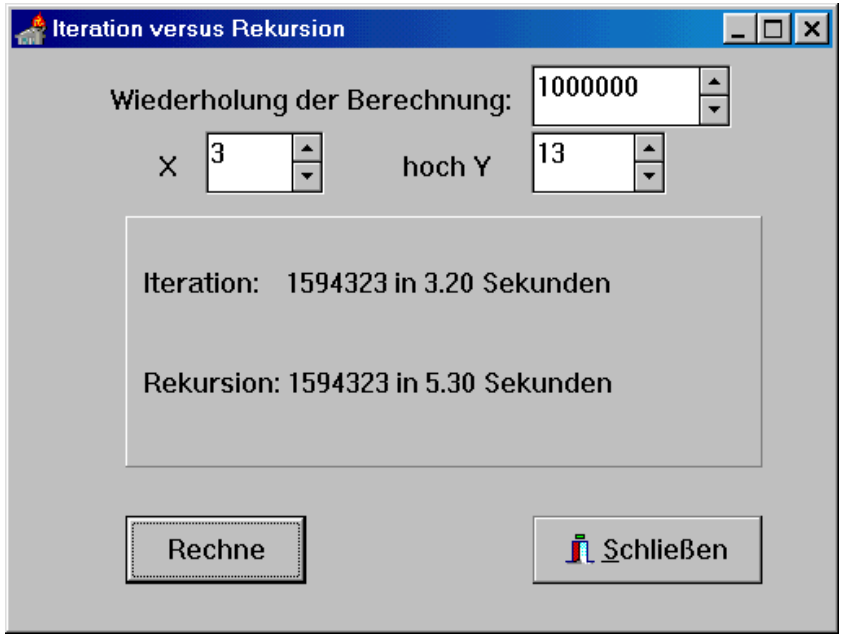
Aufgabenstellung (für alle in dieser Stunde zu schaffen):

Entwickelt ein Programm, welches **x hoch y** rekursiv und iterativ berechnet und die Rechenzeiten für beide Varianten vergleicht:



Da die Berechnung (auch mit hohen Zahlen) innerhalb von Nanosekunden abläuft, muß sie N-Mal durchgeführt werden, um vergleichbare Zeiten zu erhalten.
Der Taktgeber für die PC-Uhr liefert nämlich nur alle 18ms ein Signal, so daß die Zeitangabe allenfalls im Zehntel-Sekunden-Bereich ernst genommen werden kann.
Daß das Ergebnis den longint-Bereich überschreiten kann, braucht nicht beachtet zu werden.

Evtl. benötigt:

```
function Time: TDateTime;
```
Die Funktion Time gibt die gegenwärtige Uhrzeit zurück.

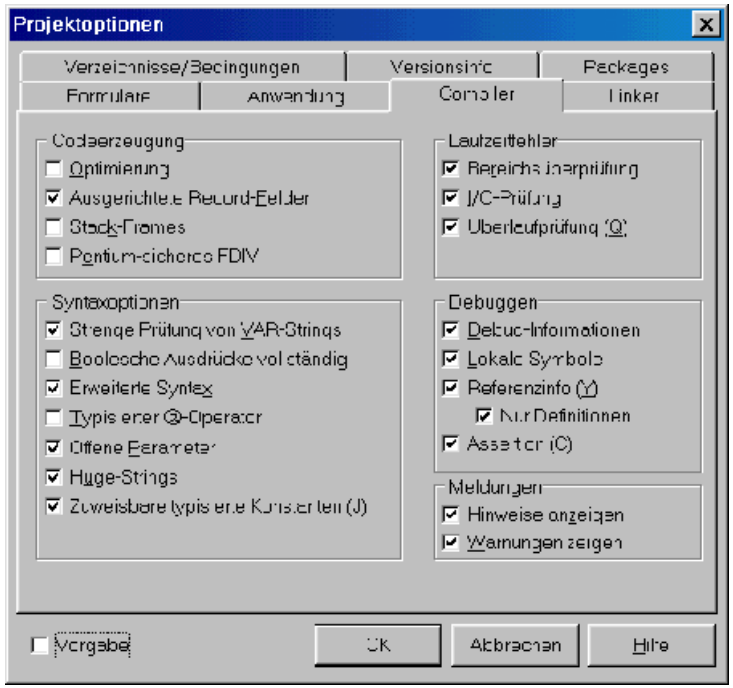
```
procedure DecodeTime (Time: TdateTime;
                      var Hour, Min, Sec, MSec: Word);
```
Die Prozedur DecodeTime teilt den als Parameter Time angegebenen Wert in Stunden, Minuten, Sekunden und Millisekunden auf.

```
Screen.Cursor:= crHourGlass;
```
Diese Eigenschaft verändert den Cursor.

