Тестирование и отладка программ

Всякая программа содержит ошибки. Задача разработчика – свести их количество к минимуму и не допустить серьезных сбоев при эксплуатации программы.

После программирования программист переходит к тестированию и отладке программы.

Тестирование – проверка работоспособности программного продукта при всевозможных вариантах его эксплуатации *с целью обнаружения ошибок*.

Отладкой называется процесс поиска и устранения ошибок.

После отладки необходимо повторить весь процесс тестирования, так как устранение одних ошибок нередко приводит к появлению других.

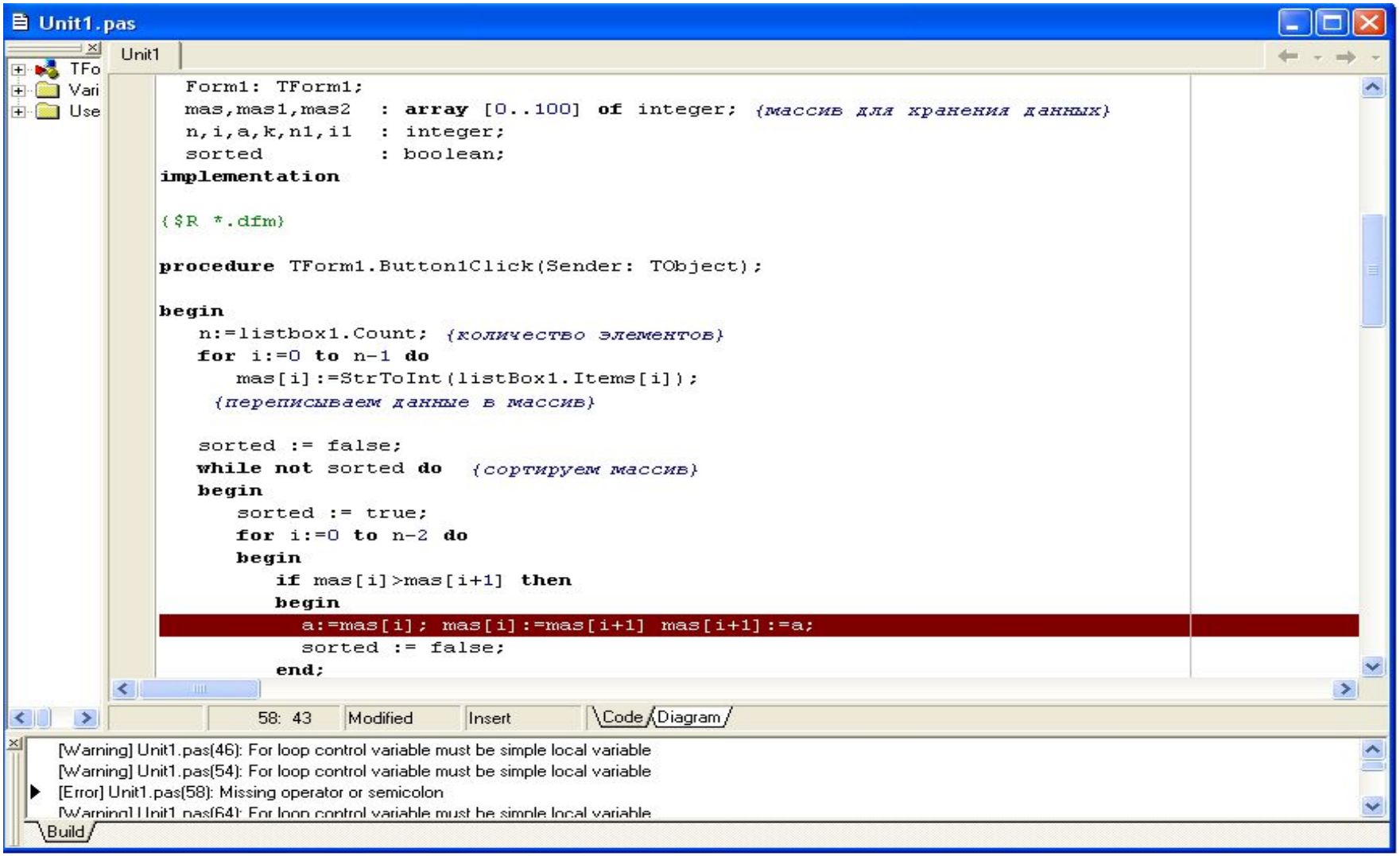
Типы ошибок в программах

