Codeknacker

Das erste etwas längere Programm soll Ihnen einige Elemente der Programmierung spielerisch näher bringen. Es soll darum gehen, eine „Geheimzahl“ zu raten, also einen Code zu „knacken“. Jede Variante sollten Sie zunächst versuchen selbstständig zu lösen. Wenn Sie keine Lösungsansätze finden, suchen Sie sich einen Tischnachbarn und versuchen es gemeinsam. Wenn Sie gemeinsam nicht weiterkommen (und die PDFs zu den Grundlagen gelesen haben) rufen Sie als 2er Team den Lehrer (dies ist Teil der mündlichen Mitarbeit: entscheiden können ob sie Hilfe brauchen und diese einfordern. Das ist wichtig!!! Ein „Ich wusste ja nicht wie das geht ist dann nämlich eine mdl schlechte Note!)

Variante 1

Für die erste Variante müssen Sie sich eine Geheimzahl ausdenken. Der Benutzer soll nun eine Zahl eingeben und bekommt von ihrem Programm gesagt „Die Zahl ist richtig“ oder die Zahl ist falsch

(Sie üben dabei: Eingabe/Ausgabe, einfache bedingte Anweisungen, Benutzung von Variablen)

Variante 2

Nun erweitern Sie ihre Ausgabe um die Bemerkung „Die eingegebene Zahl ist zu groß“ oder „Die eingegebene Zahl ist zu klein“ oder „Die eingegebene Zahl ist richtig“

(Sie üben dabei zusätzlich zur Variante 1 komplexere bedingte Anweisungen, Vergleiche >, <,)

Variante 3

Für die nächste Variante erweitern Sie das Programm um eine Schleife, und geben so lange neue Zahlenversuche ein, bis die Zahl gefunden wurde.
(Sie üben dabei zusätzlich zur Variante 2 den Einsatz einer wiederholten Anweisung (while-Schleife))

Variante 4

Für die nächste Variante erweitern Sie das Programm so, dass Sie nachher dem Benutzer angeben können, wie viele Versuche er gebraucht hat.
(Sie üben dabei zusätzlich zur Variante 3 den Einsatz eines Zählers)

Variante 5

Nun kommt der Zufall ins Spiel. Dies ist ein sehr großer Schritt, gerade jetzt, zu Beginn ihrer Programmierer-Laufbahn. Aber es erhöht deutlich den Reiz dieses Programms wenn man eben selber auch nicht weiß welche Zahl geraten werden muss. Zunächst benötigt man ein neues Element, welches eine solche Zufallszahl erzeugt. Wie eine solche Funktion in C# heißt und wie sie benutzt wird, kann man leicht auf
https://docs.microsoft.com/de-de/dotnet/api/system.random.next?view=netframework-4.8 finden. Bauen Sie diesen Programmcode ein, und versuchen Sie die die zufällige Zahl herauszufinden. Aus welchen Zahlenbereich kommt die Zufallszahl, wenn sie keine weiteren Angaben machen? (Sie üben dabei zusätzlich zur Variante 4 den Einsatz eines Zufallszahlengenerators)

Variante 6

Was für eine Zufallszahl kann dort überhaupt bei herauskommen? Auf der obigen Internetseite steht etwas von 0 bis MaxValue – klickt man auf MaxValue erfährt man, dass dies 2147483647 sein kann. Wie kann man so eine Zahl aber verändern, sodass man Zufallszahlen zwischen 1 und 100 herausbekommt, oder zwischen 1000 und 10000?

Dazu gibt es einen sogenannten Modulo Operator, das % -Zeichen.

