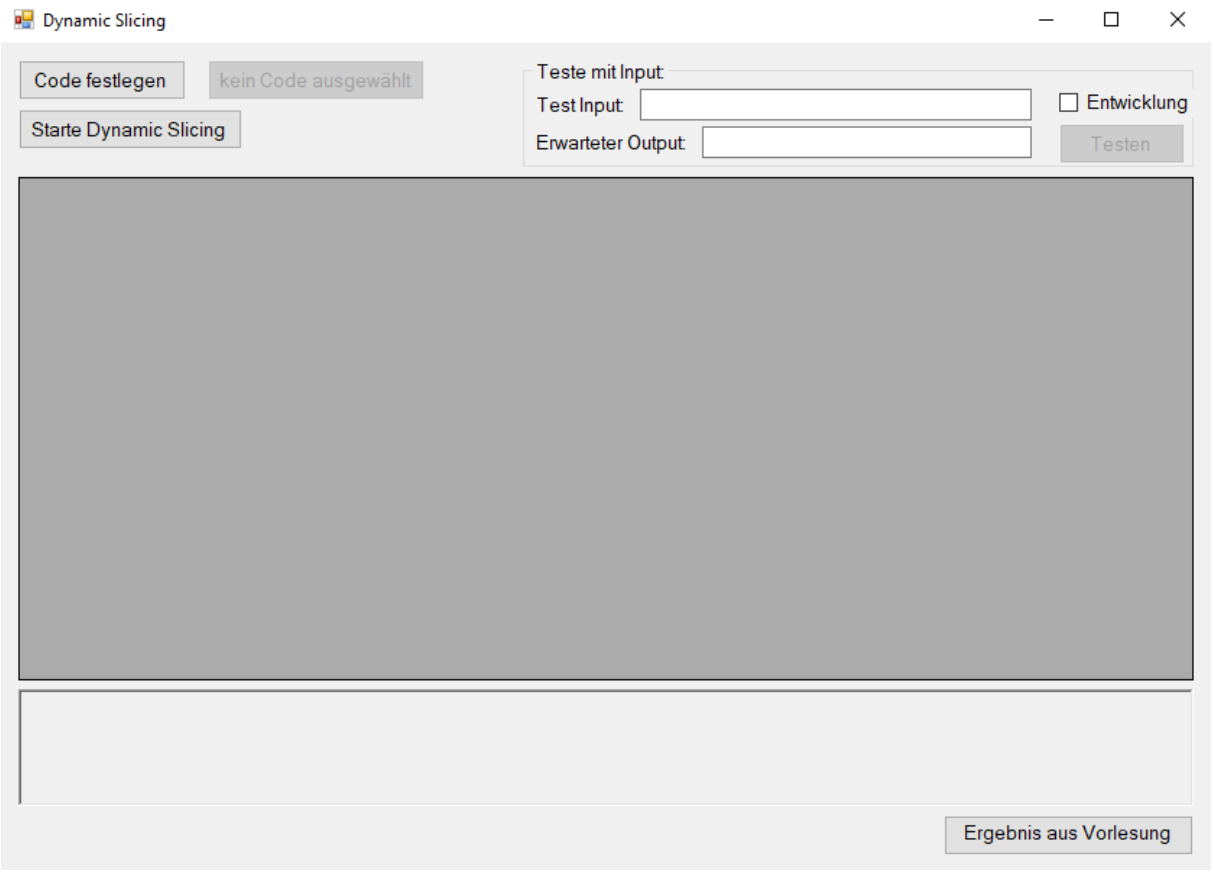
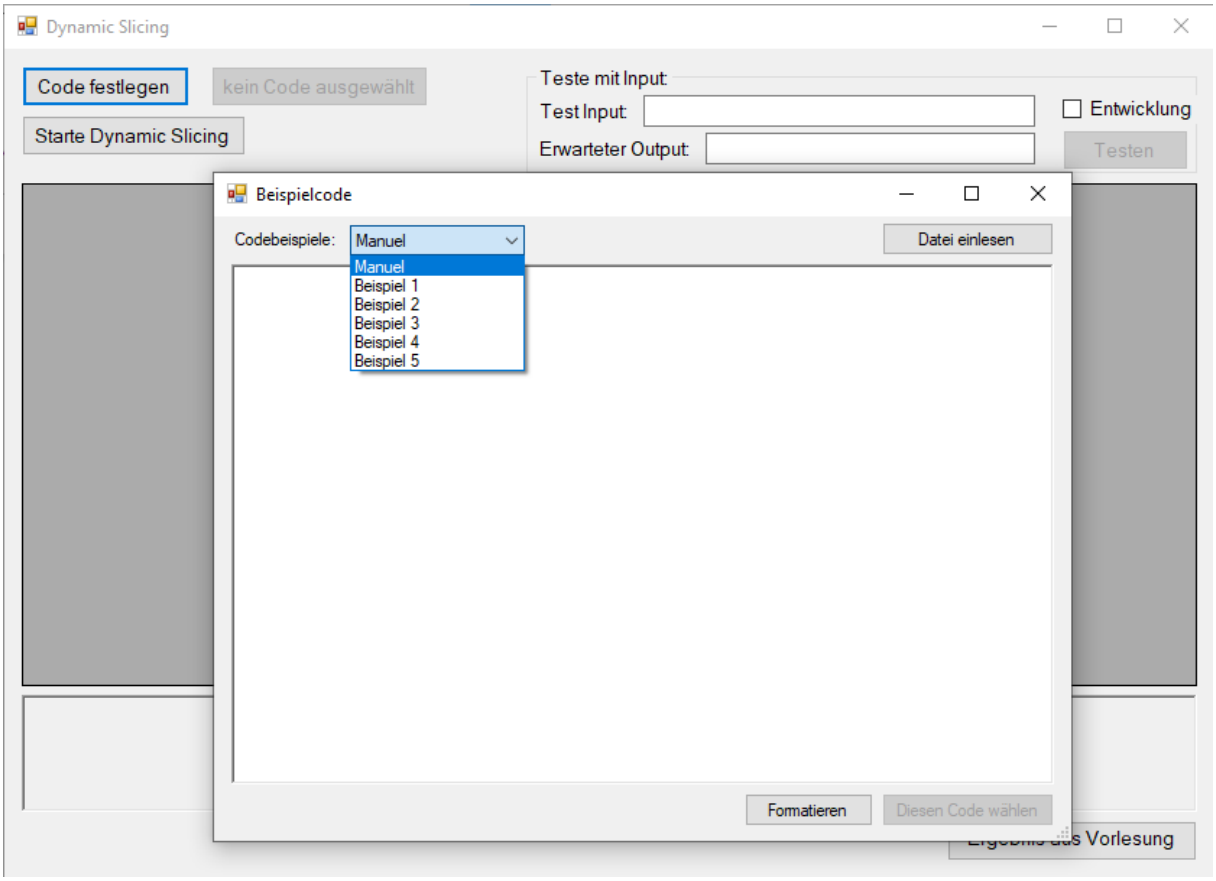
Aktueller Stand: Dynamic Slicing (14.02.2020)

