Ein Bild, das Text, Schrift, Screenshot, Electric Blue (Farbe) enthält.

Automatisch generierte Beschreibung

Assignment 02  
Programmieren I  
29.10.2024

Norwin Bertram Wiesecke

10057811

Timon Wellhausen  
10041137

Aufgabe 1:

1. Implementieren Sie das Programm zum Zahlenraten wie beschrieben. Achten Sie darauf, dass der Stack nach erfolgreicher Beendigung des Spiels leer ist (ohne die Verwendung von clear).

*Siehe .pf-Datei.*

1. Spielen Sie das Spiel selbst fünf Mal. Notieren Sie jeweils wie viele Züge Sie gebraucht haben um die Zahl zu erraten. Schreiben Sie ebenfalls auf wie viele Züge Sie in Durchschnitt gebraucht haben.

|  |  |
| --- | --- |
| Durchlauf Nummer | Anzahl Versuche |
| Durchlauf 1 | 6 |
| Durchlauf 2 | 7 |
| Durchlauf 3 | 5 |
| Durchlauf 4 | 5 |
| Durchlauf 5 | 1 |
| Durchschnittliche Versuche | 4,8 |

1. Beschreiben Sie eine Strategie um die Zahl zu erraten, mit der möglichst wenige Züge benötigt werden.

Man beginnt mit einer Eingabe von 50. Anhand der Antwort des Programms kann man entweder die oberen oder unteren 50% der Werte direkt ausschließen. In den nächsten Schritten wählt man immer den Wert, der mittig liegt. Dieser Algorithmus wird „Divide and Conquer“ genannt. Auf diese Art und Weise kann man frühzeitig große Teile der möglichen Werte ausschließen.

Aufgabe 2:

1. Entwickeln Sie eine Funktion wochenstunden, die für eine Vorlesung mit einer bestimmten Anzahl LP an der Uni berechnet, wieviel Aufwand Sie ungefähr pro Woche investieren sollten.

*Siehe .pf-Datei.*

Aufgabe 3:

1. Für die Lesbarkeit von Quelltext ist die Formatierung sehr wichtig, denn Quelltext wird häufiger gelesen, als geschrieben. Formatieren Sie den Quelltext.

*Siehe .pf-Datei.*

1. Erklären Sie das Verhalten der Funktion f möglichst kurz und prägnant. Welcher Wert liegt nach Ausführung des gegebenen Codes auf dem Stack?

Die Funktion gibt im Output Fenster die Nachricht „called f“ aus und prüft dann, ob der gegebene Parameter „i“ < 0 ist. Wenn dies der Fall ist, wird „i“ mit -1 multipliziert, andernfalls wird er mit 2 multipliziert.  
Nach Ausführen des Codes liegt 6:Int im Stack, da -3:Int < 0 ist und mit -1 multipliziert wird, wobei 3:Int auf den Stack gelegt wird. Danach wird 3:Int mit 2 multipliziert, da der Wert größer als 0 ist.

1. Das Heron-Verfahren wurde in der iterativen Variante in PostFix implementiert Leider haben sich einige Fehler in die Funktion geschlichen. Formatieren Sie zuerst die Funktion, sodass sie gut lesbar ist. Finden Sie dann die Fehler und korrigieren Sie diese. Beschreiben Sie die Fehler.

*Siehe .pf-Datei.*

1. Die Variable x wird an zwei Stellen nicht richtig definiert.  
2. Die Postfix Notation wurde in den Rechnungen ignoriert.   
3. „break if“ muss durch „breakif“ oder „{break]} if“ ersetzt werden.  
4. Am Ende der Funktion muss x abgerufen werden, um auf den Stack gelegt zu werden. Das ist das Äquivalent zu „return x“.

Aufgabe 4:

1. Welche Bezeichner verweisen auf Funktionen?

Der Identifier “fun” sorgt dafür, dass der betroffene Code als Funktion compiled wird.

