Бесплатная электронная книга

УЧУСЬ AutoHotkey

Free unaffiliated eBook created from
Stack Overflow contributors.

#autohotkey

1
1: AutoHotkey 2
2
2
AutoHotkey 1.0. * - AutoHotkey Basic, Classic, Vanilla 2
(2011 , : 2009) 2
AutoHotkey 1.1. * - AutoHotkey_L. 2
( ) 2
AutoHotkey 2.0-a * 3
(-) 3
Examples 3
3
, 4
«Hello World» 4
, SplashTextOn 5
5
2: 7
7
Examples 7
7
7
RegEx 7
8
3: 9
9
Examples 9
, SetTimer 9
SetTimer: 9
Gui 10
11
11
11

OnClipboard fucntion / label 12
Gui 12
4: 15
Examples 15
..... 15
..... 15
N 15
..... 16
..... 16
..... 16
Array () 17
5: 18
..... 18
Examples 18
Windows 18
« SelectFile» 18
Windows Drag n 'Drop 19
6: 20
..... 20
..... 20
..... 21
Examples 21
..... 21
..... 21
7: , ... 23
Examples 23
Hello World 23
" !" 23
"Hello World!" MyString 23
" !" 23
" !" 23

"Hello World!" 23
«, !» (stdout) 23
«, !» enter 24
HelloWorld.txt 24
«Hello», , «Hello 24
8: 25
..... 25
..... 25
Examples 25
..... 25
..... 26
..... 26
- 26
..... 27
..... 27
..... 28

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: autohotkey

It is an unofficial and free AutoHotkey ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official AutoHotkey.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с AutoHotkey

замечания

AutoHotkey - бесплатный пользовательский язык сценариев с открытым исходным кодом для Microsoft Windows, первоначально предназначенный для обеспечения удобных сочетаний клавиш или горячих клавиш, быстрой макроконтента и автоматизации программного обеспечения, которая позволяет пользователям большинства уровней компьютерного навыка автоматизировать повторяющиеся задачи в любом приложении Windows. Пользовательские интерфейсы могут быть легко расширены или изменены с помощью AutoHotkey (например, переопределение командных клавиш управления Windows по умолчанию с их эквивалентами Emacs). Установка Autohotkey включает в себя собственный обширный файл справки с постоянно обновляемой веб- версией.

Вы можете писать макросы мыши или клавиатуры , переделывать ключи , создавать горячие клавиши , расширять аббревиатуры, изменять содержимое буфера обмена и делать исполняемые файлы для запуска сценариев hotkey на компьютерах без установки AutoHotkey.

Версии

AutoHotkey 1.0. * - также ретроактивно известна как AutoHotkey Basic, Classic, Vanilla и т. Д.

(Разработка была прекращена в 2011 году, последняя стабильность: 2009 год)

Версия	Дата выхода
v1.0.48.05	2009-09-26
v1.0.97.02	2011-04-14

AutoHotkey 1.1. * - ранее известный как AutoHotkey_L.

(Стабильный и регулярно получает обновления)

Версия	Дата выхода
v1.1.24.00	2016-05-22

Версия	Дата выхода
v1.1.24.01	2016-08-02

AutoHotkey 2.0-a *

(Все еще в альфа-стадии)

Версия	Дата выхода
v2.0-A069	2015-10-24
v2.0-A070	2015-11-09
v2.0-A071	2015-12-25
v2.0-A072	2015-12-25
v2.0-A073	2016-02-05
v2.0-A074	2016-03-11
v2.0-A075	2016-06-03

Examples

Установка или настройка

Из документации сайта Autohotkey

1. Перейдите на домашнюю страницу AutoHotkey .
2. Нажмите Загрузить , после загрузки запустите исполняемый файл.
3. Во время установки AutoHotkey вам будет предложено выбрать из UNICODE или ANSI. Короче говоря, вы, вероятно, захотите выбрать UNICODE. Он поддерживает неанглийские буквы и цифры (символы).
4. Продолжайте движение, пока не увидите кнопку установки.
5. Сделав это, отлично!

