РАБОТА С ТЕКСТОВИ ФАЙЛОВЕ ВЪВ VISUAL BASIC

За да осъществим входни и изходни операции с текстов файл (т.е., последователен достъп) във Visual BASIC използваме изречения със следни синтаксис:

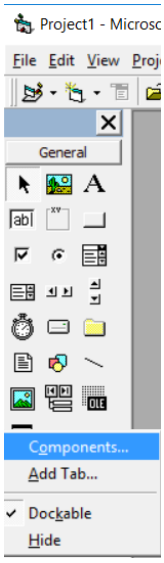
- **Отваряне на файла:**

Open *низ-път за достъп* **For** { **Input** / **Append** / **Output** } **As #** *естествено число*

– *Пътят за достъп* до файла е с правилата на операционната система (символът \ във VB не е специален, той не се удвоява в низовите константи, както в C-подобния синтаксис). Ако файлът е в текущата папка, пътят е самото име (с разширението) на файла. Ако ни е необходим пълният път до папката на изпълнимия файл (все едно – до работната ни папка, ако в момента разработваме приложението), можем да използваме свойството `Path` на обекта `App` (т.е. `App.Path`). Внимание, резултатът е низ, който обикновено НЯМА за последен символ \, но ИМА последен символ \, ако става дума за основната папка на някое устройство. Понеже във VBA на MS Office често ще ползваме тази идея, запомнете я: намираме пълния път за достъп `p` със сигурен ЕДИН символ \ накрая така:

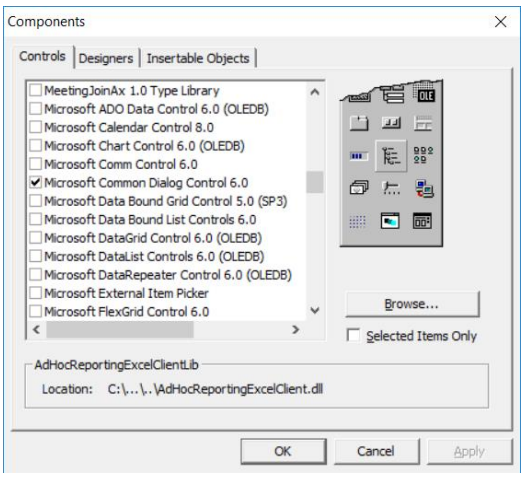
```
Dim p As String
…
p=App.Path
If Right(p,1)<>"\" Then p=p & "\"
…
```

Ако искаме потребителят да задава път за достъп до файл, можем да ползваме, разбира се, обект от типа TextBox, функцията InputBox или други стандартни графични инструменти, като внимаваме с контрола на въведеното. По принцип контролът си е въшкава работа, затова сега ще опишем начин за достъп до стандартни диалози на операционната система. Тъй като те не са част от най-стандартните графични класове във VB 6.0 (които са в ToolBox след стандартна инсталация), ще ни се наложи да ги добавим към ToolBox. Ето как:



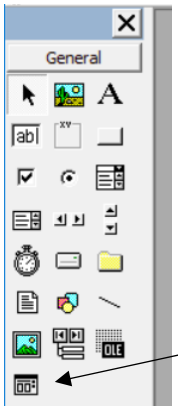
✓За тази операция трябват администраторски права.

✓Контекстно меню (десен бутон на мишката) върху ToolBox, избираме Components.



✓ В появилия се диалог избираме класа Microsoft Common Dialog Control 6.0.

✓ Натискаме бутона OK.



В ToolBox се появява иконка на новия графичен клас, той става част от инструментите на средата ви.

Обекти от този клас осигуряват достъп до графичните диалози на ОС, които дават по-голям контрол. Веднъж регистриран, употребата на такъв клас по-нататък не изисква администраторски права.

Създаването на такъв клас не е приоритет само на разработчиците на ОС, то може да става чрез повечето езици, които поддържат Common Object Model (идеологията преди .NET) на Windows. По-късно и ние ще можем да разработваме такива – даже елементарен език като VB 6.0 има инструменти за това. Ако искате, обаче, да използвате разработени от другиго такива класове (файлове с разширение ocx), бъдете нащрек: те съдържат изпълним код, който може да е зловреден. Просто трябва да сте сигурни в източника на такива инструменти.

– Изборът на една от трите думи (**Input**, **Append** или **Output**) зависи от това, за каква операция искаме да отворим файла: съответно за четене, добавяне или запис с изтриване на предишно съдържание, ако е имало такова.

– *Естественото число*, което задаваме, е номер на описателя на отворения файл (могат да бъдат отворени няколко файла едновременно). В рамките на работата с този файл, до затварянето му,

той се идентифицира с този номер. Често работим с един отворен файл: #1. Ако го затворим, можем да отворим друг със същия номер #1.

- **Работа с отворен вече файл**
  В зависимост от операцията, за която файлът е отворен, използваме следните изречения (ще смятаме, че файлът има описател #1):

– `Line Input #1,`<име_на_променлива-низ>
  Чете от файл, отворен за четене, един ред (до край_на_ред) и прочетеното става съдържание на променливата-низ. Съответства на getline при потоците в C++.

– `Print #1,`< низ1> ; <низ2> ; <низ3> ; …
  Записва във файл, отворен за запис или добавяне, един ред (завършващ с край_на_ред). Съответства при потоците на C++ на *поток << низ1 << низ2 << низ3 <<… <<* **endl**;. Броят на низовете – параметри, **отделени с точка и запетая**, може да бъде произволен – всичко се превръща в един изходен ред (все едно, че низовете са слети с &). Често срещана грешка: ако низовете се отделят със запетая, изречението пак работи, но между низовете в изхода се изпраща табулатор. Ако искаме да създадем един ред чрез няколко изречения Print (най-често в цикъл), изходите преди последното извеждане трябва да завършват с точка и запетая.

– Функция `Eof`(<номер_на_описател>)
  Логическа функция, която връща True, когато е прочетен и последният ред на файла (съответства на метода eof() при входните потоци в C++). Така фрагментът

```
Dim s As String
…
Open "Data.txt" For Input As #1
Do While Not Eof(1)
 Line Input #1,s
 …
Loop
…
Close #1
```

  ще чете на всяка стъпка в цикъла по един ред от файла Data.txt в папката на приложението, от първия до последния ред включително. Ако файлът е празен, Eof(1) ще е True още при отварянето му и цикълът няма да започне, т.е., логически всичко ще е наред.

- **Затваряне на файл**
  `Close #`<номер_на_описател>
  Не оставяйте отворени файлове! Особено е важно това при запис, тъй като ОС, с цел по-голямо бързодействие, използва буфериране и реално записва информацията на изходното устройство при запълване на буфера или при затваряне на файла. Ако пропуснете затваряне, последните записани редове може да липсват в записания файл. И, изобщо, описателите на файловете са общ ресурс на ОС и работата с тях трябва да е „чиста“ (както с паметта или входния буфер в C/C++) – използваният общ ресурс се нуждае от „хигиена“: отваряме-затваряме, искаме памет-връщаме памет, получаваме чист буфер-оставяме чист буфер.

Vusual BASIC поддържа още инструменти за работа с последователни файлове, но описаните по-горе са достатъчни и не са свързани със специфика на самия език.