FRIEDRICH-SCHILLER-UNIVERSITÄT JENA

JENAER SCHRIFTEN

ZUR

MATHEMATIK UND INFORMATIK

Eingang: 12.01.2006 Math/Inf/01/06 Als Manuskript gedruckt

Mit Puck einfach Programmieren lernen

– Eine Einführung mit fünf Beispielen –

Lutz Kohl
Abteilung Didaktik
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena
D – 07743 Jena
Lutz.Kohl@uni-jena.de

Mit Puck einfach Programmieren lernen
– Eine Einführung mit fünf Beispielen –

„Ja, ja, ihr Jungen. Ihr wollt immer alles besser machen. Und wissen Sie was?
Ihr macht es auch besser. Aber ich bin für die neuen Methoden zu alt.“[1]

Puck ist ein visuelles System, mit dem es möglich ist, grundlegende Strukturen der Programmierung zu erlernen. Der Benutzer kann Bausteine zu Programmen verknüpfen. So können einfache Anweisungen, Kontrollstrukturen, das Variablenkonzept und schließlich Prozeduren mit Parametern schrittweise kennen gelernt werden. Da Syntaxfehler durch die visuelle Konstruktion nicht möglich sind, können die erstellten Programme stets direkt ausprobiert werden. Um das Arbeiten mit textuellen Sprachen vorzubereiten, ist es während des Programmierens jederzeit möglich, den Quelltext einer imperativen Programmiersprache anzusehen.

Für wen ist dieser Artikel gedacht?

„Welchen Leser ich wünsche? Den unbefangensten, der mich, /
Sich und die Welt vergisst und in dem Buche nur lebt.“[2]

Dieser Artikel wurde für zwei Zielgruppen geschrieben. Einerseits für den Lehrer, der bisher Erfahrungen mit einer textuellen imperativen Programmiersprache hat und der das Puck-System ausprobieren möchte. Er kennt Anweisungen, Kontrollstrukturen, Variablen- und Prozedurkonzept. Er weiß, wie dies alles anhand einfacher Beispiele vermittelt werden kann. Für diesen Leser ist es sicher interessant, wie die Programmieraufgaben, die sonst mit einem syntaktisch korrekten textuellen Programm gelöst werden, mit Puck visuell gelöst werden können. Es wurde nachfolgend Wert darauf gelegt, die Verknüpfung und Modifikation der Bausteine möglichst vollständig zu beschreiben. Somit ist es möglich, die Aufgaben und deren Lösungen in diesem Artikel nachzuvollziehen.
Die zweite Zielgruppe, die angesprochen werden soll, sind die Schüler. Ihnen soll der Artikel als Nachschlagewerk dienen, in dem die Bedeutung einzelner Begriffe sowie die Bedienung des Puck-Systems erklärt sind. Ein Artikel dieses Umfangs kann nicht als alleiniges Lehrbuch dienen und es kann auch keinen guten Unterricht ersetzen.
Nach einem kurzen Kapitel zur Installation werden fünf Beispiele beschrieben, die verschiedene Aspekte der imperativen Programmierung verdeutlichen. Am Ende folgt eine Übersicht über alle Bausteine, die im Puck-System zur Verfügung stehen.

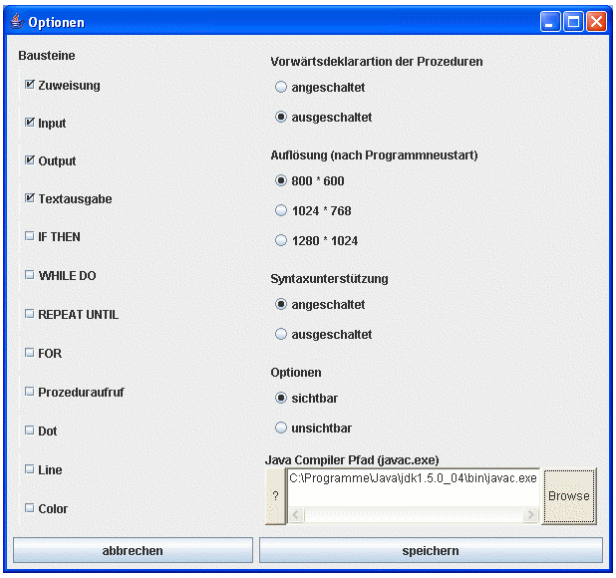
Puck installieren

„Wohl begonnen, ist halb gewonnen.“[3]

Um Puck nutzen zu können, muss ein *JDK* installiert sein (*JRE* reicht nicht aus)[4]. Dieser kann unter http://java.sun.com/j2se/1.5.0/download.jsp heruntergeladen werden. Im Folgenden wird von einer Standardinstallation unter einem Windows-System ausgegangen, bei dem ein *JDK* in dem Verzeichnis C:\Programme\Java\jdk1.5.0_04 installiert ist. Puck kann nun von der Internetseite www.uni-jena.de/puck.html heruntergeladen werden[5].

[1] Lehrer Bömmel in dem Film: Die Feuerzangenbowle, Deutschland, 1943.
[2] Aus dem Gedicht „Der berufene Leser“ von Friedrich Schiller [Sc 04].
[3] Sprichwörtlich bei vielen Völkern.
[4] Mit dem JDK (Java Development Kit) ist es möglich Java-Programme zu kompilieren und auszuführen
[5] Dieser Artikel bezieht sich auf Puck in der Version 1.8.1.

Das Puck-System befindet sich in einer ZIP-Datei. Diese muss mit einem Programm, wie z.B. WINZIP, an einen beliebigen Ort auf dem Computer entpackt werden.
Als Erstes sollte die Datei *SetuPuck.jar* aus dem Puck-Ordner mit einem Doppelklick gestartet werden. Es öffnet sich ein Fenster, in dem auf der linken Seite eingestellt werden kann, welche Bausteine bei der visuellen Programmierung zur Verfügung stehen sollen. Für das erste Programm muss hier nur ein Häkchen neben dem Baustein Line gesetzt werden. Die vier vorhandenen Häkchen können zunächst entfernt werden. Im Installationsverzeichnis des JDK befindet sich ein Unterordner *bin*, in dem die Datei *javac.exe* enthalten ist. In der rechten Hälfte des Fensters muss mit dem Browse-Button diese Datei ausgewählt werden, damit die entwickelten visuellen Programme auch direkt kompiliert und ausgeführt werden können. Um die Optionen direkt beim Programmieren ändern zu können, sollte *Optionen - sichtbar* ausgewählt sein[6]. Durch einen Klick auf den *speichern*-Button schließt sich das Fenster und die eingestellten Optionen werden in die Datei *PuckOptions.opt* geschrieben, auf die dann auch das Puck-System zugreift.



Auf den folgenden Seiten werden fünf verschiedene Beispiele zur Arbeit mit Puck vorgestellt. Die in den Programmen benötigten Bausteine müssen vor jeder Aufgabe ausgewählt werden. Die Optionen können im Programm über die Menüleiste *?* - *Optionen* eingestellt werden. Am Ende des Artikels findet sich eine Übersicht, in der alle Bausteine erläutert sind.

Wie werden Aufgaben in der Informatik gelöst?

„Wenn ein Wesen zur Reflexion unfähig ist,
lässt es sich ausschließlich von Zufall und Notwendigkeit beherrschen.“[7]

Beim Lösen einer Aufgabe in der Informatik sollten zumindest die drei Phasen **Entwurf, Implementierung** und **Reflexion** unterschieden werden[8].
Beim Entwurf wird das gegebene Problem analysiert, in Teilprobleme zerlegt und vollständig durchdacht. Es ist oft sinnvoll, mit einer Skizze oder einer Lösungsstrategie auf dem Papier zu beginnen. Nachdem die verwendeten Datenstrukturen und spezielle Unterprogramme festgelegt wurden, sollten die wesentlichen Teile des Algorithmus angegeben werden.
Die Implementierungsphase schließt neben der Umsetzung des Entwurfes in ein Computerprogramm auch dessen Test mit ein. Treten beim Testen Probleme auf, kann es zweckmäßig sein zu Papier und Bleistift zurückzukehren.
Funktioniert das Programm, sollte in der Reflexionsphase überprüft werden, ob das Ausgangsproblem wirklich vollständig gelöst wurde. Qualität und Einsatzmöglichkeiten sollten kritisch hinterfragt und die beim Lösen der Aufgabe gewonnenen Erfahrungen sollten aufgearbeitet werden.

