



KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Architektura systemów komputerowych z programowaniem niskopoziomowym [S1S1E>ASK]

Przedmiot

Kierunek studiów

Sztuczna inteligencja/Artificial Intelligence

Rok/Semestr

1/2

Studia w zakresie (specjalność)

–

Profil studiów

ogólnoakademicki

Poziom studiów

pierwszego stopnia

Język oferowanego przedmiotu

angielski

Forma studiów

stacjonarne

Wymagalność

obligatoryjny

Liczba godzin

Wykład

15

Laboratorium

15

Inne

0

Ćwiczenia

0

Projekty/seminaria

0

Liczba punktów ECTS

3,00

Koordynatorzy

dr hab. inż. Tomasz Żok prof. PP
tomasz.zok@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student powinien posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł i wykazywać gotowość do pracy w zespole.

Cel przedmiotu

Przekazanie wiedzy o niskopoziomowych aspektach programowania w języku C. Rozwijanie u studentów świadomości wyzwań i potencjalnych trudności podczas projektowania aplikacji niskopoziomowych. Zapoznanie z architekturą CPU x86 wraz z rozszerzeniami.

Przedmiotowe efekty uczenia się

Wiedza:

1. Ma uporządkowaną wiedzę na temat programowania w języku C.
2. Zna najczęstsze błędy i problemy związane z projektowaniem aplikacji niskopoziomowych.
3. Zna architekturę systemów komputerowych x86 wraz z rozszerzeniami.

Umiejętności:

1. Umie zaprojektować i zaimplementować aplikację w języku C rozwiązującą problemy takie jak przetwarzanie danych tekstowych lub binarnych.
2. Umie korzystać z debugera i z programów do analizy wycieków pamięci by rozwiązać najczęstsze problemy związane z tworzeniem aplikacji niskopoziomowych.
3. Umie stworzyć aplikację w asemblerze x86 z uwzględnieniem optymalizacji wykorzystania operacji SIMD.

Kompetencje społeczne:

1. Jest świadomy, że znajomość architektury systemów komputerowych i umiejętność tworzenia aplikacji niskopoziomowych przekłada się na pełniejsze zrozumienie każdego rodzaju rozwiązań informatycznych.
2. Rozumie, że niskopoziomowe rozwiązania są kluczowe z punktu widzenia bezpieczeństwa systemów informatycznych.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Weryfikacja wiedzy nabytej w ramach wykładu odbywa się poprzez zaliczenie pisemne zawierające pytania otwarte lub pytania wielokrotnego wyboru. Próg zaliczeniowy: 50%.

Umiejętności nabyte w ramach zajęć laboratoryjnych weryfikowane są poprzez ocenę kilku projektów lub zadań praktycznych. Studenci mogą pracować w parach. Każdy projekt oceniany jest osobno. Do zaliczenia laboratoriów wymagane jest otrzymanie co najmniej oceny 3.0 z każdego projektu.

Treści programowe

Wykład:

1. C, Basics, Types, Literals, Expressions, Statements
2. Functions, Arrays, Pointers
3. Structures, Unions and Bit-Fields
4. Dynamic Memory Management, Input and Output
5. Multithreading, Floating-point Numbers
6. Computer Architecture
7. x86 Assembly
8. Test

Laboratorium:

1. Practical guide to C programming and debugging
- 2-3. Project 1: Text File Parsing
- 4-5. Project 2: Binary File Parsing
- 6-7. Project 3: x86 Assembly
8. Late Project Reporting

Tematyka zajęć

Wykłady stanowią kompleksowe wprowadzenie do podstaw informatyki. Tematy obejmują historię i ewolucję informatyki, znaczenie abstrakcji w projektowaniu komputerów oraz prawo Moore'a i jego przyszłe implikacje. Wykłady omawiają również architekturę komputerów, obejmując hierarchie pamięci, potokowanie, przewidywanie i niezawodność systemu. Wprowadzenie do języka programowania C obejmuje typy danych, zmienne i struktury kontrolne.

Ponadto wykłady zagłębiają się w podstawy programowania komputerowego, koncentrując się na zestawach instrukcji, języku asemblera, operacjach i językach wysokiego poziomu, takich jak C i Java. Wyjaśniają struktury danych, instrukcje przesyłania danych i koncepcję programów przechowywanych.

Specyficzne dla programowania w języku C wykłady obejmują literały, wyrażenia, struktury kontrolne i instrukcje. Tematy obejmują liczby całkowite, zmiennoprzecinkowe, znaki i literały łańcuchowe, operatory, pierwszeństwo, konwersje typów, instrukcje if-else, przypadki przełączania i różne pętle. Omawiane są również operatory przypisania, inkrementacji i dekrementacji oraz instrukcje skoku, takie jak break, continue i goto, a także wywołania i zwroty funkcji.

Dalsze wykłady wyjaśniają definicje i wywołania funkcji, inicjalizację i dostęp do tablic oraz użycie

wskaźników, w tym manipulację pamięcią i wskaźniki do funkcji. Omówiono również rolę słowa kluczowego `restrict` w operacjach na wskaźnikach.

Wyjaśniono operacje arytmetyczne w obliczeniach, w tym dodawanie i odejmowanie liczb całkowitych, mnożenie i dzielenie oraz reprezentację zmiennoprzecinkową. Wykłady dotyczą ograniczeń liczb komputerowych, takich jak przepełnienie i niedopełnienie oraz wprowadzają struktury C, związki i pola bitowe wraz z ich zastosowaniami.

Zaawansowane tematy programowania w języku C obejmują algorytmy, funkcje daty i czasu, zarządzanie pamięcią, operacje wejścia/wyjścia oraz obsługę błędów. Wykorzystanie standardowych funkcji bibliotecznych do sortowania (`qsort`) i wyszukiwania (`bsearch`), manipulacja datą i czasem oraz alokacja pamięci (`malloc`, `calloc`, `free`, `realloc`) są szczegółowo opisane. Omówiono również operacje wejścia/wyjścia przy użyciu `printf`, `scanf`, `fread`, `fwrite` oraz metody losowego dostępu do plików (np. `ftell`, `fgetpos`, `fsetpos`, `fseek`, `rewind`), wraz z przykładami i fragmentami kodu.

Na koniec wykłady omawiają hierarchię pamięci i wirtualizację. Wyjaśniają korzyści ekonomiczne wynikające z wykorzystania lokalności i kompromisów między kosztami a wydajnością, projekt hierarchii pamięci, lukę między wydajnością procesora a pamięcią, zużycie energii i operacje pamięci podręcznej. Analizowane są również wskaźniki braku pamięci podręcznej, optymalizacje, pamięć wirtualna i monitory maszyn wirtualnych (VMM) w celu zwiększenia niezawodności, izolacji i bezpieczeństwa systemu.

Podczas sesji laboratoryjnych studenci opanują podstawy programowania w języku C i asemblerze x86. Będą oni realizować różne projekty, które wykorzystują koncepcje omawiane na wykładach.

Metody dydaktyczne

Wykład: prezentacja multimedialna

Ćwiczenia laboratoryjne: prezentacja multimedialna, opracowywanie przykładów przy tablicy, praca w parach

Literatura

Podstawowa

1. Peter Prinz, Tony Crawford „C in a nutshell”

2. David Patterson, John Hennessy „Computer organization and design”

Uzupełniająca

1. Gynvael Coldwind „Zrozumieć programowanie”

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	75	3,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	30	1,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwium/egzaminu, wykonanie projektu)	45	1,50