



Girls Who Code At Home

Das Labyrinth debuggen
Offline-Aktivität

Übersicht über die Aktivität

Als **Informatikerin** arbeitest du daran, **Codes** oder Befehle für einen Computer zu schreiben, damit dieser Aufgaben erledigt. Es mag überraschen, dass die meisten **Programmierer** tatsächlich die meiste Zeit mit dem **Debuggen** verbringen, also mit dem Suchen und Beseitigen von Problemen im Code! Debugging erfordert Hartnäckigkeit und die Bereitschaft, nach neuen Wegen bei der Lösung von Problemen zu suchen. Bei dieser Aktivität musst du einen Code debuggen, der eine Figur durch ein Labyrinth führt. Du kannst auch deinen eigenen Code schreiben und einen Freund herausfordern, diesen zu debuggen. Bevor du mehr über Debugging erfährst, solltest du Brittany Wenger kennenlernen, die bei unserem Spotlight zu Frauen aus der Technikbranche vorgestellt wird. Im Alter von 15 Jahren entwickelte sie ein Tool, das bei der Diagnose von Brustkrebs hilft, sodass frühestmöglich mit einer Behandlung begonnen werden kann.

Materialien

- kleine Spielfigur oder Objekt (optional)
- Arbeitsblatt mit Lösungen zum Labyrinth-Debugging

Spotlight zu Frauen in der Technikbranche: Brittany Wenger



Wenn du auf ein Problem stößt, wie suchst du nach einer Lösung? Wo wirst du zuerst suchen, wenn du die Antwort nicht kennst? Brittany Wengers Idee war es, für ihre Suche nach Antworten auf wichtige Fragen zu Brustkrebs Code einzusetzen.

Als Brittany 15 Jahre alt war, wurde bei ihrer Cousine Brustkrebs diagnostiziert. Etwa 1 von 8 Frauen in den USA erkrankt im Laufe ihres Lebens an Brustkrebs. Wird lokal begrenzter Brustkrebs früh erkannt, liegt die Überlebens-Chance bei 100 %. Brittany entschied sich zum Handeln und entwickelte einen Algorithmus, der Daten von einem Patienten sammelt und abschätzt, ob ein Knoten in der Brust

gut-oder bösartig ist. Ihr Algorithmus [Cloud4Cancer](#) konnte Knoten in der Brust mit einer Genauigkeit von 99,11 % bestimmen.

Schau dir dieses [Video](#) über Brittany Wenger an. Dabei erfährst du, wie mit Code Antworten auf wichtige Fragen gefunden werden, die Menschenleben betreffen.

Überlegung

Informatikerin sein, bedeutet mehr als einfach nur gut in der Code-Entwicklung zu sein. Nimm dir etwas Zeit und denke darüber nach, was Brittany's Arbeit mit den Stärken zu tun hat, die du als großartige Informatikerin brauchst – Belastbarkeit, Hartnäckigkeit, Kreativität und sinnvolle Ziele.



HARTNÄCKIGKEIT

Brittany konnte Technologien nutzen, um Frauen weltweit zu helfen, Brustkrebs früh zu erkennen und sofort mit einer Behandlung zu beginnen.

Erinnere dich daran, wie du bei einer Aktivität oder einem Projekt alles gegeben hast. Was hat dich motiviert, dich so sehr zu bemühen?

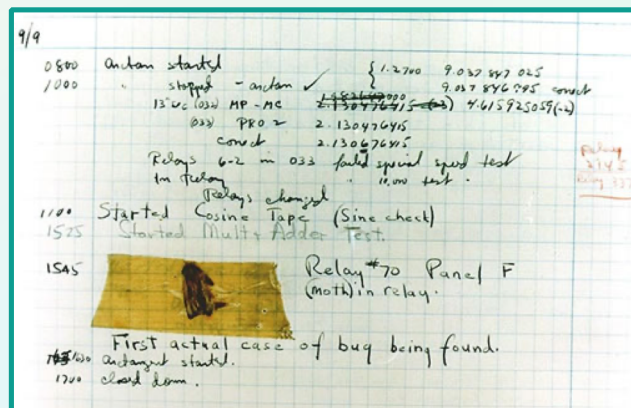
Teile deine Antworten mit einem Familienmitglied oder einer Freundin. Ermutige andere, mehr über Brittany zu lesen und sich am Gespräch zu beteiligen!

