Kriterien der Leistungsbewertung in der Sekundarstufe I

Informatik

Stufenspezifische Kompetenzerwartungen

siehe schulinternes Curriculum

Schriftliche Arbeiten

- 2 Arbeiten pro Halbjahr
- Zeitvorgabe: 45-90 Minuten

Einmal im Schuljahr kann eine Klassenarbeit durch eine andere Form der Leistungsüberprüfung (z.B. Portfolio, Projektarbeit) ersetzt werden.

Klassenarbeiten können auch praktische Programmierarbeiten enthalten.

Mit der Rückgabe der Klassenarbeit erhalten alle Schülerinnen und Schüler eine Lösung der Aufgabenstellungen in geeigneter Form. Ob darüber hinaus eine Berichtigung anzufertigen ist, entscheidet die jeweilige Fachlehrerin bzw. der jeweilige Fachlehrer.

Auch die Entscheidung, ob und wann eine Schülerin bzw. ein Schüler bei Versäumnis eine Klassenarbeit nachzuholen hat, ist in das Ermessen der Fachlehrerin bzw. des Fachlehrers gestellt.

Sonstige Mitarbeit

Zu den „Sonstigen Leistungen“ zählen beispielsweise:

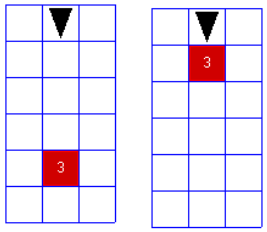
Beiträge zum Unterrichtsgespräch, das Aufzeigen von Zusammenhängen, Präsentieren und Bewerten von Ergebnissen, kooperative Leistungen in Form von Partner- und Gruppenarbeiten, im Unterricht eingeforderte Leistungsnachweise (z. B. vorgetragene Hausaufgaben, Protokolle, Heftführung, Upload von am PC bearbeiteten Aufgaben)

Auch kurze schriftliche Überprüfungen können im Bereich der „Sonstigen Leistungen“ berücksichtigt werden.

Beispiele für Klassenarbeiten

siehe Anhang

1. Robot Karol holt Material



Problem: Karol will einen Stapel Ziegel (3 Stück) bearbeiten. Dazu muss er ihn zuerst holen.

Aufgabe: Schreibe ein einfaches Programm für dieses Problem.

Hinweise: Das Hauptprogramm sollte wegen der besseren Lesbarkeit in die Anweisungen `Programm ... *Programm` eingeschlossen werden.

Denke hier und auch sonst in der Klausur an deutliche Einrückungen und an angemessene Kommentierung deines Programm-Codes.

2. Karol lernt neue Anweisungen

Implementiere (d.h. programmiere) die unten aufgeführten Anweisungen, indem du die bekannte Syntax verwendest (`Anweisung ... *Anweisung`).

a)	`Umdrehen`	Karol dreht sich (um 180°) um.
b)	`LegeZiegelUnten`	Karol legt einen Stein unter sich, seine Blickrichtung ändert sich nicht.
c)	`SchrittRueckwaerts`	Karol geht einen Schritt zurück, seine Blickrichtung ändert sich nicht.
d)	`GeheBisZiegel`	Karol geht vorwärts, bis er vor einem Ziegelstein steht.
e)	`HebeStapelAuf`	Karol hebt alle Ziegel auf, die vor ihm liegen, egal wie viele es sind.
f)	`SchritteRueckwaerts(X)`	wie c) X steht für die Zahl der Schritte

3. Eine Programm-Bibliothek benutzen

Schreibe das Programm aus Aufg. 1 eleganter, indem du die neuen Anweisungen benutzt.

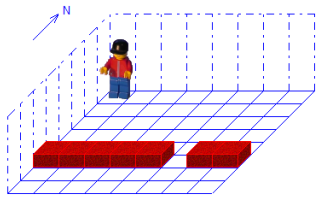
Sie sind in der Datei `KlausurBibliothek.kdp` abgespeichert, die mit

`Einfügen ... *Einfügen`

eingebunden werden muss.

Benutze die Befehle aus der Bibliothek (Aufg. 2) auch in den nächsten Aufgaben, wenn möglich...