Использование в качестве переносного программного обеспечения

1. Перейдите на страницу загрузки AutoHotkey.
2. Найдите раздел «Портативный», выберите «UNICODE 32, 64 или ANSI» и скачайте.
3. При выборе папки назначения выберите любое правильное хранилище внешнего или нет.
4. Теперь вы можете выбрать ассоциировать файлы .ahk с Autohotkey.exe

5. Создайте простой текстовый файл и дайте ему расширение .ahk
6. Затем щелкните правой кнопкой мыши файл .ahk в проводнике и выберите «Свойства».
7. В файле «Свойства» нажмите кнопку «Изменить» рядом с опцией «Открывается с».
    - После нажатия кнопки «Изменить» вам будет предоставлен список программ для открытия файла, выберите программу, которую вы хотите использовать, а затем нажмите «OK» или «Применить».
    - Если программа, которую вы хотите выбрать, не указана, нажмите кнопку «Обзор» и найдите исполняемый файл Autohotkey (.exe) и нажмите «OK», чтобы выбрать эту программу.
8. Теперь файлы .ahk будут работать, как если бы был установлен autohotkey, отлично!

Если вы установили шоколад, запустите следующую команду в качестве администратора

```
choco install autohotkey
```

В качестве альтернативы, он может быть построен из исходного кода. Подробнее см. Здесь:
https://github.com/Lexikos/AutoHotkey_L/

Привет, мир

Показать "Привет мир!" в окне сообщений.

```
MsgBox, Hello World!
```

Показать "Привет мир!" в подсказке.

```
#Persistent
Tooltip, Hello World!
```

Показать "Привет мир!" сообщение в редакторе лотка.

```
#Persistent
TrayTip,, Hello World!
```

Печать «Hello, World» на стандартный вывод (stdout).

```
FileAppend, % "Hello, World", *
```

Показать «Hello World» в графическом интерфейсе

```
Gui, Add, Text,, Hello World!
Gui, Show, w200 h200
return
```

```
GuiClose:
ExitApp
```

Получите эффект, похожий на SplashTextOn

```
Gui, +AlwaysOnTop +Disabled -SysMenu +Owner  ; +Owner avoids a taskbar button.
Gui, Add, Text,, Some text to display.
Gui, Show, NoActivate, Title of Window  ; NoActivate avoids deactivating the currently active
window.
```

Как создать скрипт

После того, как вы установили AutoHotkey, вы, вероятно, захотите, чтобы это делалось. AutoHotkey не волшебство, мы все хотели, но это не так. Поэтому нам нужно будет сказать, что делать. Этот процесс называется «Scripting».

1. Щелкните правой кнопкой мыши на рабочем столе.
2. Найдите «Новое» в меню.
3. Нажмите «AutoHotkey Script» в меню «Новое».
4. Дайте сценарию новое имя. Примечание. Он должен заканчиваться расширением . ahk. Ex. MyScript.ahk
5. Найдите вновь созданный файл на рабочем столе и щелкните его правой кнопкой мыши.
6. Нажмите «Редактировать сценарий».
7. Должно появиться окно, вероятно, Блокнот. Если да, УСПЕХ!

Итак, теперь, когда вы создали скрипт, нам нужно добавить материал в файл. Список всех встроенных команд, функций и переменных см. В разделе 5. Вот очень простой скрипт, содержащий горячую клавишу, которая набирает текст с помощью команды «Отправить» при нажатии горячей клавиши.

```
^j::
   Send, My First Script
Return
```

Позднее мы получим более подробную информацию. До тех пор вот объяснение приведенного выше кода.

- Первая строка. `^j::` это горячая клавиша. `^` означает `CTRL` , `j` - буква j. Все, что слева от `::` - это клавиши, которые нужно нажать.
- Вторая строка. `Send, My First Script` - это то, как вы `SEND` нажатия клавиш. `SEND` - это команда, все после запятой (,) будет напечатано.
- Третья строка. `Return` . Возвращение станет вашим лучшим другом. Это буквально код STOPS, идущий дальше, к строкам ниже. Это предотвратит многие проблемы, когда вы начнете иметь много вещей в своих сценариях.

8. Сохраните файл.
9. Дважды щелкните файл / значок на рабочем столе, чтобы запустить его. Откройте блокнот или (все, что вы можете ввести) и нажмите Ctrl и J.
10. Гип-гип ура! Ваш первый скрипт сделан. Пойдите, получите некоторые наградные закуски, затем вернитесь к чтению остальной части этого учебника.