1. Was machen die einzelnen Funktionen? Geben sie an, was (welche Datentypen) die einzelnen Funktionen als Parameter auf dem Stack erwarten, was (welche Datentypen) sie als Ergebnis auf den Stack legen und was sie auf der Ausgabe ausgeben. Ergänzen Sie in der Funktionssignatur (innerhalb der pf-Datei, keine extra Tabelle) den Typ der Parameter und ggf. den Typ des Rückgabewertes.

**trennen:**

Die Funktion erwartet einen Integer auf dem Stack, der in diesem Beispiel vermutlich die Anzahl der Eier darstellen soll. Dieser Wert wird zwei Mal in einen Array gelegt, welcher dann auf den Stack gelegt wird. Im Output wird kein Text ausgegeben.

**vermengen:**

Die Funktion erwartet 2 Integer auf dem Stack, die in diesem Beispiel vermutlich die Gewichtsangaben der entsprechenden Zutaten in Gramm darstellen soll. Diese beiden Werte werden addiert und als Integer auf den Stack gelegt. Im Output wird folgender Text ausgegeben:

vermenge  
...  
<:Int>vermengt

**Ei-zugeben:**

Die Funktion nimmt den Array aus der trennen-Funktion, dessen Werte die Anzahl der Eigelbe und Eiweiße darstellen soll. Der erste Wert aus dem Array stellt die Menge der Eigelbe dar und wird mit 10 multipliziert, was vermutlich das Gewicht der Eigelbe in Gramm ergeben soll. Dieser Wert wird dann auf den Integer-Wert des Gesamtgewichts der bereits vermischten Zutaten aufaddiert. Dasselbe passiert auch mit dem zweiten Wert des Arrays, wobei hier der Wert mit 20 multipliziert wird. Im Output wird folgender Text ausgegeben:

<:Arr(0)>Eigelb hinzufügen   
...  
<:Arr(1)>Eiweiß hinzufügen

**backen:**

Die Funktion entnimmt einen Integer Wert vom Stack, der in dieser Implementierung die Gesamtmenge der vermischten Zutaten darstellt, sowie die Integer-Werte der beiden Werte „BEI\_200°C“ und „20\_MINUTEN“. Die Werte werden teilweise genutzt, um im Output angegeben zu werden und teilweise um ein sleep Kommando das Programm für eine bestimmte Zeit pausieren zu lassen. Im Output wird dann folgender Text ausgegeben:

vorheizen auf <BEI\_200°C>°C  
vorgeheizt!  
Backen  
...  
<:Int>Kuchen gebacken

1. Ergänzen Sie die Funktionsdefinitionen mit einer aussagekräftigen Parameterbeschreibung. Dazu gehören ein Purpose Statement, die Beschreibung der einzelnen Parameter sowie die Beschreibung des Rückgabewertes.

*Siehe .pf-Datei.*

1. Beschreiben Sie in 1-2 Sätzen, was das Programm nachfolgend nach dem Kommentar #Rezept macht.

Das Programm ruft die Funktionen mit den vorher definierten Werten ab und stellt damit den Teil des Codes dar, der den funktionellen Teil des Programms ausführt.

1. Was ist Ihre Meinung zu dem Code hinter dem Kommentar #Rezept. Sollten Programme so geschrieben sein, sodass der Programmcode annähernd wie natürliche Sprache klingt? Begründen Sie kurz.

Es hilft, Namen von Variablen, Funktionen, etc. nach der entsprechenden Funktion zu benennen. Dadurch wird der Code verständlicher, wenn man ihn später liest. Dennoch kann es eine schlechte Angewohnheit sein, wie in diesem Beispiel sehr kleine Codestücke in eigene Funktionen umzuwandeln, da dadurch die Funktionalität hinter obskuren Funktionsnamen versteckt wird, bei denen man sich die tatsächlichen Funktionsmechanismen nur schwer erschließen kann.   
In diesem Fall würde wahrscheinlich eine Funktion „kuchen\_backen“ ausreichen.