

1 Variablen und Ausgabe

1.1 Beschreiben Sie *in Ihren eigenen Worten*, was eine Variable in der Mathematik ist?

1.2 Beschreiben Sie *in Ihren eigenen Worten*, was eine Variable beim Programmieren ist? Wie unterscheidet sie sich von einer Variablen in der Mathematik?

Betrachten Sie den rechts abgebildeten Quelltext.

1.3 Was gibt dieser Quelltext aus?

```
1 int var;  
2 var = 5;  
3 var = 7;  
  
4 printf("%i", var);  
5 printf("%i", var);  
6 printf("%i", var);
```

1.4 Hat die Variable `var` nach dem Ausführen von Zeile 3 *ausschließlich den Wert 5*, hat sie *ausschließlich den Wert 7*, oder hat sie *sowohl den Wert 5 als auch 7*? Begründen Sie.

1.5 Welchen Wert hat die Variable nach Ausführen von Zeile 5? Begründen Sie.

→ Überprüfen Sie Ihre Antworten, indem Sie den Quelltext kompilieren und ausführen. Hierfür können Sie die Website <https://ideone.com/fork/HY1kYl> nutzen. Mit dem rechts abgebildeten QR-Code gelangen Sie direkt zu dieser Seite. Diskutieren Sie alternativ Ihre Antworten mit einer Tutorin oder einem Tutor.



Betrachten Sie den folgenden Quelltext.

```
1 int var = -5;  
2 if (var > 0) {  
3     printf("Variable ist positiv.");  
4 }  
  
5 var = 10;  
6 printf("Programm zu Ende.");  
7 if (var > 0) {  
8     printf("Variable ist positiv.");  
9 }
```

1.6 Was wird ausgegeben? Begründen Sie.

Lesen Sie die folgende Diskussion dreier Studierender zum Quelltext auf der vorigen Seite.

- Alice : *Zwar hat var zunächst den Wert -5, die Ausgabe ist aber trotzdem »Variable ist positiv.Programm zu Ende.Variable ist positiv.« ,If‘ bedeutet übersetzt ‚wenn‘, d.h. wenn die Variable positiv wird, wird »Variable ist positiv.« ausgegeben. Die Änderung der Variable in Zeile 5 sorgt dafür, dass Zeile 3 nachträglich ausgeführt wird.*
- Bob : *„Nein, das Programm wird von oben nach unten ausgeführt. Es wird nie zurückgesprungen. Das Programm gibt »Programm zu Ende.Variable ist positiv.« aus. Es ist aber sicher schlechter Programmierstil, dass zuerst gesagt wird, das Programm sei zu Ende, dann aber noch etwas ausgeführt wird.“*
- Carol : *„Das liegt bestimmt daran, dass die Ausgabe hier auf Deutsch passiert. Die Befehle wie ‚if‘ sind ja alle auf Englisch. Hätte da »End of Program.« anstatt »Programm zu Ende.« gestanden, dann wäre das Program auch in Zeile 6 geendet.“*

1.7 Welchen Aussagen stimmen Sie zu, welchen nicht? Begründen Sie.

→ Überprüfen Sie Ihre Antworten, indem Sie den Quelltext kompilieren und ausführen. Hierfür können Sie die Website <https://ideone.com/fork/pLYiho> nutzen. Mit dem rechts abgebildeten QR-Code gelangen Sie direkt zu dieser Seite. Diskutieren Sie alternativ Ihre Antworten mit einer Tutorin oder einem Tutor.



2 Schleifen und Ausgabe

Betrachten Sie den rechts abgebildeten Quelltext.

2.1 Was gibt dieser Quelltext aus?

```

1 int zaehlvariable=0;
2 while (zaehlvariable<5) {
3     printf("%i\n", zaehlvariable);
4 }
```

Lesen Sie die folgende Diskussion dreier Studierender zum Quelltext.

- Alice : *In der Schleife, also in Zeile 3 wird der aktuelle Wert der Variable zaehlvariable ausgegeben. zaehlvariable ist zu Anfang 0 und die while-Schleife zählt bis 5. Es müsste also »012345« ausgegeben werden.*
- Bob : *„Ich denke hier handelt es sich um eine Endlosschleife. zaehlvariable wird nie erhöht, sondern bleibt immer beim Wert 0. Folglich wird unendlich oft 0 ausgegeben.“*
- Carol : *„Aber die while-Schleife soll ja zählen solange zaehlvariable<5 ist. Folglich ist logisch, dass zaehlvariable erhöht werden soll. Es wäre ja dumm, wenn der Compiler dann nicht dafür sorgen würde, dass zaehlvariable auch erhöht wird.“*

2.2 Welchen Aussagen stimmen Sie zu, welchen nicht? Begründen Sie.

→ Überprüfen Sie Ihre Antworten, indem Sie den Quelltext kompilieren und ausführen. Hierfür können Sie die Website <https://ideone.com/fork/sCMROm> nutzen. Mit dem rechts abgebildeten QR-Code gelangen Sie direkt zu dieser Seite. Diskutieren Sie alternativ Ihre Antworten mit einer Tutorin oder einem Tutor.