Schritt 1: Debugging verstehen (2 Min.)

Erinnere dich daran, wie du einmal versucht hast, ein Problem zu lösen. Das kann eine schwierige Aufgabe in der Schule sein oder der Versuch, etwas wiederzufinden. Hattest du die Lösung sofort parat? Wenn wir versuchen, ein Problem zu lösen, müssen wir oft zuerst Fehlschläge in Kauf nehmen, bevor wir Erfolg haben. Informatiker verbringen ihre Tage weitgehend damit, Probleme in ihrem Code zu finden und sie zu lösen.

Einen Fehler in einem Computerprogramm oder in der Hardware nennt man einen **Bug**. Der Prozess zur Bestimmung und Beseitigung von Fehlern oder Bugs aus Hardware oder Software heißt **Debugging**. Diese Begriffe gehen auf Grace Hopper zurück, eine der ersten Heldinnen der Informatikgeschichte. Bei der Arbeit mit einem der ersten Computer suchte Grace Hoppers Team nach der Ursache für einen Fehler und fand eine Motte in der Maschine – eine echte Motte! Der Tagebucheintrag von Grace mit der aufgeklebten Motte gilt heute als die erste Aufzeichnung eines Computer-Bugs in der Geschichte. Er ist im Smithsonian Museum of American History in Washington D.C. zu sehen.

Bei dieser Aktivität wirst du einen Code debuggen, der eine Figur vom Startpunkt bis zum Endpunkt durch ein Labyrinth bewegt und dabei versucht, einen Stern aufzusammeln. Du kannst auch deinen eigenen Code schreiben und einen Freund herausfordern, diesen zu debuggen.






Der erste dokumentierte Computer-Bug

Schritt 2: Die Spielregeln (2 Min.)

Ziel des Spiels ist es, eine Figur von einem Startpunkt aus zu bewegen; sie muss alle Sterne aufsammeln und an einem Endpunkt ankommen. Dazu muss man alle **Bugs** (oder Fehler) in den Befehlen finden und beheben.

Schritt 2: Spielregeln (Fortsetzung)

Regeln

1. Deine Figur muss am Startpunkt beginnen .
2. Deine Figur muss am Endpunkt ankommen . Sobald deine Figur am Endpunkt ankommt, ist das Spiel zu Ende und es kommen **keine** weiteren Befehle mehr.
3. Du musst den Stern  aufsammeln, **bevor** du zum Endpunkt kommst.
4. Zum Aufsammeln des Sterns muss sich die Figur am gleichen Ort befinden wie der Stern.
5. Du kannst die Befehle nur **bearbeiten** oder **löschen**. Einen Schritt bearbeiten bedeutet, dass Werte in den Befehlen durch einen anderen Wert ersetzt werden oder dass ein **Rechtschreibfehler** behoben wird.
6. **NUR** diese Befehle sind erlaubt. Rechtschreibung und Großschreibung müssen genauso sein, wie in dieser Liste.
 - a. nach links
 - b. nach rechts
 - c. nach oben
 - d. nach unten
 - e. [1–5] Mal wiederholen:
 - f. Wiederholen beenden
 - g. Stern aufsammeln
7. Du kannst **keine** eigenen Befehle hinzufügen.

Schritt 3: Überprüfen der Gültigkeit der Befehle (2 Min.)



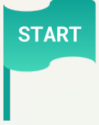
Wie beim Erlernen einer Sprache (wie Spanisch, Französisch, Deutsch etc.) gibt es beim Codieren eine Reihe von eigenen Regeln. Markiere in der folgenden Auswahl, welche Codezeilen gültig sind. **Denke daran, dass nur die oben aufgelisteten Befehle gültig sind (Regel Nr. 6).** Achte auch auf Rechtschreibung und Großschreibung.

- | | | | |
|------------------------|-----------------------|-----------------------|------------------------|
| A. Drehe nach rechts | B. Rücke vorwärts | C. Sternee aufsammeln | D. 3 Mal wiederholen: |
| E. 10 Mal wiederholen: | F. Rücke zur Linken | G. nach oben | H. Wiederholen beenden |
| I. Sterne aufsammeln | J. Stern fallenlassen | K. Gehe zu Ziel | L. nach unten |

Vergleiche deine Antworten mit den Lösungen im Arbeitsblatt am Ende des Dokuments.

