# Глава 8 Основы программирования в JAWS и NVDA

## 8.1. Основные понятия программирования

### 8.1.1. Основные алгоритмические конструкции.

Любая программа (или скрипт) всегда реализует какой-либо алгоритм. Это значит, что начинать написание программы или скрипта надо с разработки алгоритма, т.е. подробного описания действий, которые должна будет выполнять разрабатываемая программа.

При разработке алгоритмов используют три основные алгоритмические конструкции:

* Линейная;
* Ветвления;
* Циклическая.

Рассмотрим подробнее эти алгоритмические конструкции.

Большинство алгоритмов содержат фрагменты, в которых команды должны быть выполнены последовательно одна за другой. Некоторые алгоритмы полностью образованы последовательно выполняемыми командами. Такие алгоритмы (или фрагменты) называются линейными.

Таким образом, линейный алгоритм– это такой алгоритм, в котором все команды выполняются последовательно одна за другой и только один раз. Например, приводимые в этой книге алгоритмы линейные.

На практике часто встречаются алгоритмы, в которых в зависимости от первоначальных условий или результатов промежуточных вычислений необходимо выполнять те или иные команды. Эти алгоритмы должны содержать структуру ветвления, т.е. фрагмент, в котором осуществляется выбор одного из двух возможных направлений дальнейшей работы. В них выбор направления продолжения работы осуществляется по результатам проверки заданного условия. Структура ветвления в большинстве языков программирования задаётся оператором IF (условный оператор).

Например, структуру ветвления содержит алгоритм решения квадратного уравнения. Если вычисленное при работе алгоритма значение дискриминанта меньше нуля, то такое уравнение не имеет действительных корней и должно быть выдано сообщение об этом, а работа алгоритма прекращена. В противном случае уравнение имеет два (возможно совпадающих) корня и далее должны быть применены формулы для их вычисления.

Для решения многих задач применяются алгоритмы с отдельными многократно повторяющимися участками. Такие алгоритмы (или фрагменты) называются циклическими, а соответствующая конструкция – циклом.

Таким образом, цикл – это последовательность команд алгоритма, которая повторяется до тех пор, пока не будет выполнено заданное условие. Использование такой конструкции значительно снижает трудоемкость разработки алгоритмов и написания программ.

Например, Алгоритм перевода десятичного числа в двоичное является циклическим. Вспомним, что получив одну двоичную цифру как остаток от деления на 2, мы переходим к делению на два следующего числа – целой части от предыдущего деления. Так продолжается пока в качестве целой части будет получено число 0.

### 8.1.2. Примеры простейших программ.

Приводимые здесь примеры программ будут написаны на языке программирования Python, поскольку этот популярный язык изучается в школе. Программа невизуального доступа NVDA также написана на этом языке и поддерживает разработку на нём дополнений (скриптов).

Программа на языке Python представляет собой обычный текстовый файл, содержащий инструкции этого языка. Например, следующий текст можно назвать программой на языке Python:

print(“Привет! Мир!”)

print(2\*\*50)

Этот текст содержит две инструкции print(), первая из которых выводят строку (текст в кавычках), а вторая выводит результат числового выражения (2 в степени 50) на экран компьютера.

Для написания программ на языке Python удобно предварительно создать папку, в которой будут хранится разрабатываемые программы. Для этого в Total Commander можно поступать по следующему алгоритму:

1. Перейти в папку, внутри которой будет создана рабочая папка (например, в корневую папку диска D: ).

2. Ввести команду создания папки, нажав клавишу F7.

3. В раскрывшемся диалоге в поле редактирования ввести название создаваемой рабочей папки (например, «программирование»).

4. После нажатия клавиши Enter рабочая папка для хранения разрабатываемых программ будет создана.

Далее необходимо создать файл, в котором будет располагаться текст программы. Для этого можно поступать по следующему алгоритму, предназначенному также для файлового менеджера Total Commander:

1. Перейти в ранее созданную рабочую папку «программирование».

