swissinformatics.org

ECDL

locally certified – globally accepted

ECDL Syllabus
Testinhalte ECDL Modul Computing

Computing
Syllabus 1.0

Herausgeber
Digital Literacy AG
Bollwerk-Promenade 5
CH-4051 Basel

Tel. +41 61 270 88 77

info@ecdl.ch
www.ecdl.ch

ECDL in der Schweiz – unter Lizenz der Schweizer Informatik Gesellschaft (SI)
www.swissinformatics.org

Die männliche Form steht stellvertretend für beide Geschlechter.

Die elektronische Version dieses ECDL Syllabus finden Sie unter **www.ecdl.ch.**

ECDL Computing: Von der Problemanalyse bis zum fertigen Programm

Liebe Leserin, lieber Leser

Der vorliegende ECDL Syllabus beschreibt, über welche Kenntnisse Sie verfügen sollten, wenn Sie die Prüfung zum ECDL Standard Modul Computing ablegen wollen. Diese Broschüre dient Ihnen auch als Checkliste, mit der Sie überprüfen können, welche Kenntnisse Ihnen noch fehlen.

Dank den Fertigkeiten aus dem Modul Computing können Sie einfache Probleme der Informationsverarbeitung analysieren und eine systematische Vorgehensweise (Algorithmus) zu deren Lösung finden. Sie kennen die Grundlagen der Erstellung eines Programmes und können so einem Computer direkt die Anweisungen zur Lösung geben. Beim Kodieren der Lösung nutzen Sie Abläufe, logische Tests und Variablen in geeigneter Datendarstellung. Sie können dabei Schleifen, bedingte Anweisungen und Funktionen einsetzen, sowie Programme testen und Fehler bereinigen.

Es freut uns, dass wir mit dem ECDL Modul Computing unsere bis anhin auf Anwenderkenntnisse fokussierte Modulpalette um Wissen und Fertigkeiten im Grundlagenbereich der Informatik erweitern können. Damit können wir Ihnen ein noch besseres Zertifizierungsangebot anbieten und auch einen Beitrag zur Förderung der Informatik in der Schweiz und in Liechtenstein leisten.

Computerkurse, die Ihnen die Inhalte der ECDL Module vermitteln, werden von den meisten ECDL Test Centern angeboten. Sie können die entsprechenden Prüfungen an einem der rund 300 ECDL Test Center in der Schweiz und in Liechtenstein ablegen, unabhängig davon, ob Sie dort einen Kurs besucht haben.

Weitere Informationen zu den ECDL Zertifikaten sowie eine Übersicht der ECDL Test Center finden Sie auf **www.ecdl.ch.** In unserem Webshop können Sie zudem geeignete Lehrmittel zur Vorbereitung auf die ECDL Prüfungen erwerben.

April 2018

ECDL Module und Zertifikate

Base Module	Standard Module	Advanced Module	Typing Modul*
Computer-Grundlagen	Datenbanken anwenden	Text-verarbeitung	Typing Skills
Online-Grundlagen	Präsentation	Tabellen-kalkulation	
Text-verarbeitung	Online-Zusammenarbeit	Datenbank	
Tabellen-kalkulation	IT-Sicherheit	Präsentation	
	Bildbearbeitung		
	Computing		

ECDL Base Zertifikat

4 Base Module

ECDL Standard Zertifikat

4 Base Module + 3 Standard Module nach Wahl

ECDL Advanced Zertifikat

1 Advanced Modul nach Wahl

ECDL Expert Zertifikat

3 Advanced Module nach Wahl

ECDL Profile Zertifikat

mind. 4 ECDL Module nach Wahl

Typing Skills Zertifikat*

1 Typing Modul

* Dieses Modul/Zertifikat wurde von der Österreichischen Computer Gesellschaft entwickelt und ist von der ECDL Foundation anerkannt.

Computing

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

Modulziele

Die Kandidaten müssen

- Grundlagen des Computing und typische Schritte beim Erstellen eines Programmes verstehen,
- Techniken des Computational Thinking wie Problemzerlegung, Mustererkennung, Abstraktion und Algorithmen zur Problemanalyse und Lösungsentwicklung verstehen und anwenden können,
- Algorithmen für ein Programm unter Verwendung von Flussdiagrammen und Pseudocode schreiben, testen und bearbeiten können,
- wesentliche Grundsätze und Schlüsselbegriffe des Codings und die Bedeutung von gut strukturiertem und dokumentiertem Code verstehen,
- Programmkonstrukte wie Variablen, Datentypen und Logik in einem Programm verstehen und verwenden können,
- Effizienz und Funktionalität verbessern können, indem Schleifen (Iteration), bedingte Anweisungen, Prozeduren und Funktionen sowie Ereignisse (Events) und Anweisungen (Commands) in einem Programm eingesetzt werden,
- ein Programm testen, Fehler bereinigen (Debugging) und vor der Freigabe sicherstellen können, dass die Anforderungen erfüllt sind.