---

[6] Das Verstecken der Optionen ist für den Einsatz im Unterricht vorgesehen. Der Lehrer kann vorgeben, welche Bausteine verwendet werden sollen, und die Optionen unsichtbar schalten. Damit kann der Schüler nichts an den Einstellungen des Programms ändern.
[7] Zitat nach M. Czikzentmihalyi.
[8] Die Aufteilung der Programmentwicklung in drei Phasen wurde aus [Fo 02] übernommen.

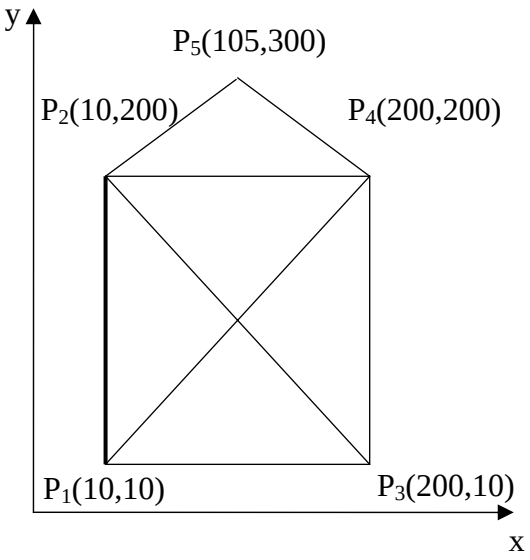
Das Haus vom Nikolaus

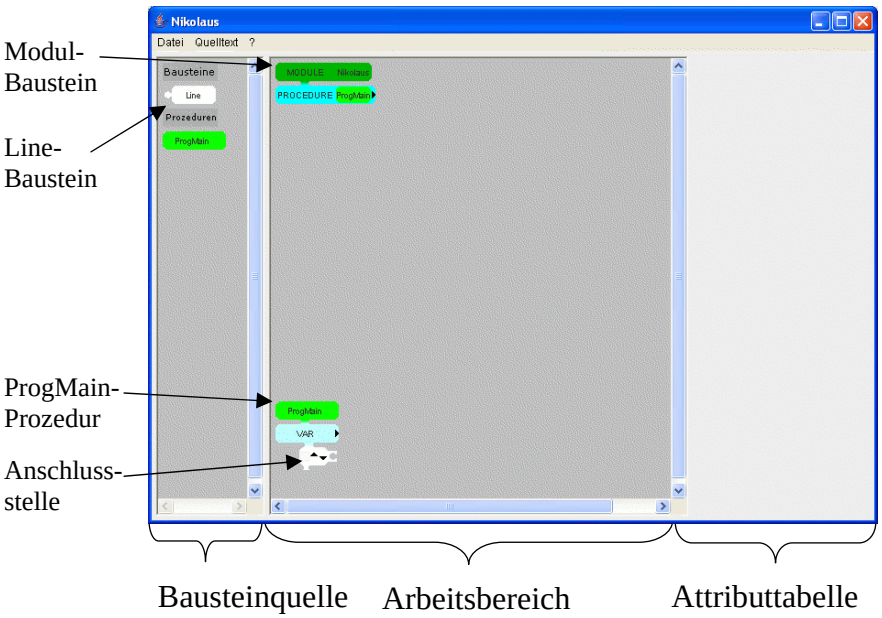
„Computer-Programmierung ist die Kunst, dass ein Computer das macht, was du willst.“[9]

Es ist für einen Menschen einfach, mit acht Linien das Haus des Nikolaus zu zeichnen. Diese Aufgabe kann auch ein Computer lösen. Wie, das soll in diesem Abschnitt gezeigt werden.

Im Puck-System ist es mit Hilfe des Line-Bausteins möglich Linien zu zeichnen. Eine Linie wird durch ihren Anfangspunkt und ihren Endpunkt charakterisiert. Der Computer muss also den Befehl bekommen, die acht in der Abbildung rechts dargestellten Linien zwischen den Punkten zu zeichnen.

Um Puck zu starten, muss die Datei *Puck.jar* mit einem Doppelklick geöffnet werden. Nach dem Start wird der Benutzer aufgefordert, einen Namen für das zu erstellende Modul einzugeben. Hier bietet sich für das erste Programm der Name *Nikolaus* an.





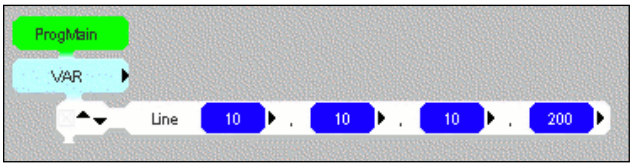
Im Arbeitsbereich der Oberfläche befindet sich nach dem Bestätigen mit *OK* schon ein dunkelgrüner Modul-Baustein, in den automatisch der Name Nikolaus eingetragen wurde. Außerdem ist bereits eine Prozedur mit dem Namen ProgMain erstellt, an deren weißer Anschlussstelle Bausteine angehängt werden können. Wenn das Programm ausgeführt wird, werden vom Computer die Anweisungen abgearbeitet, die mit den Anschlussstellen der Prozedur ProgMain verbunden sind.

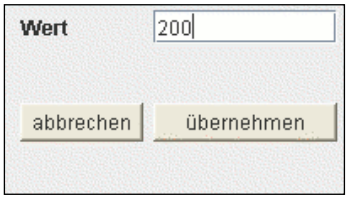
Um den Befehl zum Zeichnen der ersten Linie zu erzeugen, muss der Benutzer nun also den Line-Baustein aus der Bausteinquelle mit gedrückter linker Maustaste an eine freie Stelle des Arbeitsbereiches ziehen und die linke Maustaste dort loslassen[10]. Anschließend muss der Baustein, der inzwischen um vier blaue Felder erweitert wurde, per Drag & Drop mit der Anschlussstelle der Prozedur ProgMain verbunden werden. Das Modul, die Prozeduren und die Bausteine können im Arbeitsbereich verschoben werden.

[9] Aus: Programmieren lernen von Alan Gauld [www1].

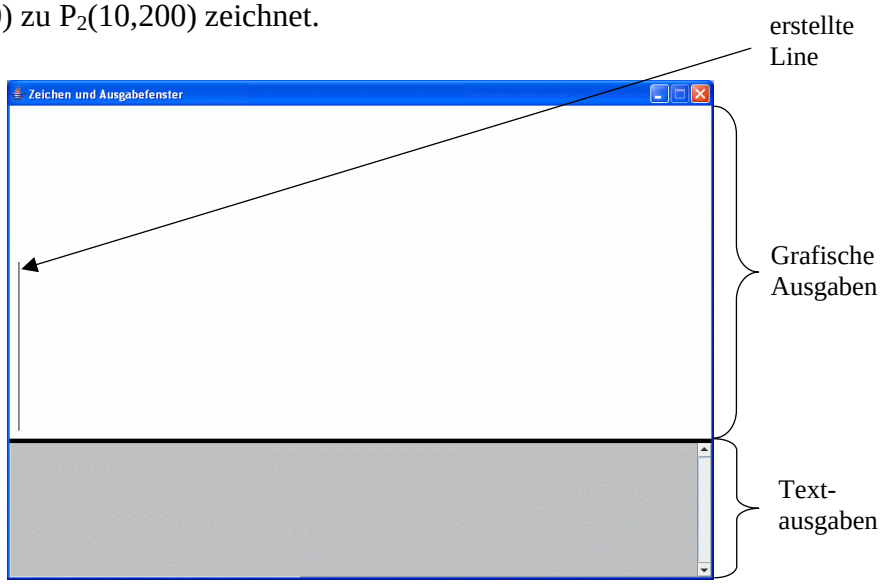
[10] Der Vorgang des Verschiebens eines Bausteins bei gedrückter linker Maustaste wird im Folgenden Drag & Drop genannt.