2. Ввести команду Shift +F4 для создания текстового файла.

3. В раскрывшемся диалоге создания файла в поле редактирования ввести имя создаваемого файла латинскими буквами без пробелов с расширением «py» (например, script.py) и, нажав Enter, создать файл будущей программы. Откроется окно редактора AkelPad и в нём будет загружен созданный (пока пустой) файл.

4. В меню «Файл» редактора AkelPad выбрать команду "Сохранить как…».

5. В раскрывшемся диалоге сохранения файла используя клавишу Tab найти поле «Кодировка:».

6. Используя курсорные стрелки выбрать кодировку «65001 (UTF-8)» и нажать Enter.

После выполнения указанных операций курсор окажется в поле редактирования редактора AkelPad, а файл будет сохранен в нужной кодировке в нужную папку. Заметим, что кодировка UTF-8 необходима для того, чтобы текстовые сообщения, генерируемые программой, без ошибок выводились на экран компьютера.

Обратите внимание, что файлы с программами на языке Python должны иметь расширение «py».

Теперь, указанные выше две инструкции программы следует ввести в окно редактора. Это можно сделать следующим образом:

1. Курсор уже находится в окне редактирования, где необходимо ввести следующие три строки:

print("Привет! Мир!")

print(2\*\*50)

input()

(Зачем нужна третья строка будет объяснено ниже).

2. Поскольку файл на диске уже создан и его имя и кодировка указаны, можно дать команду сохранения файла с помощью клавиши F2.

3. Для закрытия редактора можно использовать клавишу Escape. После закрытия окна редактора фокус окажется на имени созданного файла script.py.

Теперь можно выполнить созданную программу, т.е. выполнить все инструкции в файле от начала до конца. Программы запускаются несколькими способами. Самое простое – это, находясь в окне Total Commander, нажать клавишу Enter на имени файла с текстом программы. Раскроется окно командной строки, в котором ниже приглашения «C:\WINDOWS\py.exe» будут выведены две строки с результатами выполнения двух инструкций print():

Привет! Мир!

1125899906842624

Результат работы программы будет виден только в том случае, если в ней не было допущено ошибок. Когда интерпретатор языка Python находит ошибку, он прекращает работу и выводит на экран сообщение о найденной ошибке. Однако, особенности работы операционной системы Windows таковы, что после прекращения работы интерпретатора окно командной строки с сообщением об ошибке закроется и увидеть это сообщение будет нельзя.

Если же программа не содержала ошибок, то интерпретатор выполнит две её строки с инструкцией Print() и остановится ожидая ввода с клавиатуры. Это ожидание ввода есть результат выполнения третьей строки с инструкцией Input(). В этой ситуации можно изучить окно командной строки, где и будет выведен результат работы программы. Когда результаты работы программы будут проверены, нажмите Enter и тогда окно командной строки закроется.

Приведём алгоритм, применяя который можно изучать окно с результатами работы программы используя JAWS for Windows:

1. Запустить программу нажатием клавиши Enter на её имени в Total Commander.

2. Виртуализировать окно командной строки введя Alt +Ins +W или дать команду «JAWS к Pc» введя Ins +- (минус на дополнительной клавиатуре).

3. Используя курсорные стрелки изучить результаты работы программы.

4. Нажать клавишу Escape для отключения виртуализации окна или включить pc-курсор командой + (плюс на дополнительной клавиатуре).

5. Нажав клавишу Enter (этим будет осуществлён пустой ввод) закрыть окно командной строки.

Рассмотрим второй способ запуска программ на языке Python. При использовании этого способа вы сможете увидеть сообщения об ошибках при работе интерпретатора. Для этого можно использовать следующий алгоритм:

1. В программе Total Commander Перейти в свою рабочую папку, например, «программирование».

2. В этой папке создать файл start.cmd введя команду Shift +F4.

3. В раскрывшемся окне редактора ввести в первой строке имя файла с текстом программы на языке Python (например, script.py), а на следующей строке ввести команду pause.