3 Funktionen und Ausgabe

3.1 Formulieren Sie einen Satz, der beschreibt, was der Befehl `return` macht.

3.2 Darf der Befehl `return` mehrfach in einer Funktion auftreten? Begründen Sie Ihre Antwort.

Betrachten Sie den rechts abgebildeten Quelltext.

3.3 Werden beim Kompilieren dieses Quelltextes Fehlermeldungen auftreten? Wenn ja, welche?

3.4 Was wird ausgegeben, wenn dieser Quelltext (und somit die Funktion `main`) ausgeführt wird? Überspringen Sie alle Stellen, die nicht kompilieren würden.

3.5 Erläutern Sie, wie sich die Funktionen `a` und `b` unterscheiden.

3.6 Erläutern Sie, wie sich die Funktionen `c` und `d` unterscheiden.

```

1  int a() {
2      printf("%i", 1);
3      printf("%i", 2);
4  }

5  int b() {
6      return 3;
7      return 4;
8  }

9  int c() {
10     printf("%i", 5);
11     return 6;
12 }

13 int d() {
14     return 7;
15     printf("%i", 8);
16 }

17 int main() {
18     printf("%i", d());
19     printf("%i", c());
20     printf("%i", b());
21     printf("%i", a());
22 }
    
```

3.7 In welcher Reihenfolge werden die Funktionen ausgeführt? Ist es richtig zu sagen, dass die Funktion `main` vor allen anderen Funktionen ausgeführt wird? Begründen Sie Ihre Antwort.

→ Überprüfen Sie Ihre Antworten, indem Sie den Quelltext kompilieren und ausführen. Hierfür können Sie die Website <https://ideone.com/fork/WbSdk3> nutzen. Mit dem rechts abgebildeten QR-Code gelangen Sie direkt zu dieser Seite. Diskutieren Sie alternativ Ihre Antworten mit einer Tutorin oder einem Tutor.



Die Antworten auf die Fragen auf dieser und der nächsten Seite unterschieden sich zwischen verschiedenen Programmiersprachen und Compilern bzw. Compileroptionen. Überprüfen Sie ggf. Ihre Antworten mit verschiedenen Compilern. Sollten Sie mehr als eine Programmiersprache beherrschen bzw. lernen, lohnt es sich, diese Aufgaben auch für die anderen Sprachen zu lösen und die Antworten zu vergleichen.

4 Programmfluss

Betrachten Sie die folgenden vier Varianten einer Funktion, die den Betrag der übergebenen Variable zurückgibt.

Variante 1:

```

1 double betrag(double a) {
2     return a;
3     if (a < 0) {
4         a = -a;
5     }
6 }
```

Variante 2:

```

1 double betrag(double a) {
2     if (a < 0) {
3         a = -a;
4     }
5     return a;
6 }
```

Variante 3:

```

1 double betrag(double a) {
2     if (a < 0) {
3         return -a;
4     }
5     return a;
6 }
7
```

Variante 4:

```

1 double betrag(double a) {
2     if (a < 0) {
3         return -a;
4     } else {
5         return a;
6     }
7 }
```

- 4.1 Betrachten Sie Ihren Antwort auf Frage 3.2 erneut und passen Sie diese ggf. an.
- 4.2 Sammeln Sie in der folgenden Tabelle Vor- und Nachteile der 4 Varianten der Funktion betrag. Wird bei allen Varianten der Funktionen der Betrag zurück gegeben? Welche sind leicht zu lesen, welche nicht?

	Variante 1	Variante 2	Variante 3	Variante 4
Vorteile				
Nachteile				

- 4.3 Betrachten Sie die folgende Diskussion dreier Studierender. Welche Aussagen sind richtig, welche nicht?