Синтаксические ошибки, их также называют ошибками времени компиляции (Compile-time error), наиболее легко устранимы. Их обнаруживает компилятор, а программисту остается только внести изменения в текст программы и выполнить повторную компиляцию

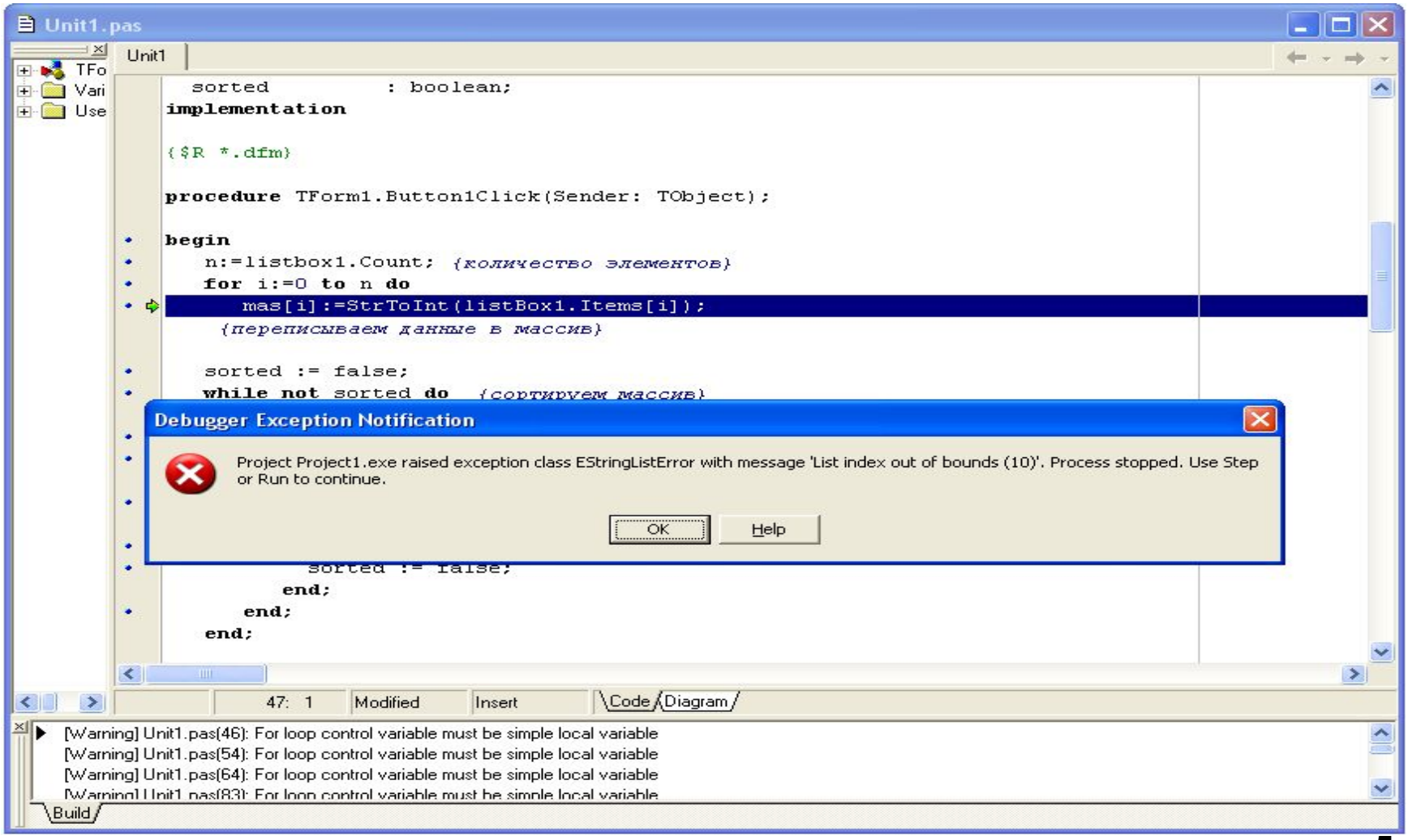
Ошибки времени выполнения (Run-time error) возникают не при каждом запуске программы, а лишь при определенном наборе входных данных (например, делении на ноль или вводе некорректной даты). Для их выявления требуется тщательно подготовить тестовые примеры. Если причиной являются не программные ошибки, а действия пользователя, то в программе должна быть предусмотрена обработка исключительных ситуаций