Dieses % ist eine so genannte Modulo Rechnung, die Sie aus der Grundschule kennen:
7:3= 2 Rest 1 schreibt ein Grundschüler bei einer Division hin, die nicht komplett aufgeht. Das was dort als Rest herauskommt, ist genau das, was bei einer Modulo-Rechnung als Ergebnis angezeigt wird. Damit kann man die Zufallszahlen von 0 bis 2147483647 auf einen beliebigen kleineren Zahlenraum abbilden. Tun Sie das und verändern Ihr Programm so, dass Sie Zahlen zwischen 1 und 1000 eingeben können. Überlegen Sie dann wie sie dann auch Zahlen zwischen 100 und 200 herauskriegen können.
(Sie üben dabei zusätzlich zur Variante 5 die Modulo-Rechnung)

Variante 7

Natürlich ist das „old-School“ heutzutage liefert natürlich der Zufallsgenerator zielgenau die Werte die man braucht, indem man entsprechende Parameter in den Funktionsaufruf eingibt. Nutzen Sie dies um ihr Programm für die nächste Aufgabe fit zu machen: Der Benutzer gibt nun den Zahlenraum selber ein, indem er die Grenzen selber festlegt, zwischen denen die Zufallszahlen gewählt werden sollen.

(Sie üben hier die Verwendung von Funktionen MIT Parametern, und lernen das Wort Parameter richtig ein zu setzen 😉, und Sie üben den richtigen Einsatz von Variablen um Informationen vom Benutzer zu speichern)

Variante 8

Mit der richtigen Strategie kann man eine solche Zufallszahl erstaunlich schnell herausbekommen – wie viele Schritte benötigt man für das obige Beispiel (1000 bis 10000), wenn man die richtige Strategie benutzt? Geben Sie nun dem Benutzer eine qualifizierte Rückmeldung zu der Anzahl von Versuchen: Hatte er Glück und brauche weniger Versuche als eigentlich nötig? Dann gratulieren Sie ihm, liegt er im Durchschnitt, hat er das Prinzip verstanden, braucht er mehr Versuche animieren Sie ihn es neu zu versuchen.

(Sie üben dabei zusätzlich zur Variante 7 noch mal die bedingten Anweisungen – haben sich aber zum ersten Mal Gedanken über einen guten Algorithmus gemacht.)

Variante 9

Überlegen Sie sich nun, wie Sie herausfinden, welche Anzahl von Versuchen „gut“ und welche Anzahl tatsächlich eine schlechte Anzahl von Versuchen darstellt. Bauen Sie diese korrekte Bewertung in Ihr Programm ein.

Überlegen Sie nun: Haben Sie das Problem programmtechnisch gelöst oder mathematisch?

Rufen Sie nun den Lehrer und erläutern Sie ihm in einem kurzen Fachgespräch ihre Lösung

(Sie lernen zusätzlich zu Variante 9 selber einen Lösungsalgorithmus zu entwickeln und sie lernen, wie ein Fachgespräch zur Notenfindung bei den späteren, größeren Programmen aussehen wird.)

Codeknacker – Zusatzaufgabe 1

Nachdem nun in allen Varianten des Codeknackers der Computer die Zahl vorgegeben hat, dürfen jetzt Sie sich die Geheimzahl selber ausdenken – der Computer soll im letzten Programm nun Ihre Zahl raten. Sie geben genauso wie der Computer in den letzten Varianten an, ob die geratene Zahl zu groß, zu klein oder richtig ist. Natürlich sollten Sie ebenfalls die Grenzen des Bereichs angeben in dem der Computer suchen darf.

(Für die Implementierung müssen Sie nun den Algorithmus, den Sie zur Suche des richtigen Wertes im Kopf haben wieder in einen Programmierten Algorithmus umsetzen).

Codeknacker – Zum Nachdenken: Zusatzaufgabe 2

Der Zufallsgenerator kann nur Zahlen zwischen 0 und ca. 2147483647 erzeugen. Was müssen Sie tun, um einen zufälligen Code aus einem größeren Zahlenbereich (z.B. 0-100.000.000.000) zu erzeugen?

Wenn Sie eine Idee haben, überprüfen Sie mal ob ihre Zufallszahlen genauso „gut“ sind, wie die von rand();

Erzeugen Sie dafür 200 Zahlen mit ihrem Verfahren, kopieren Sie diese in Excel, sortieren sie und schauen, ob es Zahlenbereiche gibt, aus denen Sie mehr Zahlen bekommen, und Zahlenbereiche wo weniger Zufallszahlen getroffen werden.

Finden Sie einen Algorithmus, der wieder gleich verteilte Zufallszahlen aus dem größeren Bereich findet?