Beim Start des Programms liegt diese Ansicht vor:



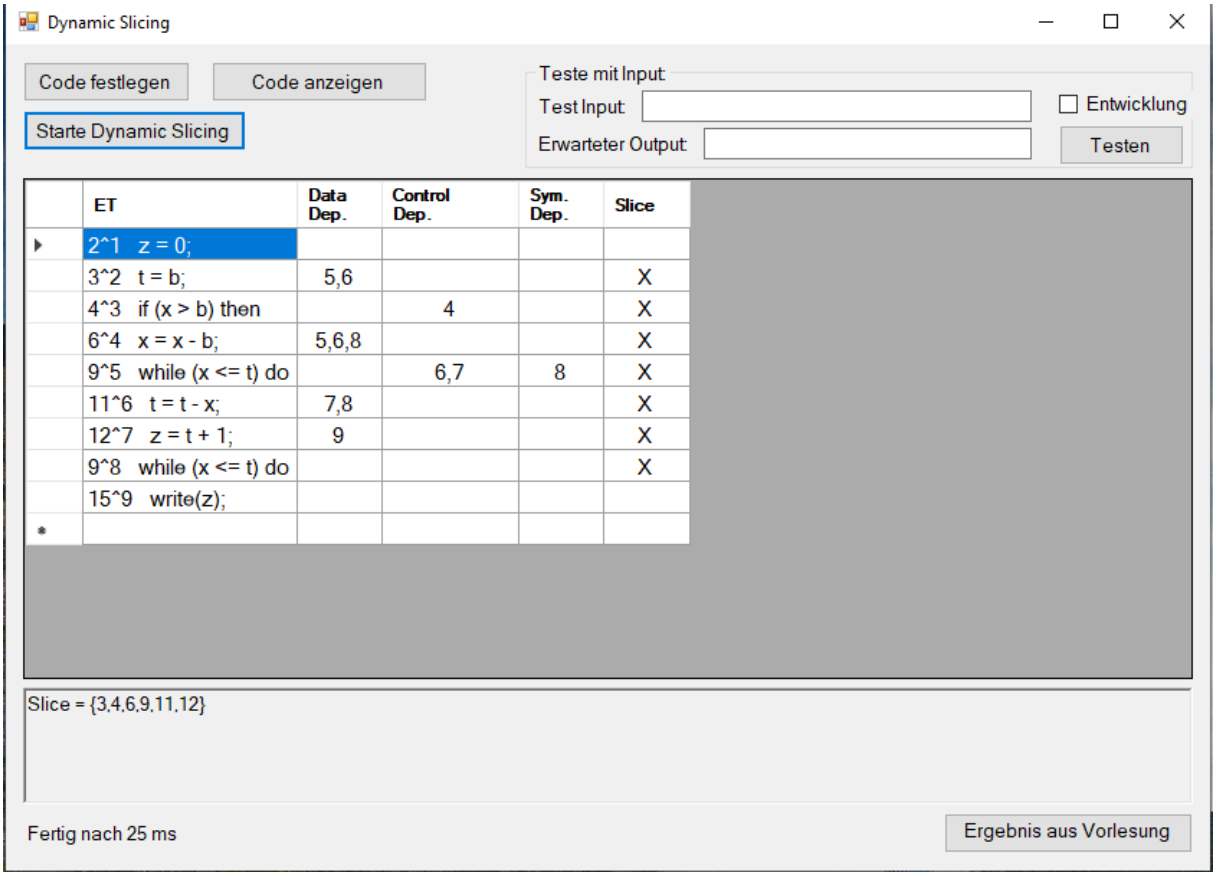
Zuerst wird erwartet, dass man einen Code festlegt. Dazu drückt man auf den entsprechenden Button. Dann öffnet sich ein zweites Fenster.