Алгоритмические ошибки. Компиляция программы, в которой есть алгоритмическая ошибка, завершается успешно. При пробных запусках программа ведет себя нормально, однако результата получается неверный. Для того чтобы устранить алгоритмическую ошибку, приходится анализировать алгоритм, вручную "прокручивать" его выполнение

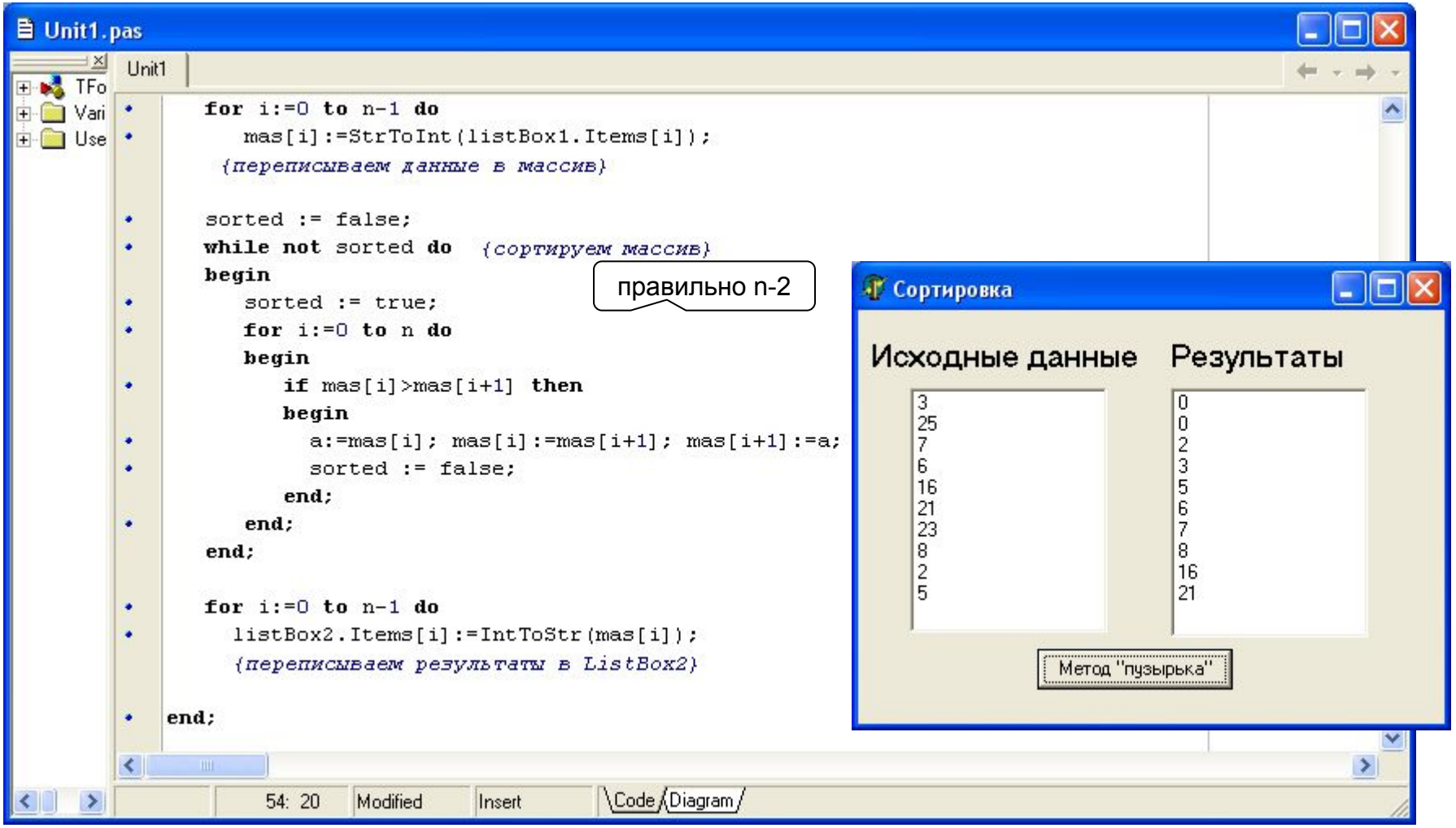
Синтаксические ошибки



Ошибки времени выполнения



Алгоритмические ошибки



Методы тестирования программ

Авторское тестирование (еще его называют методом «белого ящика») – проверка программы исходя из ее логики. Автор, зная внутреннюю логику программы, подбирает тестовые примеры так, чтобы проверить работу всех ее блоков.

Неавторское тестирование (стороннее, по методу «черного ящика») – проверка программы с точки зрения пользователя. Тестовые примеры подбираются исходя из реальных ситуаций, возникающих в ходе эксплуатации.

В крупных фирмах – разработчиках ПО тестированием занимается специальный персонал. В небольших коллективах практикуется «перекрестное тестирование»