4. Сохранить файл с помощью команды F2 и закрыть редактор клавишей Escape.

Теперь для запуска программы следует нажимать Enter на файле start.cmd. При этом окно командной строки в открытом состоянии будет держать команда pause этого файла. Обратите внимание, что последняя строка input() в файле script.py уже не нужна, задержит окно на экране команда pause, которая выполнится уже после того, как интерпретатор Python выдаст сообщение об ошибке или результаты работы программы и прекратит работу.

Изучать окно с результатами работы программы можно также опираясь на приведённый выше алгоритм. В дальнейшем вы самостоятельно будете выбирать наиболее удобный способ запуска программ на языке Python.

Рассмотрим еще один пример. Пусть необходимо вывести на экран компьютера 10 раз слово «Привет!» и результат возведения числа 2 в степень 10.

Решение: Для вывода строки и результата возведения в степень на экран используем всё ту же функцию print(). Для десятикратного повторения слова умножим строку «Привет!» на десять, а для возведения в степень воспользуемся уже встречавшейся командой \*\* (две звезды).

Вот текст этой программы:

# Программа выводит 10 раз слово «Привет!» и возводит 2 в степень 10.

Print(“Привет!”\*10) # Повторяет вывод 10 раз.

Print(2\*\*10) # Возводит 2 в степень 10.

input() # Строка для задержки результатов на экране.

Эта программа содержит в конце строку input(), такую же, как в первом примере, поэтому её можно запускать нажимая Enter на имени файла с текстом программы.

Символ # (решётка) означает комментарий. Комментарий – это текст, следующий за символом #. Комментарии могут занимать отдельную строку или добавляться в строку с программным кодом, правее его. Текст, следующий за символом #, интерпретатором игнорируется, как комментарий, добавленный для человека. В качестве комментария в текст программы добавляются пояснения для программиста, помогающие ему понять смысл программы.

Обратите внимание, что в этом случае непосредственного запуска программы (без командного файла start.cmd) при наличии ошибок сообщения о них прочитать не удастся.

Контрольные вопросы

1. Какие из перечисленных ниже правил являются алгоритмами? Ответ обоснуйте:  
   А) орфографические правила русского языка;  
   Б) правила выполнения арифметических операций;  
   В) правила техники безопасности;  
   Г) правила перевода чисел из одной системы счисления в другую.
2. Используя известные вам математические алгоритмы, приведите примеры:  
   А) линейного алгоритма;  
   Б) ветвящегося алгоритма;  
   В) циклического алгоритма.
3. Что такое Python?
4. Расскажите, как создать папку в Total Commander.
5. Расскажите, как создать файл для написания программы.
6. Расскажите, какие способы запуска программ на Python вы знаете. В чём их разница?
7. Какая инструкция Python выводит информацию на экран?

Упражнения для самостоятельного выполнения

1. Создайте на диске «D:» рабочую папку «программирование». Наберите каждый пример из этого параграфа в отдельный файл и добейтесь их работоспособности. Все файлы сохраните в созданной папке.
2. Напишите программу, выводящую на экран 8 раз слово «Победа!».
3. Напишите программу, вычисляющую:  
   А) 2 в степени 20;  
   Б) 1\*2\*3\*4\*5\*6.  
   Каждое вычисление реализуйте в отдельном фале и сохраните их в папку «Программирование».
4. У исполнителя «УТРОИТЕЛЬ» две команды, которым присвоены номера:  
   1. вычти 1  
   2. умножь на 3  
   Первая из них уменьшает число на экране на 1, вторая – увеличивает его в три раза. Запишите порядок команд в программе получения из числа 3 числа 16, содержащей не более 5 команд, указывая лишь номера команд. Например, программа 21211 это программа:  
   умножь на 3  
   вычти 1  
   умножь на 3  
   вычти 1  
   вычти 1  
   которая преобразует число 1 в 4.
5. У исполнителя «ДваПять» две команды, которым присвоены номера:  
   1. отними 2  
   2. раздели на 5  
   Выполняя первую из них, «ДваПять» отнимает от числа на экране 2, а выполняя вторую, делит это число на 5 (если деление нацело невозможно, «ДваПять» отключается). Запишите порядок команд в программе, которая содержит не более 5 команд и переводит число 152 в число 2. В ответе указывайте лишь номера команд, пробелы между цифрами не ставьте. Так, для программы:  
   раздели на 5  
   отними 2  
   отними 2  
   нужно написать 211. Эта программа преобразует, например, число 55 в число 7.
6. Автомат получает на вход трёхзначное число. По этому числу строится новое число по следующим правилам:  
   1. Складываются первая и вторая, а также вторая и третья цифры исходного числа.  
   2. Полученные два числа записываются друг за другом в порядке убывания (без разделителей).  
   Пример. Исходное число: 348. Суммы: 3 +4 =7; 4 +8 =12. Результат: 127.  
   Сколько существует чисел, в результате обработки которых автомат выдаст число 1715?

