МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования «Крымский федеральный университет имени В.И. Вернадского»

**ТАВРИЧЕСКИЙ КОЛЛЕДЖ**

(структурное подразделение)

**КУРСОВАЯ РАБОТА**

по **МДК.1.2 Прикладное программирование**

на тему:

**«Реализация бота выполняющего функции нажатия клавиш клавиатуры и мыши по алгоритму заданному пользователем»**

Выполнил: **Халевин Кирилл Александрович**

Специальность: 09.02.03 Программирование в компьютерных системах

Группа **3ПКС10**

Научный руководитель: **Михерский Ростислав Михайлович** \_\_\_\_\_\_\_\_\_\_\_

Г. Симферополь, 2018

Оглавление

[Введение 3](#_Toc503898962)

[Глава 1 – Автоматизация 7](#_Toc503898963)

[Глава 2 – Подготовка плана 10](#_Toc503898964)

[Глава 3 – Создание бота 11](#_Toc503898965)

[Выводы 20](#_Toc503898966)

[Список литературы 21](#_Toc503898967)

# Введение

Каждый день мы выполняем большое количество действий на компьютере. Эти действия мы выполняем при помощи устройств ввода - клавиатуры и мышки. И бывает, что новые компьютерные операции мы выполняем снова и снова, и нам это надоедает. И после нескольких выполненных повторяющихся действий, разные люди выбирают разные пути решения своих задач. Один человек будет повторять действия снова и снова, другой же увидит в этом задачу. Первый продолжит повторения, пока второй начнёт копания в решении задачи. Он будет искать методы решения и испытывать их на практике, он уделит время на поиск решения, пока первый делает то что ему нужно и получает свой результат решения задачи, который не получает второй пытаясь решить проблему другим путём. Но потом второй может перегнать первого по результатам и сэкономить своё время, конечно в зависимости от твёрдости повторных действий.

Интересно, как же это будет выглядеть в формулах и графиках?

Пусть есть два человека - Пётр и Василий. Обоим нужно делать одни и те же повторяющееся действия. Пётр делает действия снова и снова, а Василий ищет другие решения и пока не получает результат. Внесём обозначения, h1 – число часов Петра, а h2 – число часов Василия, которые они уделили на выполнение повторяющихся действий. Следовательно r1 – результат Петра и r2 – результат Василия. Запишем формулы результатов Петра и Василия.

h1 = r1 – результат Петра

h2 = r2 – результат Василия

Но исходя из моего примера, Василий не получает результат. Значит r2 = 0 и количество часов работы h2 = 0. Допустим Пётр проработал уже 10 часов, тогда h1 = 10. Отобразим прямолинейные функции на графике (Рисунок 1).

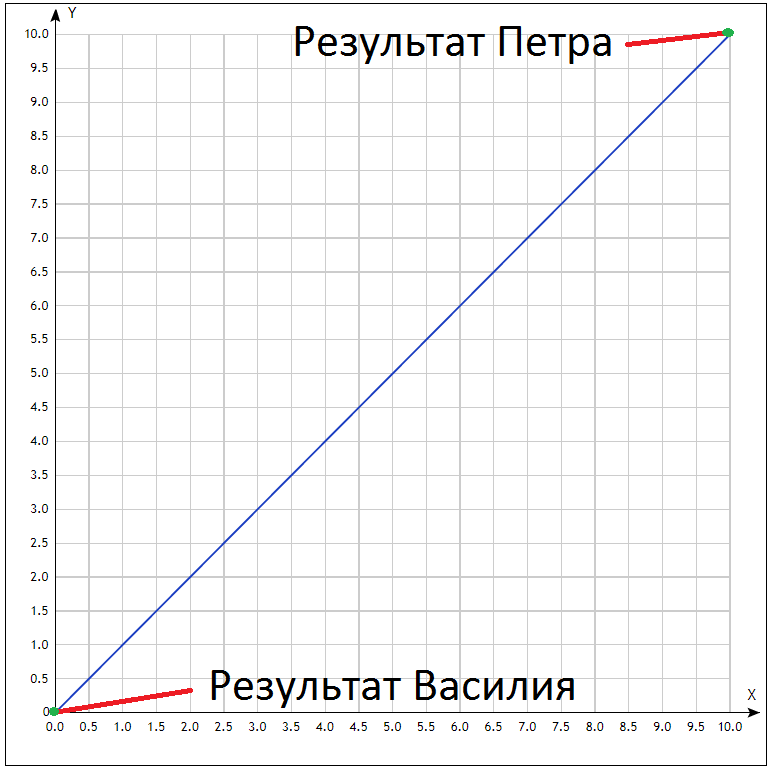


Рисунок 1 – График функции результата \*yotx.ru - Построение графиков функций онлайн

Результат Петра больше результата Василия и получается что Василию не выгодно искать решения по данным правилам. Но однако с другим решением действуют другие законы.