Массовое тестирование. Для продуктов, выпускаемых на рынок, используют тестирование широким кругом потенциальных пользователей. Для этого выпускают так называемую **«***бета-версию***»** продукта и распространяют ее (обычно бесплатно) без гарантий надежной работы. Сбор информации об ошибках и отказах дает неоценимый материал для отладки.

Методы отладки программ

Трассировка — это процесс выполнения программы по шагам (step-by-step), инструкция за инструкцией. Во время трассировки программист дает команду: выполнить очередную инструкцию программы.

Метод точек останова – заключается в том, что программист помечает некоторые инструкции программы (ставит точки останова), при достижении которых программа приостанавливает свою работу, и можно начать трассировку или проконтролировать значения переменных.

Наблюдение значений переменных

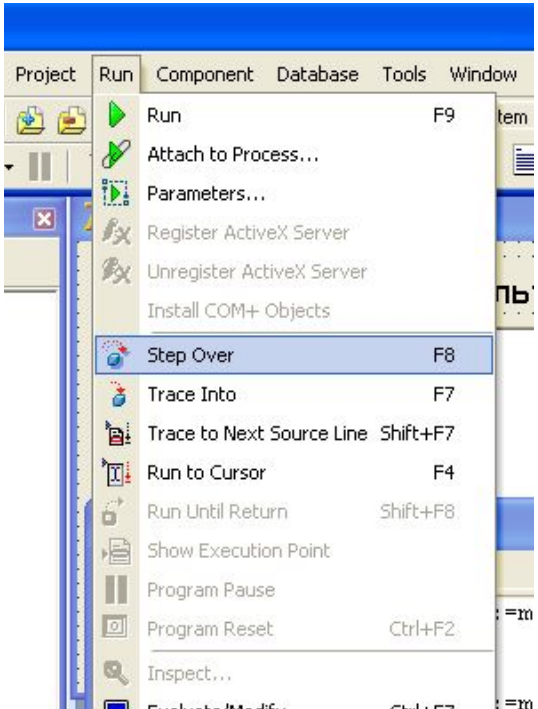
Как правило все методы используются совместно