Прочитайте Начало работы с AutoHotkey онлайн:
https://riptutorial.com/ru/autohotkey/topic/3532/начало-работы-с-autohotkey

глава 2: Встроенные переменные и функции

замечания

AutoHotkey поставляется со многими встроенными функциями и переменными, которые можно использовать в любом месте скрипта.
Полный список, включая пояснения, см .:

- Список встроенных переменных
- Список встроенных функций

Examples

Определение времени ожидания пользователя

```
if(A_TimeIdlePhysical > 60000) { ; 60,000 milliseconds
    WinClose, ahk_class Chrome_WidgetWin_1
    MsgBox, Google Chrome was closed due to user inactivity.
}
```

Эта проверка может выполняться периодически, например, с помощью `SetTimer` .

Автоматическое вставка имени текущего дня недели

Этот пример вставляет / отправляет текущий день полного имени недели (например, *воскресенье*), когда нажата `комбинация клавиш Ctrl` + `Alt` + `D` :

```
^!d::Send, %A_DDDD%
```

Извлечение строк с использованием RegEx

```
myDebt := 9000
index  := RegExMatch("You owe me $42", "\$(\d+)", dollars)
if(index > 0) { ; indices are usually 1-based in AHK
    myDebt += dollars1
    MsgBox, Current debt: %myDebt%
}
```

Результат:

> Текущий долг: 9042

Обрезать строку

```
myString := "  hello, Trim()! "
trimmed  := Trim(myString)
FileAppend, % trimmed "`n", TrimmedStrings.txt
```

Обратите внимание, что `Trim()` не будет обрабатывать исходную строку, а возвращает новую, которая должна быть где-то сохранена или выведена.

Прочитайте Встроенные переменные и функции онлайн:
https://riptutorial.com/ru/autohotkey/topic/4469/встроенные-переменные-и-функции

глава 3: Использовать функции вместо меток

замечания

AutoHotkey часто использовал лейблы до версии 1.1.20. Опора на этикетки имела очень серьезные недостатки. Основной из них заключается в том, что метки обычно выполняются в глобальном масштабе, что означает, что любая переменная, определенная внутри метки, будет доступна по всему миру. Это звучит здорово, пока вы не поймете, что, например, вы не можете просто использовать библиотеки других народов, не убедившись, что их переменные не мешают вам.
Работа в глобальном масштабе, когда это не необходимо, - это просто плохая практика.

Таким образом, здесь появляются функции. Начиная с версии 1.1.20, каждая команда AutoHotkey, которая принимает имя метки как параметр, теперь альтернативно принимает имя функции.

Examples

Очень простой пример, демонстрирующий использование функции в SetTimer.

```
;Sends the keystroke for the letter "a" every 3 seconds.
#Persistent
SetTimer, SendLetterA, 3000
return

SendLetterA() {
    Send, a
}
```

Расширенное использование SetTimer: вызов одной и той же функции с различными параметрами

Это пример того, что было бы невозможно с помощью ярлыков. Если вы выполняете одну и ту же метку несколько раз в одно и то же время, и они полагаются на переменные, которые определяются внутри них, они, скорее всего, мешают и вызывают неожиданное поведение.

Вот как это сделать с помощью функций:

```
; This script will switch between showing "Hello 1" and "Hello 2"
```

```
#Persistent
DisplayMessage_Hello1 := Func("DisplayMessage").bind("Hello 1")
SetTimer, %DisplayMessage_Hello1%, 2000

Sleep, 1000

DisplayMessage_Hello2 := Func("DisplayMessage").bind("Hello 2")
SetTimer, %DisplayMessage_Hello2%, 2000

DisplayMessage(messageToDisplay) {
    TrayTip ; remove other traytips
    TrayTip, Message to display:, %messageToDisplay%
}
```

Вот **как это сделать** (с метками):

```
;This script will never display the message "Hello 1". It will always show "Hello 2".

#Persistent
messageToDisplay := "Hello 1"
SetTimer, DisplayMessage, 2000

Sleep, 1000

messageToDisplay := "Hello 2"
SetTimer, DisplayMessage, 2000

DisplayMessage:
    TrayTip ; remove other traytips
    TrayTip, Message to display:, %messageToDisplay%
Return
```

