



Sicher. Aktuell. ECDL

ECDL Computing

Lernzielkatalog

Der spezialisierte ECDL

17

Der Europäische Computer Führerschein ist eine Initiative von ECDL Foundation und Österreichischer Computer Gesellschaft.

The European Computer Driving Licence Foundation Ltd.

Portview House
Thorncastle Street
Dublin 4
Ireland
Tel: +353 1 630 6000
Fax: +353 1 630 6001
E-Mail: info@ecd.org
Web: www.ecdl.org

Österreichische Computer Gesellschaft (OCG)

Wollzeile 1
A-1010 Wien
Tel: +43 1 512 02 35-0
Fax: +43 1 512 02 35-9
E-Mail: info@ocg.at
Web: www.ocg.at



Alle Rechte vorbehalten. Diese Publikation darf nur im Rahmen der ECDL Initiative verwendet werden.

Hinweis

Die aktuelle deutschsprachige Version der ECDL Lernzielkataloge für Österreich ist auf der ECDL Website www.ecdl.at veröffentlicht.

Haftung

Die OCG hat dieses Dokument mit Sorgfalt erstellt, kann aber weder Richtigkeit und Vollständigkeit der enthaltenen Informationen zusichern noch Haftung für durch diese Informationen verursachte Schäden übernehmen.
In Zweifelsfällen gilt die Version der ECDL Foundation, veröffentlicht auf www.ecdl.org.

Urheberrecht

© ECDL Foundation, Adapted by OCG and SI

ECDL Computing

Lernzielkatalog,

Dieses Modul behandelt grundlegende Kenntnisse und Fertigkeiten, die erforderlich sind, um Computational Thinking und Coding zur Erstellung einfacher Computerprogramme anzuwenden.

ZIELE

Die Kandidatinnen und Kandidaten sollen

- Grundlagen des Computing und typische Schritte beim Erstellen eines Programmes verstehen,
- Methoden des Computational Thinking wie Problemzerlegung, Mustererkennung, Abstraktion und algorithmisches Design zur Problemanalyse und Lösungsentwicklung verstehen und anwenden,
- Algorithmen für ein Programm unter Verwendung von Flussdiagrammen und Pseudocode schreiben, testen und bearbeiten,
- Wesentliche Grundsätze und Schlüsselbegriffe des Codings und die Bedeutung von gutstrukturiertem und dokumentiertem Code verstehen,
- Programmierbegriffe wie Variablen, Datentypen und Logik in einem Programm verstehen und verwenden,
- Effizienz und Funktionalität verbessern, indem Iteration, bedingte Anweisungen, Prozeduren und Funktionen sowie Events und Commands in einem Programm eingesetzt werden,
- Programm testen, Fehler bereinigen (debugging) und vor der Auslieferung sicherstellen, dass die erforderlichen Bedingungen erfüllt sind.

Kategorie	Wissensgebiet	Nr.	Lernziel
1 Begriffe im Bereich Computing	1.1 Schlüsselbegriffe	1.1.1	Begriffswelt im Bereich Computing kennenlernen
		1.1.2	Identifikation bestimmter Denkweisen als Computational Thinking
		1.1.3	Den Begriff Programm kennen
		1.1.4	Den Begriff Code kennen; zwischen Quellcode und Maschinencode unterscheiden

