Ma 14/27 St_Online-Ergänzung

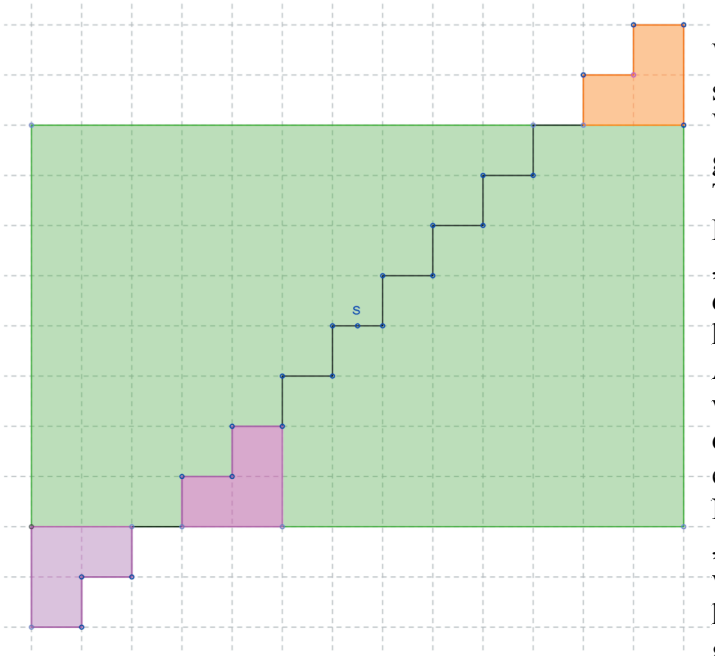
Zu einem Satz von J. J. Sylvester

Gerhard Klinger

Online-Ergänzung

Für das Rechnen „von Hand“ ist die Unterscheidung in die beiden Fälle vorzuziehen, doch für Programmierer ist das Verschwinden einer Fallunterscheidung angenehm, und es kann wirklich alles mit einem dieser Fälle berechnet werden, z.B. mit dem ersten Fall. Das soll im Folgenden erklärt werden.

Die Abbildung gehört zu $n = 52 = 4 \cdot 13$, daher ist $x = 4$, $u = 13$,
und es wird von $a = -6$ bis zu $a = 6$ mit $u = 13$ Summenden addiert.



Weil für dieses Rechteck $u > 2x$ ist, sollte der Fall B verwendet werden. Wenn trotzdem der Fall A genommen wird, dann darf die Treppe unten und oben über das Rechteck hinausreichen. Die unten „hängenden“ Stufen entsprechen dann negativen Summenden und können durch direkt benachbarte Anfangsstufen ausgeglichen werden. Diese wiederum werden durch die „übersteigenden“ Stufen ersetzt.
Natürlich kann man auch den „Unterbau“ gegen den „Überbau“ verrechnen, doch erscheint in konkreten Summationen der zuerst gewählte Weg einfacher.

Weil nur Stufen mit positiven Summanden verrechnet werden, hat die Treppe dann auch nur $2x$ Stufen.

Es folgt hier der „Kern“ eines kleinen Lazarus- bzw. Delphi-Programms:

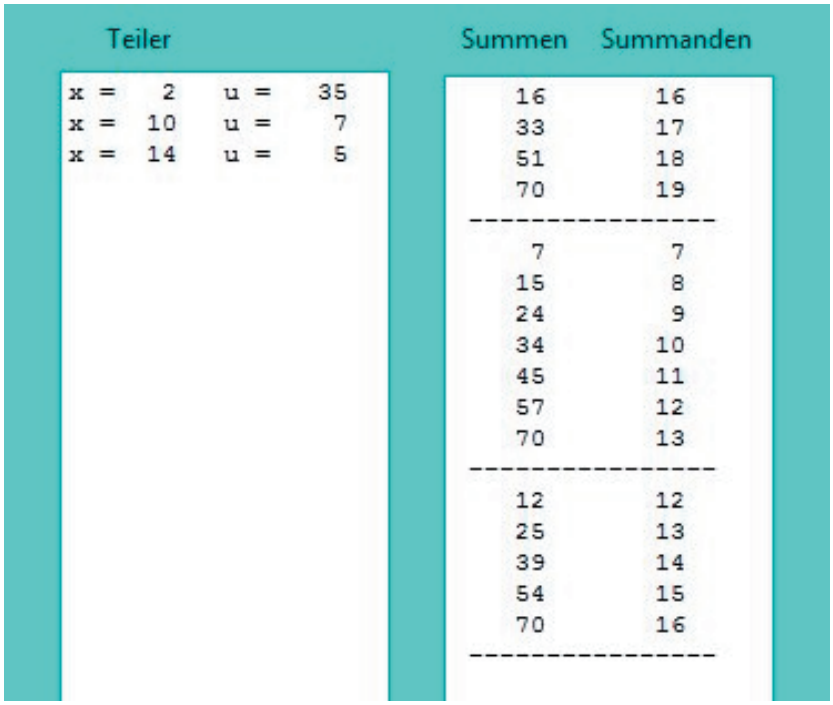
MNU Journal – Ausgabe 2.2016 – ISSN 0025-5866 – © Verlag Klaus Seeberger, Neuss

```
var
  Form1: Tform1;           // Kommentare hinter solchen Doppelstrichen //
  n,u,a,b,r,s,t,z: integer;  // Variable als Ganzzahlen

implementation
{$R *.lfm}
{ TForm1 }
procedure TForm1.Button1Click(Sender: TObject);
var i,j,k: integer;          // Variable als Ganzzahlen
begin
n:=StrToInt(Edit1.Text); ListBox2.Clear;  ListBox1.Clear;  // Eingabe von n
k:=0; z:=0; Label4.Caption:=''; Label5.Caption:='';        // Variable k und z auf 0 setzen
     for j:=1 to n div 2 do                                // j läuft von 1 bis (n-1)/2; Im 'Beweis' steht x statt j
      Begin if (n mod j = 0) and odd(n div j) then         // j teilt n und Quotient ist ungerade
          BEGIN u:=n div j; s:=0; a:=u div 2; b:=j-1; i:=-b; inc(z); // u ist ungerader Teiler, j ist Teiler von n
             // Summe s:=0; a:=(u-1)/2; u für die eine und 2*j für die andere Rechteckseite
             // daher ist mit j die Hälfte dieser Seite und mit b:=j-1 die reduzierte Hälfte gegeben
           repeat s:=s+a+i;  // nun wird s aufgefüllt, anfangs durch s:=0+a+(-b)
              if s>0 then begin ListBox2.Items.Add(FSt(IntToStr(s),6)+'    '+FSt(IntToStr(a+i),6)); inc(k); end;
              // Ausgabe nichtnegativer Summenwerte
              inc(i)  // Erhöhung der Variablen i um 1; daher s:=(a-b)+a+(-b+1)  usw.
           until i>b+1; // von i=-b bis i=0 sind es b+1 Summanden;
                  // von i=1 bis i=b+1 sind es b+1 Summanden, zusammen also 2*b+2 = 2*j Summanden
          ListBox2.Items.Add(' ================ ');
          ListBox1.Items.Add(FSt(IntToStr(j),5)+'   '+FSt(IntToStr(u),6));
          END;
      End;
          if z=0 then Label4.Caption:='Zu dieser Zahl gibt es keine Treppe';
          if(z>0)and(k<2)and(odd(n)=true) then Label5.Caption:='Das ist eine Primzahl';
          if z>1 then Label6.Caption:='Es gibt '+IntToStr(z)+' Lösungen' else
          if z=1 then Label6.Caption:='Es gibt 1 Lösung' else Label6.Caption:='Es gibt keine Lösung';
end;
end.
```

[Im Programmkern kann FSt(IntToStr(j),5)+' u = '+Fst(IntToStr(u),6) durch IntToStr(j)+' u = '+IntToStr(u) ersetzt werden, denn das FSt ist eine kleine Format-String-Anweisung, die FSGK.pas voraussetzt und die Ausgabe verschönert, aber sonst nicht verändert.
Ohne diese Änderung muss die Pascal-Unit FSGK.pas (s.u.) in dem Verzeichnis stehen, in dem auch das Treppenprogramm steht und im Programm selbst ist im Abschnitt **uses** hinter … Dialogs, Buttons, StdCtrls, der Eintrag **FSGK** einzufügen. (Sie müsste in Delphi bzw. Lazarus als 'neues Formular' eingefügt und gespeichert werden.)
Die Pascal-Unit FSGK.pas besteht nur aus den folgenden Zeilen:

```
unit FSGK;
  interface
  function FSt(s: string; k: integer): string;
implementation
  function Fst(s: string; k: integer): string;
    var i,r: integer; sv: string;
    begin
       r:=length(s);
       sv:='';  for i:=1 to k-r do sv:=sv+' ';
       FSt:=sv+s;
    end;
end.    ]
```



Ein Ausschnitt der Lazarus-Ausgabe als Beispiel:
Zum Fall A: $u < 2x$ Hier werden u Zahlen addiert.
Zum Fall B: $u > 2x$ Es werden $2x$ Zahlen addiert.
So ergeben sich für $n = 70 = 2 \cdot 35 = 10 \cdot 7 = 14 \cdot 5$ die Summen mit $2 \cdot 2 = 4$, 7 und 5 Summanden.

MNU Journal – Ausgabe 2.2016 – ISSN 0025-5866 – © Verlag Klaus Seeberger, Neuss