Projekt Brüche kürzen

Schreiben Sie ein Programm "Brüche kürzen", welches 4 passende Edit-Komponenten und zwei Knöpfe enthält. Die Edit-Komponenten sollen Zähler und Nenner von 2 Brüchen darstellen. Auf Knopfdruck soll ein in den ersten beiden Editfeldern stehender Bruch gekürzt in den anderen beiden Editfeldern erscheinen (siehe Abbildung).

1. Starten Sie Delphi, wählen Sie **Datei Neue Anwendung** und speichern Sie mit **Projekt speichern unter**. Legen Sie dabei einen **Ordner "Brueche"** unter Ihrem Namen an.

---

2. Geben Sie der FORM die **Caption "Brüche kürzen"** und den **Namen "FoBrueche".**
3. **Erstellen Sie die FORM analog zum nebenstehenden Layout (max. 15min)**

Verwendete Komponenten:
- **Panel** für die Überschrift,
- **GroupBox**
  mit **Edit** Feldern für die Eingabe
  (EdZaehler1, EdNenner1)
  und **Edit** Feldern für die Ausgabe
  (EdZaehler2, EdNenner2)
  (**Edit.Text** bitte mit numerischen Startwerten belegen; Ausgabefelder ReadOnly und grau unterlegt!)
- **Button** für
  - Berechnungsstart (BuKuerzen)
  - und Schließen (BuSchliessen).

---

4. Implementieren Sie im "Hauptprogramm" die **Ereignisbehandlungsroutine** (Aufruf im Entwurfsmodus durch Doppelklick auf den Button Kürzen)

```
procedure TFoBrueche.BuKuerzenClick(Sender: TObject);
```

- die Umwandlung und Zuweisung der String-Parameter der Edit-Eingabefelder in Ganzzahlwerten z, n mittels der Standardkonvertierungsfunktion

```
StrToInt(s: String): Integer;
```

- die Umwandlung der Integer-Ergebniswerte z und n mit der inversen Standardkonvertierungsfunktion

```
IntToStr(x: Integer): String;
```

und dessen Ausgabe im entsprechenden Edit-Ausgabefeld.

---

5. Implementieren Sie eine **eigene Prozedur**

```
procedure TFoBrueche.kuerze(var zaehler, nenner: integer);
```

der sie Zähler z und Nenner n des zu kürzenden Bruchs übergeben (**`kuerze(z,n)`**); die Prozedur soll dann die gekürzten Werte zurückliefern.

Wie kürzt man eigentlich einen Bruch? Indem man Zähler **z** und Nenner **n** so lange durch ihre gemeinsame Faktoren teilt, bis sie eben keine gemeinsamen Faktoren mehr haben. Die folgende Tabelle stellt das Vorgehen am Beispiel des Bruches 168/180 dar:

Schritt	Zähler	Nenner	Teiler	t teilt Z	t teilt N	z DIV t	n DIV t	Aktion
1.	168	180	2	**ja**	**ja**	84	90	**Kürzen mit t=2**
2.	84	90	2	**ja**	**ja**	42	45	**Kürzen mit t=2**
3.	42	45	2	ja	nein	--	--	t erhöhen
4.	42	45	3	**ja**	**ja**	14	15	**Kürzen mit t=3**
5.	14	15	3	nein	ja	--	--	t erhöhen
6.	14	15	4	nein	nein	--	--	t erhöhen
7.	14	15	5	nein	ja	--	--	t erhöhen
...	...	...	...	...	...	...	...	...
16.	14	15	14	ja	nein	--	--	Ende

Es ist natürlich wenig effizient, den Teiler 4 oder 8 zu probieren, wenn zuvor schon sichergestellt wurde, dass Zähler und Nenner keinen Primfaktor 2 mehr enthalten. Das sture Vorgehen, alle Teiler t zu testen, die nicht größer als die kleinere der beiden Zahlen z und n ist, hat aber den Vorteil, einerseits korrekt und andererseits leicht programmierbar zu sein!