- Alice : *„Variante 2 der Funktion betrag ist am besten geschrieben, da sie nur ein einziges return ganz am Ende hat. Dadurch ist leicht erkennbar, welche Variable zurückgegeben wird.“*
- Bob : *„Ich finde das in Variante 1 noch besser umgesetzt. Am Anfang der Funktion ist das return noch leichter zu sehen.“*
- Carol : *„Ich finde Variante 4 am besten. Diese Variante gleicht am ehesten der mathematischen Definition der Betragsfunktion.“*

- 4.4 Wenn die Varianten der betrag-Funktion das Gleiche machen, ist es dann egal welche Sie verwenden? Welche Gründe gibt es, sich für eine bestimmte Variante zu entscheiden?

Dieses Tutorial behandelt einfache und verschachtelte Schleifen. Um die Beispiele kurz und einheitlich zu halten, werden hier ausschließlich for-Schleifen verwendet. Alles in diesem Tutorial Diskutierte lässt sich auch auf while-Schleifen anwenden. Die Verwendung und Unterschiede zwischen for- und while-Schleifen werden im Tutorial *Schleifenarten* diskutiert.

1 Einfache Schleifen

Betrachten Sie nachfolgenden Quelltext.

```

1  for (int i = 1; i < 5; i++) {
2      printf("%i", i);
3      int quadrat = i * i;
4      printf(" zum Quadrat ist %i\n", quadrat);
5  }
```

1.1 Was gibt dieses Programm aus?

1.2 Formulieren Sie *einen* Satz, der beschreibt, was Zeile 3 „macht“.

1.3 Formulieren Sie *einen* Satz, der beschreibt, was dieses Programm „macht“.

1.4 Ist es grundsätzlich immer möglich einen Satz zu formulieren, der beschreibt was ein Programm „macht“?

Die folgende Abbildung zeigt, wie ein Student mit einem sog. „Handbogen“ angefangen hat den oben abgebildeten Quelltext nachzuverfolgen. Liest man die Spalten unter „Schritt“ von links nach rechts, sieht man in welcher Reihenfolge die Zeilen des Programms ausgeführt werden. Mit Schritt 5 wird z. B. das Ende der Schleife erreicht, weswegen in Schritt 6 erneut Zeile 1 ausgeführt wird. Unter dem Quelltext sind die Variablen angegeben. Wird in einem Schritt einer Variablen ein Wert zugewiesen, wird dies im Raster unten eingetragen. So wird z. B. in Schritt 1 der Variablen i der Wert 1 zugewiesen. Die Ausgabe des Programms ist ganz unten eingetragen.

		Quelltext	Schritt																			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Zeile	1	for (int i = 1; i < 5; i++) {	X					X														
	2	printf("%i", i);		X					X													
	3	int quadrat = i * i;			X					X												
	4	printf(" zum Quadrat ist %i\n", quadrat);				X					X											
	5	}					X															
Variable	i		1					2														
	quadrat			1						4												
Ausgabe			Notizen																			
1 zum Quadrat ist 1 2 zum Quadrat ist 4																						

1.5 Vervollständigen Sie den oben abgedruckten Handbogen.

1.6 Stimmen Ihre Antworten auf Fragen 1.2 und 1.3 mit dem überein, was Sie beim Nachverfolgen der Programmausführung beobachtet haben? Passen Sie Ihre Antworten ggf. an.

2 Verschachtelte, Unabhängige Schleifen

Sehen Sie sich den Quelltext rechts an.

2.1 Was wird ausgegeben? Begründen Sie Ihre Antwort.

```

1 for (int y=1; y<=2; y++) {
2     for (int x=1; x<=3; x++) {
3         printf("0");
4     }
5     printf("\n");
6 }
    
```

2.2 Ein Student beschreibt die Ausführung dieses Quelltextes wie folgt:

Die Schleife, die in Zeile 1 beginnt, wird zweimal ausgeführt. Als erstes in der Schleife steht eine weitere Schleife (Zeilen 2 bis 4). Diese innere Schleife wird dreimal ausgeführt. Jedes mal wird eine '0' ausgegeben. Folglich werden zuerst zweimal drei, also sechs '0' ausgegeben. Danach kommt der Befehl in Zeile 5, der einen Zeilenumbruch ausgibt. Wegen der Schleife, die in Zeile 2 beginnt, wird auch dieser zweimal ausgeführt, sodass zwei Zeilenumbrüche nacheinander ausgegeben werden.

- a. Stimmen Sie seiner Lösung zu?
- b. Erläutern Sie, wie die Lösung des Studenten zu ihrer Lösung von Aufgabe 2.1 passt.