Kategorie	Wissensgebiet	Nr.	Lernziel
1 | Begriffe im Bereich Computing	1.1 | Schlüsselbegriffe	1.1.1	Begriffswelt im Bereich Computing kennen
		1.1.2	Identifikation bestimmter Denkweisen als Computational Thinking
		1.1.3	Das Konzept eines Programms verstehen
		1.1.4	Das Konzept Code verstehen; zwischen Quellcode und Maschinencode unterscheiden
		1.1.5	Wissen, wozu die Programmbeschreibung und wozu die Programmspezifikation dienen
		1.1.6	Erforderliche Schritte bei der Erstellung eines Programms kennen und anwenden: Analyse, Entwurf, Programmierung, Testen, Erweiterung

Kategorie	Wissensgebiet	Nr.	Lernziel
		1.1.7	Unterschied zwischen einer formalen und einer natürlichen Sprache verstehen
**2	Methoden des Computational Thinking**	2.1 | Problemanalyse	2.1.1	Typische Methoden des Computational Thinking erläutern und anwenden: 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	Muster in den zerlegten Teilproblemen identifizieren können und Standardlösungen verwenden
			2.1.4	Abstraktion verwenden, um unnötige Einzelheiten bei der Problemanalyse auszublenden
			2.1.5	Die Rolle von Algorithmen beim Computational Thinking verstehen
	2.2 | Algorithmen	2.2.1	Das Konzept Sequenz im Algorithmus verstehen
			2.2.2	Möglichkeiten der Problemdarstellung verwenden 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	Algorithmus unter Verwendung von Flussdiagramm oder Pseudocode herleiten
			2.2.6	Fehler in einem Algorithmus beseitigen wie: fehlendes Programmelement, falsche Sequenz, falsches Entscheidungskriterium bei der Verzweigung
**3	Coding**	3.1 | Erste Schritte	3.1.1	Stileigenschaften eines gut strukturierten und dokumentierten Programmcodes kennen wie: Einrückung, geeignete Kommentare und aussagekräftige Bezeichnungen
			3.1.2	Einfache arithmetische Operatoren verwenden, um Rechenschritte in einem Programm auszuführen: +, –, /, *

Kategorie	Wissensgebiet	Nr.	Lernziel
		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 verstehen
		3.1.5	Verwendung von Kommentaren in einem Programm verstehen und erläutern
		3.1.6	Zweckmässige Kommentare in einem Programm setzen
	3.2 | Variablen und Daten	3.2.1	Konzept Variable verstehen und erläutern; Variablen in einem Programm verwenden
		3.2.2	Deklaration, Initialisierung und Verwendung einer Variablen unterscheiden
		3.2.3	Zuweisung eines Wertes an eine Variable
		3.2.4	Geeignete Variablennamen 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 Daten auf dem Bildschirm ausgeben
4 | Elemente der Programmierung	4.1 | Logik	4.1.1	Zweckmässige Verwendung eines logischen Tests in einem Programm verstehen und erläutern
		4.1.2	Logikaussagen mit Booleschen Variablen, Vergleichsoperatoren (>, <, >=, <=, <>, !=, ==) und Booleschen Operatoren (AND, OR, NOT) formulieren
		4.1.3	Logikaussagen in einem Programm einbauen
	4.2 | Schleifen (Iteration)	4.2.1	Zweckmässige Verwendung von Schleifen in einem Programm verstehen und erläutern

Kategorie	Wissensgebiet	Nr.	Lernziel
		4.2.2	Verschiedene Arten von Schleifen unterscheiden wie: FOR, WHILE, REPEAT
		4.2.3	Schleifen wie FOR, WHILE, REPEAT in einem Programm einbauen
		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	Zweckmässige Verwendung einer bedingten Anweisung in einem Programm verstehen und erläutern
		4.3.2	Mehrweganweisung IF… THEN… ELSE in einem Programm einbauen
	4.4 | Prozeduren und Funktionen	4.4.1	Konzept einer Prozedur verstehen; zweckmässige Verwendung einer Prozedur in einem Programm erläutern
		4.4.2	Teile eines Programms in einer Prozedur zusammenfassen und benennen
		4.4.3	Konzept einer Funktion verstehen; zweckmässige Verwendung einer Funktion in einem Programm erläutern
		4.4.4	Teile einer Berechnung in einer Funktion zusammenfassen 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: Mausklick, 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 Fehlerbehebung zur Erreichung eines möglichst korrekten Programms richtig einschätzen
		5.1.2	Verschiedene Arten von Fehlern in einem Programm kennen und unterscheiden wie: Programmsyntax und Programmlogik

Kategorie	Wissensgebiet	Nr.	Lernziel
		5.1.3	Programm ausführen
		5.1.4	Syntaxfehler in einem Programm beheben wie: falsche Schreibweise, fehlende Trennzeichen
		5.1.5	Logikfehler in einem Programm suchen und beheben wie: inkorrekter Boolescher Ausdruck, inkorrekter 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 suchen, die einen zusätzlichen Nutzen bringen würden

ECDL

locally certified – globally accepted

Weitere Informationen zum ECDL erhalten Sie bei den ECDL Test Centern und bei der Digital Literacy AG.

Ihr ECDL Test Center:

1618_V1 Design by www.rippstein.ch

Digital Literacy AG, Bollwerk-Promenade 5, CH-4051 Basel
Telefon +41 61 270 88 77, info@ecdl.ch, www.ecdl.ch