Institut für Informatik
Prof. Dr. Stefan Conrad

Sommersemester 2020
Übung 7

Hardwarenahe Programmierung
Gruppe 05 (Florian)

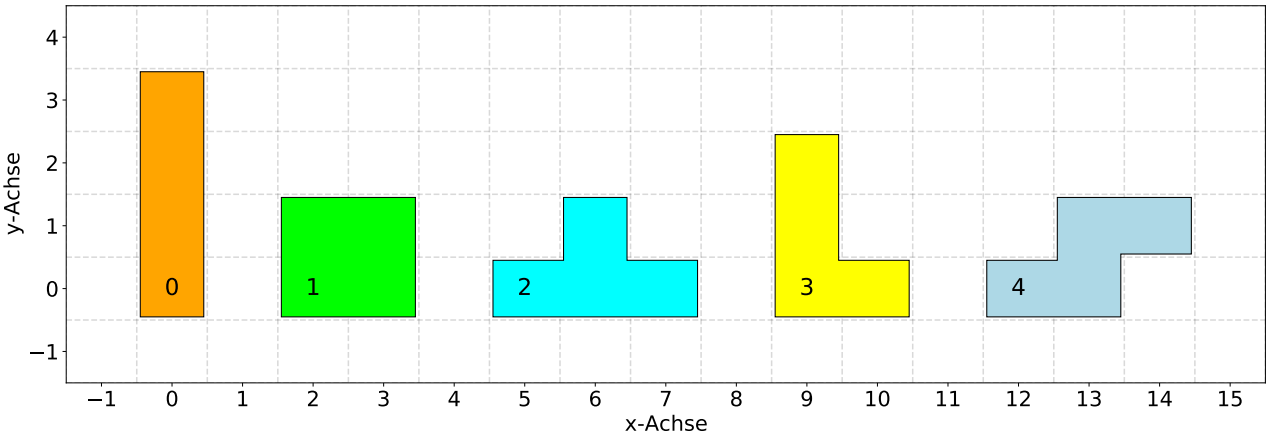
Bei diesem Abschlussprojekt bearbeiten Sie eine größere C-Programmieraufgabe, in der Sie Ihre gesammelten Fähigkeiten anwenden sollen. Die Aufgabe ist erstmals nicht kleinschrittig für Sie vorbereitet. Wir empfehlen daher, dass Sie zunächst einen groben Ablaufplan in Form von Kommentaren oder Funktionsaufrufen in Ihre Codedatei schreiben. Ersetzen Sie diese nach und nach durch funktionsfähigen Code. Testen Sie die bisherige Funktionalität häufig!

Wichtig:

- *Die Abgabe ist diesmal nicht um 23:55 Uhr, sondern um 8:00 Uhr.*
- *Sie müssen das Abschlussprojekt Ihrem Tutor in einem Einzelgespräch erklären. Denken Sie daran, dazu einen Termin zu vereinbaren. Testen Sie **vorher** Ihre technische Ausstattung (Webcam/Mikrofon oder Smartphone). Halten Sie einen Lichtbildausweis bereit.*

Aufgabe 1 *Abschlussaufgabe – Tetrominos*

In dieser Aufgabe geht es um Tetrominos. Tetrominos bestehen aus vier quadratischen Blöcken, die an den Seiten miteinander verbunden sind. Um Ihnen diese Aufgabe zu vereinfachen nehmen wir an, dass es nur die folgenden fünf Tetrominosorten gibt, die nicht gedreht werden können.



- Das Tetromino der Sorte 0 ist ein senkrechter Balken.
- Das Tetromino der Sorte 1 ist ein quadratischer 2x2 Block.
- Das Tetromino der Sorte 2 hat die Form des umgedrehten Buchstaben T.
- Das Tetromino der Sorte 3 ist L-förmig.
- Das Tetromino der Sorte 4 ist S-förmig.

Wir definieren den unteren linken Block in jedem Tetromino als Referenzblock, um die Position besser angeben zu können. Zum Beispiel wurde das graue Tetromino der Sorte 4 in dem obigen Beispiel an Position $(12, 0)$ platziert und belegt die Felder $(12, 0)$, $(13, 0)$, $(13, 1)$ und $(14, 1)$.

Aufgabenstellung:

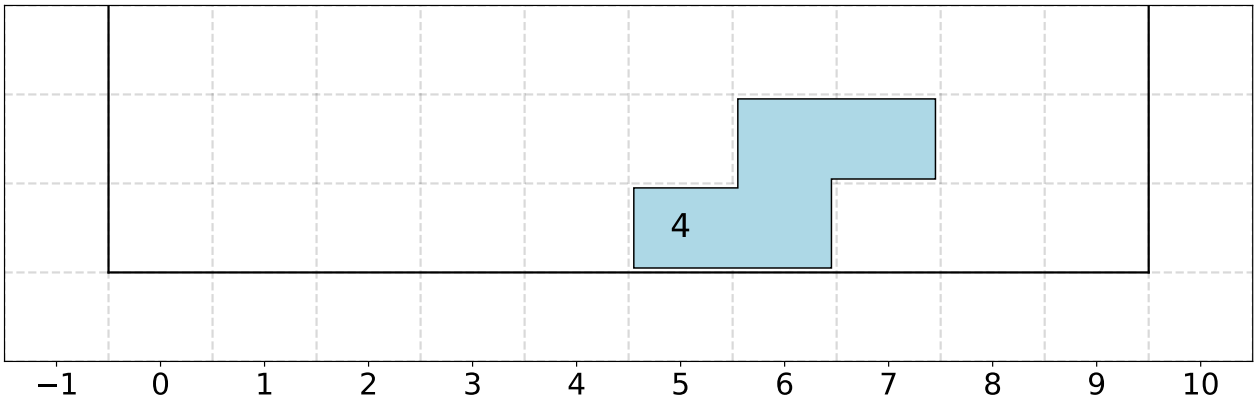
```
4 5
0 0
3 4
2 7
0 3
0 6
1 8
4 1
3 3
2 4
```

- Ihr Programm soll eine Datei entgegennehmen, in der eine Liste von Tetromino-Sorten und die x-Koordinate des Referenzblocks angegeben sind.
- Diese Tetrominos sollen nacheinander auf einem Spielfeld mit einer Breite von 10 Kästchen und unbegrenzter Höhe platziert werden. In anderen Worten, gültige Koordinaten für Tetromino-Blöcke sind $(x, y) \in \{0, \ldots, 9\} \times \{0, \ldots\}$.
- Hierbei wird angenommen, dass die Tetrominos senkrecht vom Himmel fallen und dann auf dem Tetromino, auf dem sie landen, liegen bleiben.
- Abschließend soll das Spielfeld in eine Ausgabedatei geschrieben werden.

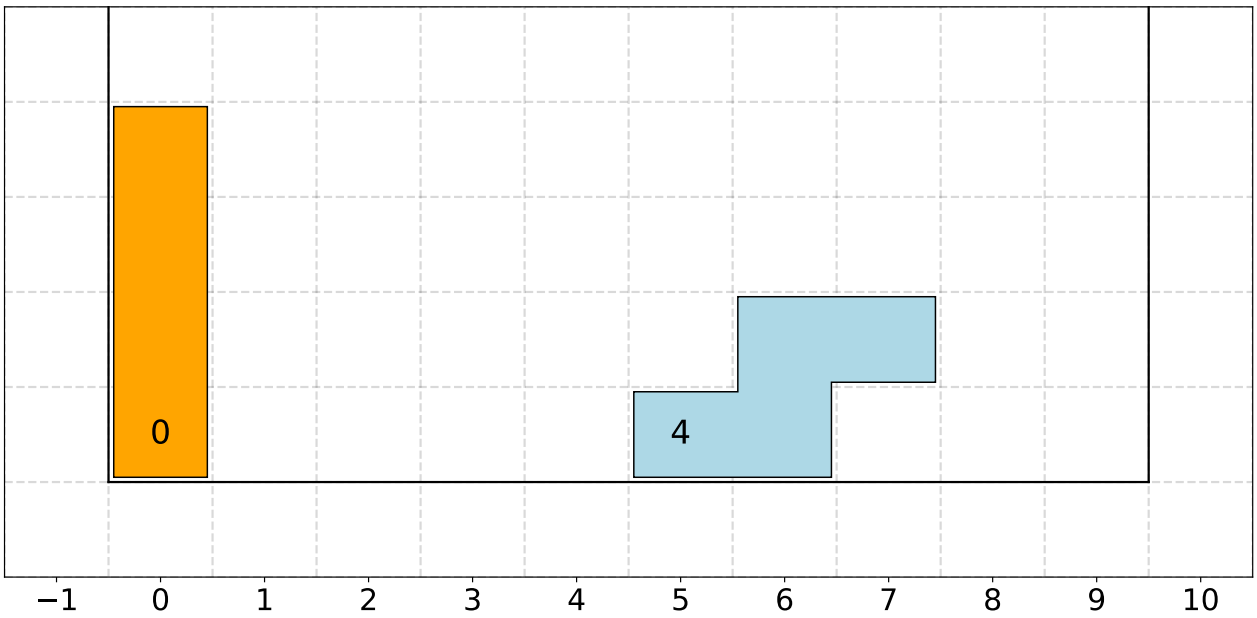
Beispiel:

Es werden nacheinander die Tetrominos aus der vorangegangenen Liste an der gegebenen x-Position eingefügt.