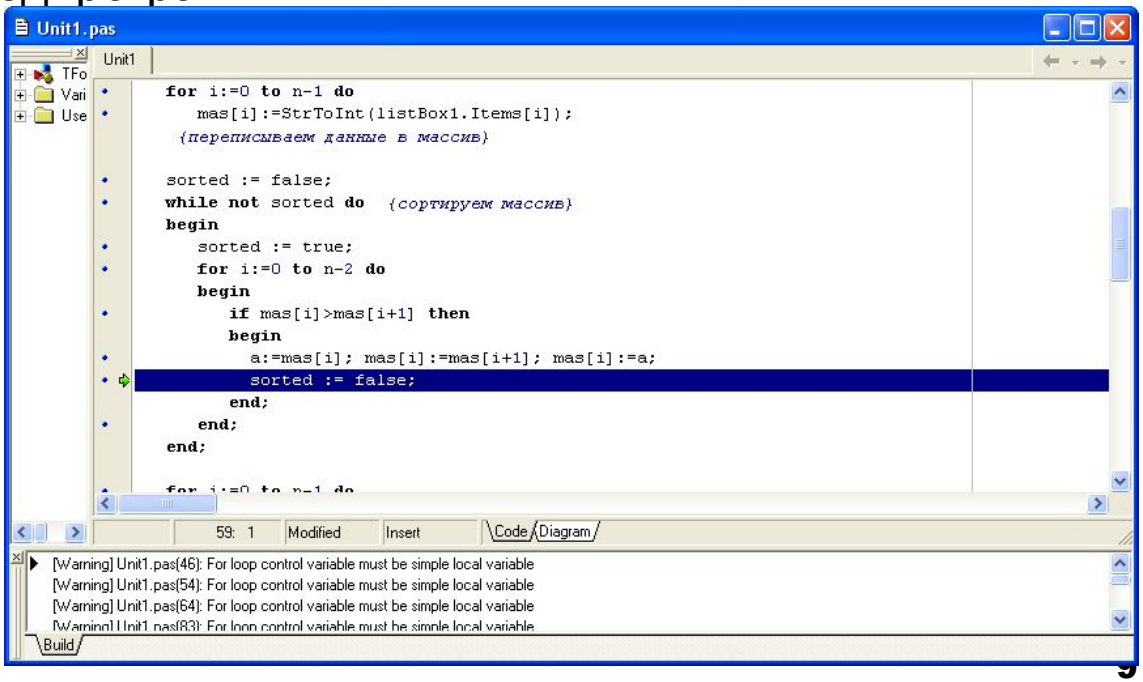
Средства отладки в Delphi: трассировка

Delphi обеспечивает два режима трассировки: без захода в процедуру (Step over) и с заходом в процедуру (Trace into).

