Обработка исключительных ситуаций. Защищенные блоки. Стандартные исключения и примеры их использования. Создание собственных исключительных ситуаций.

Обработка исключительных ситуаций. Защищенные блоки.

Во время работы программы могут возникать те или иные ошибки выполнения (деление на ноль, некорректная операция при работе с файлами и т.п.), называемые исключительными ситуациями (ИС). При появлении подобных ошибок возникает окно с сообщением, соответствующим конкретной ИС.

Программист вправе самостоятельно обработать любую ИС, т.е. отследить её возникновение и указать собственный алгоритм реакции на данную ошибку. В этом случае окно с сообщением об ошибке не возникает[14].

Для обработки ИС в Object Pascal предусмотрен механизм защищенного блока. Существует два типа защищенных блоков - except и finally:

Блок ***try..finally..end***	Блок ***try..except..end***

try
  <операторы>
finally
  <операторы>
end;
	try
	  <операторы>
	except
	  <обработчики ИС>
	else
	  <оператор>
	end;

В блоке *try..finally..end* вначале выполняются операторы секции *try..finally*. Если при выполнении операторов не возникло ИС, то затем выполняются операторы секции *finally..end*. Если в одном из операторов секции *try..finally* возникла ИС, то оставшиеся операторы пропускаются и управление сразу передается в секцию *finally..end*. Обычно конечную секцию блока составляют те операторы, выполнение которых должно произойти как при нормальном ходе программы, так и в случае ИС.[15]

В блоке *try..except..end* также вначале выполняются операторы секции *try..except*. Если операторы выполнены без возникновения ИС, то работа блока на этом завершается и управление передается оператору, стоящему после *end*. Секция *except..end* состоит из операторов вида:

on <класс исключительной ситуации> **do** <оператор-обработчик>;

Если в секции *try..except* произошла ИС, то среди операторов секции *except..end* будет выполнен первый по очереди, класс которого соответствует типу ИС. Если такового не окажется то

[14] Однако при запуске проекта в среде Delphi в целях отладки все равно появляется окно с сообщением об ИС. Для отключения этого режима снимите флажок с переключателя *Break on Exception* на странице *Preference* меню *Tools/Environment Options*.

[15] К таким операторам можно отнести процедуры закрытия файлов, разрушения временных объектов и т.п. Поэтому часто защищенный блок *finally* называют блоком защиты ресурсов.

управление передается оператору, стоящему за *else*, или, если секции *else..end* нет, то возникнет окно с сообщением о возникновении конкретной ИС.

Все классы ИС порождены от класса *Exception* и содержат свойство *Message* с текстом сообщения об ошибке. В приведенной ниже таблице представлены названия классов и условия возникновения наиболее важных ИС.

ИС работы с памятью	
EOutOfMemory	Недостаточно места в куче памяти.
EInvalidPointer	Недопустимый указатель (обычно Nil).
EAccessViolation	Попытка обратиться к недоступной для программы области памяти или обращение к полям несуществующего объекта.
ИС целочисленной математики (порождены от EIntError)	
EDivByZero	Попытка деления на ноль.
ERangeError	Выход значения за допустимый диапазон.
EIntOverflow	Целочисленное переполнение.
ИС математики с плавающей точкой (порождены от EMathError)	
EInvalidOp	Неверная операция.
EZeroDivide	Попытка деления на ноль.
EOverflow	Переполнение с плавающей точкой.
EUnderflow	Исчезновение порядка.
Другие ИС (но далеко не все)	
EStackFault	Ошибка стека.
EPrinter	Ошибка принтера.

При возникновении ИС автоматически создаются и уничтожаются объекты классов обработчиков ИС. Если программист пожелает использовать поля или методы класса обработчика, он должен поименовать автоматически созданных объект, указав это имя перед названием класса:

```
on EObject:EClassName do ...
```

Например, стандартный обработчик ошибок ввода-вывода кроме свойства *Message* содержит целочисленное свойство *ErrorCode* с номером ошибки. В примере ниже в случае ошибки ввода-вывода её номер будет выведен в специальное окно-сообщение

```
try
    try
      Reset(F);
      while not EOF(F) do
      begin
        ...
      end;
    except
      on E:EInOutError do
        ShowMessage('При выполнении файловой операции произошла ошибка №'
```

```
            +IntToStr(E.ErrorCode);
    end;
finally
      CloseFile(F);
end;
```

Вызвать ИС (т.е. инициировать соответсвующую реакцию программы) можно и не совершая ошибки выполнения. Для этого используется зарезервированное слово *raise*. В секциях *try..except* и *try..finally* такой вызов передаст управление секциям *except..end* и *finallyt..end*, а применение данного вызова в завершающих секциях защищенного блока *except..end* и *finallyt..end* завершит работу блока.

Сам по себе вызов *raise* приведет к генерации ИС общего класса *Exception*. Если возбуждается ИС конкретного вида, то вызывается соответствующий конструктор, например:

```
raise EInOutError.Create('Ошибка ввода-вывода!');
```

Создание собственных исключительных ситуаций.

Программист вправе создавать собственные классы ИС, порождая их, например, от классов *Exception* или *EAbort*. Объявление нестандартной ИС когда необходимо научить программу распознавать некорректные наборы данных и соответствующим образом на них реагировать.

Добавим к нашему примеру (класс *TDot*) собственную исключительную ситуацию. Для этого в модуль *Dots.pas* добавим описание нового класса *EDotError*:

```
type EDotError = class (EAbort)
    end;
```

Как видите, новый класс ничем кроме названия не отличается от своего родителя *EAbort*. В конец метода *MoveTo* внесем строки, возбуждающие соответствующую ИС:

```
if (x=0) and (y=0) then raise EDotError.Create('Dot in Zero!');
if (x<0) and (y<0) then raise EDotError.Create('Negative coords!');
```

Теперь при переносе точки в начало координат или в третью четверть будет возбуждаться ИС класса *EDotError* с соответствующим текстом. В программе, использующей данный модуль, те операторы, которые могут изменить координаты точки можно оформить в виде защищенного блока, например[16]:

```
try
  ...<операторы>;
except
  on e:EDotError do begin
    Writeln(E.Message);
    MoveTo(1,1);
  end;
end;
```

Очевидно, что оформление ИС можно заменить системой проверок if...then, но во многих случаях такая замена будет довольно громоздка и запутанна, а внешний вид текста программы будет безусловно испорчен.

[16] Можно, конечно, и не оформлять защищенного блока, но в этом случае при возникновении ИС будет происходить останов программы.