Gui с функциями вместо ярлыков

Пример, показывающий, как создать Guis, используя функции вместо меток.

```
Gui, Add, Button, gCtrlEvent vButton1, Button 1
Gui, Add, Button, gCtrlEvent vButton2, Button 2
Gui, Add, Button, gGoButton, Go Button
Gui, Add, Edit, vEditField, Example text
Gui, Show,, Functions instead of labels

CtrlEvent(CtrlHwnd:=0, GuiEvent:="", EventInfo:="", ErrLvl:="") {
    GuiControlGet, controlName, Name, %CtrlHwnd%
    MsgBox, %controlName% has been clicked!
}
GoButton(CtrlHwnd:=0, GuiEvent:="", EventInfo:="", ErrLvl:="") {
    GuiControlGet, EditField
    MsgBox, Go has been clicked! The content of the edit field is "%EditField%"!
}

GuiClose(hWnd) {
    WinGetTitle, windowTitle, ahk_id %hWnd%
    MsgBox, The Gui with title "%windowTitle%" has been closed!
    ExitApp
}
```

Горячие клавиши с функциями вместо меток

Примеры использования функций с помощью горячих клавиш:

```
Hotkey, a, MyFunction ; Calls MyFunction() when a is pressed

MyFunction() {
    MsgBox You pressed %A_ThisHotkey%.
}
```

Или же:

```
a::MyFunction()

MyFunction() {
    MsgBox You pressed %A_ThisHotkey%.
}
```

Действия меню лотка с функциями

```
#Persistent

Menu, Tray, NoStandard ; remove default tray menu entries
Menu, Tray, Add, MyDefaultAction, OnDefaultTrayAction ; add a new tray menu entry
Menu, Tray, Add, Exit, Exit ; add another tray menu entry
Menu, Tray, Default, MyDefaultAction ;When doubleclicking the tray icon, run the tray menu
entry called "MyDefaultAction".


OnDefaultTrayAction() {
    MsgBox, You double clicked the tray icon of this script or you clicked the MyDefaultAction
entry!
}

Exit() {
    MsgBox, You clicked the Exit entry! The script will close itself now.
    ExitApp
}
```

Общий пример функций с метками

Я продемонстрирую базовое использование функций с помощью меток + gosub.
В этом примере мы реализуем простую функциональность, чтобы добавить два числа и сохранить их в переменной.

С функциями:

```
c := Add(3, 2) ; function call
MsgBox, Result: %c%

Add(a, b) { ; This is a function. Put it wherever you want, it doesn't matter.
    ; the a and b inside of this function are set by the function call above
```

```
    Return a+b ; the function will return the result of the expression "a+b"
}
```

С ярлыками (пожалуйста, не делайте этого):

```
a := 3
b := 2
GoSub, Add ; execute the label "Add" then jump back to the next line here
MsgBox, Result: %c%
Return ; without this, the label would be executed again for no reason.

Add: ; This is a label. Please put them at the bottom of your script and use "Return" in a
line above.
    c := a+b
Return
```

OnClipboardИзменить с fucntion / label

Следующий код взят из официальной документации AutoHotkey:

Реализация функции:

```
#Persistent
OnClipboardChange("ClipChanged")
return

ClipChanged(Type) {
    ToolTip Clipboard data type: %Type%
    Sleep 1000
    ToolTip  ; Turn off the tip.
}
```

Выполнение этикеток:

```
#Persistent
return

OnClipboardChange:
ToolTip Clipboard data type: %A_EventInfo%
Sleep 1000
ToolTip  ; Turn off the tip.
return
```

Более сложный пример Gui с несколькими списками с использованием той же функции обратного вызова событий

Этот скрипт демонстрирует, как получать сложные события GUI из разных элементов управления в одной и той же функции обратного вызова. Для этого мы будем использовать два элемента управления ListView.
Теперь каждый раз, когда действие обнаруживается в одном из элементов управления ListView, мы хотим получить точное описание того, что произошло, и зарегистрировались в