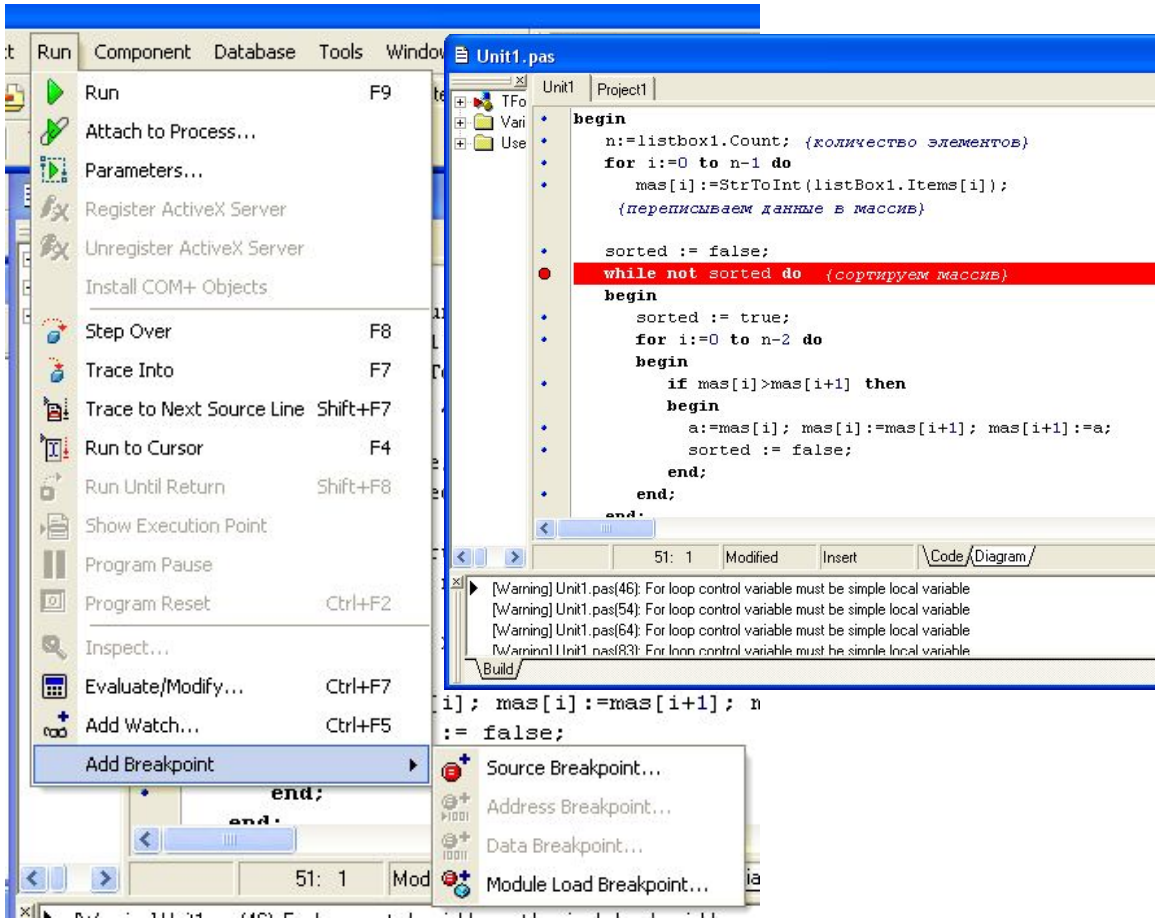
Режим трассировки без захода в процедуру выполняет трассировку только главной процедуры, при этом трассировка подпрограмм не выполняется, вся подпрограмма выполняется за один шаг.

В режиме трассировки с заходом в процедуру выполняется трассировка всей программы, т. е. по шагам выполняется не только главная программа, но и все подпрограммы.