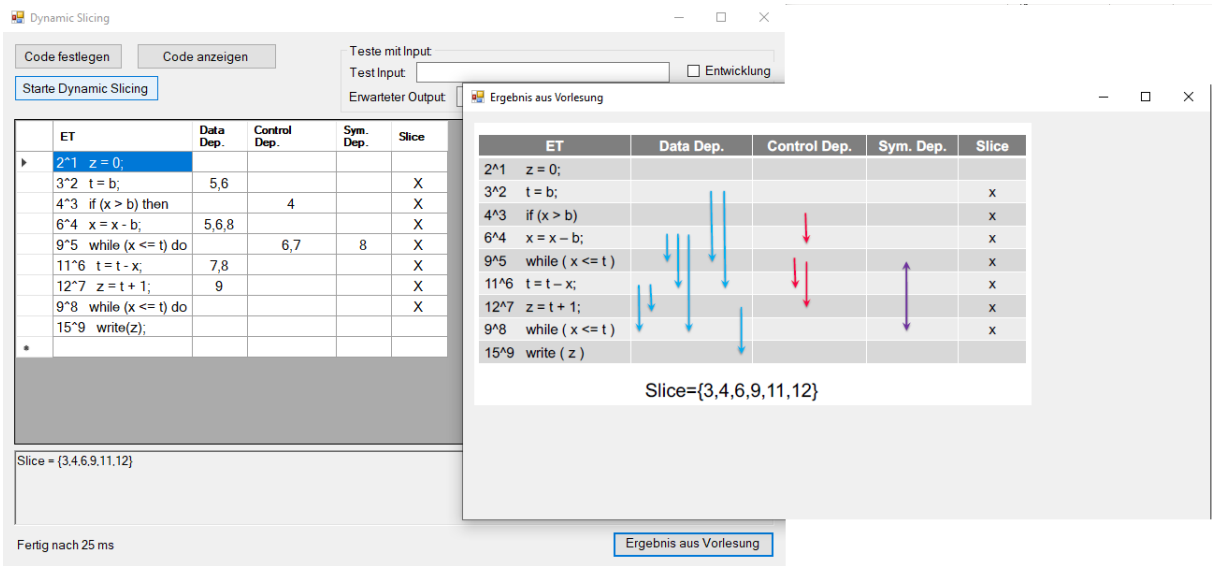
Man kann nun den Code selber eintragen, einen der fünf vorbereiteten Codes auswählen oder eine Datei einlesen, welche den Code enthält. Im Anschluss muss auf den Button „Formatieren“ geklickt werden. Der Code wird nun auf syntaktische Fehler überprüft und nach folgenden Regeln formatiert:

- Geschwungene Klammern werden entfernt
- Nur Tabulatoren werden für das Kennzeichnen einer Ebene verwendet
- Datentypen werden entfernt
- Leere Zeilen werden entfernt