## 8.2. Программирование скриптов в JAWS

### 8.2.1. Скрипты программы JAWS for Windows.

При работе на персональном компьютере пользователь постоянно выполняет определенные действия, например, вводит клавиатурные команды, устанавливает «флажки» в диалогах, вводит текст в поля редактирования и т.д. Наборы таких часто используемых операций можно объединять в скрипты (или макросы).

скрипты - это последовательности отдельных операций, которые можно использовать для управления широким диапазоном компьютерных процессов, например, вставка подписи под письмом с помощью одного нажатия клавиш или озвучивания определенной области экрана при появлении там текста и т.д.

Программа невизуального доступа JAWS for Windows использует много различных скриптов, которые определяют ее поведение в той или иной ситуации. Например, в главном файле скриптов, который называется «файл скриптов по умолчанию» (default), есть скрипт, Привязанный к клавише Стрелка вниз и этот скрипт сообщает JAWS, что необходимо прочитать новую строку. Этот конкретный скрипт носит название "SayNextLine» и активизируется при нажатии стрелки вниз. Он довольно сложен, поскольку должен проанализировать в окне какого типа находится фокус и если это текстовый редактор, то озвучить строку, на которую переместился фокус при нажатии стрелки вниз. В этом файле скриптов есть функция «SayLine ()», она вызывается после того, как фокус перемещается на следующую строку. Именно эта функция скрипта фактически выполняет чтения строки. Функция - это скрипт, который не закреплен за конкретной клавишей и который при вызове возвращает определенное значение.

JAWS всегда находится под управлением скриптов. Файлы скриптов JAWS - это наборы отдельных скриптов, которые загружаются для работы с данным конкретным приложением Windows. Эти файлы автоматически загружаются всякий раз, когда пользователь запускает новое приложение.

Есть два типа файлов скриптов - по умолчанию (Default) и для приложений. Файл скриптов по умолчанию загружается при запуске JAWS и является активным во всех сессиях. Файл скриптов для приложения размещается поверх файла скриптов по умолчанию, когда приложение загружается. JAWS знает какой файл скриптов приложения загружать, поскольку имя файла скриптов для приложения (без расширения) совпадает с именем приложения. Таким образом, файл скриптов по умолчанию называется default.jss (текст скрипта), а приложение notepad.exe будет иметь файл скриптов notepad.jss. Когда пользователь закрывает данное приложение, то файл его скриптов будет выгружен и активными станут все настройки по умолчанию, пока не будет загружен другой файл скриптов приложения.

### 8.2.2. Диспетчер скриптов.

Для модификации имеющихся или создания собственных скриптов следует использовать диспетчер скриптов JAWS. Вызвать его можно клавиатурной командой Ins +0. Команды диспетчера скриптов организованы в виде классического меню. Для входа в меню следует использовать клавишу Alt. В окне диспетчера будет загружен файл скриптов того приложения, которое было активно в момент запуска диспетчера. Если же активное приложение не имеет скриптов, то загружен будет файл по умолчанию Default.jss.