Hinweise zu Compiler-Optionen:

Damit Euer Programm auf unterschiedlichen Compilern gleich compiliert wird, achtet auf folgende Einstellungen:

unter *Projekt/Optionen/Compiler*



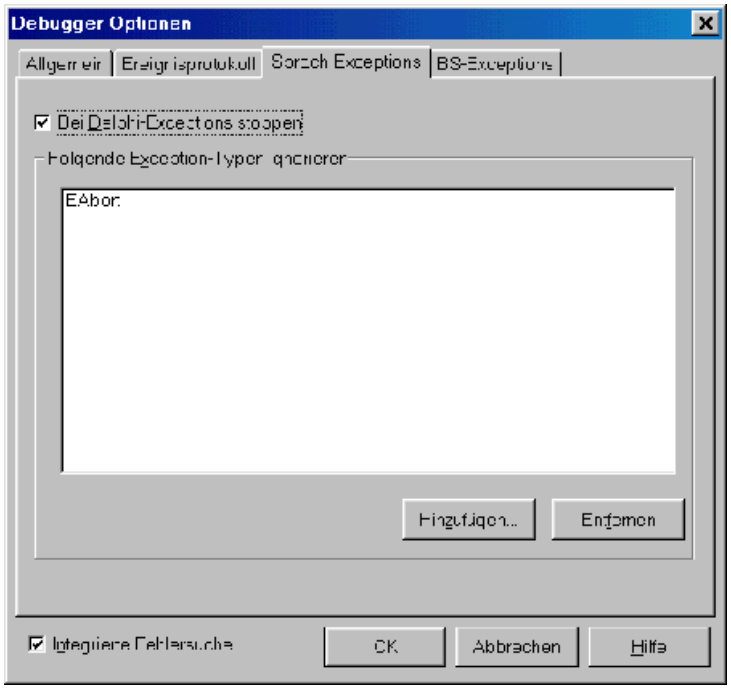
Dies ist eine sinnvolle Einstellung für Übungen. Wie alle anderen Projektoptionen weden auch diese in der *.dof _Datei gespeichert. Sie gelten dann für alle Units des Projekts. Man kann die obigen Einstellungen auch direkt im Quelltext überschreiben:

Codeerzeugung	Auswirkung
Optimierung	Aktiviert die Compileroptimierung. Entspricht {$O}. Mit Optimierung könnt Ihr beim Debuggen keine Variableninhalte kontrollieren.

Laufzeitfehler	Auswirkung
Bereichsüberprüfung	Prüft, ob Array- und String-Sammlungen ohne Bindungen bestehen. Entspricht {$R}.
I/O-Prüfung	Prüft auf I/O-Fehler nach I/O-Aufrufen. Entspricht {$I}.
Überlaufprüfung	Prüft auf Überläufe bei Integer-Operationen. Entspricht {$Q}.

Also schreibt am Anfang einer Unit : *{$O-,R+,I+,Q+}*

Unter: *Tools/Debugger-Optionen/Sprach-Exceptions*



könnt Ihr das Exception-Handling steuern. Während der Übungen solltet Ihr Delphi bei Exceptions stoppen lassen und die integrierte Fehlersuche anschalten.
Erst wenn Ihr mit Exceptions arbeitet (zum Ende der Übungen) könnt Ihr bei einem fertig ausgetesteten Programm Delphi die Fehlersuche abnehmen.