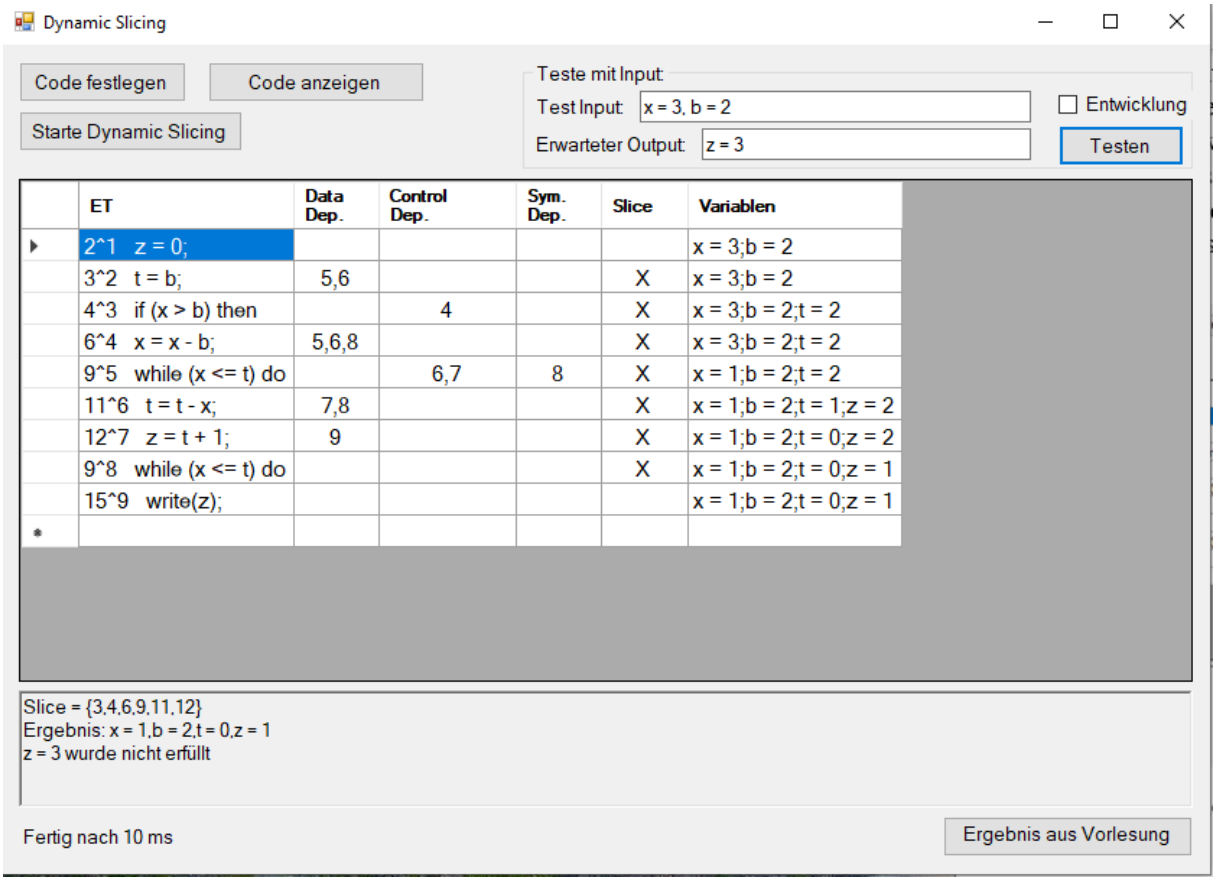
Wenn die Prüfung und Formatierung erfolgreich waren, kann der Button „Diesen Code wählen“ geklickt werden und das Fenster schließt sich.