In den blauen Feldern können nun jeweils die x- und die y-Koordinate zweier Punkte angegeben werden. Dies geschieht, indem der Benutzer mit der linken Maustaste auf ein solches blaues Feld klickt. Daraufhin öffnet sich in der Attributtabelle ein Eingabefeld, in dem eine neue Zahl eingetragen werden kann. Wenn der neue Wert mit der *Enter-Taste* oder einem Klick auf den *übernehmen-Button* bestätigt wurde, so erscheint die eingegebene Zahl in dem blauen Feld. Um die in der Vorüberlegung dick gezeichnete Linie zu erhalten, müssen x_1, y_1, x_2 und y_2 mit Werten belegt werden. Die Abbildung rechts oben zeigt ein Programm, das eine Linie von $P_1(10,10)$ zu $P_2(10,200)$ zeichnet.

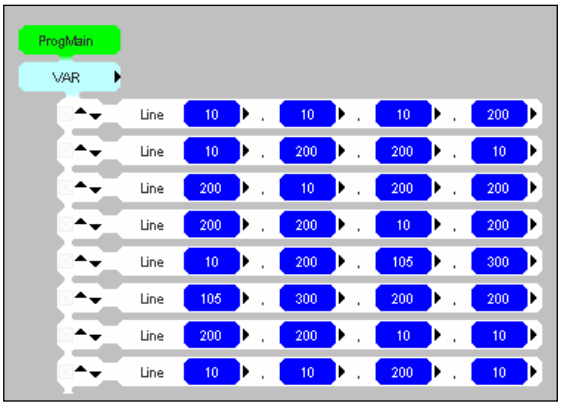




Soll dieses Programm zum Testen ausgeführt werden, so kann dies durch die Tastenkombination *STRG + E* oder über die Menüleiste *Quelltext - Programm ausführen* geschehen. Der Computer übersetzt nun das Programm in eine für ihn verständliche Form und führt es anschließend direkt aus. Es öffnet sich ein Ausgabefenster, wie in der Abbildung rechts, in das die gewünschte Linie gezeichnet wurde. Das Fenster kann nach dem Test geschlossen werden.



Um das entwickelte Programm zu vervollständigen, müssen noch weitere sieben Linien gezeichnet werden. Dafür ist es nötig, zu der Prozedur *ProgMain* noch einige Anschlussstellen hinzuzufügen. Dies geschieht, indem eines der kleinen schwarzen Dreiecke in der Anschlussstelle mit der linken Maustaste angeklickt wird. Dadurch entsteht - je nach Pfeilrichtung - eine weitere Anschlussstelle oberhalb oder unterhalb der bereits vorhandenen.



Nun müssen nur noch die weiteren Line-Bausteine mit den Anschlussstellen verbunden und mit den richtigen Werten versehen werden. Das komplette Programm ist in der Abbildung links unten dargestellt. Es kann - wie oben beschrieben - ausgeführt werden. Mit der Tastenkombination *Strg + S* oder dem Menüpunkt *Datei - Speichern* ist es möglich das Programm zu speichern. So kann es beim nächsten Starten von Puck durch *Strg + O* oder den Menüpunkt *Datei - Öffnen* wiederhergestellt werden.

Es ist möglich, zum entwickelten Programm den Quelltext der Programmiersprache Oberon-2 anzeigen zu lassen (*Quelltext anzeigen*). Dieser kann mit der Entwicklungsumgebung *POW!* ausgeführt werden. Bevor mit der nächsten Aufgabe begonnen werden kann, müssen die Bausteine Input, Output und Zuweisung in den Optionen *(? - Optionen)* aktiviert werden.

Einmal Quadrieren bitte!

„Woher kommt es, dass mich niemand versteht und jeder mag? Dein Computer“[11]

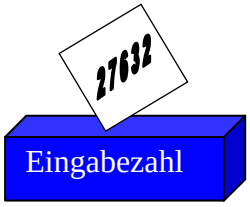
Das Ausführen unseres ersten Nikolaus-Programms ergibt immer genau die gleiche Ausgabe. Wenn dem Benutzer nun eine Möglichkeit zur Interaktion gegeben wird, so kann ein Programm auch auf verschiedene Eingaben des Benutzers unterschiedlich reagieren.

> Das EVA-Prinzip beschreibt die Arbeit eines Computerprogramms:
> Eingabe → Verarbeitung → Ausgabe

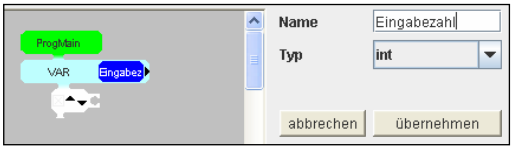
In diesem Abschnitt soll ein neues Programm entwickelt werden, das zu einer einzugebenden Zahl die entsprechende Quadratzahl berechnet. Dafür ist es nötig, die Eingaben des Benutzers zu speichern: Der Computer nutzt hierfür Variablen.

> Man kann sich Variablen wie einen Behälter mit einem Namen vorstellen, in dem Daten aufbewahrt werden. Es können bestehende Daten aus den Behältern *gelesen* und neue Daten in die Behälter *geschrieben* werden. Je nach Datentyp können in einem Behälter Zahlen, Zeichen, Zeichenketten, Wahrheitswerte oder komplexere Strukturen aufbewahrt werden. In Puck kann der Datentyp eines Behälters an seiner Farbe erkannt werden.

In dem gewählten Beispiel ist es nötig, ganze Zahlen zu speichern. Der erforderliche Datentyp heißt Integer und ist in Puck mit einem blauen Behälter gekennzeichnet. Außerdem gibt es in Puck noch graue Variablen. Diese können nur die Werte wahr oder falsch annehmen. Der entsprechende Datentyp heißt Boolean.



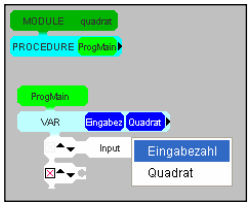
Variablen werden in Puck in dem hellblauen VAR-Baustein unterhalb der Prozedur erstellt. Hierfür muss das kleine Dreieck auf der rechten Seite angeklickt werden. Dadurch entsteht eine erste Integer-Variable. Diese hat den Namen „a“. Durch einen Klick auf den blauen Behälter werden dessen Name und Datentyp in der Attributtabelle angezeigt. Der Name kann hier zum Beispiel in Eingabezahl geändert werden. Nach dem Übernehmen der Änderung ist der blaue Behälter in der Prozedur nun auch mit dem neuen Namen gekennzeichnet. Anschließend sollte noch die Variable Quadrat angelegt werden, in der später das Ergebnis gespeichert werden kann.



> Immer wenn im Puck-System ein Feld in einem Baustein rot markiert ist, fehlt an der entsprechenden Stelle etwas und die Anweisung wird nicht abgearbeitet.

Um den Benutzer zu einer Eingabe aufzufordern, gibt es in Puck den Input-Baustein.

> Bei der Abarbeitung des Input-Bausteins wird der Benutzer mit einem übergebenen Text aufgefordert eine Zahl einzugeben, die nach der Bestätigung in einer ausgewählten Variable gespeichert wird.

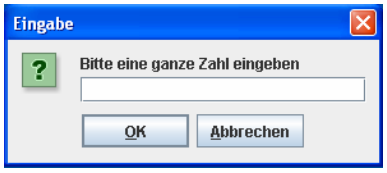


Der Input-Baustein enthält ein rotes Feld, welches signalisiert, dass er nicht weiß, wohin er die vom Benutzer eingegebene Zahl speichern soll. Wenn nun der Input-Baustein mit einer Anschlussstelle der Prozedur ProgMain verbunden wurde, kann durch einen Klick mit der rechten Maustaste anstelle des roten Feldes die Variable Eingabezahl (einer der gerade angelegten Behälter) ausgesucht werden.