Пусть, Василий, за эти 10 часов сделал программу которая выполняет нужные ему повторяющиеся действия. И пусть задача имеет идеальную твёрдость алгоритма f = 100%. То есть алгоритм, в котором не требуется осознанный выбор человека и который машина сможет спокойно выполнить. Добавим также переменную время машины – m и коэффициент полезного действия η который равен:

η = результат машины / результат Василия

Тогда теперь новая программа Василия делает новый результат по следующей формуле

r2 = η\*f\*m

Например КПД машины будет равно 200%, то наша функция будет выглядеть таким образом(Рисунок 2)

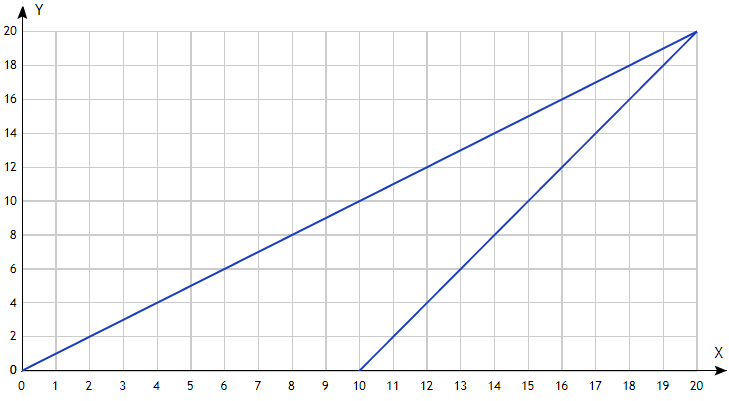


Рисунок 2 – Функция результата Василия за счёт машины \*yotx.ru - Построение графиков функций онлайн

На графике мы видим, что Василий за счёт машины догнал в результате Петра за 10 часов. И в дальнейшем Василий обгонит Петра по результатам. Василий изменивший правила и создавший новые условия быстрее поднял свой результат, но даже если КПД будет 100%, он освобождает своё время, которое в дальнейшем может вложить в решения других задач или просто отдохнуть. Конечно в моём примере очень идеальная модель и в реальной жизни имеется большое множество различных факторов-переменных, каждые из которых оказывают своё влияние на ситуацию которые в разные моменты могут как и сильно помогать, так и сильно ухудшать ситуацию.

Моя цель - сделать программу, в которой пользователь может задать нужные ему действия из списка для взаимодействия с графическим интерфейсом операционной системы Windows и по нажатию горячих клавиш, запустить заданный алгоритм. Чтобы пользователь мог без знаний языков программирования выполнять повторяющиеся действия и при этом не участвуя самому в процессе. Дать возможность пользователю составлять свой лист действий и функцию сохранения этого листа. Благодаря чему он сможет в дальнейшем использовать свой лист действий после закрытия программы, либо открыть его на другом компьютере.

Правда одним из недостатков программы является таким – программа работает с графическим интерфейсом. А для удобства пользователя, графический интерфейс является гибкой системой. Так например пользователь может указать алгоритм действий, которые включат музыку в браузере:

1. Открыть браузер;
2. Зайти в социальную сеть vk.com;
3. Нажать на плеер музыки – играть.

В этом листе может возникнуть проблема на каждом шаге. Значок браузера не может быть перемещён на другое место рабочего стола или меню пуска, иначе первый блок придётся переделывать. Так же нам важно чтобы никакие программы нам не перекрывали видимость нашего браузера. На втором шаге, когда бот успешно открыл браузер может возникнуть ситуация в которой браузер восстанавливается по прошлой длине и ширине, если пользователь самостоятельно сидел в браузере. В этом случае бот не сможет нажать на нашу вкладку vk.com. Однако если же первые блоки проработали как надо, то на самом сайте могут быть изменения. Изменения могут быть как со стороны сервера или же разработчиков если они изменят расположение плеера, так и со стороны самого пользователя если он в браузере вдруг поменял масштаб страницы.