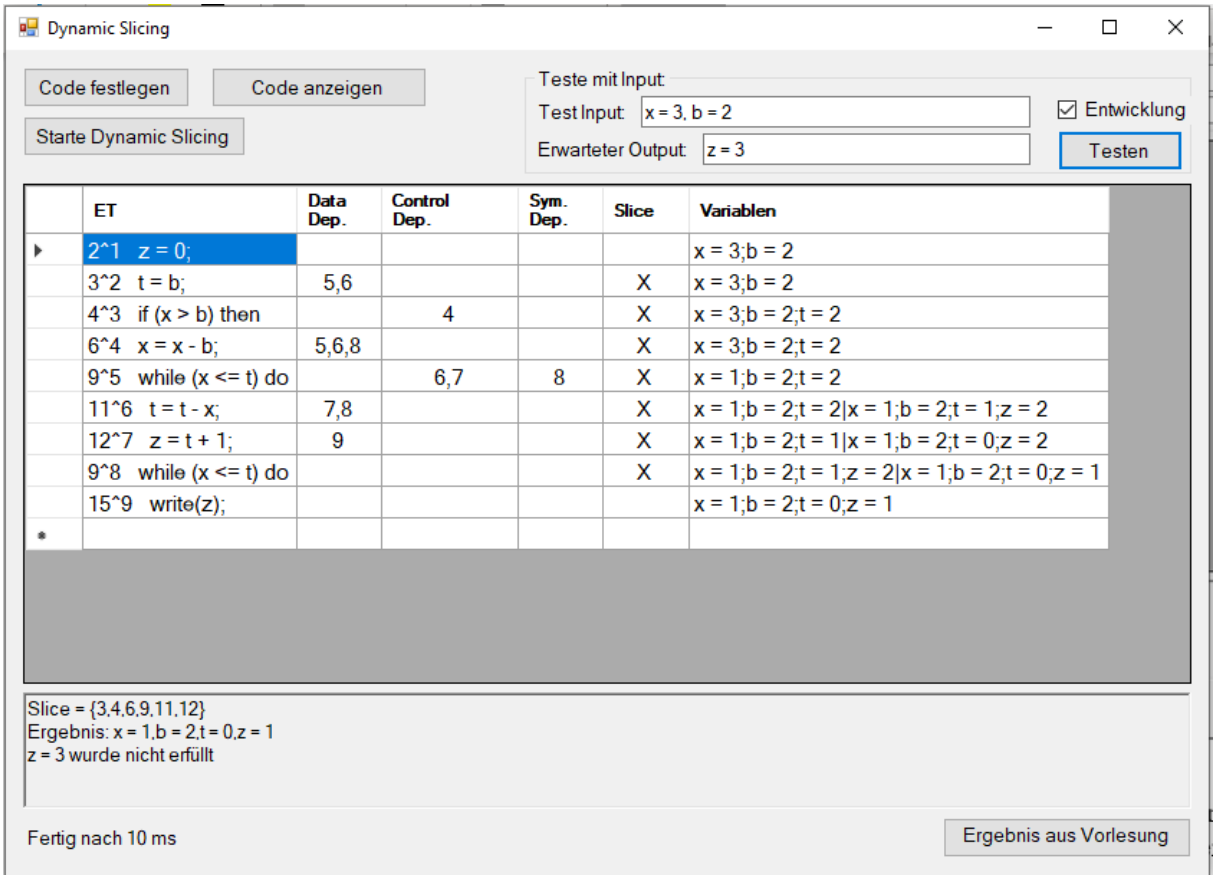
Nun klicken wir auf „Starte Dynamic Slicing“. Sofort werden für den ausgewählten Code der Execution Trace (ET) generiert und die einzelnen Dependencies und die für die letzte Zeile im ET verantwortlichen Zeilen bestimmt. Nun lässt sich schon erkennen, wie die Zeilen untereinander in Verbindung stehen, dass z.B. Zeile 2 im ET eine Data Dependency zur Zeile 5 im ET aufweist. Bisher habe ich noch nicht die Pfeile visualisiert. Trotzdem funktioniert der Algorithmus schon mit verschachtelten Schleifen (for und while) und if-else-Statements. Da ich diesen Code aus der Vorlesung habe, existiert zum Vergleich eine fertige Tabelle, die man sich als Vorschau mit Klick auf den Button „Ergebnis aus Vorlesung“ anzeigen lassen kann.



Man sieht hier, dass der Algorithmus für das Beispiel die richtige Tabelle liefert.



Hier habe ich nun mal als Input „x = 3, b = 2“ gewählt und erwarte den Output: „z = 3“. Ich klicke auf den Button „Testen“ und sofort wir der Input auf den Code angewandt. Man kann nun erkennen, welche Variablen in jeder Zeile bekannt sind und welchen Wert sie haben. Im Fall von Schleifen ändern sich die Werte, bevor sie am Ende angezeigt werden. Deswegen habe ich die CheckBox „Entwicklung“ hinzugefügt. Jetzt kann man jeden Schritt nachvollziehen.



Wir haben für den Input ein Ergebnis erhalten und der Output wurde nicht erfüllt. Wir wissen aber, welche Zeilen dafür verantwortlich sein könnten.

Ausblick:

Aktuell versuche ich, verschiedene Exceptions zu händeln. Das Tool ist noch sehr abhängig vom eingegebenen Code des Users. Noch werden nicht alle syntaktischen und semantischen Fehler gefunden. Sobald das „Code Festlegen“ sicher funktioniert, werde ich die Pfeile visualisieren, wie es auch in der Vorlesung gemacht wurde. Sicher kann man auch an der Darstellung noch ein wenig verbessern.