2.3 Füllen Sie einen Handbogen aus und überprüfen Sie so ihre Lösung. Einen leeren Handbogen finden Sie am Ende dieses Tutorials.

2.4 Formulieren Sie *einen* Satz, der beschreibt, was die gesamte Schleife in Zeilen 2 bis 4 „macht“. Dieser Satz sollte auch die Veränderungen der Zählvariable (x) beschreiben.

2.5 Formulieren Sie *einen* Satz, der beschreibt, was die gesamte Schleife in Zeilen 2 bis 4 „macht“. Dieser Satz soll die Veränderungen der Zählvariable (x) weder direkt noch indirekt enthalten.

2.6 Formulieren Sie *einen* Satz, der beschreibt, was der Befehl in Zeile 5 „macht“.

2.7 Vergleichen Sie die Sätze, die Sie in 2.5 und 2.6 formuliert haben. Worin unterscheiden sich die zwei Sätze? Worin ähneln sie sich?

2.8 Formulieren Sie *einen* Satz, der beschreibt, was der gesamte Quelltext „macht“.

→ Besprechen Sie Ihre Antworten mit einer Tutorin oder einem Tutor.

3 Verschachtelte, Abhängige Schleifen

Sehen Sie sich den Quelltext rechts an.

3.1 Was wird ausgegeben? Begründen Sie Ihre Antwort.

```

1 for (int y=1; y<=3; y++) {
2     for (int x=y; x<=3; x++) {
3         printf("0");
4     }
5     printf("\n");
6 }
```

3.2 Lesen Sie die folgende Diskussion zwischen drei Studierenden. Welchen Aussagen würden Sie zustimmen, welchen nicht?

Alice : *„Eine for-Schleife wird verwendet, wenn etwas mit einer bestimmten Anzahl wiederholt werden soll. In diesem Fall zählen beide Schleifen bis drei. Die äußere Schleife zählt die Reihen und die innere die '0'en in jeder Reihe. Deshalb wird eine Quadrat von 3x3 '0'en ausgegeben.“* $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Bob : *„Ich stimme dir zu: for-Schleifen sind dazu da, eine Aktion mit einer gewissen Anzahl zu wiederholen. Wie oft dabei die innere Schleife ausgeführt wird hängt allerdings von der äußeren Schleife ab. Jedes Mal, wenn die innere Schleife aufgerufen wird, wird die Variable x auf den aktuellen Wert von y gesetzt. Deshalb wird ein Dreieck von '0'en ausgegeben.“* $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \\ 0 & 0 & \end{pmatrix}$

Carol : *„Aber beide Schleifen erhöhen ihre Zählvariable doch mit dem '++'-Befehl. Also muss in den unteren Reihen, in denen y größer ist auch x größer sein. Das Dreieck muss also unten breiter sein als oben.“* $\begin{pmatrix} 0 & & & & \\ 0 & 0 & & & \\ 0 & 0 & 0 & & \end{pmatrix}$

3.3 Verwenden Sie einen Handbogen um den Quelltext nachzuverfolgen und Ihre Antworten zu überprüfen. Einen leeren Handbogen finden Sie am Ende dieses Tutorials.

3.4 Im Folgenden soll die Tabelle rechts ausgefüllt werden, um die Werte der beiden Zählvariablen x und y zu visualisieren.

a. Was ist der erste Wert, der y zugewiesen wird?

b. Was ist der erste Wert, der x zugewiesen wird? Markieren Sie die Kombination aus den beiden Anfangswerten der Zählvariablen, indem Sie in der Tabelle ein „A“ an die entsprechende Position eintragen.

		x				
		0	1	2	3	4
y	0					
	1					
	2					
	3					
	4					

c. Wie verändern sich die Zählvariablen, wenn das Programm weiter ausgeführt wird? Fügen Sie jedes mal wenn x ein neuer Wert zugewiesen wird einen weiteren Buchstaben („B“, „C“,...) in die Tabelle ein.

d. Wie passen die Werte aus der Tabelle zu Ihren Antworten der Fragen 3.1 und 3.2?

e. Welche Aspekte Ihrer Tabelleneinträge kann man in der Ausgabe des Quelltextes wiedererkennen und welche nicht?

3.5 Formulieren Sie *einen* Satz, der beschreibt, was die Schleife in Zeilen 2 bis 4 „macht“. Gehen Sie dabei auf die Werte der Zählvariable (x) ein.