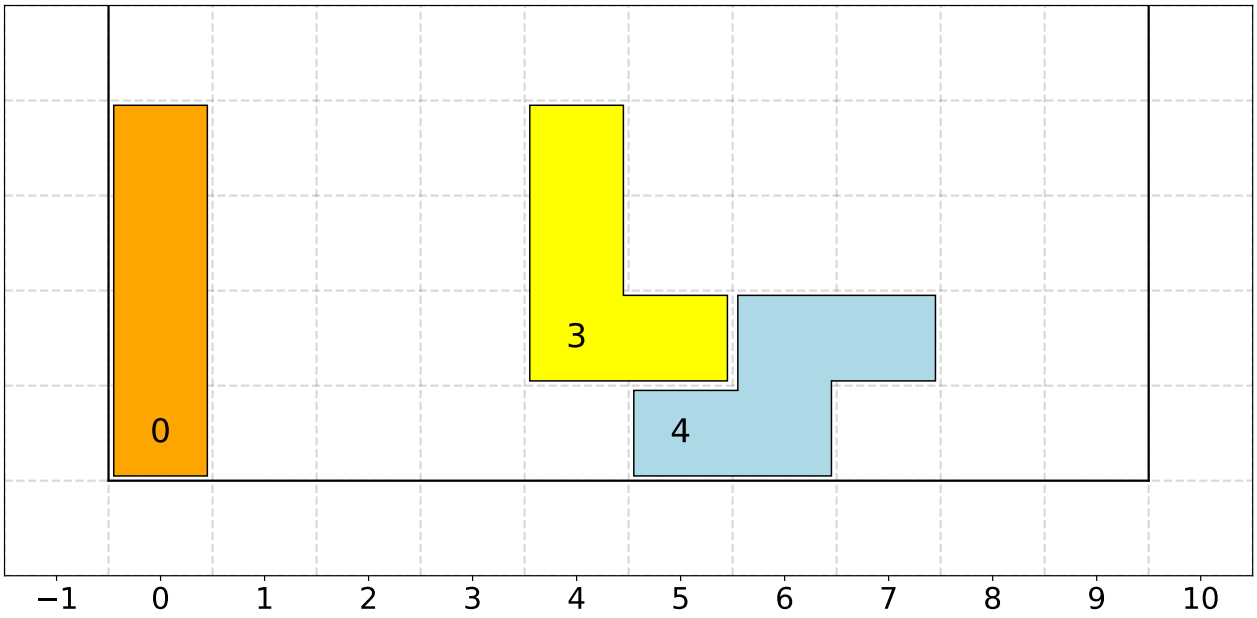
Das Tetromino der Sorte 4 mit vorgegebener x-Position 5 landet auf y-Position 0:



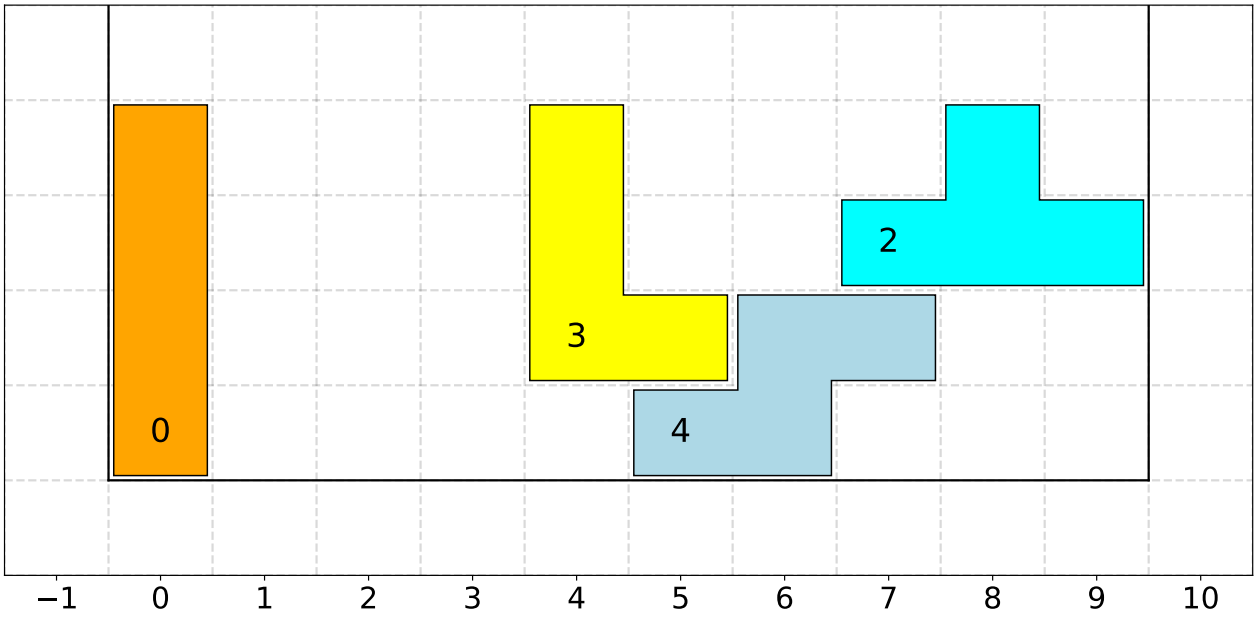
Das Tetromino der Sorte 0 mit vorgegebener x-Position 0 landet auf y-Position 0:



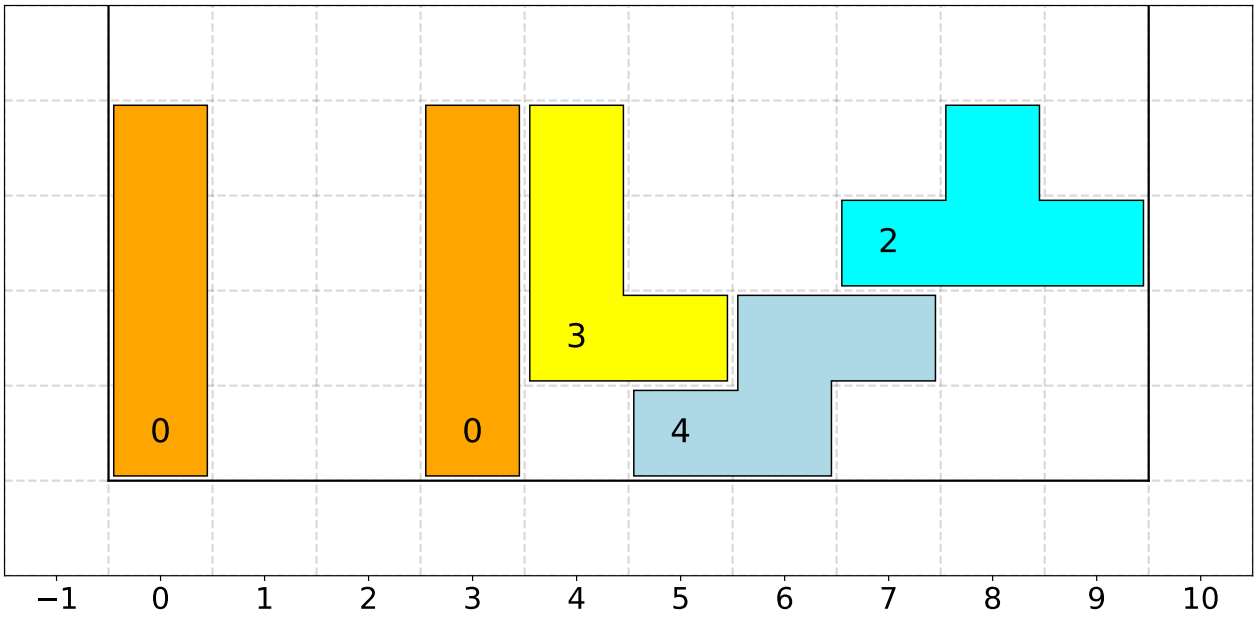
Das Tetromino der Sorte 3 mit vorgegebener x-Position 4 landet auf y-Position 1:



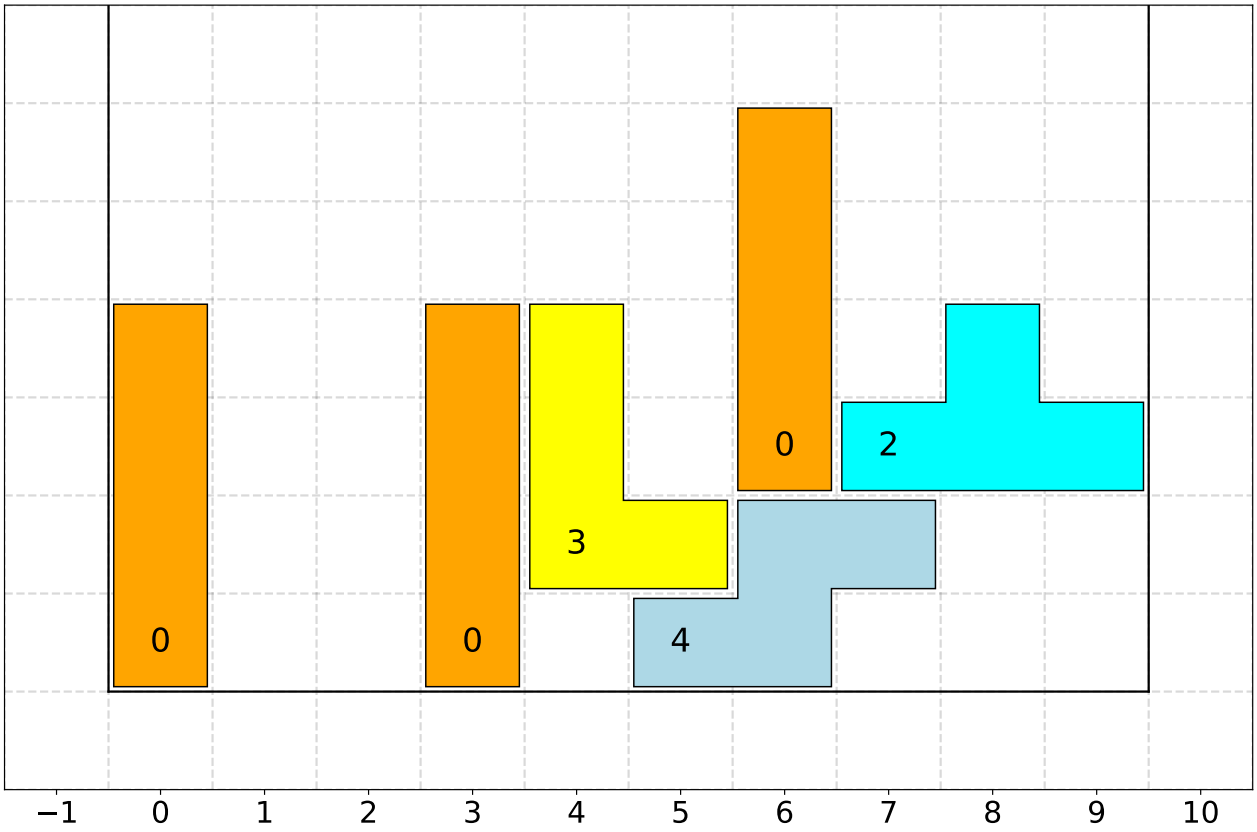
Das Tetromino der Sorte 2 mit vorgegebener x-Position 7 landet auf y-Position 2:



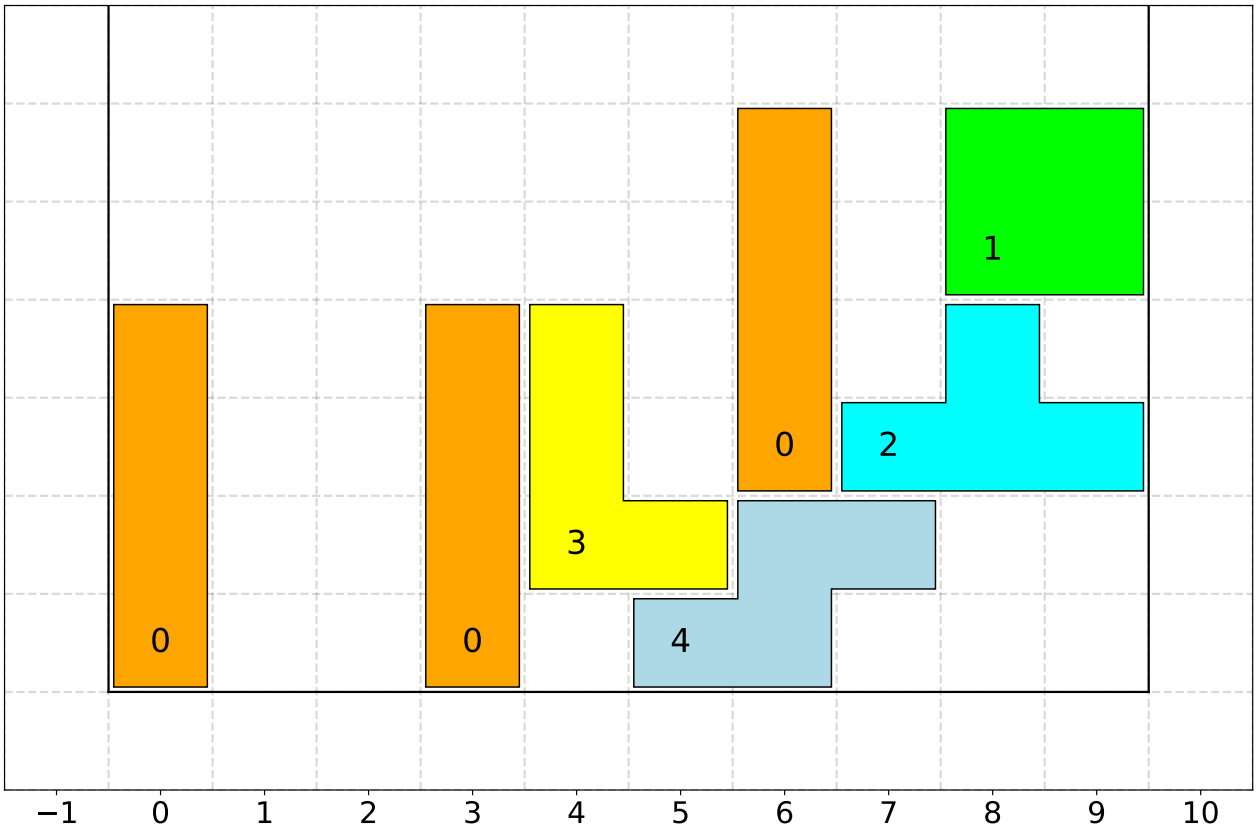
Das Tetromino der Sorte 0 mit vorgegebener x-Position 3 landet auf y-Position 0:



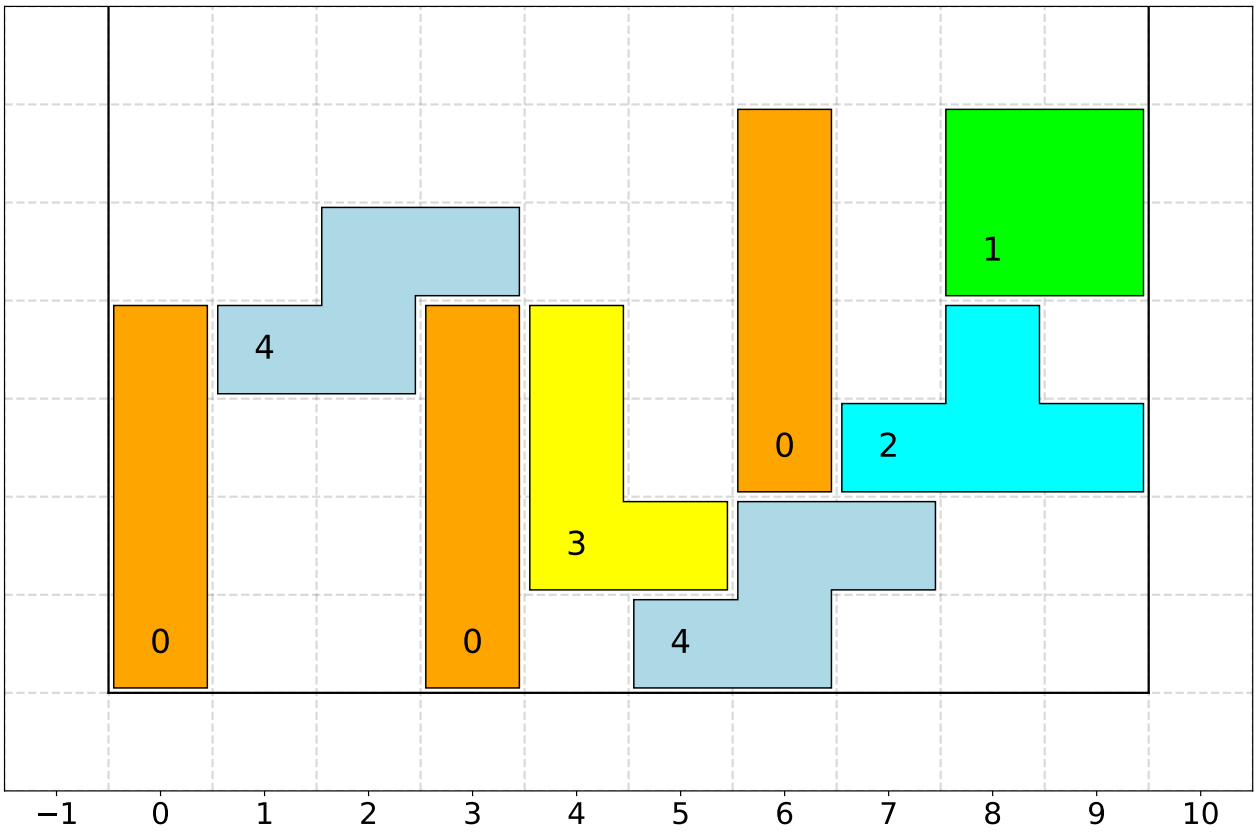
Das Tetromino der Sorte 0 mit vorgegebener x-Position 6 landet auf y-Position 2:



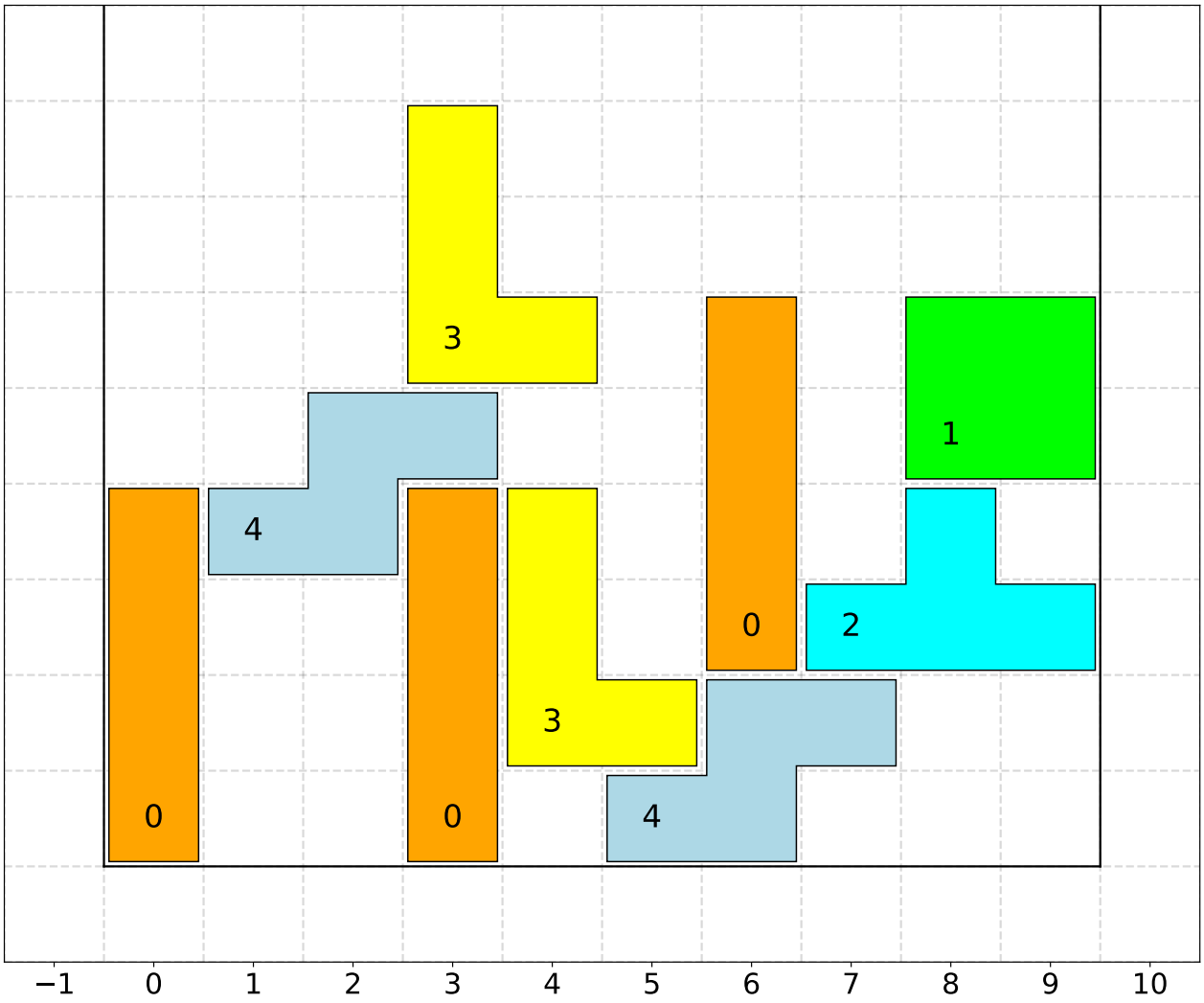
Das Tetromino der Sorte 1 mit vorgegebener x-Position 8 landet auf y-Position 4:



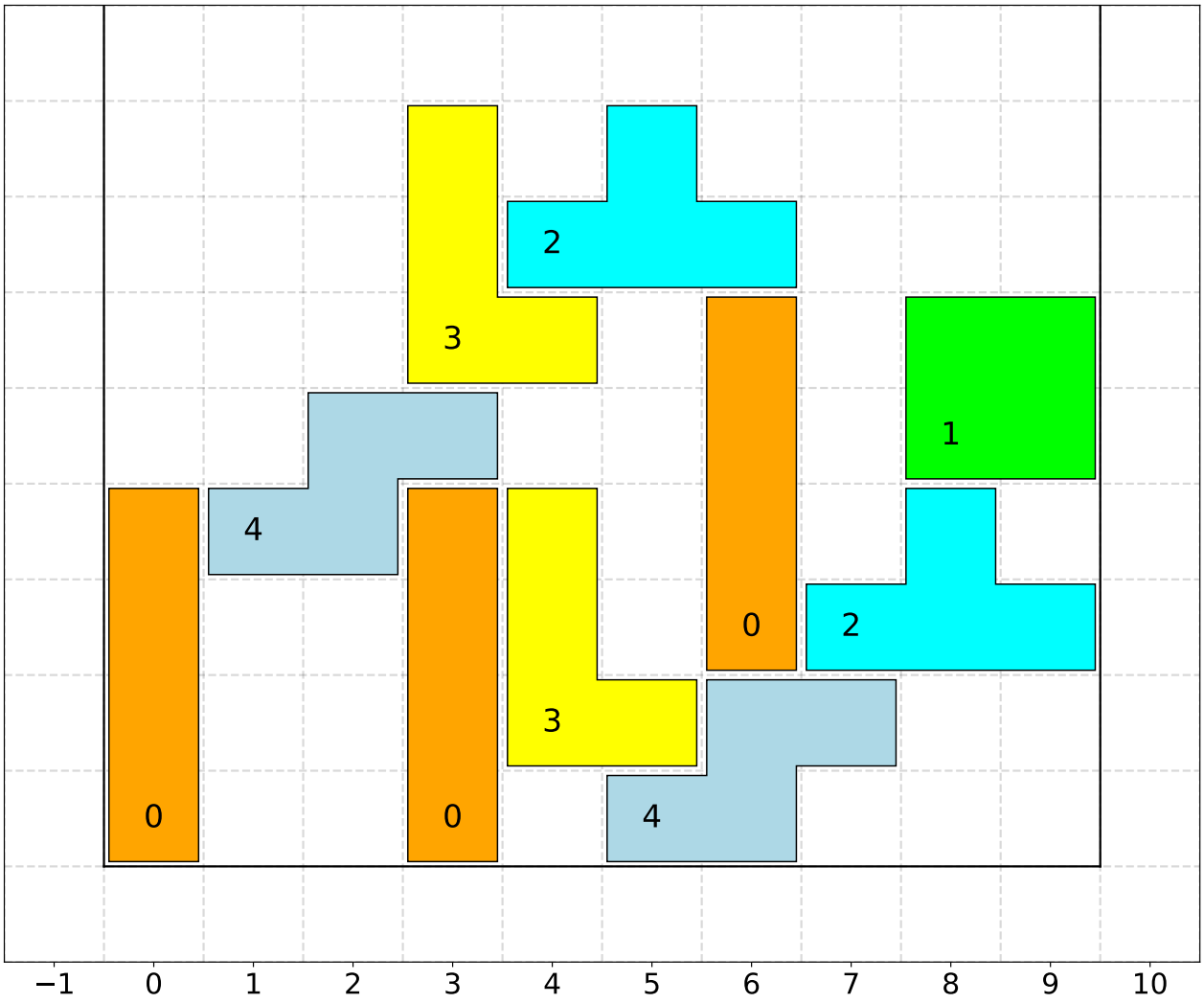
Das Tetromino der Sorte 4 mit vorgegebener x-Position 1 landet auf y-Position 3:



Das Tetromino der Sorte 3 mit vorgegebener x-Position 3 landet auf y-Position 5:



Das Tetromino der Sorte 2 mit vorgegebener x-Position 4 landet auf y-Position 6:



Erstellen des Programms: Nennen Sie Ihr Programm `tetrominos`. Wir haben Ihnen ein Makefile zur Verfügung gestellt, mit dem Sie Ihr Programm kompilieren, ausführen und testen sollen.

```
4 5
0 0
3 4
2 7
0 3
0 6
1 8
4 1
3 3
2 4
```

Prüfen des Programms: Prüfen Sie mit `make diff`, ob Ihr Programm die richtige Ausgabe produziert, und mit `make valgrind`, ob die Speicherverwaltung fehlerfrei ist. Mit `make run` können Sie beides überprüfen. Wir haben diese targets im Makefile bereits für Sie erstellt.

Eingabe: Die Eingabe erfolgt über eine Text-Datei mit zwei Spalten und jeweils einer Zeile pro Tetromino. In jeder Zeile steht zuerst die Tetromino-Sorte (0-4), dann folgt ein Leerzeichen, die x-Position (0-9) und schließlich ein Zeilenumbruch (`\n`).

Ausgabe: Die Ausgabe des Spielfelds erfolgt ebenfalls über eine Text-Datei. Hierbei soll für jede Koordinate des Spielfelds ausgegeben werden, welche Sorte von Tetromino sich dort befindet, oder ein Leerzeichen, wenn dort nichts ist.
Jede Zeile der Ausgabe soll exakt die Breite des Spielfelds haben, gefolgt von einem Zeilenumbruch. Weiterhin soll die Ausgabe so hoch sein, dass alle Spielsteine enthalten sind, aber nicht höher (den abschließenden Zeilenumbruch ausgenommen).
Auf der rechten Seite sehen Sie die Textausgabe für das vorherige Beispiel.

```
   3 2
   3222
   33 0 11
  44  0 11
04403 0 2
0  03 0222
0  03344
0  0 44
```

Das Programm soll im Fehlerfall eine kurze, aussagekräftige Meldung ausgeben und folgende **Rückgabewerte** haben:

- 0, wenn alles ordnungsgemäß funktioniert.
- 1, wenn nicht genau zwei zusätzliche Parameter übergeben werden.
- ein beliebiger Wert größer 1, wenn
  - die Ein- oder Ausgabedatei nicht geöffnet werden kann,
  - die Tetrominos außerhalb des Spielfelds eingefügt würden,
  - es Fehler beim Lesen oder Schreiben von Dateien gab,
  - die Eingabedatei nicht dem vorgegebenen Format entspricht.

Aufruf des Programms: Das Programm bekommt beim Aufruf die Namen der Eingabe- und Ausgabedatei übergeben. Dabei steht die Eingabedatei immer vor der Ausgabedatei:

```
./tetrominos eingabe.txt ausgabe.txt
```

Dateien: Alle Dateien (Testdateien, Makefile, Skript zur Überprüfung) für diese Aufgabe finden Sie im ILIAS.

Wichtige Anforderungen an das Programm:

- Der Speicher ihre Datenstrukturen muss dynamisch verwaltet werden.
- Insbesondere darf dieser also nicht statisch alloziert werden.
- Vor Beendigung des Programms muss der gesamte Speicher wieder freigegeben werden, und geöffnete Dateien müssen geschlossen werden.
- Valgrind darf weder Speicherzugriffsverletzungen noch Speicherlecks oder sonstige Fehler melden.
- Alle Funktionen sowie gegebenenfalls Datenstrukturen sollen in einer separaten Headerdatei deklariert werden.
- Ihr Algorithmus soll mit den von Ihnen allozierten Speicherbereichen arbeiten, und Dateizugriffe nur für das Einlesen in und Schreiben aus dem Speicher verwenden. Ihr Programm darf nie mehr als eine Datei gleichzeitig geöffnet haben, und die Ausgabe-Datei darf nur einmal geöffnet und geschlossen werden.
- Das Programm muss alle Tests fehlerfrei bestehen, d.h. `make diff` und `make valgrind` dürfen keine Fehlermeldung liefern.
- Ein Testdurchlauf für alle Testdateien darf nicht länger als fünf Minuten dauern. Selbst eine mittelmäßige Implementierung benötigt allerdings nur Sekundenbruchteile für `make diff` und einige Sekunden für `make valgrind`.
- Achten Sie auf die Einhaltung der Coderichtlinien. Insbesondere müssen Sie Ihren Code in Funktionen gliedern!
- Zum Bestehen müssen Sie Ihren Code in einem Einzelgespräch erklären. Vereinbaren Sie dazu mit Ihrem Tutor einen Termin.

Viel Erfolg!