# Глава 1 – Автоматизация

Автоматизация – это процесс в котором занятность человека переходит в руки машины или механизма(хотя у большинства машин и механизмов нет рук, но да ладно) и способствует улучшению продуктивности. Автоматизация в наше время это уже обыденность и используется в различных отраслях. Например Вы могли участвовать в процессе автоматизации, если пополняли счёт телефона или продлевали домашний Интернет через терминал. В этом случае терминал узнал у Вас куда Вы хотите пополнить деньги и на какой счёт. Просканировав деньги и убедившись в том, что они действительны он отправляет их через Интернет. В данном случае терминал сам передал информацию о нашем платеже и сам его оплатил.

А в бизнесе автоматизацию может выполнять например веб-сайт. Веб-сайт убирает необходимость делать определённые шаги которые находятся в бизнес модели. Он увеличивает продуктивность, число заказов и доход предпринимателя. Так например клиенты заходят в Интернет, вбивают в поисковик нужный им продукт или услугу, заходят на сайт и после этого сайт позволяет не делать два шага. Первым шагом сайт выполняет функции презентации. На нём клиент узнает характеристики продукта, его стоимость и интересные предложения которые он может получить купив продукт. Если же презентация рассказала о выгодном предложении за минимальное время и привлекла клиента, то следующий шаг который сделает сайт автоматически – это разделение клиентов на группы. Разделение на тех, кого не устраивает предложение и которые уходят с сайта. И на группу людей, которые стали потенциальными клиентами и оставили свои контактные данные для уточнения некоторых вопросов и для дальнейшей покупки. То есть сайт сам прошёл за предпринимателя или отдела продаж этап презентации и выявил группу потенциалов. А дальше остаётся только работа с возражениями которые возникли после презентации.

Но мы сейчас рассмотрим варианты автоматизации в информационной области, в которой задействована моя тема. В операционной системе Windows есть несколько способов как автоматизировать работу:

* Один из способов – это написание скриптов. В Windows можно сделать скрипт на VBS и этот скрипт нажмёт нужные клавиши. Или можно скачать бесплатный продукт AutoIt и написать скрипты под него. Недостатком этого способа заключается в том, что пользователю необходимо уметь писать скрипты и знать минимальную документацию языка.
* Скачать программу которая выполняет действия решающие похожие задачи между собой. Например калькулятор решает разные математические примеры требуя лишь входные данные. Или например программа-чистильщик которая по нажатию на кнопку – ‘Очистить’, сама находит временные файлы на компьютере и после, избавляется от них чтобы освободить место.
* А можно скачать готовую программу в которой не нужны знания языков программирования. Внутри которой пользователь сам задаёт действия которые ему нужны и по нажатию горячих клавиш, программа переводит пользовательский список действий в реальные действия выполняющихся в системе или графическом интерфейсе. Такие программы сложно настраивать, но я бы назвал такую программу универсальной, так как ограничения зависят только от фантазии, желания и терпения пользователя. Хотя универсальной в ограниченном смысле.

Если в Интернете не находится уже готовых решений под вашу задачу, то лучше использовать первый вариант и писать скрипты, которые автоматизируют работу. Так как если человек умеет писать скрипты, то он может сделать что только не придумает. И нажать клавишу мыши или клавиатуры. Может также открыть файл, как в графическом интерфейсе так и во внутреннем потоке если это текстовый файл. Скрипт сможет редактировать эти текстовые файлы, а также создавать новые папки и по необходимости их удалять. Информацию в файлах можно сортировать, а любые действия можно загнать в цикл и повторять их снова и снова. Но если человек не знает языков программирования или плохо разбирается в программировании и у него есть задача – сделать клики там-то и там, при том что такие действия являются прямолинейными и они не меняются, то ему подойдёт программа в которой можно задавать свой алгоритм.

Ещё я думаю что моя программа которая относится к третьему варианту, так же развивает логическое мышление. Как я пришёл к такому выводу? Когда я сам пробовал сделать определённые действия при помощи программы, то передо мной возникла задача и я думал как её решить. Мне пришло в голову несколько вариантов и в некоторых из них я при помощи логики выявил, что могут быть некоторые нюансы при которых они могут не заработать. Другими словами это тоже относится к программированию – составлению алгоритмам.

# Глава 2 – Подготовка плана

Создадим план по созданию бота-программы. Пусть пользователь сможет задавать несколько действий, я добавлю 4 основных действия с своими поддействиями:

* Виртуальное нажатие клавиши на клавиатуре
  + Нажать и оставить клавишу
  + Отпустить нажатую клавишу
  + Нажать и сразу отпустить нажатую клавишу
* Виртуальное нажатие клавиши на мышке
  + Нажать и оставить нажатой
  + Отпустить нажатую клавишу
  + Нажать и сразу отпустить нажатую клавишу
* Задержка перед новым действием
* Повтор действий, которые уже есть в списке действий.

