



**ECDL**  
Deutschland

**SYLLABUS**

---

# ECDL MODUL COMPUTING

**Syllabus Version 1.0**

DLGI  
Dienstleistungsgesellschaft für Informatik  
Am Bonner Bogen 6  
53227 Bonn  
Tel.: 0228- 688-448-0 Fax: 0228- 688-448-99  
E-Mail: [info@dlgi.de](mailto:info@dlgi.de), URL: [www.dlgi.de](http://www.dlgi.de)

In Zweifelsfällen gilt die Version der ECDL Foundation ([www.ecdl.com](http://www.ecdl.com)). Dieser Syllabus darf nur in Zusammenhang mit der ECDL Initiative verwendet werden. Im Zusammenhang mit der ECDL Initiative ist dieser Syllabus zur Verwendung und Vervielfältigung freigegeben.

# ECDL Computing

Dieses Modul behandelt die wesentlichen Kenntnisse und Fertigkeiten, die für "informatisches Denken" und Programmieren benötigt werden, um kleine einfache Computerprogramme zu entwickeln.

## Module Goals

Kandidatinnen und Kandidaten sollen:

- Die grundlegenden Konzepte von *Computing* und typische Aktivitäten im Zusammenhang mit Programmerstellung verstehen.
- Informatisches Denken (*computational thinking*) wie Problemzerlegung, Muster- bzw. Strukturerkennung, Abstraktion und Algorithmen verstehen bzw. anwenden, um ein Problem zu analysieren und entsprechende Lösungen zu entwickeln.
- Algorithmen für ein Programm unter Verwendung von Flussdiagrammen und Pseudocodes schreiben, testen und ändern.
- Die wichtigsten Grundsätze und Begriffe im Zusammenhang mit Programmierung, und die Relevanz von gut strukturiertem und dokumentiertem Code verstehen.
- Programmierkonstrukte wie Variablen, Datentypen und Logik in einem Programm verstehen und verwenden.
- Effizienz und Funktionalität durch Verwendung von Iteration (mehrfache Zuweisung), bedingte Anweisungen, Prozeduren und Funktionen, sowie Ereignisse und Befehle in einem Programm verbessern.
- Ein Programm vor der Freigabe testen und debuggen, sowie sicherstellen, dass es alle Anforderungen erfüllt.

Kategorie	Wissensgebiet	Nr.	Lernziel
<b>1 Begriffe der Programmierung</b>	1.1 Grundlegende Begriffe	1.1.1	Den Begriff Programmieren definieren.
		1.1.2	Den Begriff <i>computational thinking</i> definieren.
		1.1.3	Den Begriff Programm definieren.
		1.1.4	Den Begriff Code definieren. Zwischen Quellcode und Maschinencode unterscheiden können.
		1.1.5	Die Begriffe Programm-Beschreibung und Programm-Spezifizierung und deren Unterschied verstehen.
		1.1.6	Typische Arbeiten bei der Programmentwicklung kennen: Analyse, Design, Programmierung, Testen, Verbesserung.
		1.1.7	Den Unterscheid zwischen formaler und natürlicher Sprache verstehen.

Kategorie	Wissensgebiet	Nr.	Lernziel
<b>2 Methoden des Computational Thinking</b>	2.1 Problemanalyse	2.1.1	Typische Methoden des <i>computational thinking</i> aufzeigen können: Dekomposition, Mustererkennung, Abstraktion, Algorithmus.
		2.1.2	Problemdekomposition zur Aufschlüsselung von Daten oder Prozessen verwenden, oder ein komplexes Problem in kleinere Einzelteile zerlegen.
		2.1.3	Muster in kleineren, zerlegten Problemen erkennen.
		2.1.4	Abstraktion anwenden, um unwichtige Details bei der Problemanalyse herauszufiltern.
		2.1.5	Verstehen, wie Algorithmen im <i>computational thinking</i> verwendet werden.
	2.2 Algorithmen	2.2.1	Den Begriff Sequenz beim Programmplanungskonstrukt definieren. Sinn und Zweck der Sequenzierung bei dem Entwurf von Algorithmen darlegen.
		2.2.2	Mögliche Methoden zur Problempäsentation kennen, wie: Ablaufdiagramme, Pseudocode.
		2.2.3	Symbole von Ablaufdiagrammen kennen, wie: Start/Stop, Prozess, Entscheidung, Input/Output, Connector, Pfeil.
		2.2.4	Die Sequenz einzelner Operationen in einem Ablaufdiagramm oder Pseudocode darlegen.
		2.2.5	Einen fehlerfreien Algorithmus auf Basis einer Beschreibung unter Verwendung von Ablaufdiagramm und/oder Pseudocode schreiben.
2.2.6		Fehler in Algorithmen beheben, wie: fehlende Programmelemente, falsche Sequenzen, falsches Entscheidungsergebnis.	
<b>3 Programmieren</b>	3.1 Grundlagen	3.1.1	Eigenschaften eines gut strukturierten und dokumentierten Codes beschreiben, wie: Einrückung, richtige Kommentare, beschreibende Benennung.
		3.1.2	Einfache arithmetische Operatoren für Berechnungen im Programm kennen: +, -, /, *.

