



ECDL Podstawy programowania

Sylabus - wersja 1.0

Przeznaczenie Sylabusa

Dokument ten zawiera szczegółowy Sylabus dla modułu Podstawy programowania. Sylabus opisuje, poprzez efekty uczenia się, zakres wiedzy i umiejętności, jakie musi opanować Kandydat, żeby zdać wymagany egzamin. Sylabus zawiera podstawy teoretyczne do pytań i zadań egzaminacyjnych z tego modułu.

Copyright © 1997 - 2017 Fundacja ECDL

Wszystkie prawa zastrzeżone. Żadna część poniższego opracowania nie może być wykorzystana bez zgody Fundacji ECDL. Wszystkie podmioty zainteresowane wykorzystaniem opracowania powinny kontaktować się bezpośrednio z Fundacją ECDL.

Oświadczenie

Mimo tego, że podczas opracowania powyższego dokumentu Fundacja ECDL dołożyła wszelkich starań by zawierał on wszystkie niezbędne elementy, to Fundacja ECDL, jako wydawca opracowania nie udziela gwarancji i nie bierze odpowiedzialności za ewentualne braki. Fundacja nie bierze również odpowiedzialności za błędy, pominięcia, nieścisłości, straty lub szkody wynikające z tytułu użytkowania poniższej publikacji. Wszelkie zmiany mogą zostać dokonane przez Fundację ECDL na jej odpowiedzialność, bez konieczności zgłaszania tego faktu.

ECDL Podstawy programowania

Ten moduł sprawdza znajomość podstawowej wiedzy i umiejętności zastosowania myślenia komputacyjnego i kodowania do tworzenia prostych programów komputerowych.

Założenia modułu

Aby zaliczyć moduł Kandydat musi:

- Znać i rozumieć podstawowe pojęcia dotyczące myślenia komputacyjnego i typowe działania związane z tworzeniem programu.
- Rozumieć i wykorzystywać techniki myślenia komputacyjnego, takie jak dekompozycja problemu, rozpoznawanie wzorców, abstrakcja i algorytmy w celu analizy problemu i opracowania rozwiązań.
- Umieć zapisywać algorytmy przy użyciu schematów blokowych oraz pseudokodów, testować je i modyfikować.
- Znać i rozumieć podstawowe zasady i pojęcia związane z kodowaniem i rozumieć znaczenie dobrze ustrukturyzowanego i udokumentowanego kodu.
- Rozumieć znaczenie i używać konstrukcji programistycznych takich jak: zmienne, typy danych i logika w programie.
- Umieć poprawiać wydajność i funkcjonalność programu poprzez zastosowanie iteracji, instrukcji warunkowych, procedur i funkcji, jak również zdarzeń w programie.
- Umieć testować i debugować program, aby upewnić się przed wdrożeniem produkcyjnym, że spełnia on postawione wymagania.

Osoba posiadająca daną kwalifikację:

KATEGORIA	OBSZAR WIEDZY I UMIEJĘTNOŚCI	NR	ZADANIE
1 Terminologia	1.1 <i>Kluczowe pojęcia</i>	1.1.1	Wyjaśnia pojęcie: computing.
		1.1.2	Wyjaśnia pojęcie: myślenie komputacyjne.
		1.1.3	Wyjaśnia pojęcie: program.
		1.1.4	Wyjaśnia pojęcie: kod. Rozróżnia kod źródłowy i kod maszynowy.
		1.1.5	Wyjaśnia pojęcia: opis programu i specyfikacja.

KATEGORIA	OBSZAR WIEDZY I UMIEJĘTNOŚCI	NR	ZADANIE
2 Metody myślenia komputacyjnego	2.1 <i>Analiza problemu</i>	1.1.6	Rozpoznaje typowe działania przy tworzeniu programu: analiza, projektowanie, programowanie, testowanie, ulepszanie.
		1.1.7	Wyjaśnia różnice pomiędzy językiem formalnym a językiem naturalnym.
		2.1.1	Określa typowe metody stosowane w myśleniu komputacyjnym: dekompozycja, rozpoznawanie wzorców, abstrakcja, algorytmy.
		2.1.2	Używa metody dekompozycji problemu, aby podzielić dane, procesy lub złożone problemy na mniejsze części.
		2.1.3	Znajduje wzorce wśród mniej złożonych, zdekomponowanych problemów.
		2.1.4	Wykorzystuje abstrakcję, aby wyeliminować niepotrzebne szczegóły podczas analizy problemu.
	2.1.5	Wyjaśnia, jak algorytmy są wykorzystywane w myśleniu komputacyjnym.	
	2.2 <i>Algorytmy</i>	2.2.1	Definiuje sekwencję jako konstrukcję algorytmiczną. Określa cel stosowania sekwencji przy projektowaniu algorytmów.
		2.2.2	Rozpoznaje możliwe metody prezentacji problemu takie jak: schemat blokowy, pseudokod.
		2.2.3	Rozpoznaje elementy schematu blokowego takie jak: start/stop, proces, decyzja, wejście/wyjście, łącznik, strzałka.
		2.2.4	Wyjaśnia operacje reprezentowane przez schemat blokowy, pseudokod.
		2.2.5	Wykorzystuje schemat blokowy lub pseudokod do stworzenia dokładnego algorytmu na podstawie opisu.