Сложность заключается в том, что каждое действие имеет разные параметры, а значит интерфейс программы будет содержать разное количество полей. В некоторых случаях пользователю нужно будет заменить одно действие на другое, а в некоторых случаях ему потребуется удалить действие. Сложно сделать гибкость. Чем большую гибкость имеет модель программы, тем сложнее и запутанней становится код. Сложный код повышает шансы на ошибки, а исправление этих ошибок может привести к менее устойчивому коду, что понижает в будущем гибкость кода.

Поля для параметров одного действия будут находится на одной панели. На ней так же будет кнопка удаления действия из списка и надпись с числом действия по последовательности в списке. При наведении на поля ввода, я добавлю подсказки для пользователей. Кроме этого я так же попробую сделать локализацию на два языка: русский и английский. В моей программе нужна ещё возможность сохранения и загрузки скрипта.

# Глава 3 – Создание бота

Писать код и делать проект я решил в среде разработки Delphi 7 (Рисунок 4). Я выбрал Delphi 7 по нескольким причинам. Delphi использует язык Pascal и в своей практике программирования я больше всего пользовался компилятором Turbo Pascal 7.0 (Рисунок 3) и средой разработки Delphi 7 (Рисунок 4)

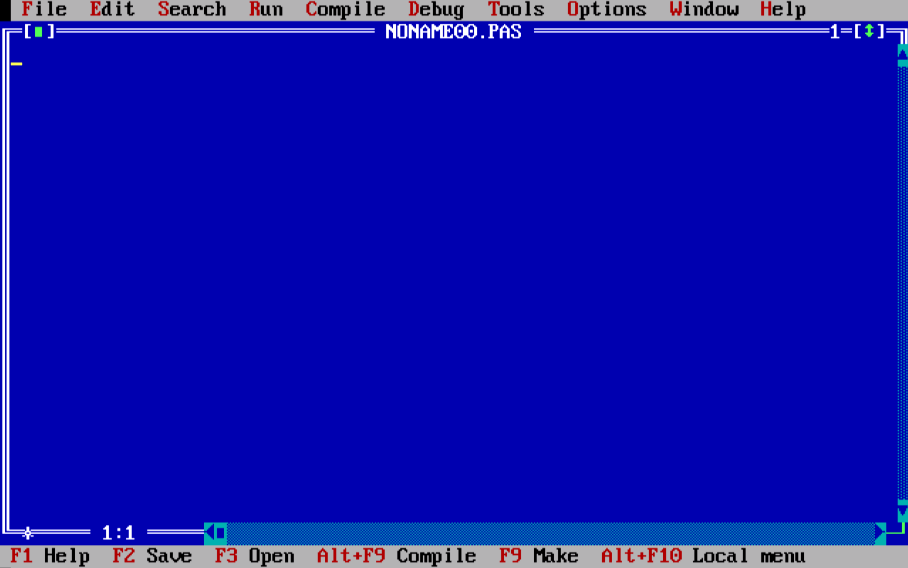


Рисунок 3 – Компилятор Turbo Pascal 7.0

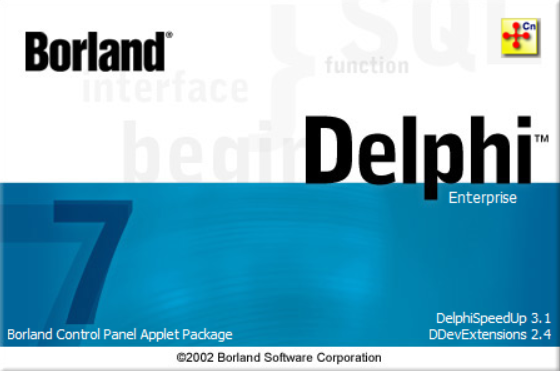


Рисунок 4 – Среда разработки Delphi 7

В новом проекте я добавил несколько панелей с 4 действиями: клавиатура(Keyboard); повторение(Repeat); мышь(Mouse); задержка(Sleep) и в каждой панели добавил нужное количество полей и выпадающих списков (Рисунок 5). Сделал для себя такую визуализацию, чтобы понимать как будет выглядеть моя программа.

