Übungen zu: 7. Algorithmen

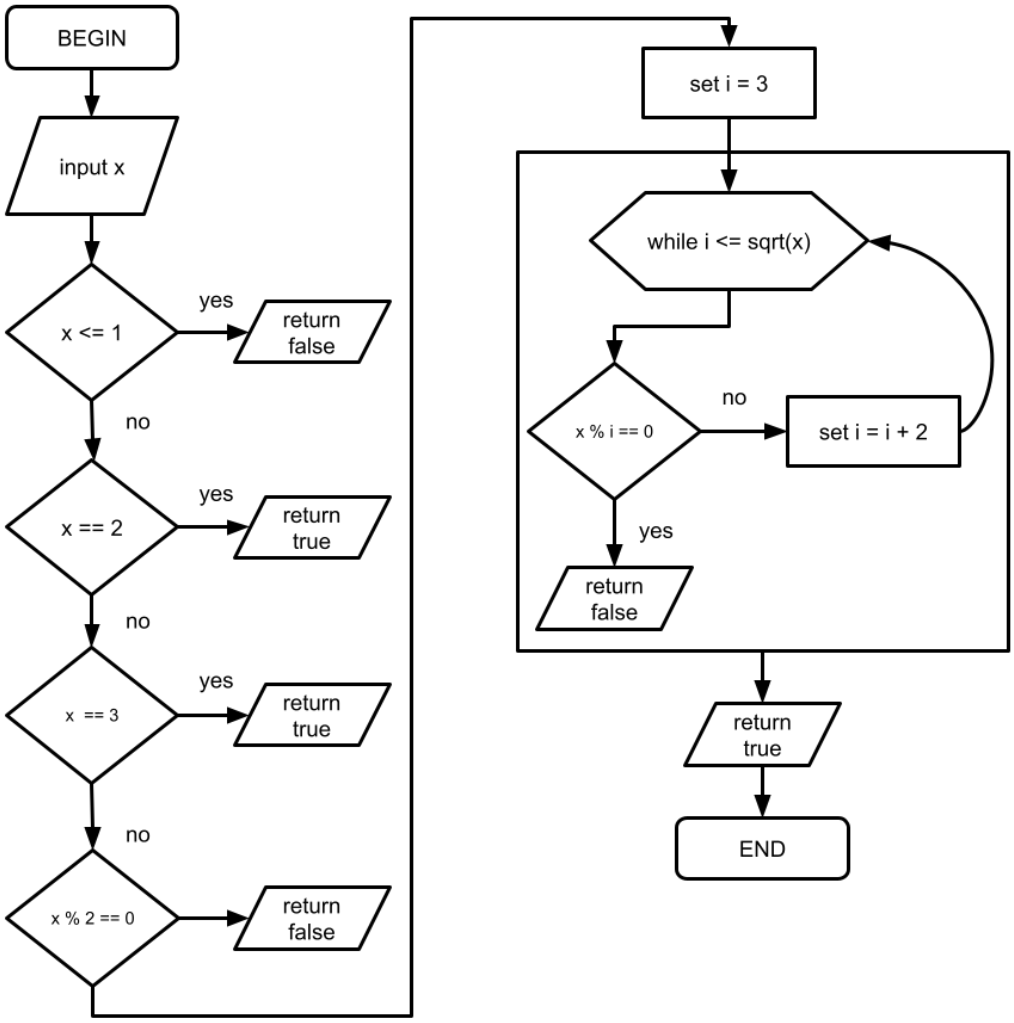
Digitalisierung und Programmierung

Übungen

1. Woher stammt der Begriff "Algorithmus"?
2. Definiere, was ein Algorithmus ist, und gib drei verschiedene Beispiele aus dem Alltag, die keinen direkten Bezug zu Computern haben.
3. Erläutere, was mit der Komplexität eines Algorithmus gemeint ist. Warum ist die Komplexität eines Algorithmus wichtig? Wie wird sie angegeben?
4. Welche Komplexitätsklassen kennst du? Bringe sie in eine Reihenfolge von der geringsten zur höchsten Komplexität.
5. Berechne den GGT von 56 und 98 mithilfe des euklidischen Algorithmus und dokumentiere jeden Schritt!
6. Berechne die Quadratwurzel von 25 mit der babylonischen Methode und dokumentiere jeden Schritt!
7. Erkläre die Funktionsweise des babylonischen Algorithmus zur Berechnung der Quadratwurzel. Verwende dazu visuelle Hilfsmittel. Warum konvergiert der Algorithmus gegen den exakten Wert der Quadratwurzel?
8. Vergleiche die Ergebnisse der babylonischen Methode nach 3, 5 und 7 Iterationen mit dem exakten Wert der Quadratwurzel.
9. Erkläre die Monte Carlo Methode zur Schätzung von π und beschreibe, wie Zufallszahlen zur Annäherung von π verwendet werden können.
10. Du hast kennengelernt, wie unterschiedliche Informationen in einem Computer repräsentiert werden. Das ist wichtig, um die Eingabe und die Ausagbe von Algorithmen zu beschreiben. Überlege, wie die Ein- und Ausgabe des Dijkstra-Algorithmus repräsentiert werden könnte. Was benötigt der Algorithmus als Eingabe und was gibt er als Ausgabe zurück?

Pseudocode

Betrachte den Pseudocode in der Abbildung unten und beantworte die folgenden Fragen!



1. Beschreibe die Ein- und Ausgabeinformationen für den gezeigten Algorithmus. Verwende dazu das visuelle Problemlösungsschema aus der Vorlesung!
2. Wie werden die Ein- und Ausgabeinformation für diesen Algorithmus im Computer repräsentiert? Wie viele Bits benötigen sie?
3. Welches Problem löst der Algorithmus? Beschreibe den Ablauf des Algorithmus in eigenen Worten! Gehe dazu den Pseudocode Schritt für Schritt für verschiedene Beispiele durch!

4. Wie würdest du die Komplexität des Algorithmus beschreiben? In welche Komplexitätsklasse ordnest du ihn ein?

5. Implementiere den Algorithmus in Python. Schreibe eine Funktion, die die Eingabe als Parameter entgegen nimmt und die korrekte Ausgabe mit `return` zurückgibt. Teste die Funktion mit verschiedenen Eingaben!