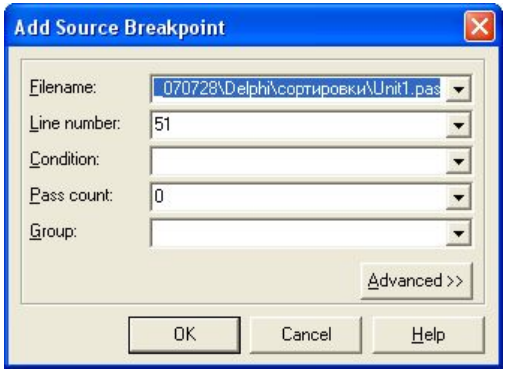




Средства отладки в Delphi: точки останова

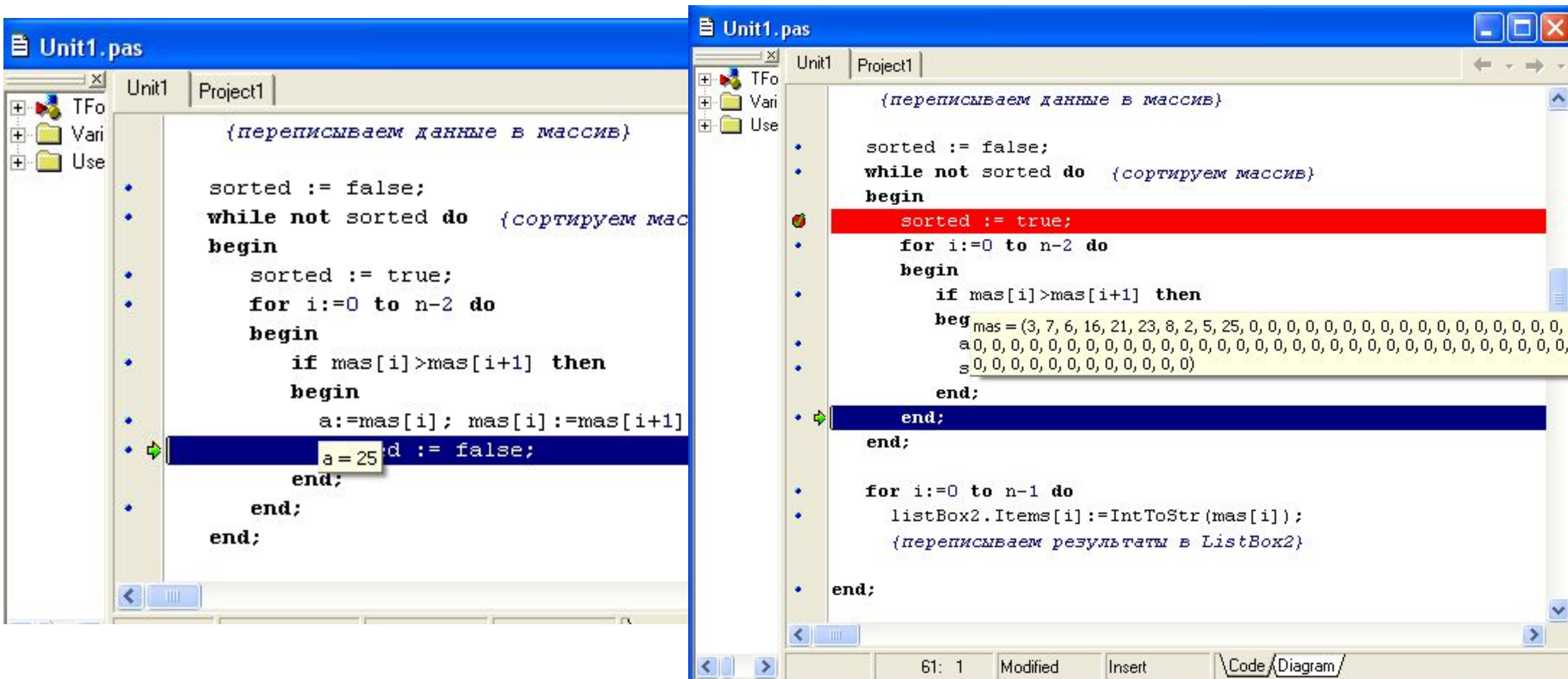


- Программа доходит до указанной точки и останавливается. Затем можно выполнить трассировку
- Для точки останова можно задать некоторые дополнительные параметры при помощи диалогового окна **Add Source Breakpoint** меню **Run.**



Средства отладки в Delphi: наблюдение значений переменных

Во время трассировки можно увидеть текущее значение любой переменной в данный момент, наведя курсор мыши на имя переменной



Обеспечение надежной работы программы. Обработка исключительных ситуаций

Синтаксические и алгоритмические ошибки должны быть устранены в ходе тестирования и отладки.

Ошибки времени выполнения не могут быть устранены программистом, т.к. связаны с внешними действиями (ошибками пользователей, отказами оборудования и т.п.)

Для исключения ошибок времени выполнения в языке Object Pascal существуют средства **обработки исключительных ситуаций**, позволяющие «перехватывать» и обрабатывать возникающие ошибки.

Это конструкции *try…finally* и *try … except*.