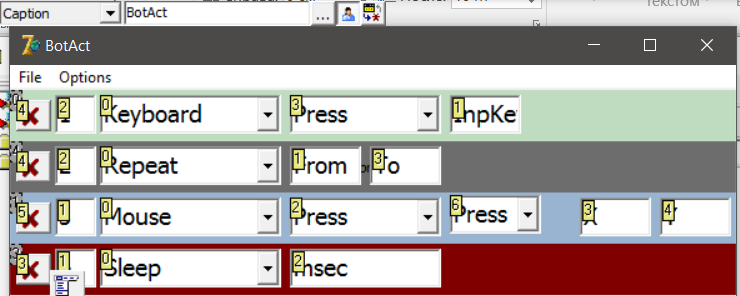


Рисунок 5 – Главная форма с 4 панелями

Добавил функцию с параметром **addAct(ActPos : integer)**, которая добавляет панель с кнопкой удаления; полем ввода в котором находится номер действия; выпадающий список действий и кнопка для добавления новой панели с возможностью выбора нового действия. Параметр **ActPos** – позволяет узнать с какой панели пользователь создал новое действия. Так например если у пользователя в списке будет 10 действий и он добавит новое действие нажав на кнопку на пятой панели, то сработает функция addAct(5) и у нас появится пустое действие на шестой позиции(Рисунок 6)

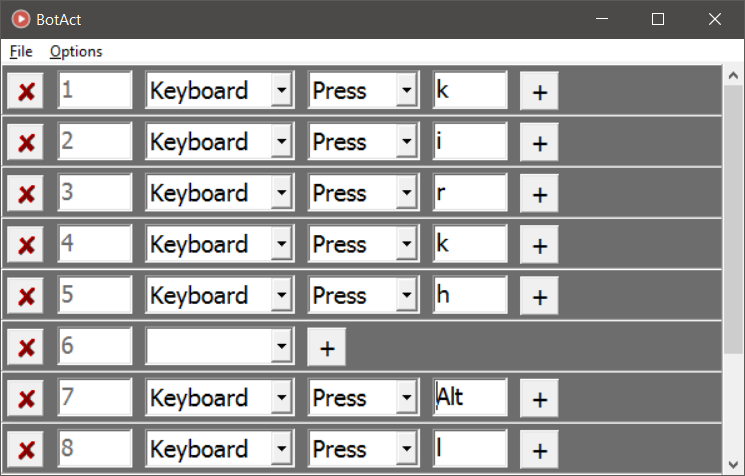


Рисунок 6 – Добавление действия после 5-го действия

Так же по нажатию на кнопку – ‘+’ срабатывает моя другая функция - **ShiftActsFrom(Frm: integer)** которая сдвигает действия ниже на позицию i+1. Так например на рисунке 6 после добавления действия на пятой панели, данные с 6 панели(6-Keyboard-Press-Alt) перенеслись на седьмую и за местом седьмой(7-Keyboard-Press-l) стало 7-Keyboard-Press-Alt и так далее с остальными. Аналогичным образом работает моя другая функция **ShiftActsBack(B : integer)** которая сдвигает действия в обратном порядке и срабатывает она по удалению действия.

При добавлении нового действия у пользователя находится пустой объект – выпадающий список. В нём он может выбрать четыре действия(Рисунок 7) и в зависимости от выбора создадутся новые поля. Поля обретут имена со своим индексом для того чтобы к ним можно обратиться по индексу в массиве. Но имя у объекта это строка и объекты не могут называться одним и тем же именем и для этого случая я сделал функцию **GetNumOfPanel(NameObj: string) : integer**, которая парсит цифры в конце строки и получает номер действия.

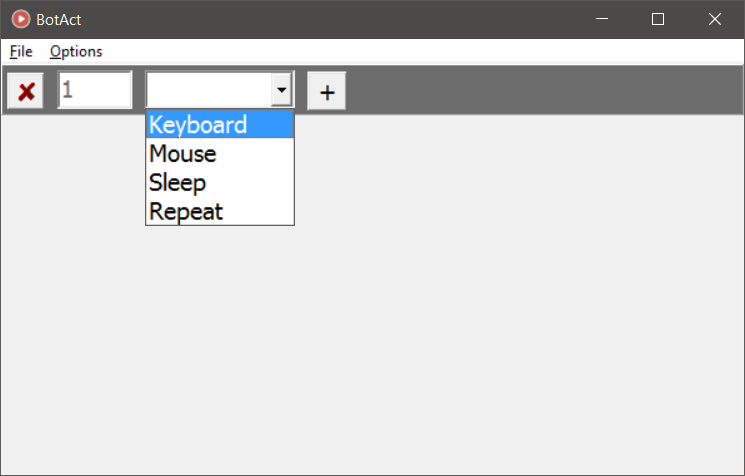


Рисунок 7 – Выпадающий список с выбором действия

После выбора действия, у объекта срабатывает событие onChange(на изменение) в котором при помощи оператора case по разному создаются объекты. Размеры объектов определяются по заданным мною константам, а расположение зависит от местонахождения визуально предыдущего объекта.