KATEGORIA	OBSZAR WIEDZY I UMIEJĘTNOŚCI	NR	ZADANIE
3 Początki programowania	<i>3.1 Podstawy</i>	2.2.6	Poprawia błędy w algorytmie takie jak: brak elementu programu, niepoprawna kolejność, nieprawidłowy wynik decyzji.
		3.1.1	Tworzy dobrze zbudowany i udokumentowany kod poprzez: wcięcia, odpowiednie komentarze, opisowe nazewnictwo.
		3.1.2	Używa prostych operatorów arytmetycznych do wykonywania obliczeń: +, -, /, *.
		3.1.3	Objaśnia pojęcia: priorytet operatorów i kolejność wykonywania działań. Używa nawiasów do konstruowania złożonych wyrażeń.
		3.1.4	Objaśnia pojęcie: parametr. Określa cel stosowania parametrów w programie.
		3.1.5	Definiuje pojęcie: komentarz. Określa cel stosowania komentarzy w programie.
	<i>3.2 Zmienne i typy danych</i>	3.1.6	Używa komentarzy w programie.
		3.2.1	Definiuje pojęcie: zmienna. Wyjaśnia cel użycia zmiennej w programie.
		3.2.2	Definiuje i inicjalizuje zmienne.
		3.2.3	Przypisuje wartości do zmiennych.
		3.2.4	Używa odpowiednio nazwanych zmiennych w programie do wykonywania obliczeń i przechowywania wartości.
		3.2.5	Używa w programie różnych typów danych: liczba całkowita, liczba zmiennoprzecinkowa, string (ciąg znaków), znak, wartość logiczna.
		3.2.6	Używa strukturalnych typów danych takich jak tablica, lista, krotka (tupla).

KATEGORIA	OBSZAR WIEDZY I UMIEJĘTNOŚCI	NR	ZADANIE
4 Programowanie	4.1 Logika	3.2.7	Korzysta w programie z danych wprowadzonych przez użytkownika.
		3.2.8	Wypisuje wyniki programu na ekran.
		4.1.1	Definiuje pojęcie: warunek logiczny. Określa cel stosowania warunków logicznych w programie.
		4.1.2	Wykorzystuje wyrażenia logiczne w celu wygenerowania wartości prawda/fałsz za pomocą operatorów: =, >, <, >=, <=, <>, !=, ==, AND, OR, NOT.
		4.1.3	Używa w programie wyrażenia logicznych.
		4.2 Iteracja	4.2.1
	4.2.2		Rozpoznaje typy pętli wykorzystywanych do iteracji: for, while, repeat.
	4.2.3		Wykorzystuje w programie pętle typu: for, while, repeat.
	4.2.4		Wyjaśnia pojęcie: pętla nieskończona.
	4.2.5		Wyjaśnia pojęcie: rekurencja.
	4.3 Warunki	4.3.1	Definiuje pojęcie: instrukcja warunkowa. Określa cel i korzyści ze stosowania instrukcji warunkowej w programie.
		4.3.2	Używa instrukcji warunkowej typu if...then ...else w programie.
	4.4 Procedury i funkcje	4.4.1	Definiuje pojęcie: procedura. Określa cele i korzyści z zastosowania procedury w programie.
		4.4.2	Tworzy i nazywa procedury w programie.

KATEGORIA	OBSZAR WIEDZY I UMIEJĘTNOŚCI	NR	ZADANIE
		4.4.3	Definiuje pojęcie: funkcja. Określa cel i korzyści z zastosowania funkcji w programie.
		4.4.4	Tworzy i nazywa funkcje w programie.
	4.5 Zdarzenia	4.5.1	Definiuje pojęcie: zdarzenie. Określa cel wykorzystania zdarzeń w programie.
		4.5.2	Korzysta z obsługi zdarzeń, takich jak: kliknięcie myszą, wejście z klawiatury, kliknięcie przycisku, zegar.
		4.5.3	Używa dostępnych bibliotek takich jak: math, random, time.
5 Testowanie, debugowanie i wersja produkcyjna	5.1 Uruchamianie, testowanie i debugowanie	5.1.1	Wyjaśnia zalety testowania i debugowania programu w celu znalezienia błędów.
		5.1.2	Rozróżnia typy błędów w programie takie jak: składniowe, logiczne.
		5.1.3	Uruchamia program.
		5.1.4	Znajduje w programie i poprawia takie błędy składniowe jak: niepoprawna pisownia, opuszczony znak interpunkcyjny.
		5.1.5	Znajduje w programie i poprawia takie błędy logiczne jak: niepoprawne wyrażenie logiczne, niepoprawny typ danych.
	5.2 Wersja produkcyjna	5.2.1	Sprawdza zgodność programu z wymaganiami początkowymi.
		5.2.2	Opisuje gotowy program, określając jego cel i wartość.
		5.2.3	Identyfikuje możliwe ulepszenia i doskonalenia programu, które mogą zaspokajać dodatkowe, powiązane z celem głównym potrzeby.