Kategorie	Wissensgebiet	Nr.	Lernziel
		3.1.3	Den Vorrang von Operatoren und die Reihenfolge von Evaluierung in komplexen Ausdrücken verstehen. Die Verwendung von Klammern bei komplexen Ausdrücken verstehen.
		3.1.4	Den Begriff Parameter verstehen. Sinn und Zweck von Parametern in einem Programm darlegen.
		3.1.5	Den Begriff Kommentar in der Programmierung definieren. Sinn und Zweck eines Kommentars innerhalb eines Programms darlegen.
		3.1.6	Kommentare in einem Programm verwenden.
	3.2 Variablen und Datentypen	3.2.1	Den Begriff Variable in einem Programmkonstrukt definieren. Sinn und Zweck einer Variablen innerhalb eines Programms darlegen.
		3.2.2	Eine Variable definieren und initialisieren.
		3.2.3	Einer Variablen einen Wert zuordnen.
		3.2.4	Richtig benannte Variablen zur Berechnung und Speicherung von Werten in Programmen verwenden.
		3.2.5	Datentypen in Programmen verwenden: String (Zeichenkette), Zeichen, Integer, Float, Boolean.
		3.2.6	Aggregierte Datentypen im Programm verwenden, wie: Array, List, Tupel.
		3.2.7	Daten-Input eines Nutzers in einem Programm verwenden.
		3.2.8	Daten-Output auf einem Monitor in einem Programm verwenden.
4 Code verwenden	4.1 Logisch	4.1.1	Den Begriff <i>logic test</i> (Test mit logischen Ausdrücke) beim Programmierungskonstrukt definieren. Sinn und Zweck eines logischen Tests in einem Programm darlegen.
		4.1.2	Verschiedene Arten Boolescher Ausdrücke kennen, um einen richtigen oder falschen Wert zu erzeugen, wie: =, >, <, >=, <=, <>, !=, ==, AND, OR, NOT.

Kategorie	Wissensgebiet	Nr.	Lernziel
		4.1.3	Boolesche Ausdrücke in einem Programm verwenden.
	4.2 Durchlauf	4.2.1	Den Begriff Schleife in einem Programm-Konstrukt definieren. Sinn und Vorzug einer Schleife in einem Programm darlegen.
		4.2.2	Verschiedene Arten von Schleifen für den Durchlauf kennen: <i>For</i> -Schleife, <i>While</i> -Schleife, <i>Repeat</i> -Schleife.
		4.2.3	Schleifen bei einem Programm verwenden, wie: <i>for</i> , <i>while</i> , <i>repeat</i> .
		4.2.4	Den Begriff Endlosschleife ( <i>infinite loop</i> ) verstehen.
		4.2.5	Den Begriff Rekursion verstehen.
	4.3 Bedingtheit	4.3.1	Den Begriff bedingte Anweisung im Programm-Konstrukt verstehen. Sinn und Zweck einer bedingten Anweisung in einem Programm darlegen.
		4.3.2	Bedingte Anweisungen IF...THEN...ELSE in einem Programm verwenden.
	4.4 Prozeduren und Funktionen	4.4.1	Den Begriff Prozedur verstehen. Sinn und Zweck einer Prozedur in einem Programm darlegen.
		4.4.2	Eine Prozedur in einem Programm schreiben und benennen.
		4.4.3	Den Begriff Funktion verstehen. Sinn und Zweck einer Funktion in einem Programm darlegen.
		4.4.4	Eine Funktion in einem Programm schreiben und benennen.
	4.5 Ereignisse und Kommandos	4.5.1	Den Begriff Ereignis verstehen. Sinn und Zweck eines Ereignisses darlegen.
		4.5.2	Event-Handler verwenden, wie: Mouseklick, Tastatur-Input, Klick auf Button, Timer.
		4.5.3	Generische Bibliotheken verwenden, wie: <i>math</i> , <i>random</i> , <i>time</i> .
<b>5 Test, Debug und Release</b>	5.1 Run, Test und Debug	5.1.1	Verstehen, warum man ein Programm testen und debuggen sollte, um Fehler zu beheben.
		5.1.2	Verschiedene Fehlerarten in einem Programm verstehen, wie: Syntax, Logik.

Kategorie	Wissensgebiet	Nr.	Lernziel
		5.1.3	Ein Programm ausführen.
		5.1.4	Syntaxfehler in einem Programm erkennen und beheben, wie: falsche Schreibweise, fehlende Interpunktion.
		5.1.5	Logische Fehler in einem Programm erkennen und beheben, wie: falscher Boolescher Ausdruck, falscher Datentyp.
	5.2 Release	5.2.1	Das Programm mit den ursprünglich beschriebenen Anforderungen abgleichen.
		5.2.2	Das fertig gestellte Programm beschreiben, Sinn und Zweck kommunizieren.
		5.2.3	Erkennen von Optimierungsmöglichkeiten, Verbesserungen am Programm, die zusätzliche, in Zusammenhang stehende Bedürfnisse decken.