На рисунке 8 можно увидеть как визуально располагаются объекты при выборе каждого действии

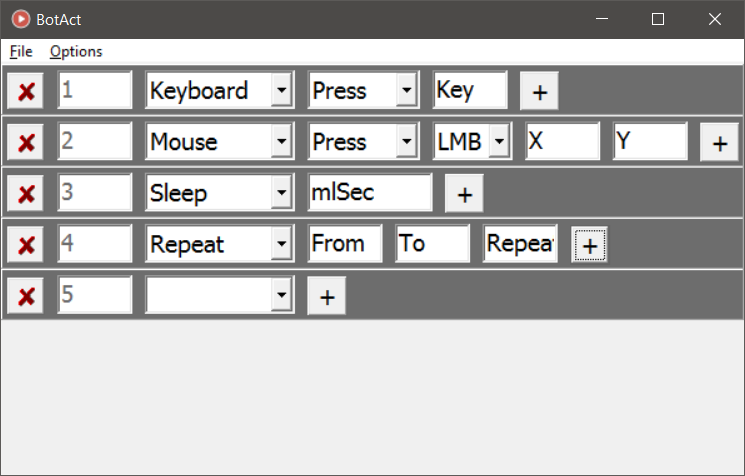


Рисунок 8 – расположение объектов в зависимости от действия

Дальше от пользователя требуется уточнения действия. Так например на первой позиции где выбрано действие – Keyboard, нужно указать клавишу в поле ввода – Key и выбрать событие Press, Down или Up. После ввода клавиши в поле Key, программа заменит это слово на нажатый символ, либо на название клавиши, а также сохранит код клавиши в другое скрытое поле по которому будет выполнятся действие нажатия клавиши клавиатуры. Событие Down позволяет зажать кнопку и не отпускать. Нужно для того, чтобы можно было использовать комбинации клавиш. Например Win+R(от английского слова Run(выполнить)) открывает окно ‘Выполнить’, но чтобы сделать эту комбинацию, нужно сначала зажать обе клавиши, что позволяет сделать событие Down и чтобы мы потом могли дальше нормально пользоваться клавиатурой, то нам нужно нажать на эти же клавиши и отжать их физически, либо использовать Up событие на эти кнопки, благодаря которому мы сможем их отжать виртуально.

Программирование действий клавиатуры разобрали, теперь перейдём к мышке(Mouse). Здесь так же присутствуют события Press, Down и Up и работают они так же как и в прошлом примере, только с мышкой. Далее пользователь выбирает какую клавишу мыши он хочет чтобы нажал бот. Он может выбрать из списка LMB(Left Mouse Button) - ЛКМ(Левую Кнопку Мыши), RMB(Right Mouse Button) – ПКМ(Правую Кнопку Мыши), либо MMB(Middle Mouse Button) – СКМ(Среднюю Кнопку Мыши). И виртуальные координаты экрана в поле – X и в поле – Y для выбора места куда кликнуть. Кнопку и координаты можно не вводить вручную, для этого я сделал так чтобы по нажатию по Ctrl, открывалась новая прозрачная форма которая разворачивается на весь экран и становится по верх всех других форм. При помощи события формы считываются координаты мыши X и Y, а событие onMouseDown(На нажатие мыши) узнает какую кнопку нужно нажать. Эта информация передаётся в основную форму и заполняется автоматически.

Следующим основным действием является задержка. Поскольку графический интерфейс сделан для облегчения работы пользователя, а любые механизмы и графическая оболочка которые облегчают работу людям и экономят их время, занимают и используют больше ресурсов компьютера. Но кроме ресурсов такая модель может занимать времени в десяток а то и более раз. Конечно для нас например отображение программы на глаз за 0,01 секунду и 0,10 секунд не особо заметна, но если делать автоматизацию нажатий в графической оболочке без задержки, то графическая оболочка не будет успевать за работой бота и автоматизация бота продолжится не по логике пользователя, а то есть не откроет приложение или не там наберёт текст и не в том приложение нажмёт клавишу мыши, но проделает весь алгоритм за малый промежуток времени. Либо пользователю просто нужно сделать следующее действие через 30 секунд. Но поскольку в любом случае нам нужно добавлять задержку после каждого действия, то я решил добавить функцию авто-задержки, которая будет срабатывать после каждого действия в алгоритме.