[11] Abwandlung eines Einsteinzitates in einem Werbefilm für das Informatikjahr 2006, vgl. [www2].

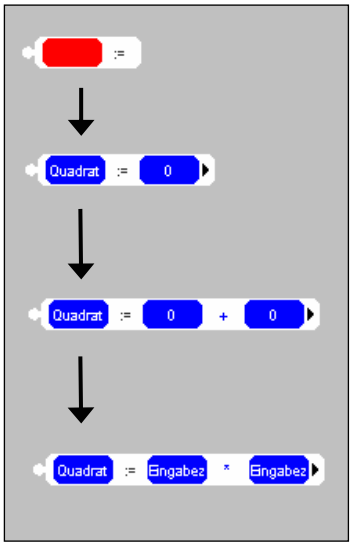
Der Eingabeaufforderung sollte noch eine Beschreibung, wie zum Beispiel *Bitte eine ganze Zahl eingeben* hinzugefügt werden. Dies kann nach einem Klick auf das Wort *Input* in der Attributtabelle erfolgen.

Wenn das so erstellte Programm nun ausgeführt wird, erscheint auf dem Bildschirm eine Eingabeaufforderung mit der entsprechenden Meldung. Der Benutzer kann nun eine Zahl eingeben, die – nachdem er die Eingabe mit *OK* bestätigt hat – in die Variable Eingabezahl gespeichert wird.



Jetzt soll der Wert der Variablen Eingabezahl quadriert werden und das Ergebnis soll in die Variable Quadrat gespeichert werden. Dafür kann der Zuweisungsbaustein genutzt werden.

> Bei einer Zuweisung wird in der Variablen auf der linken Seite der Wert des Ausdrucks auf der rechten Seite gespeichert.



Nachdem der in der Abbildung links dargestellte Zuweisungsbaustein mit einer Anschlussstelle von ProgMain verbunden wurde, kann mit der rechten Maustaste anstatt des roten Feldes die zuvor angelegte Variable Quadrat ausgewählt werden.

Dieser Variablen wird dann auf der rechten Seite ein Ausdruck des entsprechenden Datentyps zugewiesen.

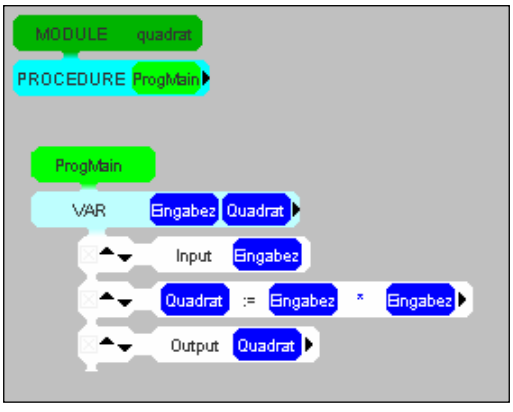
Durch das Dreieck auf der rechten Seite kann der Ausdruck um einen Operator und einen Operanden erweitert werden.

Nun ist es möglich, durch die rechte Maustaste für die Operanden die Variable Eingabezahl und für den Operator * auszuwählen.

Nach dieser Berechnung ist nun das gewünschte Ergebnis in der Variablen Quadrat gespeichert.

> Mit dem Output-Baustein kann der Wert eines Integer-Ausdrucks auf dem Bildschirm ausgegeben werden.

Um dem Benutzer des Programms das errechnete Ergebnis auf dem Bildschirm auszugeben, muss noch ein Output-Baustein mit einer Anschlussstelle der Prozedur ProgMain verbunden werden. Nun ist es möglich, anstatt der vorgegebenen 0 die Variable Quadrat auszuwählen. Anschließend kann mit einem Klick auf das Wort *Output* in der Attributtabelle noch eine Beschreibung des Ergebnisses übergeben werden, die vor der Zahl ausgegeben wird.

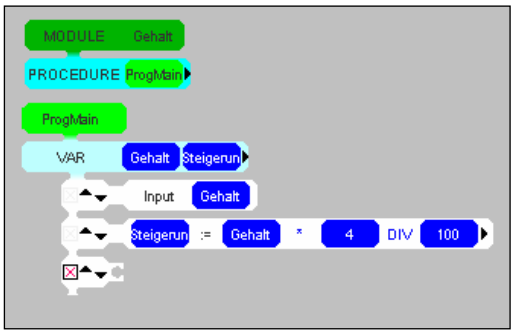


Gehaltserhöhung!

„Nicht die Aufgaben sollen einem über den Kopf wachsen,
sondern der Kopf soll über den Aufgaben wachsen.“[12]

Aufgabe: Eine IT-Firma konnte ihre Effizienz durch den Umstieg auf eine visuelle Programmiersprache steigern. Der Chef möchte deshalb seine Mitarbeiter belohnen. Die Gehälter aller Mitarbeiter sollen um 4 %, mindestens aber um 80 € im Monat erhöht werden. Es soll ein Programm entwickelt werden, das nach Eingabe des alten Monatsgehaltes das neue Gehalt berechnet und ausgibt.

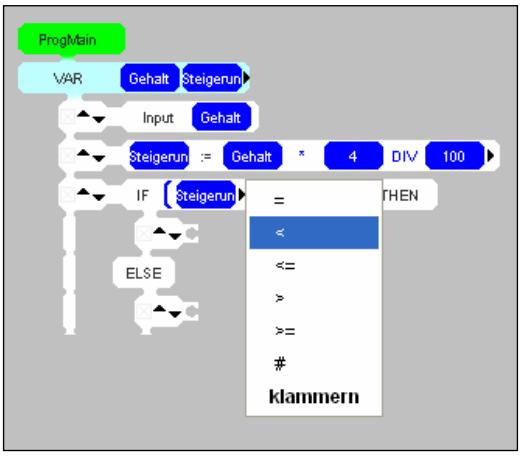
Um diese Aufgabe zu lösen, muss als Erstes das aktuelle Gehalt eingelesen und in eine Variable gespeichert werden. Dafür wird eine Variable Gehalt angelegt. Ein Input-Baustein wird mit der Prozedur ProgMain verbunden und an Stelle des roten Feldes wird Gehalt ausgewählt. Ein Text sollte bei keiner Eingabeaufforderung fehlen, damit der Nutzer weiß, was einzugeben ist. Hierfür kann nach einem Klick auf *Input* in der Attributtabelle *Ihr aktuelles Gehalt:* eingegeben werden.



Um den Wert einer Gehaltserhöhung um 4 % zu berechnen, wird eine weitere Variable Steigerung angelegt. Mit einem Zuweisungsbaustein wird in der Variablen Steigerung der Wert des Ausdrucks Gehalt * 4 DIV 100 gespeichert. Somit ist nun eine Steigerung des Gehaltes um 4 % bekannt. Anschließend muss aber, je nachdem wie hoch der Wert in der Variablen Steigerung ist, das neue Gehalt unterschiedlich berechnet werden. Dafür kann der IF-THEN-Baustein genutzt werden.

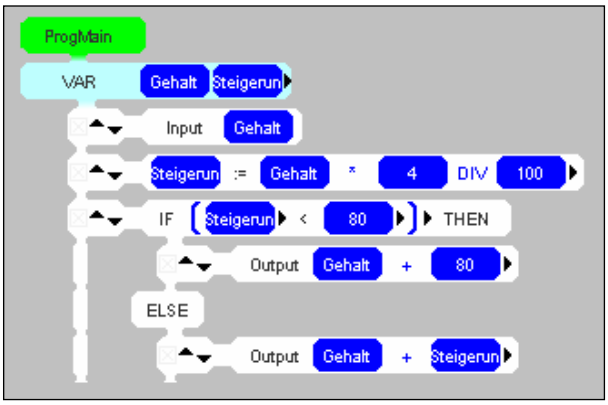
Beim IF-THEN-Baustein wird eine Bedingung überprüft. Ist die Bedingung erfüllt, so werden die Anweisungen des THEN-Zweiges ausgeführt. Ist die Bedingung nicht erfüllt, so werden die Anweisungen des ELSE-Zweiges ausgeführt.