Kategorie	Wissensgebiet	Nr.	Lernziel
		1.1.5	Wissen, wozu die Programmbeschreibung und wozu die Programmspezifikation dient
		1.1.6	Erforderliche Schritte bei der Erstellung eines Programms kennen: Analyse, Entwurf, Programmierung, Testen, Erweiterung
		1.1.7	Unterschied zwischen einer formalen und einer natürlichen Sprache kennen
2 Methoden des Computational Thinking	2.1 Problemanalyse	2.1.1	Typische Methoden des Computational Thinking erläutern: Zerlegung, Mustererkennung, Abstraktion, Algorithmen
		2.1.2	Problemzerlegung verwenden, um umfangreiche Daten und Prozesse zu bewältigen oder um ein komplexes Problem in kleinere Teile zu zerlegen
		2.1.3	Standardlösungen (Muster) in den zerlegten Teilproblemen identifizieren können
		2.1.4	Abstraktion verwenden, um unnötige Einheiten bei der Problemanalyse aus dem Weg zu räumen
		2.1.5	Die Rolle von Algorithmen beim Computational Thinking verstehen
	2.2 Algorithmisches Design	2.2.1	Rolle des Konzepts Sequenz in den Abläufen verstehen
		2.2.2	Möglichkeiten der Hilfe bei der Problemanalyse kennen, wie: Flussdiagramme, Pseudocode
		2.2.3	Symbole in Flussdiagrammen kennen, wie: Start/Stop, Prozess, Entscheidung, Ein-/Ausgabe, Verbinder, Pfeil
		2.2.4	Abfolge der wichtigsten Schritte mit einem Flussdiagramm oder mit Pseudocode beschreiben
		2.2.5	Detaillierten Ablauf (Algorithmus) unter Verwendung von Flussdiagramm oder Pseudocode beschreiben
		2.2.6	Fehler in einem Algorithmus verbessern, wie: fehlendes Programmelement, falsche Sequenz, falsches Entscheidungskriterium bei der Verzweigung
3 Coding	3.1 Erste Schritte	3.1.1	Stileigenschaften eines optisch gut strukturierten und dokumentierten Programmcodes kennen, wie: Einrückung, geeignete Kommentare und aussagekräftige Bezeichnungen

Kategorie	Wissensgebiet	Nr.	Lernziel
		3.1.2	Einfache arithmetische Operatoren verwenden, um Rechenschritte in einem Programm auszuführen: +, -, /, *
		3.1.3	Prioritäten der Operatoren und Reihenfolge der Evaluation in arithmetischen, logischen und zeichenverarbeitenden Ausdrücken kennen; Verstehen, wie Klammern zur Strukturierung komplexer Ausdrücke eingesetzt werden
		3.1.4	Verwendung von Parametern in einem Programm kennen
		3.1.5	Verwendung von Kommentaren in einem Programm verstehen und erläutern
		3.1.6	Zweckmäßige Kommentare in einem eigenen Programm setzen
	3.2 Variablen und Daten	3.2.1	Konzept Variable kennen und erläutern; Variablen in einem Programm verwenden
		3.2.2	Definition, Initialisierung und Verwendung einer Variablen unterscheiden
		3.2.3	Zuweisung von Werten an eine Variable
		3.2.4	Geeigneten Namen für Berechnungen und zur Speicherung von Werten verwenden
		3.2.5	Einfache Datentypen in einem Programm verwenden: Zeichenkette (string), Zeichen (character), Ganzzahlen (integer), Gleitkommazahlen (float), Logische Aussagen (boolean)
		3.2.6	Strukturierte Datentypen in einem Programm verwenden, wie: Array, Liste, Tupel
		3.2.7	In einem interaktiven Programm auf Dateneingaben einer Anwenderin bzw. eines Anwenders reagieren
		3.2.8	In einem interaktiven Programm die Datenausgabe auf dem Bildschirm gestalten
4 Konstruktive Verwendung von Code-Elementen	4.1 Logik	4.1.1	Korrekte Formulierung von logischen Tests beherrschen; zweckmäßige Verwendung eines logischen Tests in einem Programm kennen und erläutern
		4.1.2	Boolesche Logikaussagen mit Variablen, Vergleichsoperatoren und Booleschen Operatoren als Ausdrücke formulieren; Verwendung der Operatoren : =, >, <, >=, <=, <>, !=, ==, AND, OR, NOT

