ICDL

The Digital Skills Standard

ICDL Digital Student
COMPUTING
Syllabus 1.1

Informatické myslenie a základy programovania
Sylabus 1.1

Sylabus Document - Znenie sylabu

Účel

Tento dokument uvádza v plnom znení sylabus pre modul Informatické myslenie a základy programovania (Computing). Sylabus podrobne popisuje znalosti a zručnosti (learning outcomes), ktoré by ucházdač o skúšku z tohto modulu mal mať. Sylabus je zároveň aj východiskom pre zostavenie teoretických a praktických testov na overenie znalostí a zručností z tohto modulu.

PREHLÁSENIE (zrieknutie sa zodpovednosti)

Hoci príprave tejto publikácie bola v ICDL Foundation venovaná najvyššia pozornosť, ICDL Foundation nedáva ako vydavateľ žiadnu záruku na úplnosť informácií v tomto materiáli a ICDL Foundation nemá povinnosť ani zodpovednosť v spojení s akýmikoľvek chybami, omylmi, nepresnosťami, stratou alebo škodou, ktorá by kedykoľvek vznikla na základe informácií alebo inštrukcií obsiahnutých v tomto materiáli. ICDL Foundation si vyhradzuje právo vykonávať zmeny podľa vlastného uváženia a bez predchádzajúceho upozornenia.

Oficiálna verzia tohto materiálu je verzia zverejnená na webovej stránke ICDL Foundation: **icdl.org**, lokalizovaná verzia na webových stránkach Slovenskej informatickej spoločnosti Kancelária ICDL na **www.icdl.sk**.

Sylabus verzie 6.1 bol v roku 2024 schválený ICDL Foundation na používanie na území ČR a SR v rámci pilotného overovania.

Modul – Informatické myslenie a základy programovania (Computing), M16

Tento modul vymenúva **základné pojmy a zručnosti z oblasti informatického myslenia a programovania**, ktoré uchádzač musí ovládať, aby bol schopný využívať informatické myslenie (computational thinking) a vytvárať jednoduché počítačové programy (programming/ coding)[1].

Ciele modulu

Úspešný uchádzač by mal byť schopný:

- rozumieť kľúčovým (základným) pojmom z oblasti informatického myslenia a programovania/ kódovania a poznať a využívať typické činnosti spojené s programovaním,
- rozumieť technikám informatického myslenia a používať ich, konkrétne dekompozíciu problému (rozklad na jednotlivé podproblémy), vyhľadávanie vzorov, zovšeobecňovanie / abstrakcia a algoritmy (postupy) na analýzu problému a vytvorenie jeho riešenia,
- vedieť zapisovať, ladiť a opravovať algoritmy pre program s použitím blokových schém /vývojových diagramov a pseudokódu,
- rozumieť kľúčovým princípom a pojmom pri písaní kódu (programovaní) a chápať dôležitosť dobre štrukturovaného a zdokumentovaného (komentovaného) kódu,
- rozumieť programovacím štruktúram v programe a používať ich, konkrétne: premenné, dátové typy, logické konštrukcie,
- zvyšovať efektivitu a funkčnosť programu pomocou iterácie, podmienených príkazov, podprogramov s alebo bez návratovej hodnoty , ako aj pomocou využívania udalostí a príkazov,
- testovať a ladiť program, odstraňovať chyby v programe a vedieť posúdiť, či program spĺňa požiadavky na funkčnosť od zadávateľa pred tým ako bude zverejnený (vydaný),
- vedieť používať nástroje s technológiami umelej inteligencie na vytváranie a úpravu programového kódu.

KATEGÓRIA	SÚBOR ZRUČNOSTÍ	REF.	VYŽADOVANÁ SCHOPNOSŤ
16.1. Základné pojmy	16.1.1 Kľúčové pojmy	16.1.1.1	Rozumieť pojmu computing[2] (spracovanie údajov pomocou počítača vrátane výpočtov nad nimi).
		16.1.1.2	Rozumieť pojmu informatické myslenie[3] (computational thinking).
		16.1.1.3	Rozumieť pojmu počítačový program.
		16.1.1.4	Rozumieť pojmu programový kód (code) a rozdielu medzi pojmami zdrojový kód (source code) a strojový kód (machine code).
		16.1.1.5	Rozumieť pojmom popis účelu programu (program description) a špecifikácia funkcií programu (program specification).
		16.1.1.6	Vedieť rozpoznať typické fázy procesu tvorby počítačového programu, ako sú: analýza, návrh algoritmu (design), kódovanie, testovanie (testing), vylepšovanie.
		16.1.1.7	Chápať rozdiel medzi formálnym a prirodzeným jazykom.
16.2. Metódy informatického myslenia	16.2.1 Analýza problému	16.2.1.1	Poznať typické metódy používané v informatickom myslení: dekompozícia problému, rozpoznávanie vzorov (pattern recognition), abstrakcia, algoritmizácia.