Die Bedingung in einem IF-THEN-Baustein besteht aus einem Boolean-Ausdruck. Bei unserem Beispiel muss ein Wahrheitswert durch den Vergleich zweier Integer-Ausdrücke ermittelt werden. Dafür wird nach einem Rechtsklick auf den Wahrheitswert FALSE aus dem Kontextmenü *Vergleich* ausgewählt. Dadurch entstehen zwei Integer-Ausdrücke, die mit dem Gleichheitszeichen verglichen werden. Für den ersten Integerausdruck kann nun die Variable Steigerung ausgewählt werden. Wie in der Abbildung zu sehen ist, kann das „=“ mit einem Rechsklick durch ein „<“ ersetzt werden. Anschließend muss auf der rechten Seite des Vergleichs noch die Zahl 80 in der Attributtabelle eingegeben werden.



[12] Zitat nach Gerhard Uhlenbruck.

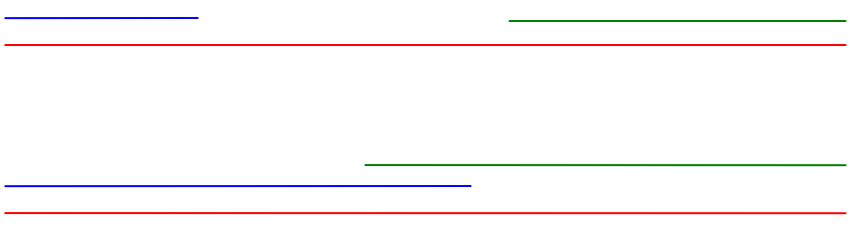
In den beiden entstandenen Zweigen des IF-THEN-Bausteins muss nun je eine Ausgabe eingefügt werden. Zwei Output-Bausteine werden mit den entsprechenden Anschlussstellen verbunden und bekommen nach einem Linksklick in der Attributtabelle die Beschreibung *Neues Gehalt:* übergeben. Im oberen THEN-Zweig wird Gehalt + 80 im unteren ELSE-Zweig Gehalt + Steigerung ausgegeben. Somit wird also überprüft, ob die Steigerung um 4 % kleiner als 80 € ist. Wenn dies der Fall ist, so wird das alte Gehalt um 80 € erhöht. Ansonsten wird das alte Gehalt um die berechnete Steigerung erhöht.



Bei Boolean-Ausdrücken wird mit Wahrheitswerten operiert. Ein Wahrheitswert kann aus einer Boolean-Variablen, einem Vergleich zweier Integer-Ausdrücke oder aus den Konstanten TRUE und FALSE ermittelt werden. Boolean-Operanden können mit dem Und-Operator & und dem Oder-Operator OR verknüpft werden. Außerdem kann mit dem Negator ~ das Gegenteil des Wahrheitswertes eines Boolean-Operanden ermittelt werden.

Aufgabe1:
Ein Großbauer möchte sein Land erweitern. Dafür bietet er den Bauern im Dorf an, ihre Felder zu pachten. Da er größere Felder mit seinen modernen Geräten besser bewirtschaften kann, ist er bereit, dafür einen höheren Preis zu zahlen. Für Felder über 10 Hektar Größe will der Großbauer 150 € Pacht pro Hektar im Jahr bezahlen. Bei allen anderen Feldern nur 130 € pro Hektar.
Entwickeln Sie für die Bauern des Dorfes ein Programm, welches zu der Eingabe einer Fläche in Hektar und der geplanten Pachtdauer in Jahren den Erlös ausgibt.

Aufgabe 2:
Nach der Eingabe der Längen von drei Strecken soll von einem Programm überprüft werden, ob aus diesen Strecken ein Dreieck konstruiert werden kann. Dafür muss die Summe der Längen von jeweils zwei Strecken größer sein als die Länge der dritte Strecke.



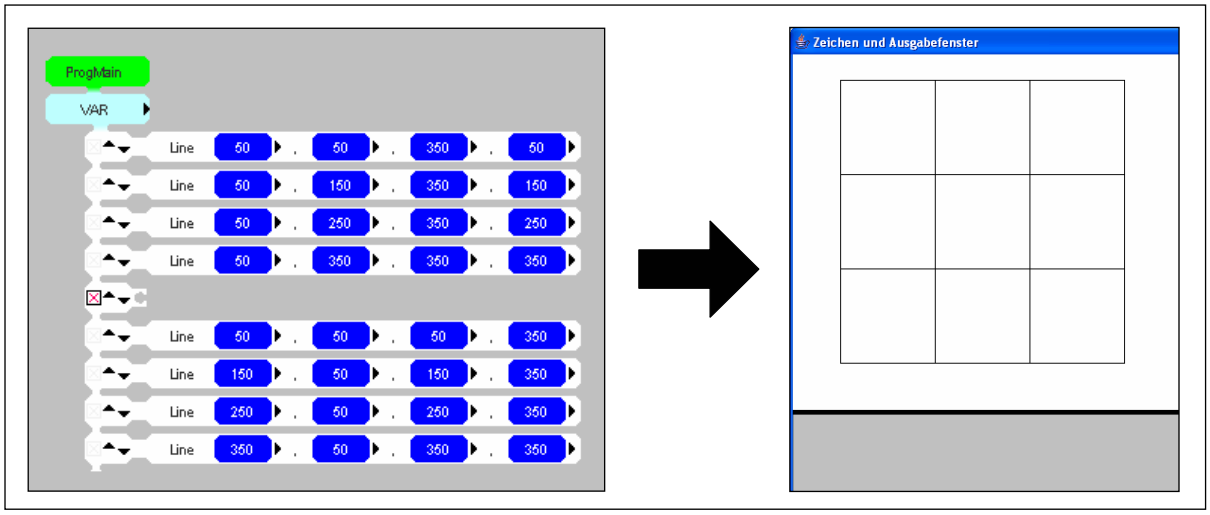
Dreieck nicht möglich

Dreieck möglich

Hinter Gittern

„Zufälligerweise ist Programmieren eine Kunst, die relativ leicht zu erlernen ist.“ [13]

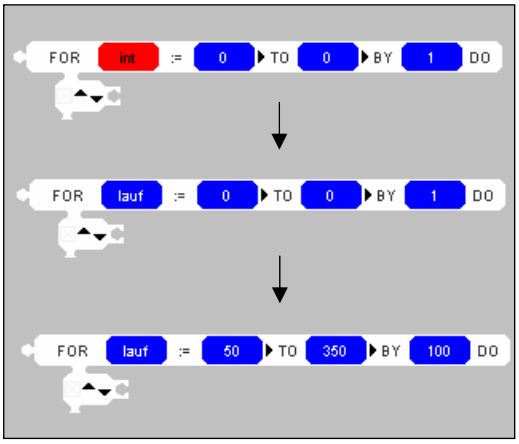
Im Ausgabefenster soll mit Hilfe von Line-Bausteinen ein Gitter erzeugt werden. Eine mögliche Lösung ist es, jeden Gitterstab mit einem Line-Befehl zeichnen zu lassen. Während diese Lösung bei einem Gitter mit 8 Stäben noch möglich erscheint, so ist sie doch bei einem Gitter mit 20 Stäben schon recht aufwändig.



Beim Betrachten der ersten vier Line-Bausteine fällt auf, dass bei den waagerechten Linien die x-Koordinaten immer die Werte 50 und 350 haben und die y-Koordinaten bei jeder Linie um 100 erhöht werden. Bei den senkrechten Linien ist es ähnlich. Das Zeichnen der Linien kann mit einem FOR-Baustein automatisiert werden.