В диспетчере скриптов доступны следующие клавиатурные команды:

Ctrl +N - создать новый файл;

Ctrl +O - открыть файл;

Ctrl +Shift +D - открыть «файл по умолчанию» (Default.jss);

Ctrl +S - скомпилировать и сохранить скрипт;

Ctrl +W - сохранить без компиляции;

Ctrl +E - создать скрипт;

Ctrl +Del - удалить скрипт;

Ctrl +I - вставить вызов функции;

Ctrl +Shift +I - вставить выполнение скрипта;

F2 - Следующий скрипт;

Shift +F2 - предыдущий скрипт;

Ctrl +G - перейти на строку с номером;

Ctrl +L - вывести список скриптов;

Alt +F4 – закрыть диспетчер скриптов.

В окне редактора диспетчера скриптов будут также доступны все клавиатурные команды редактирования текста.

В качестве примера приведем скрипт вставки своей подписи:

1. Запустить диспетчер скриптов командой Ins +0.

2. Чтобы возможность вставки подписи была в любом приложении, загрузить с помощью команды Ctrl +Shift +D файл скриптов по умолчанию.

3. В окне редактора скриптов перейти в конец текста командой Ctrl +End.

4. Открыть диалог создания скрипта командой Ctrl +E.

5. В первом поле открывшегося диалога ввести имя скрипта SignUp (имя выбирается произвольно).

6. В следующем поле клавишей пробел установить флажок, разрешающий назначение клавиатурной команды на выполнение этого скрипта.

7. В следующем поле редактирования ввести краткое описание скрипта «вставка подписи». Эта информация будет озвучиваться при включенном режиме клавиатурной помощи (включается командой Ins +1).

8. В следующем поле вводится более подробное описание скрипта, которое будет озвучено при быстро дважды выполненном нажатии команды вызова скрипта в режиме клавиатурной помощи.

9. В следующем комбинированном редакторе можно ничего не выбирать, а перейти клавишей Tab далее.

10. В этом поле необходимо нажать комбинацию клавиш, которая будет вызывать данный скрипт. Здесь можно нажать, например, комбинацию Ctrl +Alt +1 и перейти клавишей Tab на кнопку «Ok».

11. Клавишей пробел активировать кнопку «Ok», диалоговое окно закроется и фокус окажется в тексте будущего скрипта между строками «Script SignUp ()» и «EndSkript».

12. Здесь необходимо вписать текст скрипта на языке скриптов JAWS. Мы впишем всего одну команду TypeString(“Иван Иванович Иванов”).

13. Теперь следует скомпилировать (т.е. превратить в исполняемую программу) и сохранить созданный скрипт с помощью команды Ctrl +S. Если диспетчер скриптов сообщает об ошибке, проверьте правильность написания единственной строки скрипта. Обратите внимание, что в конце строки с именем функции (после круглой скобки закрыто) точку ставить не следует.

14. Закрыть диспетчер скриптов командой Alt + F4.

Скрипт создан. Теперь его можно использовать в любом текстовом редакторе вводя команду Ctrl +Alt +1.

Контрольные вопросы

1. Что такое скрипт?
2. Приведите пример скрипта программы JAWS for Windows.
3. Какие два типа файлов скриптов существуют в программе JAWS for Windows? Как они работают?
4. Как модифицируются существующие или создаются новые скрипты программы JAWS?
5. Расскажите алгоритм создания скрипта вставки подписи.

Задания для самостоятельного выполнения

1. Создайте скрипт вставки подписи по приведенному выше алгоритму и проверьте его работоспособность в нескольких текстовых редакторах.
2. Включите режим клавиатурной помощи и введите команду вызова созданного скрипта вставки подписи. Затем введите команду вызова скрипта быстро дважды. В чем разница в сообщениях JAWS?
3. Самостоятельно освойте вставку функций с помощью соответствующего диалога, вызываемого командой Ctrl +I.
4. Усовершенствуйте скрипт вставки подписи так, чтобы JAWS объявлял: «подпись вставлена». Используйте диалог вставки функций.
5. Удалите созданный ранее скрипт и создайте новый скрипт вставки подписи так, чтобы подпись вставлялась только в редакторе блокнот, а во всех остальных случаях вставлялся ваш E-Mail.