Schritt 4: Zeit zum Debugging! (5 Min.)

Unten siehst du das Spielfeld, auf dem sich der Start in Feld A3, der Stern in Feld B2 und das Ziel in Feld A1 befindet. Durch Ausführen der Code-Befehle soll die Figur vom Start, zum Aufsammeln des Sterns und dann zum Zielpunkt bei A1 gehen. Wir haben jedoch festgestellt, dass die Figur **zurzeit** den Stern nicht aufsammelt und am Punkt C1 ankommt.

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (A1) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (C1)

Das Befolgen der zuvor aufgelisteten Regeln beseitigt („debuggt“) den fehlerhaften Code und schreibt den richtigen Code in die Spalte „Debuggter Code/Befehl“.

Tip: Verwende zu Hause eine kleine Figur oder ein Objekt als Spielfigur. Du kannst sie durch das Labyrinth bewegen, während du jede Codezeile ausführst, um den Standort der Figur nachzuverfolgen.




Falscher Code/Befehl	Debuggter Code/Befehl
1. nach oben	
2. nach rechts	
3. Stern aufsammeln	
4. nach oben	
5. nach rechts	

HINWEIS: Es gibt **2** Bugs

Du kannst deine Antworten mit den Lösungen am Ende des Dokuments vergleichen.

Schritt 5: Auf zur nächsten Herausforderung (5-10 Min.)

Oh, Nein! Noch ein Bug! Hilf mit, die folgenden Befehle zu debuggen. Vergiss nicht, dir die Regeln anzuschauen.

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (B, 2) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (B, 2)

Falscher Code/Befehl	Debuggter Code/Befehl
1. nach oben	
2. nach rechts	
3. nach oben	
4. nach rechts	
5. Stern aufsammeln	
6. nach unten	
7. nach links	




HINWEIS: Es gibt **2** Bugs

BONUS: Diese Befehle bilden eine Lösung, um den Stern aufzusammeln und an das Ziel zu kommen, aber es gibt dafür **mehrere** Wege. Kannst du dir eine andere Lösung vorstellen, die zum gleichen Ergebnis führt?

Du kannst deine Antworten mit den Lösungen am Ende des Dokuments vergleichen.

Schritt 6: Debugging-Schleifen (5-10 Min.)

Schon wieder ein Bug! Aber Moment, diese Darstellung sieht aus wie in der letzten Herausforderung! Der richtige Code müsste wie der Code aus Herausforderung Nr. 2 aussehen, aber hier wird der Befehl *Wiederholen* verwendet. In der Informatik nennen wir das eine Schleife oder „Loop“. **Loops** werden verwendet, um eine Reihe von Anweisungen mehrere Male zu *wiederholen*. Befehle, die wir wiederholen möchten, werden *eingerückt* dargestellt. So weiß der Computer, welche Codezeilen zu wiederholen sind.

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (B, 2) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (B, 2)

Falscher Code/Befehl	Debuggter Code/Befehl
1. <u>1</u> Mal wiederholen:	
2. nach oben	
3. Wiederholen beenden	
4. <u>3</u> Mal wiederholen:	
5. nach rechts	
6. Wiederholen beenden	
7. Stern aufsammeln	
8. Gehe nach links	
9. nach unten	

HINWEIS: Es gibt **3** Bugs

Du kannst deine Antworten mit den Lösungen am Ende des Dokuments vergleichen.

Schritt 7: Schreibe deinen eigenen Code! (10–15 Min.)

Und nun bist du an der Reihe! Fordere einen Freund heraus, ein eigenes Code-Set zu schreiben, in dem es **maximal 4 Bugs** gibt. Fülle auch die Tabelle mit den erwarteten und den derzeitigen Aktionen für deinen Freund aus. Verwende dieses Blatt als **Lösungsschlüssel** und wenn dir deine Herausforderung gefällt, verwende das nächste Blatt, um den fehlerhaften Code, das Labyrinth und die Aktionstabelle für deinen Freund zu **kopieren**.

Hilfreicher Tipp: Schreibe die richtigen Befehle zuerst auf und ändere dann höchstens 4 Zeilen oder füge einen weiteren Bug hinzu.