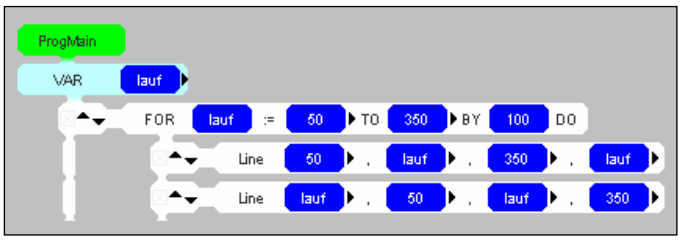
> Bei einem FOR-Baustein wird der Wert einer Integer-Variablen (die auch als Laufvariable bezeichnet wird) von einem Startwert bis zu einem Endwert um eine bestimmte Schrittweite erhöht. Die mit den Anschlussstellen verbundenen Bausteine werden in jedem Schritt abgearbeitet. Die wiederholte Abarbeitung einer Anweisungsfolge in einem Baustein wird als Schleife bezeichnet. Auf den aktuellen Wert der Laufvariable kann in den einzelnen Anweisungen zugegriffen werden.

Nachdem der FOR-Baustein per Drag & Drop in den Arbeitsbereich gezogen wurde, muss er mit einer Anschlussstelle der Prozedur ProgMain verbunden werden. Der Baustein hat nun ein rotes Feld mit der Beschriftung int, das signalisieren soll, dass an dieser Stelle eine Integer-Variable eingesetzt werden muss. Nachdem eine solche Variable mit dem Namen lauf vereinbart wurde, kann diese mit einem Klick der rechten Maustaste auf int ausgewählt werden. In den beiden folgenden blauen Integer-Ausdrücken des FOR-Bausteins können Start- und Endwert nach einem Linksklick in der Attributtabelle verändert werden. Auf die gleiche Art und Weise erfolgt auch das Festlegen der Schrittweite im letzten Feld.



Um bei jedem Durchlauf der Schleife eine

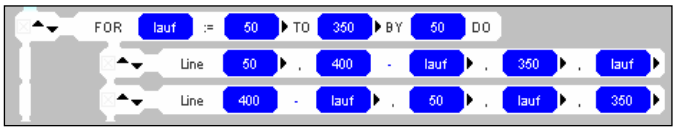
[13] Nach Joseph Weizenbaum [We 76].

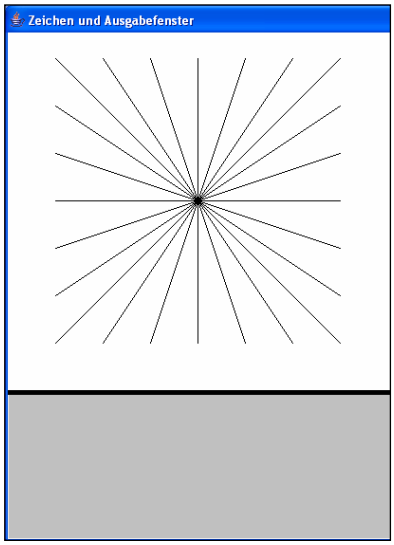


waagerechte und eine senkrechte Linie zu zeichnen, müssen in den Anschlussstellen des FOR-Bausteins noch zwei Line-Bausteine eingefügt werden.

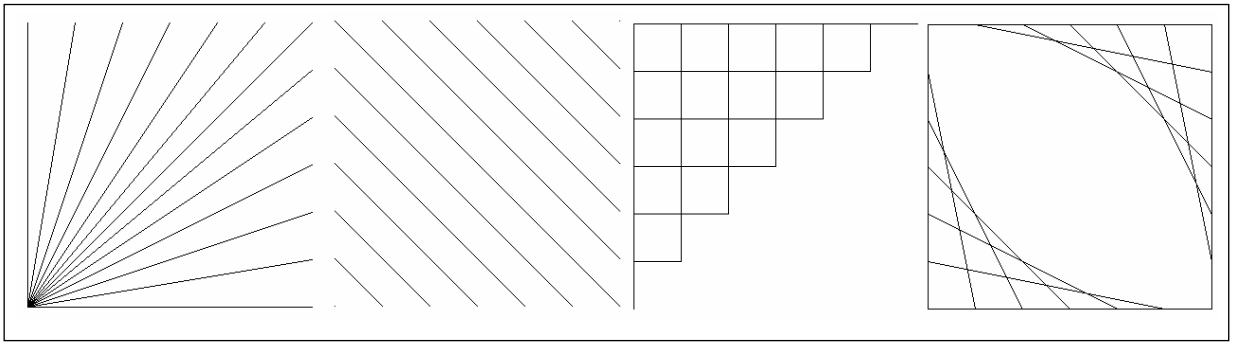
Beim Zeichnen der waagerechten Linien sind die Werte der x-Koordinaten immer gleich. 50 und 350 können nach einem Klick mit der linken Maustaste in der Attributtabelle eingegeben werden. Die y-Koordinaten ändern sich wie der Wert in der Variable lauf, die nach einem Rechtsklick aus dem Kontextmenü ausgewählt werden kann. Das Erstellen der senkrechten Linien erfolgt genauso, nur dass hier die y-Koordinaten die festen Werte bekommen und die x-Koordinaten den Wert von lauf, der bei jeder Abarbeitung der Anweisungsfolge um 100 erhöht wird. Das Programm besteht jetzt aus drei Bausteinen und liefert beim Ausführen dasselbe Ergebnis wie das Programm mit acht Bausteinen. Wenn nun die Anzahl der Gitterstäbe verdoppelt werden soll, so müssten zum ersten Programm acht weitere Line-Bausteine hinzugefügt werden. Stattdessen kann im zweiten Programm einfach die Schrittweite von 100 auf 50 geändert werden. Durch die Halbierung der Schrittweite muss die Variable doppelt so oft erhöht werden, um den Endwert zu erreichen. Damit werden auch die Anweisungen innerhalb des FOR-Bausteins doppelt so oft ausgeführt und so werden auch doppelt so viele Linien gezeichnet.

Was wird passieren wenn in den beiden Line-Bausteinen jeweils einmal lauf durch 400 - lauf ersetzt wird? Die Antwort kann durch das Anfertigen einer Skizze, oder durch die Modifikation des aktuellen Programms gegeben werden. Nach einem Rechtsklick auf die Variable lauf in einem der Line-Bausteine erscheint ein Kontextmenü. In diesem kann < *einfügen* ausgewählt werden. Dadurch werden links von der Variable lauf ein neuer Operand und ein neuer Operator eingefügt. Das + kann nach einem Rechtsklick in ein – und die 0 in der Attributtabelle in eine 400 geändert werden.





Aufgabe 3:
Programmieren Sie folgende Fadengrafiken.



Informatikanien

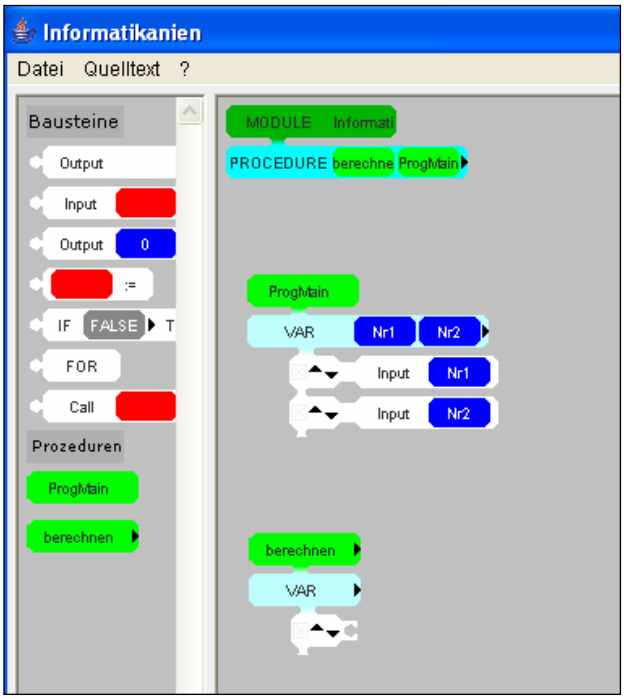
„Es ist die wichtigste Kunst des Lehrers, die Freude am Schaffen und am Erkennen zu erwecken.“[14]