Последнее действие в списке является – повторение. Цикл – это одна из основ алгоритмов. Нам на то и нужны алгоритмы, чтобы можно было их выполнять снова и снова. От пользователя требуется ввести номер действия с которого будет начинается цикл и нужно ввести номер конечного действия. При этом я сделал так, что если пользователь в цикле установил диапазон в котором присутствует этот же цикл(где позиция действия = позиции выполняющего цикла), то в данном цикле, цикл ему подобный не запускается. Это сделано для того чтобы не было бесконечного цикла, а так же можно было использовать конечную позицию, где N – позиция цикла и действия в диапазоне больше N тоже выполнялись. Да, это всё конечно прекрасно, но бывают же случаи когда нужен бесконечный цикл. Например есть у нас игра жанра кликер и мы хотим много накликать. То в третьем параметре мы задаём число повторений 0(Рисунок 9), что позволит запустить бесконечный цикл.

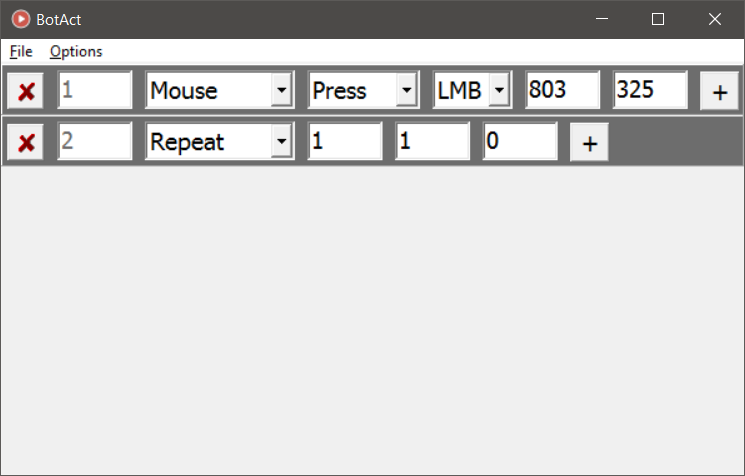


Рисунок 9 – Пример задавания бесконечного цикла

После нажатия горячих клавиш Ctrl+F1 программа убирает вкладки File и Options, а так же блокирует интерфейс программы. И создаёт новый поток, который начинает выполнять автоматические действия. Повторное нажатие горячих клавиш возвращает исходное положение программы.

Если же Вы попробуете закрыть программу, то увидите следующее сообщение(рисунок 10).

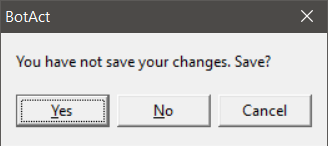


Рисунок 10 – Окно, которое предлагает сохранить новый скрипт или обновить старый

Если Вы хотите использовать Ваш алгоритм снова, то Вы можете сохранить его нажав на Yes, либо выбрав вкладку File(Файл) – Save(Сохранить). После чего появится диалоговое окно для выбора пути сохранения.

Файл имеет стандарты CSV(Comma-Separated Values) расширения, а расширение я сделал .bas(BotAct Script). На рисунке 11 можно увидеть как действия алгоритма выглядят в блокноте.