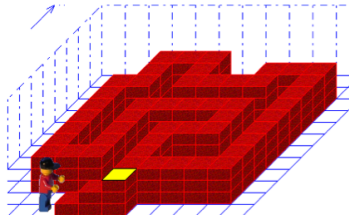
5. Loch in der Mauer



Programmiere Karol so, dass er den Durchgang sucht und findet und bis zur südlichen Wand läuft.

6. Karol auf Wache

Karol soll einen Burgwall ablaufen, ohne herunterzufallen. Dazu muss er bei jedem Schritt prüfen, in welche Richtung es weiter geht. Er darf nicht fallen!



Zunächst steigt er die Treppe hoch, deren Ende markiert ist. Zur Sicherheit vor einem Überfall baut er hinter sich die Treppe ab.

Teile die Aufgabe in sinnvolle kleinere Aufgaben.

Benutze die Kontrollstrukturen

```
wiederhole immer ... *wiederhole
wenn ... dann ... sonst ... *wenn
```

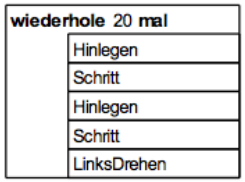
und den Sensor

```
IstMarke
```

7. Struktogramme verstehen / umsetzen

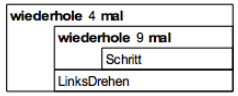
Schreibe den Programm-Code für die folgenden Struktogramme und beschreibe klar und deutlich, was die Programme tun.

a)



b)

Hauptprogramm



c)

Hauptprogramm

w=wahr, f=falsch

		max	erreicht:
Aufg. 1	*Robot Karol holt Material* Dein Programm sorgt dafür, dass Karol die Teilprobleme löst: Weg zum Stapel zurücklegen, Stapel aufnehmen, den Stapel an seinem Ziel aufbauen, Ausgangsposition wieder einnehmen. Bonus: Du hast die einzelnen Teilprobleme im Programmcode deutlich gemacht (z.B. durch Anweisungen oder durch Kommentierung)	***3***	
Aufg. 2	*Karol lernt neue Anweisungen* Du hast geeignete Anweisungen programmiert, die die gestellten Probleme lösen (je 2P.). Bonus: Du greifst auf bereits programmierte Anweisungen zurück (z.B. Umdrehen), du benutzt unterschiedliche Schleifen-Typen.	***12***	
Aufg. 3	*Eine Programm-Bibliothek benutzen* Du wendest die neuen Anweisungen aus Aufg. 2 richtig an. Dabei findest du eine möglichst einfache, elegante Lösung und vermeidest überflüssige Schritte.	***3***	
Aufg. 4	*Loch in der Mauer* Dein Programm sorgt (mithilfe Wiederholungsschleifen) dafür, dass Karol die Teilprobleme löst: Weg zur Mauer zurücklegen, Durchgang suchen, durch die Lücke in der Wand laufen. wichtig: die Wegstrecke darf nicht vorgegeben werden (vgl. Aufgabe) Bonus: - Du hast die einzelnen Teilprobleme im Programmcode deutlich gemacht (z.B. durch Anweisungen oder zumindest durch Kommentierung) - Du hast Anweisungen aus KlausurBibliothek.kdp benutzt	***6***	
Aufg. 5	*Karol auf Wache* Dein Programm sorgt dafür, dass Karol die Teilprobleme löst: Treppe hochlaufen und hinter sich abbauen, Weg auf der Mauer zurücklegen (mit Wiederholungsschleife und Prüfung, wo der Weg weitergeht) wichtig: die Wegstrecke darf nicht vorgegeben werden (vgl. Aufgabe) - Du hast die einzelnen Teilprobleme im Programmcode deutlich gemacht (z.B. durch Anweisungen oder zumindest durch Kommentierung) - Du hast Anweisungen aus KlausurBibliothek.kdp benutzt	***6***	
Aufg. 6	*Struktogramme verstehen / umsetzen* - Der Programmcode ist korrekt (je 1P.). - Die Erläuterungen sind zutreffend, genau und gut verständlich a) Karol baut einen Turm in Quaderform (3x3x5), der in der Mitte hohl ist. b) Karol geht auf dem Rand eines 10x10 Felder großen Quadrats entlang c) Karol überprüft, ob er vor einem Ziegel steht. Falls ja, nimmt er ihn und alle weiteren Steine, die dort liegen mögen, auf. Falls nein, bewegt er sich zum nächsten Ziegel.	***9***	
Du hältst dich an gängige Programmier-Konventionen: konsequentes Einrücken von Blöcken, ausführliche Kommentierung		***4***	
ggf. Abzüge für die äußere Form, z.B.: Rand nicht ausreichend oder nicht eingehalten, fehlende Übersichtlichkeit / Ordnung, Streichungen ohne Lineal, o.Ä.; sprachliche Mängel			
Summe		***47***	

