Министерство образования Республики Беларусь  
  
Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

*К защите допустить:*

Заведующий кафедрой информатики

*\_\_\_\_\_\_\_\_\_\_*С.И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОГРАММА ДЛЯ СОЗДАНИЯ И ЗАПУСКА МАКРОСОВ**

БГУИР КП 1-40 04 01 007

Студент В.П. Бычко

Руководитель Н.Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

# ВВЕДЕНИЕ

В современное время при выполнении различных задач на компьютере люди сталкиваются с повторяющимися действиями, такими как набор одних и тех же строк текста, необходимость нажатия определенных сочетаний клавиш и тому подобные. Для экономии времени существуют такое понятие как макрос – последовательность нажатий на клавиши или движений мышью, которые преобразуются в более длинные последовательности нажатий или движений мышью.

Целью данной курсовой работы является написание программы, позволяющей записывать и воспроизводить макросы в операционной системе windows. Для этого будет произведен анализ типов макросов, возможностей операционной системы windows по эмуляции ввода с клавиатуры и способов взаимодействия платформы .NET с ними.

Данную цель можно разложить на следующие задачи – изучение теории по созданию макросов, запись и имитация ввода клавиатуры и мыши средствами языка C#, создание программы с пользовательским интерфейсом, позволяющей считывать и воспроизводить ввод клавиатуры и мыши.

Для решения первой задачи необходимо осуществить поиск и изучение специализированной литературы, связанной с данной проблемой.

Для решения второй задачи необходимо определиться с технологией, используемой для создания пользовательского интерфейса, и библиотеками, используемыми для записи и воспроизведения нажатий клавиш и движений.

В результате выполнения данной курсовой работы будет получена программа с графическим интерфейсом, позволяющая осуществлять ввод определенных макросов и их запуск.

# 1 МАКРОСЫ. ИСТОРИЯ, ВИДЫ

## 1.1 Макросы

Макрос – это правило или паттерн, которое определяет, как определенная входная последовательность символов должна быть сопоставлена с определённой выходной последовательностью символов. Он может выполнять сколько угодно раз.

Применение макроса к вводу известно, как раскрытие макроса. Ввод и вывод могут представлять собой последовательность лексических токенов, символов, или синтаксическое дерево. В приложениях поддерживаются символьные макросы, чтобы упростить вызов команд. Макросы, состоящие из последовательности токенов или синтаксического дерева, поддерживаются в некоторых языках программирования для повторного использования кода или расширения возможностей языка.

Макросы используются для предоставления программисту последовательности вычислительных инструкций в виде одного оператора программы, что делает задачу программирования менее утомительной и уменьшает количество ошибок, возникающих в работе. Макросы, с греческого переводится как большой, получили именно такое название, потому что "большой" блок кода может быть раскрыт из "маленькой" последовательности символов. Макросы часто позволяют использовать позиционные или ключевые параметры, которые определяют, что генерирует препроцессор, и могут быть использованы для создания целых программ в зависимости от операционной системы, платформы или других факторов. Термин происходит от "макроинструкции", и такие расширения изначально использовались при генерации кода на языке ассемблера.

Различают такие виды макросов: макросы клавиатуры и мыши, макросы замены текста, процедурные и синтаксические макросы.

## 1.2 Макросы клавиатуры и мыши

Макросы клавиатуры и мыши позволяют преобразовать короткие последовательности нажатий клавиш и действий мыши в другие, обычно более трудоемкие, последовательности нажатий клавиш и действий мыши. Таким образом, часто используемые или повторяющиеся последовательности нажатий клавиш и движений мыши могут быть автоматизированы. Отдельные программы для создания этих макросов называются считывателями макросов.

В 1980–х годах макропрограммы – первоначально SmartKey, затем SuperKey, KeyWorks, Prokey – были очень популярны, сначала как средство автоматического форматирования сценариев, затем для различных задач пользовательского ввода. Эти программы были основаны на режиме работы terminate-and-stay-resident и применялись ко всему вводу с клавиатуры, независимо от того, в каком контексте это происходило. Они начали устаревать после появления пользовательских интерфейсов, управляемых мышью. Более широкое распространение макросов клавиатуры и мыши в текстовых редакторах и электронных таблицах, с возможностью создания для конкретных приложений, еще больше ускорило устаревание макропрограмм с режимом работы terminate-and-stay-resident.