Ob **t** ein Teiler von **zaehler** ist, testet man mit der Modulo-Funktion **`(zaehler MOD t)`**, welche den Rest einer Ganzzahldivision zurückgibt. Damit Zähler und Nenner Integerwerte bleiben, müssen Sie **`(zaehler DIV t)`** benutzen, und **nicht** **`(zaehler / t)`**!

Algorithmus Bruchkürzen / Schleife mit Anfangsbedingung WHILE (solange)

```
Setze t auf 2;
    Solange (t <= zaehler) und (t <= nenner) ist, wiederhole:
       Wenn (zaehler MOD t = 0) und (nenner MOD t = 0) sind,
       dann
          teile zaehler durch t
          teile nenner  durch t
       andernfalls
          erhöhe t um 1.
```

6. Eine wesentlich effizientere Methode besteht in der Division durch den **größten gemeinsamen Teiler (ggT) von Zähler z und Nenner n.**

Die schnellste Methode zur Berechnung des ggT wurde schon 300 v.Chr. durch Euklid beschrieben und zählt als ältester nicht–trivialer Algorithmus (→ www.wikipedia.org):

Euklidscher Divisions–Algorithmus

```
Sei a die größere
der beiden Zahlen a und b.

   Solange b größer als 0 ist,
   Wiederhole folgendes:
       berechne den Rest r,
         den a bei der Division
               durch b läßt;
       weise a den Wert von b zu;
       weise b den Wert von r zu.

Danach ist a der gesuchte ggT.
```

a) Euklid nannte sein Verfahren **„antepheiresis“ (Wechselwegnahme).**
Um zu verstehen, was es damit auf sich hat, sollten Sie den Euklidschen Algorithmus auf dem Papier durchspielen z.B. für die Zahlenpaare (560, 91) und (972, 666).

b) Deklarieren und implementieren Sie die **Prozedur "ggT"**:

```
procedure ggT(a, b: integer; var GgT: integer);
```

Sie soll 3 Parameter erhalten: In den ersten beiden sollen die Werte von Zähler und Nenner des zu kürzenden Bruchs übergeben werden (→ Werteparameter), im dritten soll die Prozedur den ggT der beiden Zahlen zurückgeben (→ Variablenparameter !!)

c) Implementieren Sie die ggT-Prozedur gemäß diesem Algorithmus! Sie benötigen dabei eine Schleife mit Abbruchbedingung, deren Syntax hierzu kurz wiederholt werden soll:

Schleife mit Zähler	Schleife mit	
	Durchlaufbedingung	Abbruchbedindung

`for i := 1 to 20 do`
`begin`
`  <Anweisung>`
`end;`
	`While <Bedingung> do`
	`begin`
	`  <Anweisung>`
	`end;`
		`repeat`
		`  <Anweisung>`
		`until <Bedingung>;`

Information zu Werte- und Variablen-Parameter:

Wenn einer Prozedur

beim Aufruf ein **Wertparameter b** übergeben wird:

```
Procedure tuwas (    a: real);
```

```
tuwas(b),
```

dann wird der Wert von **b** in der **lokalen** Variablen a gespeichert und steht nur innerhalb der Prozedur zur Verfügung. Nach dem Verlassen der Prozedur ist die Variable a verloren.

Wenn einer Prozedur

beim Aufruf ein **Variablenparameter b** übergeben wird:

```
Procedure tuwas (var a: real);
```

```
tuwas(b),
```

dann werden alle Veränderungen innerhalb der Prozedur tatsächlich an **b** vorgenommen und stehen somit auch außerhalb der Prozedur zur Verfügung.

Merke

- Will man nur **Werte** an eine Prozedur übergeben, verwendet man **Werteparamter**;
- Soll die Prozedur **äußere Variablen verändern,** verwendet man **Variablenparameter**.

Wenn man in der Parameterliste einer Prozedur sowohl Werte- als auch Variablenparameter braucht, dann werden die Deklarationen durch Strichpunkte voneinander getrennt:

```
procedure ggT(a, b: integer; var GgT: integer);
```