[1] coding a programming sú pojmy, ktoré označujú tie isté činnosti, len v istých obdobiach sa používal viac jeden a v inom období druhý; do slovenčiny oba termíny prekladáme ako programovanie

[2] Obsah pojmu je presne vymedzený, v rôznych štátoch je obsah pojmu rôzny

[3] informatické myslenie je o trochu širší pojem ako algorimické myslenie

KATEGÓRIA	SÚBOR ZRUČNOSTÍ	REF.	VYŽADOVANÁ SCHOPNOSŤ
		16.2.1.2	Vedieť používať dekompozíciu, teda rozklad na jednotlivé menšie časti, a to na údaje, procesy alebo celkový problém na menšie podproblémy.
		16.2.1.3	Vedieť rozpoznať (identifikovať) rovnaké vzory, teda spoločné znaky (vlastnosti), jednotlivých častí problému.
		16.2.1.4	Vedieť používať abstrakciu na odfiltrovanie (ignorovanie) nepodstatných detailov počas analýzy problému.
		16.2.1.5	Rozumieť, ako sa používajú algoritmy v informatickom myslení (computational thinking).
	16.2.2 Algoritmy	16.2.2.1	Rozumieť pojmu postupnosť príkazov pri vytváraní počítačového programu. Chápať účel radenia príkazov pri navrhovaní algoritmu.
		16.2.2.2	Poznať ako sa dajú zobraziť algoritmy, ako sú: blokové schémy/ vývojové diagramy (flowcharts), pseudokód (pseudocode).
		16.2.2.3	Poznať základné symboly blokovej schémy/ vývojového diagramu, ako sú: začiatok / koniec (start / stop), spracovanie (process), rozhodovanie (decision), vstup / výstup (input / output), spojnica (connector), šípka (arrow).
		16.2.2.4	Rozumieť postupnosti operácií, ktorá je znázornená blokovou schémou/ vývojovým diagramom, pseudokódom.
		16.2.2.5	Vedieť napísať konkrétny algoritmus na základe opisu problému pomocou blokovej schémy/ vývojového diagramu alebo pseudokódu.
		16.2.2.6	Vedieť opraviť chyby v algoritme ako sú: chýbajúci prvok v programe (príkaz, parameter, premenná a pod.), nesprávne poradie operácií, nesprávna logická podmienka.
16.3. Začiatok kódovania	16.3.1 Príprava kódovania	16.3.1.1	Poznať pravidlá dobre štrukturovaného a komentovaného kódu ako sú: odsadenie (indentation), vhodné komentáre, vhodné opisné názvy.
		16.3.1.2	Vedieť používať operátory +, -, /, * na vykonávanie aritmetických výpočtov ako sú sčítanie, odčítanie, násobenie a delenie.
		16.3.1.3	Poznať prioritu jednotlivých operátorov a poradie v akom sa vyhodnocujú zložitejšie výrazy. Vedieť používať zátvorky na štrukturovanie zložitejších výrazov.
		16.3.1.4	Rozumieť pojmu parameter. Poznať účel, na aký sa parameter v programe používa.
		16.3.1.5	Rozumieť programovacej štruktúre komentár. Poznať účel, na aký sa komentár v programe používa.
		16.3.1.6	Vedieť používať komentáre v programe.
	16.3.2 Premenné a dátové typy	16.3.2.1	Rozumieť programovacej štruktúre premenná. Poznať účel, na aký sa premenná v programe používa.

KATEGÓRIA	SÚBOR ZRUČNOSTÍ	REF.	VYŽADOVANÁ SCHOPNOSŤ
		16.3.2.2	Vedieť zaviesť /deklarovať / definovať[4] a inicializovať premennú.
		16.3.2.3	Vedieť priradiť premennej konkrétnu hodnotu.
		16.3.2.4	Vedieť zavádzať vhodné názvy premenných v programe, a to na účel podporných, priebežných výpočtov i ukladania hodnôt výsledkov.
		16.3.2.5	Vedieť používať základné dátové typy v programe: reťazec (string), znak (character)[5], celé číslo (integer), reálne číslo (float), logická hodnota (Boolean).
		16.3.2.6	Vedieť používať agregované / zložené dátové typy v programe, ako sú: pole (array)[6], zoznam (list), n-tica (tuple).
		16.3.2.7	Vedieť v programe prijať vstupný údaj od používateľa.
		16.3.2.8	Vedieť v programe odoslať / zobraziť výstupný údaj na monitore / obrazovke.
16.4. Vytváranie kódu	16.4.1 Logické operácie	16.4.1.1	Rozumieť programovacej štruktúre logický test. Poznať účel, na aký sa logický test v programe používa.
		16.4.1.2	Poznať typy logických výrazov, ktorých výstupom je hodnota “true” alebo “false” ako sú[7]: rovnosť (==), väčší (>), menší (<), väčší alebo rovný (>=), menší alebo rovný (<=), nerovnosť (!=, <>) , logický súčin (AND), logický súčet (OR), negácia (NOT).
		16.4.1.3	Vedieť používať logické výrazy v programe.
	16.4.2 Iterácia	16.4.2.1	Rozumieť programovacej štruktúre cyklus. Poznať účel, na aký sa cyklus v programe používa a jeho výhody.
		16.4.2.2	Poznať základné typy cyklov, ktoré sa používajú na iteráciu: daný počet opakovaní (for), opakovanie zatiaľ čo je splnená podmienka (while), opakovanie zatiaľ čo je nesplnená podmienka (repeat)[8].
		16.4.2.3	Vedieť v programe používať základné cykly, ako sú: for, while, repeat[9].
		16.4.2.4	Rozumieť pojmu nekonečný cyklus (infinite loop).
		16.4.2.5	Rozumieť pojmu rekurzia (recursion).
	16.4.3 Podmienené príkazy	16.4.3.1	Rozumieť programovacej štruktúre podmienený príkaz. Poznať, na aký účel sa podmienený príkaz v programe používa.
		16.4.3.2	Vedieť v programe používať podmienený príkaz typu IF…THEN...ELSE.