Конструкции *try…finally* и *try … except*

```
try
   <инструкция>
   <инструкция>
   …
finally
   <инструкция>
   …
end;
```

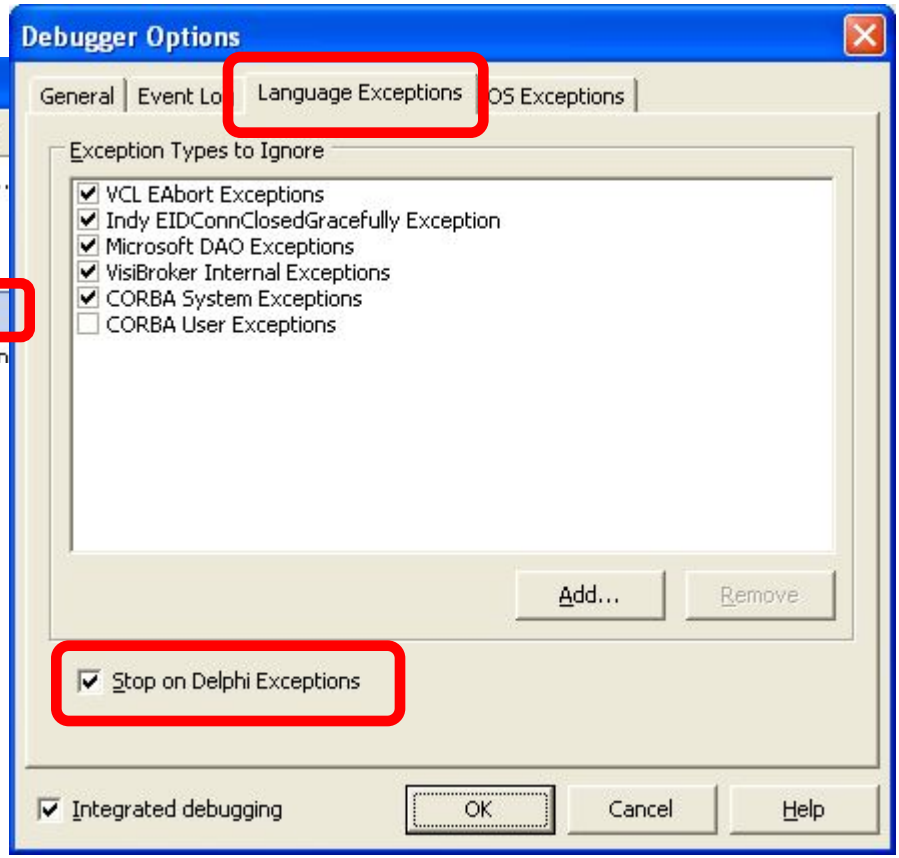
```
try
   <инструкция>
   <инструкция>
    …
except
   on Exception1 do <инструкция>
   on Exception2 do <инструкция>
   …
else {может отсутствовать}
   <инструкция>
   {обработчик ИС по умолчанию}
end;
```

Некоторые важные ИС

Ситуация	Возникает при условии
EConvertError	Невозможность преобразования данных
EOutOfResources	Нехватка системных ресурсов
ИС целочисленной математики	
EDivByZero	Попытка деления на ноль
ERangeError	Число или выражение выходит за допустимый диапазон
EIntOverflow	Целочисленное переполнение
ИС математики с плавающей точкой	
EInvalidOp	Неверная операция
EZeroDivide	Попытка деления на ноль
EOverflow	Переполнение с плавающей точкой

Обработка ИС, описанная в блоке try, будет выполняться должным образом только при запуске exe-файла приложения прямо из Windows.

Для того, чтобы обработка выполнялась и из среды Delphi, нужно выполнить следующее:



Снять флажок

Пример

Без учета ИС

С учетом, но без обработки

С обработкой ИС

```
with StringGrid1 do begin
    sum:=0;
    nt:= RowCount * ColCount;
    for i:=0 to ColCount-1 do
        for j:=0 to RowCount-1 do
        try
            sum := sum+StrToInt(Cells[i,j]);
         except
           on EConvertError do nt:=nt-1;
         end;
    end;
    StringGrid2.Cells[3,1]:=IntToStr(sum);
    StringGrid2.Cells[3,2]:=FloatToStr(sum/nt);
end;
```

Лекция окончена

Спасибо за внимание