элементе управления редактирования в том же графическом интерфейсе.

```
Gui, Add, ListView, gListCtrlEvent vMyFirstListView AltSubmit -ReadOnly R10 w310,
ColumnTitle1|ColumnTitle2|ColumnTitle3
Gui, Add, ListView, gListCtrlEvent vMySecondListView AltSubmit -ReadOnly R10 w310,
ColumnTitle1|ColumnTitle2|ColumnTitle3
Gui, Add, Text, w310, Action Log
Gui, Add, Edit, vLog R7 w310,
Gui, Show,, Functions instead of labels

; Create example entries for the first ListView
Gui, ListView, MyFirstListView
Loop, 10 {
    LV_Add("", "Column-1 | Row-" A_Index ,  "Column-2 | Row-" A_Index,  "Column-3 | Row-"
A_Index)
}
LV_ModifyCol()

; Create example entries for the second ListView
Gui, ListView, MySecondListView
Loop, 10 {
    LV_Add("", "Column-1 | Row-" A_Index ,  "Column-2 | Row-" A_Index,  "Column-3 | Row-"
A_Index)
}
LV_ModifyCol()


ListCtrlEvent(ctrlHwnd:=0, guiEvent:="", eventInfo:="", errLvl:="") {
    GuiControlGet, ctrlName, Name, %CtrlHwnd%
    whatHappened := "Action detected!`n"
    whatHappened .= "Control handle: " ctrlHwnd "`n"
    whatHappened .= "Control name: " ctrlName "`n"

    If (guiEvent = "DoubleClick") {
        whatHappened .= "`nThe user has double-clicked within the control."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "R") {
        whatHappened .= "`nThe user has double-right-clicked within the control."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "ColClick") {
        whatHappened .= "`nThe user has clicked a column header."
        whatHappened .= "`n> Column number: " eventInfo
    } Else If (guiEvent = "D") {
        whatHappened .= "`nThe user has attempted to start dragging a row or icon."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "d") {
        whatHappened .= "`nThe user has attempted to start right-click-dragging a row or
icon."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "e") {
        whatHappened .= "`nThe user has finished editing the first field of a row."
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "Normal") {
        whatHappened .= "`nThe user has left-clicked a row."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "RightClick") {
        whatHappened .= "`nThe user has right-clicked a row."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "A") {
        whatHappened .= "`nA row has been activated."
        whatHappened .= "`n> Row number: " eventInfo
```

```
    } Else If (guiEvent = "C") {
        whatHappened .= "`nThe ListView has released mouse capture."
    } Else If (guiEvent = "E") {
        whatHappened .= "`nThe user has begun editing the first field of a row."
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "F") {
        whatHappened .= "`nThe ListView has received keyboard focus."
    } Else If (guiEvent = "f") {
        whatHappened .= "`nThe ListView has lost keyboard focus."
    } Else If (guiEvent = "I") {
        whatHappened .= "`nItem changed. (A row has changed by becoming selected/deselected,
checked/unchecked, etc.)"
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "K") {
        whatHappened .= "`nThe user has pressed a key while the ListView has focus."
        whatHappened .= "`n> Key pressed: " GetKeyName(Format("vk{:x}", eventInfo))
    } Else If (guiEvent = "M") {
        whatHappened .= "`nItem changed. (A row has changed by becoming selected/deselected,
checked/unchecked, etc.)"
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "S") {
        whatHappened .= "`nMarquee. The user has started to drag a selection-rectangle around
a group of rows or icons."
    } Else If (guiEvent = "s") {
        whatHappened .= "`nThe user has finished scrolling the ListView."
    }
    GuiControlGet, Log
    GuiControl,, Log, % whatHappened "`n----------------------`n" Log
}

GuiClose(hWnd) {
    WinGetTitle, windowTitle, ahk_id %hWnd%
    MsgBox, The Gui with title "%windowTitle%" is going to be closed! This script will exit
afterwards!
    ExitApp
}
```

Прочитайте Использовать функции вместо меток онлайн:
https://riptutorial.com/ru/autohotkey/topic/4062/использовать-функции-вместо-меток

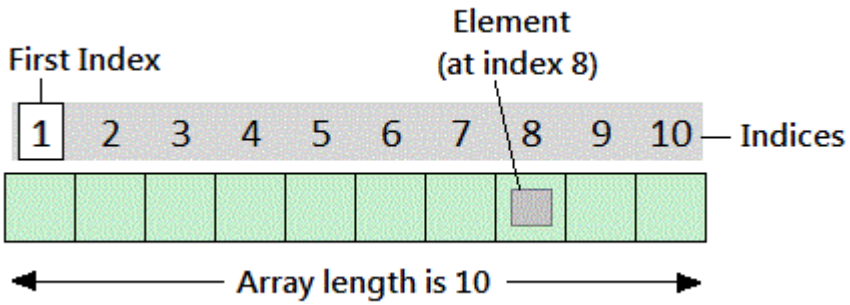
глава 4: Массивы

Examples

Создание и инициализация простых массивов

вступление

Массив - это контейнерный объект, который содержит несколько значений. На следующем изображении вы можете увидеть массив размером 10, первым элементом, индексированным 1, и последним элементом 10.