[4] SISp: napr. jazyk Python pozná len definíciu
[5] SISp: napr. jazyk Python nepozná
[6] SISp napr. v jazyku Python má iný špecifický význam. Ako pole sa používa dátový typ zoznam.
[7] SISp: v zátvorkách sú uvedené príklady značenia v rôznych jazykoch
[8] SISp: napr. jazyk Python nepozná
[9] SISp: napr. jazyk Python nepozná

KATEGÓRIA	SÚBOR ZRUČNOSTÍ	REF.	VYŽADOVANÁ SCHOPNOSŤ
	16.4.4 Podprogramy, funkcie	16.4.4.1	Rozumieť pojmu podprogram bez návratovej hodnoty (procedure). Poznať účel, na aký sa takýto podprogram v programe používa.
		16.4.4.2	Vedieť v programe vytvoriť a nazvať podprogram bez návratovej hodnoty (procedure).
		16.4.4.3	Rozumieť pojmu funkcia (podprogram s návratovou hodnotou, function). Poznať účel, na aký sa v programe používa.
		16.4.4.4	Vedieť v programe vytvoriť a nazvať funkciu (podprogram s návratovou hodnotou, function).
	16.4.5 Udalosti a príkazy	16.4.5.1	Rozumieť pojmu udalosť (event). Poznať účel, na aký sa udalosť v programe používa.
		16.4.5.2	Vedieť spracovať udalosť ako: kliknutie myši, stlačenie klávesu na klávesnici, kliknutie na tlačidlo, časovač (timer).
		16.4.5.3	Vedieť používať bežné funkcie z knižnice funkcií ako sú: math (matematické funkcie), random (generovanie náhodných čísiel), time (funkcie pre dátum a čas).
16.5 Testovanie, ladenie a vydanie programu	16.5.1 Spustenie, testovanie a ladenie programu	16.5.1.1	Rozumieť, že účelom testovania (testing) a ladenia (debugging) programu je odstránenie chýb.
		16.5.1.2	Poznať základné typy chýb v programe ako sú: syntaktické chyby, logické chyby.
		16.5.1.3	Vedieť spustiť program.
		16.5.1.4	Vedieť rozpoznať a opraviť syntaktické chyby v programe ako sú: preklepy, chýbajúce oddeľovače (vrátane odsadenia postupnosti príkazov[10]).
		16.5.1.5	Vedieť rozpoznať a opraviť logické chyby v programe ako sú: chybný logický výraz, nesprávny dátový typ.
	16.5.2 Vydať (release) program	16.5.2.1	Vedieť vyhodnotiť, či boli splnené účel program a požiadavky na jeho funkcie (špecifikácia programu).
		16.5.2.2	Vedieť opísať zhotovený program, jeho účel a význam.
		16.5.2.3	Vedieť rozpoznať možné vylepšenia a rozšírenia programu, ktoré by mohli napĺňať ďalšie súvisiace potreby.
16.6 Vytváranie kódu pomocou nástrojov s technológiami umelej inteligencie (AI)	16.6.1 Nástroje s AI	16.6.1.1	Poznať bežné nástroje, ktoré využívajú AI na generovanie a úpravu programového kódu.
		16.6.1.2	Rozumieť technologickým obmedzeniam nástrojov s technológiami AI v oblasti práce s programovým kódom a uvedomovať si potrebu kritického posudzovania výsledkov.

[10] SISp: Vyskytuje sa u niektorých programovacích jazykov, napr. Python

KATEGÓRIA	SÚBOR ZRUČNOSTÍ	REF.	VYŽADOVANÁ SCHOPNOSŤ
	16.6.2 Používanie nástrojov s AI	16.6.2.1	Používať nástroje s AI na písanie a dokumentovanie programového kódu v rôznych programovacích jazykoch na základe zadaných požiadaviek.
		16.6.2.2	Používať nástroje s AI na analýzu, úpravu a optimalizáciu programového kódu.
		16.6.2.3	Používať nástroje s AI na konverziu programového kódu do iného programovacieho jazyka.