Zadania z Technik Informatycznych

System operacyjny i podstawy sieci

1. Zaczynamy – system operacyjny, środowisko wielodostępne, praca zdalna

1. Zaloguj się na komputerze student.agh.edu.pl (polecenie **`putty lub ssh user@student.agh.edu.pl`** i odpowiedź na pytanie o hasło). Zmień sobie hasło na bardziej Ci przyjazne (polecenie: **`passwd`**). Zapisz sobie nowe hasło. Wyloguj się z systemu (polecenie **`logout`**) i zaloguj się ponownie, używając oczywiście nowego hasła.
2. Sprawdź działanie komendy **`finger`** . Do czego ona służy, jak sprawdzisz informacje o osobach o imieniu Jan? Wypróbuj działanie komendy **`finger | more`**
3. Sprawdź, czy nie ma do Ciebie poczty (**https://poczta.agh.edu.pl**). Wyślij list z pozdrowieniami do siedzącego obok Ciebie kolegi na adres @student.agh.edu.pl a także @local.student.agh.edu.pl. Uruchom program **pine** i sprawdź, czy list na drugi adres też doszedł. Jaka jest różnica pomiędzy adresem @student.agh.edu.pl a @local.student.agh.edu.pl?.
4. Rozpoznaj nazwę i położenie Twojego bieżącego katalogu (polecenie: **`pwd`**). Wypróbuj działanie komend **`mkdir cd`** oraz **`md`**. Utwórz katalogi **`adresy`** oraz **`znajomi`** Zanotuj sobie do czego służą te polecenia i czym się one różnią**.**
5. Sprawdź jakie pliki są w Twoim osobistym katalogu (**`ls`**). Dowiedz się, jakie opcje ma polecenie ls (**`man ls`**). Wykonaj polecenie **`cd /`** a następnie polecenie **`ls -la`**, wyświetlona została zawartość katalogu głównego (tzn. katalogi i pliki znajdujące się w nim) w formie prostej. Dowiedz się co oznaczają poszczególne oznaczenia na ekranie. Co oznacza literka “d” w pierwszej kolumnie? Przejdź do swojego katalogu. Wykonaj teraz polecenia: 1: **`ls`**, 2: **`ls -l`**, 3: **`ls -a`** , 4: **`ls –la`**. Zauważ, że opcja **`-a`** pozwala na oglądanie plików ukrytych tzn. których nazwa rozpoczyna się od kropki (np. **`.plan`**).
6. Po zalogowaniu się na serwerze student.agh.edu.pl uruchom Midnight Commander (polecenie: **`mc`**) – nakładkę na system operacyjny. Zapoznaj się z plikiem pomocy (F1). Zapisz sobie, do czego służą tzw. klawisze funkcyjne (F1...F10). Poświęć kilkanaście minut na zapoznanie się z objaśnieniami zawartymi w pomocy do Midnight Commandera. Obserwując zmiany w panelach Midnighta, wykonuj dalsze zadania.
7. Tworzenie plików przy użyciu polecenia **`cat > nowy_plik`.** Wykonaj następujące polecenia: przejdź do katalogu **`adresy`,** wykonaj komendę **`cat > tomek .`**(aby zakończyć naciśnij **`^D)`**, sprawdź poleceniem **`ls –la`** czy został utworzony plik. Obejrz zawartośc utworzonego pliku komendą **`cat nazwa_pliku`** oraz **`more nazwa_pliku`** Sprawdź działanie edytorów **`mcedit`** oraz **`pico`** edytując ten plik.
8. Skopiuj utworzony plik **tomek** z katalogu **adresy** do katalogu **znajomi ,** sprawdź czy plik został skopiowany. Skopiuj plik **tomek** z katalogu **adresy** do katalogu **znajomi** pod inna nazwą zmień nazwę pliku **`tomek`** na **`tomasz`** (polecenie **`mv`**).

Aby zaliczyć ćwiczenie wyślij list z pozdrowieniami do prowadzącego zajęcia oraz kopię listu (**cc**) do prowadzącego wykład. W liście podaj odpowiedź na pytanie: do czego służą komendy: **man**, **finger**, **more**, **passwd**, polecenia z poprzedniego punktu i ewentualne inne – jeśli potrafisz).

Zadanie domowe: Połącz się z domu z serwerem student.agh.edu.pl (program putty) i wyślij list do prowadzącego z użyciem programu pine lub mail.
Na następne przygotuj (pendrive, dysk sieciowy, mail?) Twoje zdjęcie (zeskanowane do pliku w formacie **gif** lub **jpg**).

2. Sieć TCP/IP – podstawowe usługi

1. Za pomocą komendy ping sprawdź łączność z serwerem **galaxy.agh.edu.pl**
2. Sprawdź jakie numery IP mają nast. hosty: student.agh.edu.pl, www.onet.pl
   Sprawdź jaką nazwę nosi host o IP: 149.156.98.1 (komenda `nslookup` albo `host`)
3. Sprawdź przez ile routerów (węzłów) prowadzi trasa do hosta: www.debian.org
4. Prześlij za pomocą scp (lub winscp) plik z Twoim zdjęciem z komputera lokalnego na serwer student.
5. Pobierz plik ze strony www przy użyciu komendy wget. Przykład:
   `wget` `http://cdn-3.famouslogos.us/images/linux-logo.jpg`
6. `Uruchom sesję ftp w okienku Midnight Commandera (F9+left+ftp link+adres) i skopiuj z serwera ftp.icm.edu.pl plik /pub/README`
7. Utwórz sobie na serwerze **student** katalog **public_html** , w tym katalogu utwórz plik **index.html** o następującej zawartości:

```
<html>
<title>Strona imie_nazwisko (w dopełniaczu) </title>
<h1> imie_nazwisko </h1>
<p><a href="http://www.agh.edu.pl/">Strona główna AGH</a></p>
<p><a href="http://mops.uci.agh.edu.pl/faq/#www">WWW w UCI</a></p>
<p><a href="http://mops.uci.agh.edu.pl/">Informacje o Mopsie</a></p>
<p><a href=" http://pl.html.net/tutorials/html/ ">Kurs HTML</a></p>
<p><i>Copyright (c) 2016</i></p>
</html>
```