Autohotkey предлагает несколько способов определения и создания массивов.

- Массив: = []
- Массив: = Массив ()

Создание и инициализация массивов с N количеством элементов

```
Array := [Item1, Item2, ..., ItemN]
Array := Array(Item1, Item2, ..., ItemN)
```

В Autohotkey возможно наличие массивов без элементов:

```
Array := [] ; works fine.
```

И впоследствии элементы могут быть назначены ему:

```
Array[0] := 1
```

Размер массива можно определить с помощью метода, называемого `length` :

```
msgbox % array.length()  ; shows 1 in this case.
```

Если массив не пуст, **MinIndex** и **MaxIndex / Length** возвращают самый низкий и самый высокий индекс, который в настоящее время используется в массиве. Поскольку самый низкий индекс почти всегда равен 1, MaxIndex обычно возвращает количество элементов. Однако, если нет целых ключей, MaxIndex возвращает пустую строку, тогда как Length возвращает 0.

Создание и инициализация многомерных массивов

Вы можете создать многомерный массив следующим образом:

```
Array[1, 2] := 3
```

Вы можете создавать и инициализировать в одно и то же время, а внутренние массивы не должны быть одинаковой длины.

```
Array := [[4,5,6],7,8]
```

Такие массивы также называются массивами массивов.

Заполнение массива

```
; Assign an item:
Array[Index] := Value

; Insert one or more items at a given index:
Array.InsertAt(Index, Value, Value2, ...)

; Append one or more items:
Array.Push(Value, Value2, ...)
```

Значение индекса для элемента массива также может быть отрицательным целым числом (-1, 0, 1, 2, 3, 4, ...)

Удаление элементов из массива

```
; Remove an item:
RemovedValue := Array.RemoveAt(Index)
```

```
; Remove the last item:
RemovedValue := Array.Pop()
```

Добавление пользовательских методов путем переопределения функции Array ()

AutoHotkey - это язык программирования на основе прототипов , что означает, что вы можете в любой момент отменить любую встроенную функцию / объект. В этом примере демонстрируется переопределение функции Array () для добавления методов в пользовательский объект класса.

```
; Overrides Array()
Array(Args*) {
    Args.Base := _Array
    Return Args
}

; Custom Class Object with Methods
Class _Array {

    ; Reverses the order of the array.
    Reverse() {
        Reversed := []
        Loop % This.MaxIndex()
            Reversed.Push(This.Pop())
        Return Reversed
    }

    ; Sums all Integers in Array
    Sum(Sum=0) {
        For Each, Value In This
            Sum += Value Is Integer ? Value : 0
        Return Sum
    }
}

; Arr == ["Hello, World!", 4, 3, 2, 1]
Arr := [1, 2, 3, 4, "Hello, World!"].Reverse()

; SumOfArray == 10
SumOfArray := Arr.Sum()
```

Прочитайте Массивы онлайн: https://riptutorial.com/ru/autohotkey/topic/4468/массивы

глава 5: Открыть файл в скрипте

Вступление

Различные способы открытия файла для работы в скрипте.

Examples

Открыть файл через проводник Windows

Внутри скрипта используйте первую строку для хранения самой первой переменной (в этом примере, `%1%`) с именем, с которым нужно иметь дело. Пример: `OpenWithFile = %1%`

Как только вы откроете файл с помощью этого скрипта через Windows (щелкните правой кнопкой мыши по любому файлу в MS Windows и выберите «Открыть с помощью ...», затем выберите **скомпилированную** версию скрипта, например script.exe), имя выбранного файла будет хранится в этой переменной, и поэтому скрипт сможет работать с ним. Пример:

```
OpenWithFile = %1%
if OpenWithFile !=
{
FileRead, content, %OpenWithFile%
msgbox %content%
return
}
```

Откройте диалоговое окно «Файл через SelectFile»

В следующем примере создается Gui с одной кнопкой, которая приносит диалоговое окно SelectFile.

```
Gui, Loader: New
Gui, Loader: Add, Button, Default Center w220 vLOAD, LOAD
Gui, Loader: Show, AutoSize Center, Loader

return

LoaderButtonLOAD:
FileSelectFile, LoadedFile, , , ,

if ErrorLevel=1
{
return
}

else
{
```

```
    FileRead, content, %LoadedFile%
    msgbox %content%
}
return
```