3.6 Formulieren Sie *einen* Satz, der beschreibt, was die Schleife in Zeilen 2 bis 4 „macht“. Gehen Sie dabei weder direkt noch indirekt auf die Werte der Zählvariable (x) ein.

3.7 Füllen Sie die folgende Tabelle aus. Geben Sie darin an, wie viele '0' en durch die Schleife (Zeilen 2 bis 4) ausgegeben werden, je nachdem welchen Wert die Variable y bei Beginn der Schleife hat.

Wert von y	-1	0	1	2	3	4	5
Ausgabe							

3.8 Überprüfen Sie, ob der von Ihnen in 3.6 formulierte Satz die in 3.7 beschriebene Abhängigkeit der Ausgabe von der Variable y enthält. Passen Sie den Satz gegebenenfalls an.

3.9 Vergleichen Sie Ihren Satz mit dem in Aufgabe 2.5.

4 Zählreihenfolgen

Sehen Sie sich den Quelltext rechts an.

4.1 Was wird durch diesen Quelltext ausgegeben? Begründen Sie Ihre Antwort.

```

1 for (int y = 1; y <= 3; y++) {
2     for (int x = 3; x >= y; x--) {
3         printf("0");
4     }
5     printf("\n");
6 }
    
```

4.2 Formulieren Sie *einen* Satz, der beschreibt, was die Schleife in Zeilen 2 bis 4 macht. Gehen Sie dabei weder direkt noch indirekt auf die Werte der Zählvariable (x) ein.

4.3 Vergleichen Sie Ihre Sätze aus 3.6 und 4.2.

4.4 Verwenden Sie einen Handbogen, um den Quelltext nachzuverfolgen und Ihre Antworten zu überprüfen. Einen leeren Handbogen finden Sie am Ende dieses Tutorials.

4.5 Halten Sie die in 3.6 beschriebene oder die in 4.2 beschriebene Schleife für leichter lesbar? Begründen Sie.

4.6 Ist es egal, welche der beiden Schleifen Sie verwenden?

Handbogen

		Schritt																																							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
	1	Quelltext																																							
	2																																								
	3																																								
	4																																								
	5																																								
	6																																								
	7																																								
	8																																								
	9																																								
	10																																								
	11																																								
	12																																								
	13																																								
	14																																								

Variable	

Ausgabe	
Notizen	

Handbogen

		Schritt																																							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
	1	Quelltext																																							
	2																																								
	3																																								
	4																																								
	5																																								
	6																																								
	7																																								
	8																																								
	9																																								
	10																																								
	11																																								
	12																																								
	13																																								
	14																																								

Variable	

Ausgabe	
Notizen	

1 Verschiedene Schleifenarten

Betrachten Sie die folgenden zwei Schleifen.

```

1 for (int x=0; x<5; x++) {
2     printf("%i\n", x);
3 }
4
5
```

```

1 int x = 0;
2 while (x<5) {
3     printf("%i\n", x);
4     x++;
5 }
```

1.1 Was geben die beiden Schleifen aus?

1.2 Zählen Sie auf, worin sich for- und while-Schleifen unterscheiden.

1.3 Welche Schleife (for oder while) halten Sie in dem hier gezeigten Fall für besser? Begründen Sie.

1.4 In welchen Fällen würden Sie die Schleife verwenden, die Sie bei Frage 1.3 für schlechter hielten?

2 Beispielprogramm

Betrachten Sie den folgenden Quelltext.

```

1 for (int zahl=1; zahl<=5; zahl++) {
2     int test=2;
3     bool teilbar = false;
4     while (test < zahl) {
5         if (zahl % test == 0) {
6             teilbar = true;
7         }
8         test += 1;
9     }
10    if (!teilbar) {
11        printf("%i ist eine Primzahl.\n", zahl);
12    }
13 }
```

2.1 Formulieren Sie einen Satz, der beschreibt, was dieses Programm „macht“.

2.2 Beschreiben Sie, wie dieses Programm funktioniert.

2.3 Nutzen Sie den Handbogen, um den Ablauf des Programmes auf der vorherigen Seite nachzuvollziehen. Passen Sie Ihre Antworten auf Fragen 2.1 und 2.2 ggf. an.

2.4 Schreiben Sie den Quelltext um, sodass die for-Schleife durch eine while-Schleife und die while-Schleife durch eine for-Schleife ersetzt wird. An der Funktion des Programmes soll sich aber nichts ändern.

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	

2.5 Vergewissern Sie sich, dass ihr Programm wie gewünscht funktioniert. Verwenden Sie hierzu einen Handbogen, oder führen Sie das Programm in einer Entwicklungsumgebung aus. Sie können auch einen Tutor um Hilfe bitten. Beheben Sie ggf. vorhandene Fehler in Ihrem Programm.