Kategorie	Wissensgebiet	Nr.	Lernziel
		4.1.3	Logikaussagen in einem Programm einbauen
	4.2 Schleifen (Iteration)	4.2.1	Korrekte Formulierung von Schleifen beherrschen; zweckmäßige Verwendung von Schleifen in einem Programm kennen und erläutern
		4.2.2	Verschiedene Arten von Schleifen unterscheiden, wie: FOR, WHILE, REPEAT
		4.2.3	Schleifen wie FOR, WHILE, REPEAT in einem Programm verwenden
		4.2.4	Konzept einer Endlosschleife verstehen
		4.2.5	Konzept der Rekursion verstehen, Unterschied zur Iteration kennen
	4.3 Bedingte Anweisung	4.3.1	Korrekte Formulierung von bedingten Anweisungen beherrschen; zweckmäßige Verwendung einer bedingten Anweisung in einem Programm kennen und erläutern
		4.3.2	Mehrwegweisung IF...THEN...ELSE in einem Programm verwenden
	4.4 Prozeduren und Funktionen	4.4.1	Konzept der Prozedur verstehen; zweckmäßige Verwendung einer Prozedur in einem Programm verstehen und erläutern
		4.4.2	In einem Programm einen Teil davon korrekt in eine Prozedur ausgliedern und benennen
		4.4.3	Konzept der Funktion verstehen; zweckmäßige Verwendung einer Funktion in einem Programm kennen und erläutern
		4.4.4	Einen geeigneten Teil eines Programms in eine Funktion ausgliedern und benennen
		4.4.5	Funktionen mit Parametern schreiben können
		4.4.6	Funktionen schreiben können, die sich selber aufrufen (Rekursion)
	4.5 Ereignisse (Events) und Aufrufe (Commands)	4.5.1	Konzept eines Ereignisses (Events) verstehen; zweckmäßige Verwendung eines Ereignisses (Events) in einem Programm erläutern
		4.5.2	Ereignisbehandlungsroutine (Event-Handler) erstellen und verwenden, wie: Mausclick, Tastatureingabe, Klick auf Schaltfläche, Timer
		4.5.3	Funktionen aus Standardbibliotheken verwenden, wie: math, random, time
5 Testen, Fehlersuche, Auslieferung	5.1 Programm ausführen, testen, Fehler beseitigen	5.1.1	Möglichkeiten von Test und Beweis zur Erreichung eines möglichst korrekten Programms richtig einschätzen

Kategorie	Wissensgebiet	Nr.	Lernziel
		5.1.2	Verschiedene Arten von Fehlern in einem Programm kennen und unterscheiden, wie: Programmsyntax und Programmlogik
		5.1.3	Programme ausführen
		5.1.4	Syntaxfehler in einem Programm suchen und beheben, wie: falsche Schreibweise, fehlende Trennzeichen.
		5.1.5	Logikfehler in einem Programm suchen und beheben, wie: inkorrekt boolescher Ausdruck, inkorrekt Datentyp
	5.2 Auslieferung des Programms	5.2.1	Erstelltes Programm mit den Anforderungen der ursprünglichen Problembeschreibung vergleichen
		5.2.2	Erstelltes Programm beschreiben, Zweck und Wert der Anwenderin bzw. dem Anwender kommunizieren
		5.2.3	Erweiterungen und Verbesserungen für das Programm vorschlagen, die einen zusätzlichen Nutzen bringen würden

Sicher. Aktuell. ECDL

TM



ECDL
in Education Austria

ECDL Standard in Education Austria

6 Pflicht-
module

+

1 Wahl-
modul

Pflicht- module

Computer-
Grundlagen

Online-
Grundlagen

Textverarbeitung

Tabellenkalkulation

Präsentation

IT-Security

Wahl- module

Datenbanken
anwenden

Online-
Zusammenarbeit

Image Editing
(Bildbearbeitung)

NEU

Computing
(Wahlmodul oder
Einzelzertifikat)

ECDL Profile in Education Austria

3 Pflicht-
module

+

1 Wahl-
modul

Pflicht- module

Computer-
Grundlagen

Online-
Grundlagen

Textverarbeitung

Wahl- module

Tabellenkalkulation

Präsentation

IT-Security

www.ecdl.at



**OESTERREICHISCHE
COMPUTER GESELLSCHAFT**®
AUSTRIAN
COMPUTER SOCIETY

Europäischer Computer Führerschein -
Eine Initiative der ECDL Foundation und
der Österreichischen Computer Gesellschaft