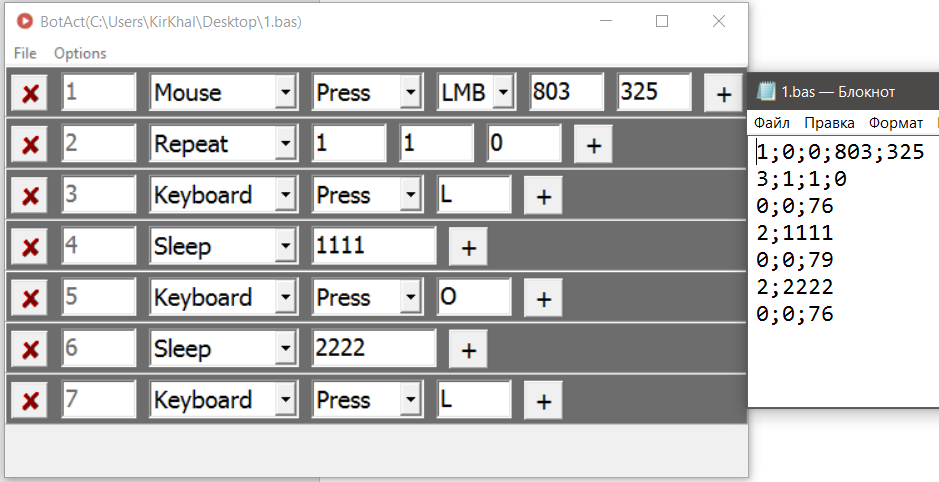


Рисунок 11 – Как выглядят действия в файле

Рассмотрим по подробнее первые четыре записи. В программировании счёт начинается с нуля и поэтому ноль будет первым элементом. Таким образом, нашим первым действием является нажатие клавиши мыши, а мышь в списке действия находится на первой позиции после нуля, поэтому в текстовом файле на первой строке в первой записи мы можем увидеть 1. Следующий 0 отвечает за параметр события нажатия кнопки. Второй 0 отвечает за выбор клавиши мыши. А остальные два числа являются координатой X и Y. Вторая запись начинается с тройки. Это по той причине, что действие Repeat находится на последнем месте в списке. Дальше идёт информация о диапазоне 1:1 и число повторений 0 – бесконечный цикл. По третей записи мы можем сказать, что действие Keyboard находится на первом месте в списке. Следующий 0 говорит нам о событии Press, а число 76 говорит нам о том, какой код бот будет использовать, чтобы нажимать виртуально клавишу. Задержка имеет только один параметр и поэтому в четвёртой записи после 2 записано только 1111 (миллисекунд) и больше ничего.

Вкладка Options(Рисунок 12), открывает форму с настройками.

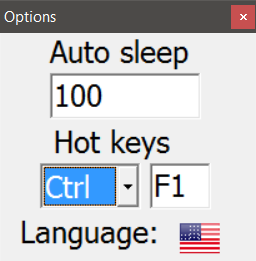


Рисунок 12 – Настройки программы

Здесь можно изменить авто задержку между действиями. Так же можно изменить горячие клавиши для запуска/остановки потока выполняющего виртуальные действий и можно поменять язык нажав на флаг. (Рисунок 13)

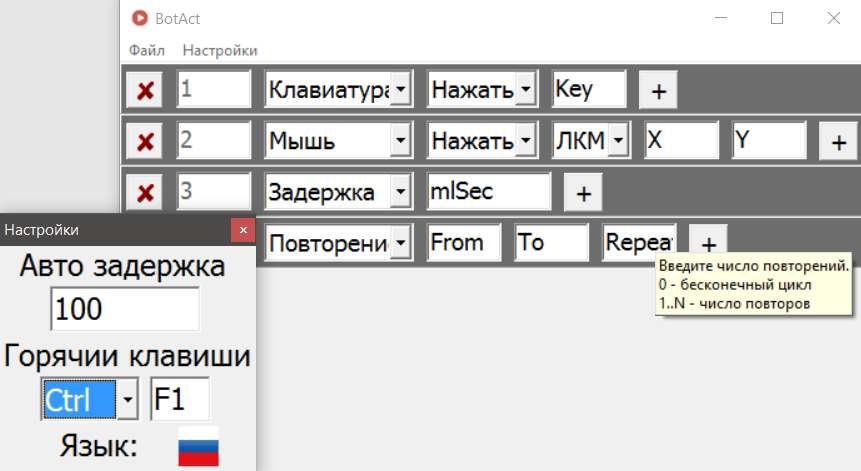


Рисунок 13 – После нажатия на флаг

Изменения в настройках сохраняются в файле по пути C:\Users\User\AppData\Local\BotAct\Options.bao.

# Выводы

И так, к какому конечному итогу я пришёл? Я достиг своей цели! Моя программа-бот позволяет пользователю выбирать действия из списка: клавиатура, мышь, задержка, повторение и создавать из них совокупный список. Этот список по нажатию горячих клавиш Ctrl+F1 включает бота и он начинает выполнять эти действия в графической оболочке Windows.

Пользователь может запрограммировать действия без знаний языков программирование. Вместе с этим я смог реализовать в программе сохранения списка в моём новом расширении .bas(BotAct Script) и добавил функцию чтение этих файлов и автоматическое их занесение в программу.

В ходе исследования я так же понял, что реализация алгоритмов в моей программе развивает логическое мышления.

Благодарю за внимание!!!

# Список литературы

1. Библия Delphi 3-издание 2015г. 686 стр. Михаил Евгеньевич Фленов. Издательство “БХВ-Петербург”
2. Книттель Брайан - Windows 7.Скрипты, автоматизацияи командная строка 2012г. Издательство “Питер”.