Aufgabe: Der Planet Informatikanien hat 10.000 Einwohner. Die Bewohner dieses Planeten haben keine Namen, sondern eine Nummer zwischen 1 und 10.000. Je kleiner die Nummer eines Bewohners ist, desto unwichtiger ist er. Der Reichtum eines Informatikaniers wird nicht in Geld gemessen, sondern er ergibt sich aus der Summe der Nummern der unterrichteten Schüler. Jeder Bewohner kann sowohl unterrichten als auch unterrichtet werden. Ein Meister unterrichtet einen Schüler genau dann, wenn der Schüler unwichtiger ist als der Meister, und die Nummer des Meisters ohne Rest durch die Nummer des Schülers geteilt werden kann. Wenn sich zwei Informatikanier treffen, streiten sie häufig, wer der reichere ist. Schreiben Sie ein Programm, das zwei Informatikaniern bei Eingabe ihrer Nummern ausgibt, wer wie reich ist.
Ein Beispiel: Die Informatikanier 25 und 24 streiten, wer reicher ist:
24 unterrichtet 1,2,3,4,6,8 und 12, sein Reichtum ist also 1+2+3+4+6+8+12=36.
25 unterrichtet 1 und 5, sein Reichtum ist also 1+5=6. 24 ist also reicher als 25.

Um die Nummern der streitenden Informatikanier zu speichern, müssen zuerst zwei Variablen Nr1 und Nr2 angelegt werden. Anschließend werden zwei Input-Bausteine mit der Prozedur ProgMain verbunden und mit der Beschreibung *Geben Sie bitte Ihre Nummer ein* versehen. Nun muss zuerst der Reichtum von Nr1 und dann der Reichtum von Nr2 berechnet werden. Da es zu aufwändig wäre, die Berechnung des Reichtums zweimal zu programmieren, soll eine Prozedur berechnen entwickelt werden, die dann zweimal aufgerufen wird.

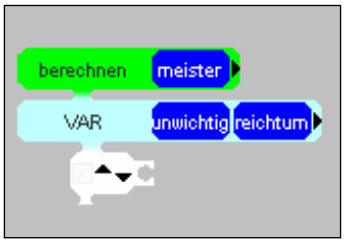
Eine Prozedur ist ein Unterprogramm. Immer, wenn mit einem Prozeduraufruf-Baustein eine Prozedur aufgerufen wird, werden die Anweisungen dieser Prozedur abgearbeitet.

Prozeduren werden in Puck im Arbeitsbereich direkt unterhalb von MODULE erzeugt. Neben dem Wort PROCEDURE befindet sich schon die Prozedur ProgMain, die bei jedem Start des Programms aufgerufen wird. Wenn das schwarze Dreieck rechts neben dieser Standardprozedur mit der linken Maustaste gedrückt wird, so entsteht links von ProgMain eine weitere Prozedur mit dem Namen Proc1. Wenn diese neue Prozedur mit der linken Maustaste angeklickt wird, so kann deren Name in der Attributtabelle in berechnen geändert werden. Die neue Prozedur befindet sich nun auch in der Bausteinquelle auf der linken Seite des Puck Systems. Sie kann per Drag & Drop genau wie die anderen Bausteine in den Arbeitsbereich gezogen werden.
Auf die Variablen aus ProgMain kann in einer anderen Prozedur nicht zugegriffen werden. Der neuen Prozedur berechnen muss deshalb die Nummer des Informatikaniers, dessen Reichtum ermittelt werden soll, übergeben werden. Dies kann mit Parametern geschehen.

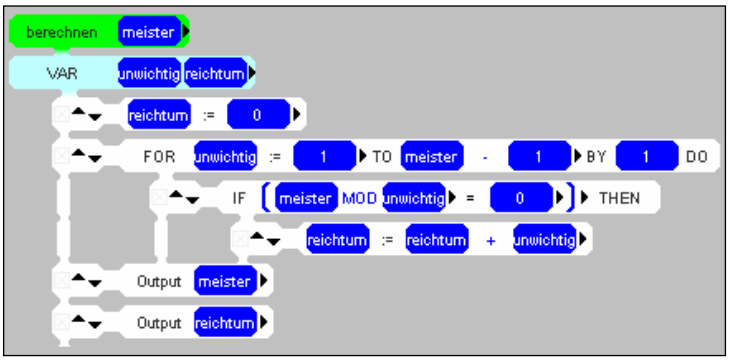


[14] Aus einer Ansprache Albert Einsteins an Kinder [Ei 05].

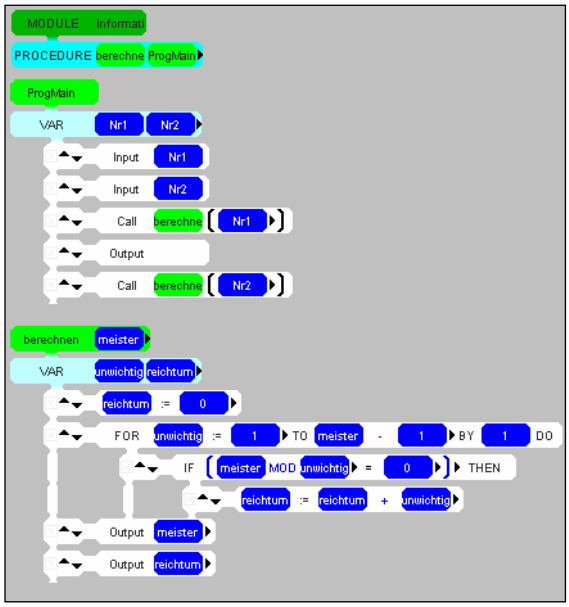
Mit Parametern ist es möglich, den Ablauf einer Prozedur zu beeinflussen.
Ein Wertparameter verhält sich innerhalb einer Prozedur wie eine Variable. Beim Aufruf einer Prozedur mit einem Wertparameter kann im Prozeduraufruf-Baustein ein Ausdruck des entsprechenden Datentyps als Startwert für diese Variable übergeben werden.



Parameter werden angelegt, indem das kleine schwarze Dreieck neben dem Namen der neuen Prozedur berechnen angeklickt wird. Der Name kann wie bei Variablen in der Attributtabelle verändert werden. Da zu der übergebenen Nummer eines Informatikaniers alle Schüler ermittelt und deren Nummern aufsummiert werden sollen, wird dem Parameter der Name meister zugeordnet. Außerdem werden noch die Variablen unwichtig und reichtum angelegt.



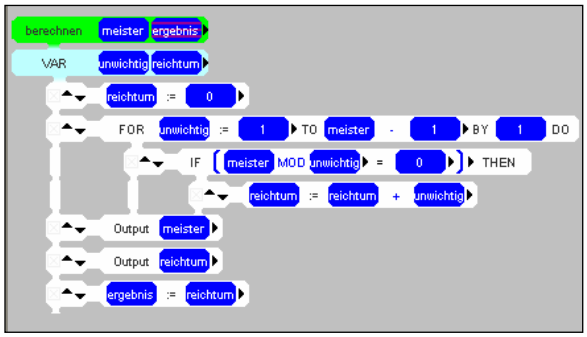
Als Erstes wird der Wert der Variablen reichtum mit einer Zuweisung auf 0 gesetzt.
Jetzt folgt die Berechnung des Reichtums. Dies kann mit einer Schleife geschehen, in der alle unwichtigeren (also kleineren) Nummern durchgegangen werden. Dafür wird ein FOR-Baustein mit der Anschlussstelle der Prozedur verbunden und unwichtig als Laufvariable ausgewählt. Für den Startwert wird 1 und für den Endwert meister - 1 eingestellt. In der so entstandenen Schleife werden nun alle die Informatikanier, die unwichtiger als der Meister sind, durchgegangen. Es sollen aber nur die Nummern zum Reichtum des Meisters addiert werden, die bei einer Division den Rest Null ergeben. Dafür wird in der Bedingung eines IF-THEN-Bausteins ein Vergleich eingesetzt. Auf der linken Seite wird der Rest bei der ganzzahligen Division von meister durch unwichtig mit der MOD-Operation berechnet. Wenn diese Operation 0 ergibt, wird mit einem Zuweisungs-Baustein der Variable reichtum der Wert des Ausdrucks reichtum + unwichtig übergeben. Am Ende der Prozedur werden mit Output-Bausteinen die Werte der Variablen meister und reichtum mit entsprechenden Beschreibungen ausgegeben.