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (,) 	<ul style="list-style-type: none"> • Stern ... • Endet am Standort (,)

Falscher Code/Befehl	Debuggter Code/Befehl
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

HINWEIS: Es gibt __ Bugs

ERSTELLER:

HERAUSFORDERER:

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none">• Stern aufsammeln• Am Endpunkt ankommen (,)	<ul style="list-style-type: none">• Stern ...• Endet am Standort (,)

Falscher Code/Befehl	Debugger Code/Befehl
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

HINWEIS: Es gibt __ Bugs

Schritt 8: Teile dein Werk! (5 Min.)




1. Mache ein Bild oder scanne dein Labyrinth und fordere einen Freund heraus, deine Befehle zu debuggen.
2. Vergiss nicht, dein Labyrinth in den Sozialen Medien zu teilen. Tagge @girlswhocode #codefromhome und wir werden dich vielleicht sogar auf unserem Profil vorstellen!

Das Labyrinth debuggen – Lösungen

Schritt 3: Überprüfen der Gültigkeit der Befehle

A. Drehe nach rechts Kein gültiger Befehl.	B. Rücke vorwärts Kein gültiger Befehl.	C. Sternee auf sammeln Rechtschreibfehler.	D. 3 Mal wiederholen:
E. 10 Mal wiederholen: Kein gültiger Befehl. Wiederholen kann nur 1 – 5 Mal ausgeführt werden.	F. Rücke zur Linken Kein gültiger Befehl.	G. nach oben	H. Wiederholen beenden
I. Sterne auf sammeln Rechtschreibfehler.	J. Stern fallenlassen Kein gültiger Befehl.	K. Gehe zu Ziel Kein gültiger Befehl.	L. nach unten




Schritt 4: Zeit zum Debugging – Lösungen

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (A, 1) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (C, 1)

Falscher Code/Befehl	Debuggter Code/Befehl
1. nach oben	1. nach oben
2. nach rechts	2. nach rechts
3. Stern aufsammeln	3. Stern aufsammeln
4. nach oben	4. nach oben
5. nach rechts	5. nach links

Schritt 5: Auf zur nächsten Herausforderung – Lösungen

	A	B	C
1			
2			
3			




Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (B, 2) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (B, 2)

Falscher Code/Befehl	Debugger Code/Befehl
1. nach oben	1. nach oben
2. nach rechts	2. nach oben
3. nach oben	3. nach rechts
4. nach rechts	4. nach rechts
5. Stern aufsammeln	5. Stern aufsammeln
6. nach unten	6. nach unten
7. nach links	7. nach links

BONUS: Diese Befehle bilden eine Lösung, um den Stern aufzusammeln und an das Ziel zu kommen, aber es gibt dafür **mehrere** Wege. Kannst du dir eine andere Lösung vorstellen, die zum gleichen Ergebnis führt?

Eine weitere Lösung wäre: nach rechts, nach rechts, nach oben, nach oben, Stern aufsammeln, nach unten, nach links. Die letzten beiden Schritte können auch ausgetauscht werden. Bei Herausforderung Nr. 4 gibt es auch eine andere Lösung, bei der du eine Schleife verwendest.

Schritt 6: Debugging-Schleifen – Lösungen

	A	B	C
1			
2			
3			

Erwartete Aktionen	Derzeitige Aktionen
<ul style="list-style-type: none"> • Stern aufsammeln • Am Endpunkt ankommen (B, 2) 	<ul style="list-style-type: none"> • Stern wird nicht aufgesammelt • Endet am Standort (B, 2)

Falscher Code/Befehl	Debuggter Code/Befehl
1. <u>1</u> Mal wiederholen:	1. <u>2</u> Mal wiederholen:
2. nach oben	2. nach oben
3. Wiederholen beenden	3. Wiederholen beenden
4. <u>3</u> Mal wiederholen:	4. <u>2</u> Mal wiederholen:
5. nach rechts	5. nach rechts
6. Wiederholen beenden	6. Wiederholen beenden
7. Stern aufsammeln	7. Stern aufsammeln
8. Gehe nach links	8. nach links
9. nach unten	9. nach unten

Diese Lösung hat mehr Schritte als die vorherige ist. WARUM ist es besser, Schleifen zu verwenden? Schleifen sind praktisch, wenn ein Befehl oder Schritt oft wiederholt wird (mehr als zwei Mal) oder wenn es mehrere Schritte gibt, die sich wiederholen. In diesem besonderen Fall sind es nicht weniger Schritte. Aber es ist oft hilfreich, den Code kürzer zu machen, weil man ihn dann besser lesen kann.