Bereich: Programmieren

Kopiere dir zuerst den ganzen Ordner mit den Klausuraufgaben (`M:\IF-Klausur`) in den Ordner `Arbeitsergebnisse` auf deinem Desktop. Von dort kannst du die einzelnen Dateien öffnen, und dort sollen sie auch gespeichert werden.

Löse möglichst viele Teilschritte der folgenden Aufgaben.
Achtung: die Schwierigkeit nimmt mit den Teilschritten zu!

Speichere jeweils nur eine Datei, nämlich die am weitesten entwickelte Version deines Programms.

Aufgabe 1 - `A1_Quadrat.pde`

Dein Programm zeichnet ein weiß gefülltes Quadrat mit Kantenlänge 25px, wobei

- ☐ das Quadrat dem Mauszeiger folgt,
- ☐ keine Spuren hinterlässt,
- ☐ es sich blau einfärbt, wenn eine Maustaste gedrückt ist,
- ☐ sich das Quadrat wieder weiß einfärbt, wenn keine Maustaste gedrückt ist

Aufgabe 2 - `A2_Smiley.pde`

Dein Programm zeichnet einen Smiley (s. rechts).

Durchmesser: 100px

Strichstärke:
10px für Augen u. Mund

Arbeite das Programm folgendermaßen aus:

- ☐ der Smiley ist gelb gefüllt (*Tipp*: Tools ➟ Color Selector)
- ☐ der Smiley wird durch eine eigene Methode `void drawSmiley()` erzeugt und dann im Hauptprogramm void draw () aufgerufen.
- ☐ der Smiley wird durch die eigene Methode `void drawSmiley(int x, int y)` aufgerufen, wobei x und y die Koordinaten des Mittelpunkts angeben.

Aufgabe 3 - `A3_Spielball.pde`

Dein Programm zeichnet einen Kreis mit Durchmesser 25px (ähnlich einem Spielball),

- ☐ der an einer festen Stelle im Fenster erscheint,
- ☐ der an einer zufälligen Stelle im Fenster erscheint,
- ☐ der sich zu Beginn nach rechts bewegt,
- ☐ der sich zu Beginn nach rechts unten bewegt,
- ☐ der von den Rändern abprallt,
- ☐ der dabei nicht zur Hälfte über den Rand herausragt,
- ☐ der seine Geschwindigkeit verändert, wenn man die Tasten '+' oder '-' drückt.

Bereich: Fehler finden, Probleme im Programmcode beheben

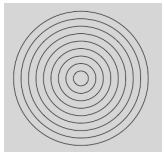
Öffne den jeweiligen Programmcode als Datei. Lies dort, was das Programm machen soll. Behebe den Fehler und speichere eine funktionierende Version ab.

Aufgabe 4: Farbverlauf - `A4_Farbverlauf.pde`

gewünschte Bildschirmausgabe:

Aufgabe 5: Zielscheibe - `A5_Zielscheibe.pde`

gewünschte Bildschirmausgabe:



Bereich: Umgang mit Variablen

Löse die folgenden Aufgaben auf dem Blatt!

Aufgabe 6:

Welche der folgenden Deklarationen ist/sind korrekt? Kreuze an!

a) `float zahl;` b) `int zahl = 3.5;` c) `int x > 22;`

d) `String message = "Hallo";`

Aufgabe 7:

Welcher Wert steht nach Ausführung der folgenden Programmzeilen in der Variable b?

```
int a = 0;
int b = 5;
a = a + 3;
b = a * 2 + b;
b = 10 - b;
```

Antwort: `b` hat den Wert _____ .

Viel Erfolg!