Макросы клавиатуры и мыши получили широкое распространение в играх жанра MMORPG, в которых часто встречаются повторяющиеся задачи. Однако, поскольку это делается без участия человека и это может сильно влиять на экономику игры, использование макросов является нарушением условий пользования или лицензионного соглашения большинства MMORPG, и их администраторы тратят значительные усилия на подавление использования программ такого рода.

Макросы клавиатуры и мыши, созданные с использованием встроенных функций приложения, иногда называются макросами приложения. Они создаются путем однократного выполнения последовательности действий и предоставления приложению возможности записывать их. Также может существовать базовый язык макропрограммирования, чаще всего язык скриптов, с прямым доступом к функциям приложения.

Текстовый редактор для программистов Emacs возводит эту идею в абсолют. По сути, большая часть редактора состоит из макросов. Изначально Emacs был разработан как набор макросов на языке редактирования TECO; позже он был портирован на диалекты Lisp.

Другой текстовый редактор для программистов, Vim (потомок vi), также имеет реализацию макросов клавиатуры. Он может записывать в регистр (макрос) то, что пользователь набирает на клавиатуре, и это можно воспроизводить или редактировать точно так же, как VBA макросах для Microsoft Office. В Vim также есть язык сценариев под названием Vimscript для создания макросов.

Visual Basic for Applications (VBA) – это язык программирования, включенный в Microsoft Office начиная с Office 97 и заканчивая Office 2019 (хотя он был доступен в некоторых компонентах Office до Office 97). Необходимо обратить внимание, что его функции развились из макроязыков, которые первоначально были включены в некоторые из этих приложений, и заменила их.

## 1.3 Макросы замены текста

Макросы замены текста. Языки, такие как C и некоторые языки ассемблера, имеют рудиментарные макросистемы, реализованные в виде препроцессоров для компилятора или ассемблера. Макросы препроцессора C работают путем простой текстовой замены на уровне токена, а не на уровне символов. Однако макросредства более сложных ассемблеров, например, IBM High Level Assembler (HLASM), не могут быть реализованы с помощью препроцессора; код для сборки инструкций и данных перемежается с кодом для сборки вызовов макросов.

Классическое использование макросов–в системе компьютерной верстки TeX и ее производных, где большая часть функциональности основана на макросах.

## 1.4 Процедурные макросы

Макросы на языке PL/I написаны в подмножестве самого PL/I: компилятор выполняет "инструкции препроцессора" во время компиляции, и выходные данные этого выполнения являются частью компилируемого кода. Возможность использовать знакомый процедурный язык в качестве макроязыка дает гораздо больше возможностей, чем у макросов подстановки текста. Макросы в PL/I, как и во многих ассемблерах, могут иметь дополнительные функции, например, устанавливать переменные, к которым могут получить доступ другие макросы.

Макросы frame technology имеют свой собственный синтаксис команд, но также могут содержать текст на любом языке. Каждый фрейм является как общим компонентом в иерархии вложенных подсборок, так и процедурой интеграции со своими подсборными фреймами (рекурсивный процесс, который разрешает конфликты интеграции в пользу подсборок более высокого уровня). Выходные данные представляют собой пользовательские документы, обычно компилируемые исходные модули. Фреймовая технология позволяет избежать распространения похожих, но слегка отличающихся компонентов–проблемы, которая преследует разработчиков программного обеспечения с момента изобретения макросов и подпрограмм.

Большинство языков ассемблера имеют менее мощные процедурные макросредства, например, позволяющие повторять блок кода N раз для развертывания цикла; но они имеют синтаксис, совершенно отличный от реального языка ассемблера.

