Programowanie proceduralne i obiektowe

Kierunek: Mechatronika Semestr: 3

Liczba godzin lekcyjnych: 30

Zasady laboratorium.

Student ma prawo do jednej nieusprawiedliwionej nieobecności w semestrze.

Każdy program ma ustalony termin oddania i jest oceniany w skali 0 - 100% jak poniżej:

- 100% - poprawny program oddany w terminie
- 60% - poprawny program oddany po terminie
- 0% - plagiat (niedopuszczalna jest sytuacja gdzie student prezentuje program i nie potrafi wyjaśnić każdej pojedynczej linii kodu)

Ocenianie programów przez prowadzącego rozpoczyna się na zajęciach odpowiadających danemu ćwiczeniu oraz na początku następnych zajęć. Późniejsze oddanie programu jest traktowane jako oddanie po terminie. Programy nie są oceniane na godzinach konsultacji za wyjątkiem ostatniego tygodnia zajęć. Programy nie są oceniane w sesji egzaminacyjnej. Oceny ze wszystkich programów maja jednakowe wagi.

Ćwiczenie 1.

Zapoznanie się z systemem operacyjnym Linux. Podstawowe komendy w systemie Linux. Tworzenie i edycja pliku źródłowego. Kompilacja prostego programu z użyciem narzędzia gcc. Opcje kompilacji. Uruchamianie programu wykonywalnego. Wyświetlanie tekstu na ekranie. Podstawowe typy zmiennych języka C. Wyświetlanie zmiennej w różnych formatach.

Ćwiczenie 2.

Struktura programu w języku C. Instrukcje warunkowe *if*, *else if*, *else*. Instrukcje *switch* i *case*. Operacje arytmetyczne na zmiennych. Porównywanie zmiennych. Napisanie prostego programu wykorzystującego instrukcje warunkowe.

Ćwiczenie 3.

Używanie pętli w języku C. Instrukcje *for, while* i *do…while*. Instrukcja *break*. Pętle zagnieżdżone. Napisanie programu wykorzystującego pętle.

Ćwiczenie 4.

Tablice statyczne w języku C. Operacje na tablicach. Napisanie programu wykorzystującego tablice i pętle.

Ćwiczenie 5.

Funkcje w języku C. Argumenty funkcji i wartość zwracana. Używanie funkcji w programie. Napisanie programu z wykorzystaniem funkcji.

Ćwiczenie 6.

Struktury w języku C. Pola struktur. Napisanie programu z wykorzystaniem struktur.

Ćwiczenie 7.

Alokacja dynamiczna w języku C. Wskaźniki. Funkcje *malloc()* i *free()*.Napisanie programu wykorzystującego alokację dynamiczną. Sprawdzanie programu pod względem wycieków pamięci z użyciem narzędzia valgrind.

Ćwiczenie 8.

Kolokwium 1.

Ćwiczenie 9.

Język C++. Różnice między językami C i C++. Klasy, pola i metody. Struktura programu w języku C++. Zmienne publiczne i prywatne. Wyświetlanie tekstu i zmiennych. Napisanie i kompilacja prostego programu w języku C++.

Ćwiczenie 10.

Napisanie klasy liczb zespolonych *Complex* oraz programu sprawdzającego poprawne działanie klasy.

Ćwiczenie 11.

Przeciążanie operatorów oraz jego wykorzystanie w klasie *Complex*. Napisanie programu sprawdzającego poprawne działanie operatorów.

Ćwiczenie 12-13.

Alokacja dynamiczna w języku C++. Instrukcje *new* i *delete*. Konstruktory i destruktory. Napisanie programu implementującego klasę macierzy dwuwymiarowej *Matrix*.

Ćwiczenie 14.

Kolokwium 2.

Ćwiczenie 15.

Oddanie zaległych programów, poprawa kolokwium, wystawienie ocen.

Wskazówki do ćwiczeń.

1. Komendy systemu Linux.

Należy zapoznać się i przetestować następujące komendy:

cd <nazwa_folderu>

cd ..

ls

ls –l

mkdir

clear

2. Narzędzie gcc

Program kompiluje się używając komendy:

gcc <nazwa_pliku_źródłowego> -o <nazwa_pliku_wykonywalnego>

Jeżeli program nie zawiera błędów, powstanie plik wykonywalny o podanej nazwie, który następnie można uruchomić komendą:

./<nazwa_pliku_wykonywalnego>

Należy zapoznać się z opcjami kompilacji takimi jak -Wall, -O3, -Werror, -pedantic

3. Narzędzie valgrind

Program sprawdza się komendą:

valgrind ./<nazwa_pliku_wykonywalnego>

W przypadku poprawnego napisania programu valgrind powinien zwrócić informację o zerowej liczbie błędów.

4. Wskazówki ogólne pisania programów

• Program ma być napisany przejrzyście i czytelnie.

• Program ma być wyraźnie podzielony na funkcje odpowiadające za poszczególne zadania.

• Unikaj długich funkcji. Staraj się pisać funkcje zawierające kilka, maksymalnie kilkanaście linii kodu.

• Unikaj duplikacji kodu.

• Nazwy funkcji mają być dobrane tak aby wyjaśniały co dana funkcja wykonuje.

• Nazwy zmiennych mają być dobrane tak aby wyjaśniały co dana zmienna reprezentuje.

• Nie ma nic złego w stosowaniu dłuższych nazw funkcji i zmiennych zbudowanych z kilku słów.

• Zastanów się jaką konwencję nazewnictwa chcesz stosować i stosuj ją konsekwentnie w swoich programach.

• Staraj się nie pisać komentarzy na rzecz lepszego nazewnictwa typów, funkcji i zmiennych.

• Nie używaj w programie "liczb magicznych".

• Poprawnie stosuj wcięcia w programie (aby uporządkować wcięcia: zaznacz blok kodu w edytorze tekstowym, wciśnij Tab lub Shift+Tab).