2.6 Welche Version des Programmes halten Sie für sinnvoller, die Ursprüngliche oder die von Ihnen Erstellte? Begründen Sie.

2.7 Lässt sich grundsätzlich jede for-Schleife durch eine while-Schleife ersetzen?

2.8 Lässt sich grundsätzlich jede while-Schleife durch eine for-Schleife ersetzen?

2.9 Warum gibt es sowohl for-Schleifen als auch while-Schleifen?

3 Der Befehl break

3.1 Beschreiben Sie in Ihren eigenen Worten, was der Befehl break macht.

Nun wird das originale Programm erneut modifiziert, indem der Befehl break eingefügt wird.

```
1 for (int zahl=1; zahl<=5; zahl++) {
2     int test=2;
3     bool teilbar = false;
4     while (test < zahl) {
5         if (zahl % test == 0) {
6             teilbar = true;
7             break;
8         }
9         test += 1;
10    }
11    if (!teilbar) {
12        printf("%i ist eine Primzahl.\n", zahl);
13    }
14 }
```

3.2 Was ändert das eingefügte break am Programm?

3.3 Ist es sinnvoll, diesen Befehl an der Stelle einzufügen?

3.4 Kann dieser Befehl auch in der von Ihnen umgeschriebenen Version des Programms eingefügt werden?

1 Funktionen

1.1 Betrachten Sie den rechts abgebildeten Quelltext.

```

1 double betrag(double x) {
2     if (x < 0) {
3         x = -x;
4     }
5     return x;
6 }
```

a. Formulieren Sie einen Satz, der beschreibt, was Zeilen 2 bis 4 „machen“.

b. Was können Sie über den Wert der Variable x nach Zeile 4 aussagen?

c. Formulieren Sie einen Satz, der beschreibt, was die Funktion `betrag` „macht“.

1.2 Dürfen zwei Variablen mit dem gleichen Namen in einem Programm definiert werden?

Die oben abgebildete Funktion wird nun im rechts abgebildeten Quelltext aufgerufen.

```

1 double x = 7;
2 double y = -4;
3 y = betrag(y);
4 printf("%f\n", x);
5 printf("%f\n", y);
```

1.3 Wird dieser Quelltext eine Fehlermeldung produzieren? Wenn ja, welche? Wenn nein, was gibt er aus?

1.4 Wie viele Variablen werden bei der Ausführung des Quelltextes insgesamt erzeugt bzw. verwendet?

1.5 Handelt es sich bei der Variable x in der Funktion und der Variable x außerhalb der Funktion um die gleiche Variable?

→ Überprüfen Sie Ihre Antworten, indem Sie den Quelltext kompilieren und ausführen. Hierfür können Sie die Website <https://ideone.com/fork/qJLFGx> nutzen. Mit dem rechts abgebildeten QR-Code gelangen Sie direkt zu dieser Seite. Diskutieren Sie alternativ Ihre Antworten mit einer Tutorin oder einem Tutor.



Eine Studentin beobachtet folgendes: *Es scheint so, als ob das, was eine Funktion zurück gibt, nur von den Werten der Variablen, die ihr übergeben werden, abhängt. Außerhalb der Funktion können Variablen mit dem gleichen Namen wie in der Funktion definiert sein, diese haben aber keinen Einfluss auf die Funktion.*

1.6 Stimmen Sie dieser Aussage zu?

2 Funktionen in Funktionen

Betrachten Sie den rechts abgebildeten Quelltext.

2.1 Was gibt dieses Programm aus, wenn es ausgeführt wird? (Sie müssen die Zahl nicht berechnen.)

```

1 double pi() {
2     return 3.1415926535897932;
3 }
4 double kreisA(double r) {
5     return pi() * r*r;
6 }
7 double zylinderV(double r, double h) {
8     return h * kreisA(r);
9 }
10 int main() {
11     printf("%f", zylinderV(5, 10));
12 }
```

Im Folgenden soll das Diagramm unten rechts ausgefüllt werden. Jeder Knoten in diesem Diagramm stellt eine „Instanz“ (sozusagen eine „Ausführung“) einer Funktion mit ihrem Namen und den übergebenen Parametern da. Ein Pfeil nach unten symbolisiert den Aufruf der Funktion und ist mit allen übergebenen Variablen beschriftet. Ein Pfeil nach oben symbolisiert die Beendigung der Funktion und ist mit ihrem Rückgabewert beschriftet.

