное состояние."""
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    context.user_data[user_id]['state'] = STATE_START
```

```
    await query.edit_message_text(text="Вы находитесь в начальном состоянии.")
    return STATE_START

async def state_intermediate(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Промежуточное состояние."""
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    context.user_data[user_id]['state'] = STATE_INTERMEDIATE
    await query.edit_message_text(text="Вы перешли в промежуточное состояние.")
    return STATE_INTERMEDIATE

async def state_finish(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Финальное состояние."""
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    context.user_data[user_id]['state'] = STATE_FINISH
    await query.edit_message_text(text="Работа завершена. Спасибо за использование
нашего бота!")
    return ConversationHandler.END

# Логика обработки нажатий кнопок
async def handle_button_click(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Обрабатываем нажатие кнопок."""
    query = update.callback_query
    data = query.data
    user_id = query.from_user.id
    current_state = context.user_data.get(user_id, {}).get('state')

    if data == 'start':
        return await state_start(update, context)
    elif data == 'middle':
        return await state_intermediate(update, context)
    elif data == 'finish':
        return await state_finish(update, context)

def main():
    application = ApplicationBuilder().token(TOKEN).build()

    conv_handler = ConversationHandler(
        entry_points=[
            CommandHandler('start', start),
        ],
        states={
            STATE_START: [CallbackQueryHandler(handle_button_click)],
            STATE_INTERMEDIATE: [CallbackQueryHandler(handle_button_click)],
            STATE_FINISH: []
        },
        fallbacks=[]
    )

    application.add_handler(conv_handler)
```

```
    print("Микробот запущен...")
    application.run_polling()

if __name__ == "__main__":
    main()
```

3. Результаты работы программы