Открыть файл через Windows Drag n 'Drop

В этом примере создается новый пустой Gui, способный перетащить событие «Drop»:

```
Gui, Dropper: New
Gui, Dropper: Font, s10 w700
Gui, Dropper: Add, Text, y80 vText1, Drag the files here
Gui, Dropper: Show, w200 h200 Center, Dropper

return

DropperGuiDropFiles:
DroppedFile:=A_GuiEvent

    FileRead, content, %DroppedFile%
    msgbox %content%

return
```

Прочитайте Открыть файл в скрипте онлайн: https://riptutorial.com/ru/autohotkey/topic/9070/открыть-файл-в-скрипте

глава 6: Поле ввода

Вступление

Чтобы получить вход пользователя и сохранить его в переменной, вы можете использовать команду InputBox. Сценарий не будет продолжать выполнять команды, пока пользователь не нажмет «OK» или «Отмена».

«OK» закроет окно и сохранит вход пользователя «Отмена» закроет окно, отбросив вход пользователя

параметры

InputBox, OutputVar [, Title, Prompt, HIDE, Width, Height, X, Y, Timeout, Default]	Что означает каждый вариант
OutputVar	Переменная, введенная пользователем, будет сохранена в
заглавие	Имя поля ввода
Незамедлительный	Текст внутри поля ввода
СКРЫВАТЬ	Отображает ввод пользователя в виде звездочек, чтобы скрыть ввод-тип HIDE для включения
ширина	Ширина поля ввода
Рост	Высота окна ввода
Икс	Количество пикселей от левого края экрана, в котором находится верхний левый угол окна ввода
Y	Количество пикселей от верхнего края экрана, в котором находится верхний левый угол окна ввода
Тайм-аут	Автоматически закрывает окно ввода и сохраняет вход пользователя после этого времени в миллисекундах

InputBox, OutputVar [, Title, Prompt, HIDE, Width, Height, X, Y, Timeout, Default]	Что означает каждый вариант
По умолчанию	Текст, который будет отображаться в редактируемом поле окна ввода при его открытии

замечания

Поле ввода - это элемент GUI, поэтому он будет рассматриваться как элемент графического интерфейса.

Список ошибок для этой команды:

Errorlevel	Что это означает
0	Пользователь нажал кнопку «OK»
1	Пользователь нажал кнопку «Отмена»
2	Время ожидания ввода

Вы можете найти страницу для этой команды в документации по AutoHotkey: https://autohotkey.com/docs/commands/InputBox.htm

Examples

Пример базового использования

```
InputBox, userinput
```

Это будет хранить то, что пользователь вводит в поле ввода в переменной с именем *userinput*

Пароли

```
InputBox, password, Enter your Password,, HIDE,, 100

Loop, {
  if (errorlevel = 1)
return

  if (password = "password") {
MsgBox, The password is correct.
    return
  } else if (password != "password") {
```

```
MsgBox, The password is incorrect.
InputBox, password, Enter your Password,, HIDE,, 100
  }
}
```

Это проверит, набрал ли пользователь пароль в поле ввода. Если пользователь вводит правильное значение, он скажет: «Пароль правильный». и закройте окно ввода. Если пользователь вводит неверное значение, он скажет: «Пароль неверен». и снова откройте окно ввода. Если уровень ошибок равен 1 (пользователь отменил отмену), он прекратит выполнение сценария.

Прочитайте Поле ввода онлайн: https://riptutorial.com/ru/autohotkey/topic/9917/поле-ввода

глава 7: Привет, мир

Examples

Примеры Hello World

Показать "Привет мир!" в окне сообщений.

```
MsgBox, Hello World!
```

Показать "Hello World!" хранится в переменной MyString в окне сообщения.

```
MyString := "Hello World!"
MsgBox, %MyString%
```

Показать "Привет мир!" в подсказке.

```
#Persistent
Tooltip, Hello World!
```

Показать "Привет мир!" сообщение в редакторе лотка.

```
#Persistent
TrayTip,, Hello World!
```

Показать "Hello World!" в окне графического интерфейса пользователя.

```
Gui, Add, Text,, Hello World!
Gui, Add, Edit,, Hello World!
Gui, Show
return

GuiClose() {
    ExitApp
}
```

Печатает «Привет, мир!» до стандартного вывода (

stdout).

```
FileAppend, % "Hello, World!", *
```

Имитация ввода «Привет, мир!» при нажатии enter

```
Enter::Send, Hello World{!}
```