2.2 Von der Funktion `zylinderV(5,10)` wird die Funktion `kreisA(5)` aufgerufen. Tragen Sie diesen Funktionsaufruf ein.

2.3 Vervollständigen Sie das Diagramm, indem Sie alle Funktionsaufrufe bis `pi()` mit den entsprechenden übergebenen Variablen eintragen.

2.4 Tragen Sie nun an den Pfeilen aufwärts die Rückgabewerte der Funktionen ein.

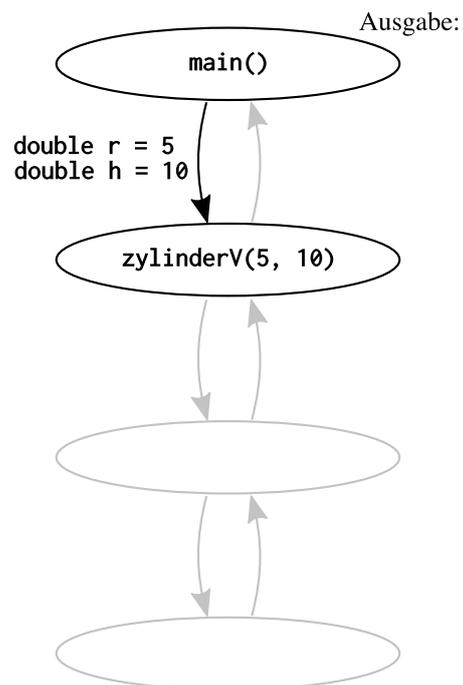
2.5 Tragen Sie die Ausgabe des Programms im Diagramm ein.

2.6 Beginnt die Ausführung der Funktion `kreisA` vor, während oder nach der Ausführung von `zylinderV`?

2.7 Endet die Ausführung der Funktion `kreisA` vor, während oder nach dem Ende von `zylinderV`?

2.8 Wird immer nur eine Funktion zur Zeit ausgeführt oder werden mehrere Funktionen zur gleichen Zeit ausgeführt? Erläutern Sie.

→ Besprechen Sie Ihre Antworten mit einer Tutorin oder einem Tutor.



3 Lineare Rekursion

Betrachten Sie den folgenden Quelltext.

```

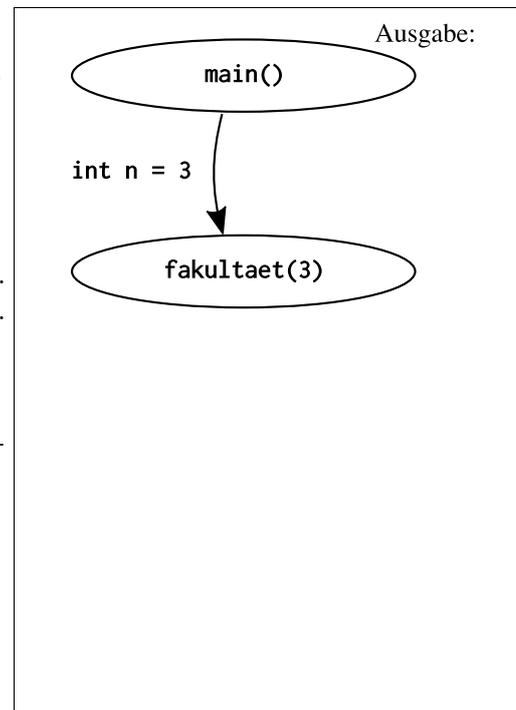
1 double fakultaet(int n) {
2     if (n == 0 || n == 1) {
3         return 1.0;
4     } else {
5         double produkt = n * fakultaet(n-1);
6         return produkt;
7     }
8 }

9 int main() {
10     printf("%f", fakultaet(3) );
11 }
```

3.1 Formulieren Sie einen Satz, der beschreibt, was die Funktion fakultaet(n) macht.

3.2 Im Folgenden soll wieder ein Diagramm aller Funktionsaufrufe erstellt werden. Bereits eingetragen ist der Aufruf der Funktion main() bei Ausführen des Programms sowie der Aufruf der Funktion fakultaet(3) mit der Variable n=3.

- a. Tragen Sie im Diagramm ein, welche Funktion von fakultaet(3) aufgerufen wird. Zeichnen Sie dazu einen neuen Knoten. Geben Sie auch die übergebene Variable mit Wert an.
- b. Tragen Sie im Diagramm ein, welche Funktion anschließend aufgerufen wird. Zeichnen Sie dazu einen neuen Knoten. Geben Sie auch die übergebene Variable mit Wert an.
- c. Tragen Sie die Rückgabewerte von unten nach oben ein.



