Обработка на грешки във Visual BASIC

Обработката на грешки е много експлоатирана съвременна идеология за програмиране. Става дума за следното:

Ако една програма се окаже в „нормална“ обстановка (има коректни входни данни, нормален достъп до ресурсите и т. п.), което е най-честият случай, тя си работи „нормално“ (така, както я е предвидил разработчикът). Да се отговори на въпроса, обаче, дали всичко е в „очаквано“ („нормално“) състояние, понякога е задача, по-трудна от проблема, който се разрешава. Във всички случаи, обаче, проверката на състоянието отнема ресурс: памет, време (а най-често – и двете), програмистки труд. Тя е и чест източник на грешки, защото е свързана с проверки, понякога доста сложни.

В съвременното програмиране има и друг подход, а именно – обработка на грешки. Няма съвременен език който да не поддържа тази идеология. Естествено, не по един и същ начин. В C-подобните езици, например, това става със специални блокови изречения **try-catch**, в Pascal – с подобни конструкции (Try-Except-Finally) и т. н. Но идеята е една: програмира се „нормален“ ход на програмата, а възникнали грешки (наричани „изключения“ – exceptions) се обработват специално. Тази проста идея премахва до голяма степен недостатъците, за които стана дума по-горе: програмата в големия процент от случаите върви именно в „нормалните“ си граници, не се губи ресурс за допълнителни проверки, опростява се кодът и се намаляват възможностите за логически грешки.

Във Visual BASIC е възприета следната идеология при обработка на грешки:

Всяка подпрограма (все едно дали е от тип Sub, Function или Property) може да се раздели на две части: „нормален“ ход и обработка на грешки. За целта се използват следните езикови конструкции:

- **Етикети**. Във Visual BASIC етикетите са идентификатори, записани на самостоятелен ред и завършващи с двоеточие (по подобие на C/C++). Управлението на изчислителния процес може да се предава на изречението, следващо етикет, с помощта на инструкцията **GoTo *етикет*** (както в C/C++). Употребата на етикети не е ограничена само за обработка на грешки, но там най-често е задължителна. Етикетите във Visual BASIC са локални идентификатори, важат за подпрограмата, в която са дефинирани, т.е. можем да ползваме едни и същи етикети при обработка на грешки (както и ще правим, за по-голямо удобство).
- Указание към изчислителния процес: изречение **On Error**…

  Преди да има опасност от възникване на грешка, подпрограмата трябва да е снабдена с указание къде да предаде управлението при възникване на грешка. Най-често използваният синтаксис изглежда така: **On Error GoTo *етикет***. Това е указание, че при възникване на грешка управлението автоматично преминава на изречението, следващо указания *етикет*.

  Специален случай на това указание е изречението **On Error GoTo 0**, което премахва възможността след него подпрограмата да обработва грешки (до евентуално срещане на друго указание). Тази форма на On Error се използва най-вече по време на разработка.

  Ще спомена и специалния синтаксис **On Error Resume Next**. Това указание се чете така: „Ако възникне грешка, премини направо на изречението, следващото това, в което е възникнала грешката.“ С други думи, да не се обръща внимание на възникнала грешка. Изглежда много привлекателно, но внимание: за да не обърнем внимание на грешка трябва да сме сигурни, че тя може да се игнорира! Такива случаи не са много, трябва добре да е помислено преди това.

  Мястото на изреченията On Error… (най-чисто е, ако използвате само едно такова, ние така ще правим) е в „нормалната“ част от подпрограмата, преди да се стигне до изречение, в което би могла да възникне грешка. Често пишем едно такова изречение като първо, непосредствено след заглавната част на подпрограмата, която пишем и смятаме да снабдим с възможности да обработва грешки.

- Обратен преход от частта за обработка на грешки към „нормалната“ част: изречения от тип **Resume** …
  Когато възникне грешка в подпрограма, обработваща грешки, информация за мястото на възникване се запазва в стек. Макар че най-често Visual BASIC се справя с изчистването на този стек и от подпрограма за обработка на грешки можем да се върнем при повиквача, това не се приема за добра идея. Правилният начин за излизане е от „нормалната“ част. За целта се използват вариантите на изречението **Resume**, които премахват информацията, записана в стека за грешки. Използваме следните му варианти:
  - ✓ **Resume** без параметри връща управлението на изречението, където е възникнала грешката. ВНИМАНИЕ: в частта за обработка трябва да сме взели мерки това изречение вече да завърши без грешка, иначе рискуваме безкраен цикъл на преходи от едната в другата част на подпрограмата!
  - ✓ **Resume** *етикет, който се намира в „нормалната“ част* връща управлението на изречението след указания етикет.
  - ✓ **Resume Next** връща управлението на изречението, непосредствено следващо това, в което е възникнала грешката.

Ето една принципна схема на подпрограма, в която е предвидена обработка на грешки:

Описание	**Пример**
Заглавна част на подпрограмата	Function Div(ByVal a As Long, ByVal b As Long) As Long
Указание, ако възникне грешка	On Error GoTo er
„Нормален“ ход	Div=a \ b 'Целочислено деление
етикет за изход	ex:
„Нормален“ изход	Exit Function
етикет за грешките	er:
Частта, обработваща грешки	MsgBox "Деление на нула",,"ГРЕШКА"
Подходящо връщане в „нормалната“ част	Resume ex 'Обратно за коректен изход
	End Function

Специфики:

Не използваме етикет *err* (с *две r*), защото има обект Err, също свързан с обработката на грешки. Именно чрез него можем изкуствено да предизвикаме грешка с метода му Err.Raise.