## 8.3. Программирование скриптов в NVDA

### 8.3.1. Дополнения NVDA.

Дополнение (или аддон) NVDA является аналогом скрипта программы JAWS. NVDA также постоянно использует какие-либо дополнения. Однако, отличия между скриптами JAWS и дополнениями NVDA всё же существуют.

Дополнение NVDA — это стандартный ZIP-архив, но с расширением «.nvda-addon». Его создание состоит из нескольких этапов:

* Создание manifest-файла;
* Добавление плагинов или драйверов;
* Создание кода, срабатывающего при установки или удалении дополнения;
* Локализация (перевод) дополнения;
* Упаковка в один файл.

Рассмотрим эти этапы в отдельности.

Любое дополнение должно содержать manifest-файл с именем «manifest.ini». В нём указываются сведенья о дополнении и его авторе. Этот файл должен иметь кодировку UTF-8. Каждая строка файла должна состоять из названия поля и через знак равно его значения.

На этапе добавления плагинов и драйверов в дополнение помещаются те программы, которые и будут работать в NVDA.

Этап создание кода, срабатывающего при установки или удалении дополнения является не обязательным. Он может быть полезен, если разработчику дополнения требуется встроить в него некоторые дополнительные возможности, например, проверку лицензии или нестандартные процедуры перемещения файлов.

Этап локализации также не обязателен и предназначен для тех случаев, когда автор дополнения реализует в нём поддержку нескольких языков.

Как было сказано выше, дополнение NVDA представляет собой стандартный ZIP-архив с изменённым расширением файла. Таким образом, на последнем этапе упаковки в один файл следует заархивировать все файлы и папки дополнения, входящие в его состав, в архив ZIP любым архиватором. Затем расширение файла следует изменить с «.zip» на «.nvda-addon», после чего дополнение будет полностью готово.

### 8.3.2. Установка аддонов в NVDA.

Установка аддонов или дополнений в NVDA является достаточно простой операцией. При ее выполнении необходимо соблюсти несколько условий:

* Дополнение должно находиться в файле с расширением "nvda-addon";
* В момент установки дополнения обязательно должна быть запущена программа невизуального доступа NVDA.

Для установки дополнения поступать нужно в соответствии со следующим алгоритмом:

1. Скачать дополнение из сети Интернет к себе на компьютер.

2. Запустить программу NVDA.

3. Открыть меню программы NVDA командой Ins +N.

4. Двигаясь стрелкой вниз найти подменю "Сервис" и в нём выбрать команду "Менеджер дополнений".

5. В раскрывшемся диалоге нажать кнопку «Установить» и выбрать на диске скаченное дополнение.

6. Далее необходимо согласиться с установкой данного дополнения.

7. Для закрытия диалогового окна менеджера дополнений следует нажать кнопку «Закрыть».

8. После закрытия окна менеджера NVDA предложит перезапуститься, чтобы дополнение вступило в силу. С этим предложением следует согласиться.

После выполнения данных операций дополнение будет установлено и всеми его функциями можно будет пользоваться.

Контрольные вопросы

1. Что такое дополнение NVDA?
2. Расскажите как создаются дополнения NVDA.
3. Что такое manifest-файл?
4. Какие условия должны быть выполнены для установки дополнения NVDA?
5. Расскажите как можно установить дополнение NVDA.

Упражнения для самостоятельного выполнения

1. Изучите в менеджере дополнений NVDA список установленных дополнений и выпишите в отдельный текстовый файл их описания.
2. Найдите в сети Интернет дополнение «Translate» и установите его.