Создайте новый файл HelloWorld.txt

```
FileAppend,, HelloWorld.txt
```

При вводе слова «Hello», за которым следует пробел или введите его, будет заменено «Hello World»

```
::Hello::Hello World
```

Прочитайте Привет, мир онлайн: https://riptutorial.com/ru/autohotkey/topic/3547/привет--мир

глава 8: Сценарии горячих клавиш

Синтаксис

- :: комбинации клавиш
- ::сокращение::
- Вернуть

параметры

Сочетания клавиш	подробности
^	Ctrl
!	Клавиша Alt
+	Клавиша Shift
#	Клавиша Windows
{войти}	отправить ключ ввода
{Вкладка}	отправить ключ вкладки
*	подстановочный знак, любая клавиша может быть нажата
~	Настроенная функция ключа не будет заблокирована
<символ	указывает левую клавишу (<+ - сдвиг влево)
символ>	указывает правую клавишу

Examples

Горячая клавиша

Чтобы создать горячую клавишу, которая отправляет последовательность клавиш «Hello World», нажав `Ctrl` + `J` на активное окно (может быть продемонстрировано в блокноте, например)

```
^j::
    Send, Hello World
Return
```

автозамены

Чтобы сценарий заменил фразу, используйте синтаксис `::abbreviation::` hotstring. Он заменит `btw by the way` когда вы введете `btw` а затем клавишу пробела.

```
::btw::by the way
```

Если вы хотите сделать скрипт входа в систему для более быстрого ведения журнала, вы можете сделать такой скрипт (файл не зашифрован, поэтому любая информация в вашем скрипте будет видна всем, у кого есть доступ к файлу).

```
::lmi::user{tab}password{enter}
```

Несколько нажатий клавиш

Для запуска скрипта при нажатии нескольких клавиш используйте `&` между клавишами.

```
Numpad0 & Numpad1::
    MsgBox You pressed 0 and 1
return
```

Контекстно-зависимые горячие клавиши и горячие клавиши

Чтобы создать горячую клавишу или горячую строку, которая запускается только когда определенные окна активны или существуют, вы можете поместить одну или несколько из следующих *директив* перед определением горячей клавиши:

```
#IfWinActive [, WinTitle, WinText]
#IfWinExist [, WinTitle, WinText]
#IfWinNotActive [, WinTitle, WinText]
#IfWinNotExist [, WinTitle, WinText]
```

Пример: Вы хотите `stackoverflow.com` быть послан всякий раз , когда вы набираете `so` (и через пробел после этого) в Google Chrome, но игнорировать строку автозамены в любом другом окне.

```
#IfWinActive, ahk_class Chrome_WidgetWin_1
::so::stackoverflow.com
```

Используя `#If [, Expression ]` , вы можете активировать горячую клавишу только тогда, когда произвольное выражение истинно, например:

```
#If A_Hour < 9
F1::
    MsgBox, It is too early to ask for help!
return
```

Клавиши быстрого доступа

Следующий пример переназначает ключ `z` на `y` и наоборот, например, если вы хотите работать с макетом QWERTY на клавиатуре QWERTZ.

```
z::y
y::z
```

Переключаемые горячие клавиши

Следующий скрипт вводит предопределенные строки на нажатиях горячих клавиш, если активна блокировка прокрутки. Это может быть полезно, если вы часто вставляете несколько повторяющихся строк. Включена горячая клавиша для обновления скриптов (например, если вам нужно отредактировать вставные строки).

```
; refresh script hotkey
Numpad9::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Reload
Return

Numpad1::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Send,         Hello
Return

Numpad2::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Send,         World
Return
;...
```

Прочитайте Сценарии горячих клавиш онлайн: https://riptutorial.com/ru/autohotkey/topic/3853/сценарии-горячих-клавиш

кредиты

S. No	Главы	Contributors
1	Начало работы с AutoHotkey	blackholyman, Community, depperm, Forivin, Joe DF, Shyam Sundar Shankar, tlm, Vijay
2	Встроенные переменные и функции	MCL
3	Использовать функции вместо меток	Forivin
4	Массивы	blackholyman, errorseven
5	Открыть файл в скрипте	freestock.tk
6	Поле ввода	Meh
7	Привет, мир	errorseven, Forivin, Goerman, mikew, vasili111
8	Сценарии горячих клавиш	depperm, MCL, user5226582