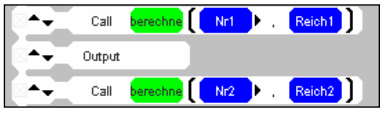
Um in der Prozedur ProgMain die Berechnung ausführen zu lassen, müssen unter den Input-Bausteinen zwei Prozeduraufruf-Bausteine mit den Anschlussstellen verbunden werden. Diese enthalten ein rotes Feld, welches signalisiert, dass noch nicht klar ist, welche Prozedur sie aufrufen sollen. Mit der rechten Maustaste kann dort in beiden Bausteinen die Prozedur berechnen ausgewählt werden. Im Ausdruck, welcher hinter dem ausgewählten Prozedurnamen erscheint, werden nun die Werte der Variablen Nr1 und Nr2 als Startwerte für den Parameter meister übergeben. Zwischen den Prozeduraufruf-Bausteinen kann noch ein Textausgabebaustein eingefügt werden, welcher den Übergang in eine neue Zeile veranlasst.

Soll durch das Programm ausgegeben werden, welcher der beiden Informatikanier der reichere ist, so ist dies bei genauerer Betrachtung nicht so einfach möglich. Denn innerhalb der Prozedur berechnen sind immer nur die Nummer und der Reichtum eines Informatikaniers bekannt. In ProgMain sind nur die Nummern der Informatikanier bekannt, denn die Berechnung des Reichtums wird innerhalb der aufgerufenen Prozedur erledigt. Es müsste also eine Möglichkeit geben, den berechneten Reichtum an ProgMain zurückzugeben. Hierfür bieten sich Referenzparameter an.

> Referenzparametern wird beim Prozeduraufruf eine Variable übergeben. Diese Variable wird in der Prozedur mit dem Namen des Parameters angesprochen. Alle Berechnungen werden mit der Variablen selbst durchgeführt. So kann der Wert der außerhalb der Prozedur definierten Variablen in der Prozedur verändert werden.



Um das Ergebnis der Prozedur berechnen zurückzugeben, wird ein weiterer Parameter mit dem Namen ergebnis angelegt. Nachdem in dessen Attributtabelle *Referenzparameter* gewählt wurde, wird dieser schraffiert dargestellt. Am Ende der Prozedur wird dem Referenzparameter ergebnis der Wert von reichtum zugewiesen. Im Prozeduraufruf-Baustein muss nicht nur die Nummer übergeben werden, sondern auch eine Variable, in welcher der Reichtum des entsprechenden Informatikaniers gespeichert wird. So ist nach dem Aufruf von berechnen auch der Reichtum der beiden Informatikanier in Variablen gespeichert. Mit einem IF-THEN-Baustein kann verglichen werden, welcher der reichere ist.



Aufgabe 4:
Welcher Informatikanier ist der reichste?
Gibt es Informatikanier, die genauso reich sind wie ihre Nummer?
Gibt es mehr Informatikanier, die reicher als ihre Nummer sind, oder mehr, die ärmer sind?
Versuchen Sie die Fragen jeweils mit Hilfe eines Computerprogramms zu beantworten.

Quellen

[Sc 04] Schiller, Friedrich: Sämtliche Werke in fünf Bänden; Alt, Peter-André / Meier Albert / Riedel Wolfgang (Hrsg.); Carl Hanser Verlag; München, Wien; 2004; Band I; S. 314.

[Ei 05] Einstein, Albert: Mein Weltbild; Seelig, Carl (Hrsg.); 28. Auflage; Ullstein Verlag; 2005; S.29.

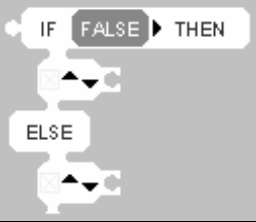
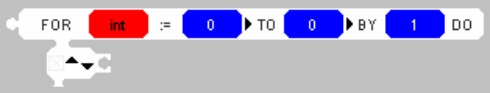
[Fo 02] Fothe, Michael: Problemlösen mit Python; Thüringer Institut für Lehrerfortbildung, Lehrplanentwicklung und Medien (Hrsg.): Reihe Materialien; Heft 72; Bad Berka; 2002.

[We 78] Weizenbaum, Joseph: Die Macht der Computer und die Ohnmacht der Vernunft; Suhrkamp Taschenbuch Verlag; Frankfurt am Main; 1978; S.362.

[www1] Gauld, Alan: Programmieren lernen;
Übersetzung ins Deutsche von Bruno Schäfer;
http://www.freenetpages.co.uk/hp/alan.gauld/german/tutwhat.htm ;
Stand: 12.01.2006.

[www2] Werbefilm für das Informatikjahr 2006;
http://www.informatikjahr.de/index.php?id=123 ; Stand: 12.01.2006.

Bausteinübersicht

Name	Beschreibung
Zuweisung 	Bei einer Zuweisung wird in der Variablen auf der linken Seite der Wert des Ausdrucks auf der rechten Seite gespeichert.
Input 	Der Benutzer wird zur Eingabe einer Zahl aufgefordert. Die Zahl wird in die ausgewählte Variable gespeichert.
Output 	Dem Anwender wird der Wert eines Integer-Ausdrucks im Ausgabefenster ausgegeben. Außerdem kann vor der Zahl noch ein Text ausgegeben werden.
Textausgabe 	Dem Anwender wird ein Text ausgegeben. Anschließend kann ein Zeilenumbruch erfolgen.
IF THEN 	Beim IF-THEN-Baustein wird eine Bedingung überprüft. Ist die Bedingung erfüllt so werden die Anweisungen des THEN-Zweiges ausgeführt. Ist die Bedingung nicht erfüllt, so werden die Anweisungen des ELSE-Zweiges ausgeführt.
WHILE DO 	Eine Bedingung wird jeweils am Anfang überprüft. Die Anweisungsfolge wird so oft wiederholt abgearbeitet, wie das Auswerten der Bedingung TRUE ergibt.
REPEAT UNTIL 	Die Anweisungsfolge wird abgearbeitet und anschließend wird eine Bedingung überprüft. Solange das Auswerten der Bedingung FALSE ergibt, wird erneut mit der Abarbeitung begonnen. Erst wenn das Ergebnis der Überprüfung TRUE ist, wird die Abarbeitung der Schleife abgebrochen.
FOR 	Bei einem FOR-Baustein wird der Wert einer Integer-Variablen von einem Startwert bis zu einem Endwert um eine bestimmte Schrittweite erhöht. Die mit den Anschlussstellen verbundenen Bausteine werden in jedem Schritt abgearbeitet.
Prozeduraufruf 	Eine Prozedur ist ein Unterprogramm. Immer wenn mit einem Prozeduraufruf-Baustein eine Prozedur aufgerufen wird, werden die Anweisungen dieser Prozedur abgearbeitet.
Dot 	Mit dem Dot-Baustein ist es möglich einen Punkt an der mit der x- und y-Koordinate angegebenen Stelle in dem Ausgabefenster zu erzeugen.
Line 	Durch den Line-Baustein können Linien im Ausgabefenster erzeugt werden. Dafür müssen die Koordinaten von zwei Punkten angegeben werden.
Color 	Mit dem Color-Baustein wird die Zeichenfarbe festgelegt. Für rot, grün und blau können jeweils Werte zwischen 0 und 255 angegeben werden.