## 1.5 Синтаксические макросы

Макросистемы, такие как описанный ранее препроцессор C, которые работают на уровне лексических токенов, не могут надежно сохранять лексическую структуру. Синтаксические макросистемы вместо этого работают на уровне абстрактных синтаксических деревьев и сохраняют лексическую структуру исходной программы. Наиболее широко используемые реализации синтаксических макросистем находятся в языках, подобных Lisp. Эти языки особенно подходят для этого стиля макросов из-за их единообразного синтаксиса. В частности, именно он упрощает определение вызовов макросов. Макросы Lisp преобразуют саму структуру программы, при этом для выражения таких преобразований доступен полный язык. Хотя синтаксические макросы часто встречаются в языках, подобных Lisp, они также доступны на других языках, таких как Prolog, Erlang, Dylan, Scala, Nemerle, Rust. Они также доступны в качестве сторонних расширений для JavaScript и C#.

**Макросы раннего Lisp.** До того, как в Lisp появились макросы, в нем были так называемые FEXPRs, функционально-подобные операторы, входными данными которых были не значения, вычисляемые с помощью аргументов, а скорее синтаксические формы аргументов, и выходными данными которых были значения, которые будут использоваться при вычислении. Другими словами, FEXPRs были реализованы на том же уровне, что и EVAL, и предоставляли окно на уровень метаоценки.

В 1963 году Тимоти Харт предложил добавить макросы в Lisp 1.5 в AI Memo 57: MACRO Definitions for LISP.

**Анафорические макросы.** Анафорический макрос – это тип программного макроса, который намеренно фиксирует некоторую форму, предоставляемую макросу, на которую может ссылаться анафора (выражение, ссылающееся на другое). Анафорические макросы впервые появились в книге Пола Грэхема "On Lisp", и их название является отсылкой к лингвистической анафоре – использованию слов в качестве замены предшествующих слов.

**Гигиенические макросы.** В середине восьмидесятых в ряде статей было введено понятие расширения гигиенических макросов (синтаксические правила), cистема на основе шаблонов, в которой синтаксические среды определения макроса и использования макроса различаются, что позволяет разработчикам макросов и пользователям не беспокоиться о случайном захвате переменных. Гигиенические макросы стандартизированы для Scheme в стандартах R5RS, R6RS и R7RS. Существует ряд конкурирующих реализаций гигиенических макросов, таких как синтаксические правила, синтаксический регистры, явное переименование и синтаксические замыкания. Как синтаксические правила так и синтаксические регистры стандартизированы в стандартах Scheme.

Racket объединил понятия гигиенических макросов с "башней вычислителей", так что время синтаксического расширения одной макросистемы является обычным временем выполнения другого блока кода, и показал, как применять чередующееся расширение и синтаксический анализ в языке без скобок.

## 1.6 Макросы для машинно-независимого программного обеспечения

Макросы обычно используются для сопоставления короткой строки (вызова макроса) с более длинной последовательностью инструкций. Другое, менее распространенное использование макросов заключается в обратном: для сопоставления последовательности инструкций со строкой макроса. Именно такого подхода придерживалась система мобильного программирования STAGE2, которая использовала элементарный компилятор макросов (называемый SIMCMP) для отображения конкретного набора команд данного компьютера в машинно-независимые макросы. Приложения (в частности, компиляторы), написанные с помощью этих машинно-независимых макросов, затем могут быть запущены без изменений на любом компьютере, оснащенном элементарным компилятором макросов. Первым приложением, запускаемым в таком контексте, является более сложный и мощный компилятор макросов, написанный на машинно-независимом макроязыке. Этот компилятор макросов применяется к самому себе в режиме начальной загрузки для создания скомпилированной и гораздо более эффективной версии самого себя. Преимущество этого подхода заключается в том, что сложные приложения могут быть перенесены с одного компьютера на совершенно другой компьютер с очень небольшими усилиями (для каждой архитектуры целевой машины достаточно написать элементарный компилятор макросов). Появление современных языков программирования, в частности C, компиляторы для которых доступны практически на всех компьютерах, сделало такой подход излишним. Однако это был один из первых примеров (если не первый) начальной загрузки компилятора.

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Краткое руководство: создание макроса [Электронный ресурс]. – Режим доступа: https://support.microsoft.com/ru-ru/office/краткое-руководство-создание-макроса-741130ca-080d-49f5-9471-1e5fb3d581a8 – Дата доступа: 24.03.2023