3.3 Wie unterscheidet sich dieses Diagramm von dem auf der vorherigen Seite?

3.4 Hätten Sie den Rückgabewert von fakultaet(3) eintragen können, bevor der Funktionsaufruf fakultaet(2) eingetragen war?

3.5 Wird die Funktion fakultaet() mehrfach gleichzeitig ausgeführt? Erläutern Sie.

4 Nicht-Lineare Rekursion

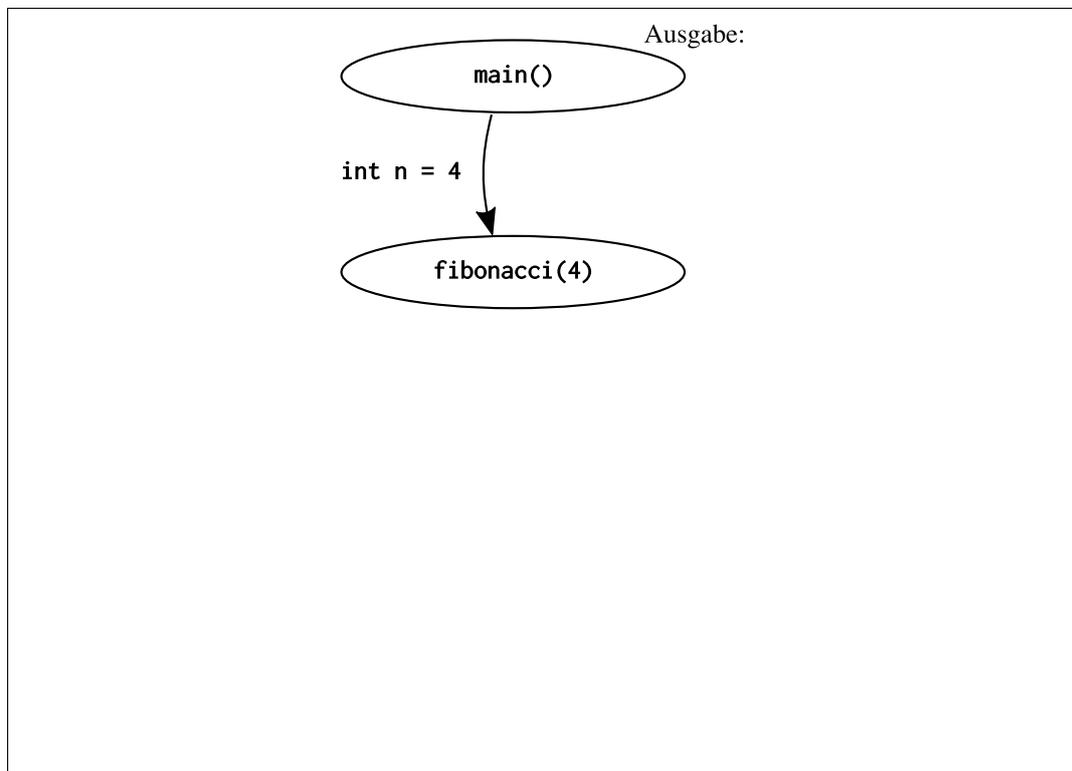
Betrachten Sie den folgenden Quelltext.

```

1 double fibonacci(int n) {
2     if (n == 0) {
3         return 0.0;
4     } else if (n == 1) {
5         return 1.0;
6     } else {
7         return fibonacci(n-1) + fibonacci(n-2);
8     }
9 }
10 int main() {
11     printf("%f", fibonacci(4) );
12 }
```

Im Folgenden soll wieder ein Diagramm aller Funktionsaufrufe erstellt werden. Bereits eingetragen ist der Aufruf der Funktion `main()` bei Ausführen des Programms sowie der Aufruf der Funktion `fibonacci(4)` mit der Variable `n=4`.

- 4.1 Tragen Sie die zwei Funktionen (mit den übergebenen Variablen) ein, die von `fibonacci(4)` aufgerufen werden.
- 4.2 Tragen Sie alle verbleibenden Funktionsaufrufe ein.
- 4.3 Tragen Sie die Rückgabewerte ein.



- 4.4 Ist diese Funktion zur Berechnung einer Fibonacci-Zahl sinnvoll? Erläutern Sie.