udostępnij dla innych zawartość tego katalogu :
(polecenie: **`chmod -R a+rx $HOME/public_html`**, oraz: **`chmod a+x $HOME`**)
W konsoli graficznej uruchom przeglądarkę www i obejrzyj stronę spod adresu:
`http://student.agh.edu.pl/~nazwa_twojego_konta`

8. Udoskonal zawartość swojej strony WWW np. wzbogacając ją o grafikę
   (Twoje zdjęcie z ćw. nr 1?)

Aby zaliczyć ćwiczenie, wyślij list do prowadzącego zajęcia (kopia do wykładowcy) i opisz działanie komend `nslookup` oraz `traceroute`. Do listu dołącz (jako załącznik) pobrany plik tekstowy **README**

Programowanie w C

3. Zaczynamy programowanie w C – stdio

1. Zaloguj się na serwerze student. Uruchom Midnight Commander. Rozpocznij edycję jakiegoś pliku. Wywołaj pomoc. Przypomnij sobie wszystkie skróty klawiszowe odnoszące się do edycji i ich znaczenie. Zapisz je sobie w pliku.
2. Utwórz katalog roboczy (np. cw03), w którym będziesz przechowywać programy napisane dzisiaj w języku C. Wykorzystując edytor Midnight Commander, przepisz poniższe programiki umieszczając je w plikach o rozszerzeniu "**c**", np. **moj01.c**. Z otrzymanych plików źródłowych utwórz programy za pomocą kompilatora **gcc**, przykładowo: **`gcc moj01.c -o moj01 .`** Wykonaj te programy wywołując je przez ich nazwę np. **moj01** lub **./moj01.** Modyfikuj teksty źródłowe programów i obserwuj jak zmienia się ich wynik.
3. Na zakończenie ćwiczeń napisz program w języku C wypisujący na ekranie w kolejnych wierszach: Twoje imię i nazwisko, nazwę Uczelni, nazwę Wydziału, rok kalendarzowy, miejsce i godzinę ćwiczeń, stopień naukowy i nazwisko osoby prowadzącej ćwiczenia.

Gdy upewnisz się, ze Twój program jest poprawny, wyślij wersję źródłową jako załącznik do sprawozdania z ćwiczeń.

```
// a010.c
// wypisywanie tekstu - Styl c
#include <stdio.h>
main()
{
  printf("====================\n");
  printf("Dzien dobry");
  printf("\n\n\n");
  printf("Dobry wieczor""\n\n");
  printf("%40s","Dobranoc\n");
  printf("%20d",2016);
return 0;
}
-------------------------------
// a020.c
// kalkulator 1 - Styl c
#include <stdio.h>
main()
{
  printf("====================\n");
// formatowanie liczb calkowitych
  printf("%10d",10+1);
  printf("%10d\n",10+2);
  printf("%10d",10+3);
  printf("%10d\n",10+4);
// formatowanie liczb rzeczywistych
  printf("%15.3f\n",3.14*10);
  printf("%15.3f",3.14*100);
  return 0;
}
```

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

4. Zmienne, pętle, instrukcje warunkowe

1. Zaloguj się na serwerze student. Utwórz katalog **cw04** i zapisuj w nim wykonywane dzisiaj prace. Jako pierwszy, napisz program (**cw04-01**), który oblicza kwadraty pierwszych dwunastu jeden liczb naturalnych z użyciem pętli for, pętli while i pętli do-while.

2. Napisz program (**cw04-02**) który, zaprezentuje na ekranie tabliczkę mnożenia do 100.

3. Napisz program (**cw04-03**) który rozwiązuje równanie kwadratowe, przy czym współczynniki równania są podane wewnątrz programu a wyniki obliczeń są wyświetlane na ekranie.

4. Napisz program (**cw04-04**), który rozwiązuje równanie kwadratowe, przy czym współczynniki równania powinny być podane przez użytkownika w odpowiedzi na pytania zadawane przez program.

5. Napisz program (**cw04-05**), rozwiązujący co najmniej jedno równanie kwadratowe. Współczynniki równania powinny być podawane przez użytkownika w odpowiedzi na pytania zadawane przez program. Po wyświetleniu wyników program powinien zadawać użytkownikowi pytanie czy chce on podać następne dane do obliczeń.

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

5. Obsługa plików, wskaźniki, tablice

1. Napisz program, który oblicza n!,
2. Napisz program, rozwiązujący kilka równań kwadratowych, wczytujący dane z pliku.
3. Utwórz plik tekstowy, (prop. nazwa: **dane**) zawierający kilkanaście liczb całkowitych. Napisz program, który wybierze największą oraz najmniejszą spośród tych liczb.
4. Napisz program, który oblicza n!, dla podanej przy jego wywołaniu wartości n (np. **`cw06-01 5`**).

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

6. Funkcje, instrukcje wyboru, liczby losowe

1. Napisz program z użyciem funkcji, która oblicza n! (skorzystaj z kodu z ćw. 5.1),
2. Napisz funkcje (i odpowiedni do jej testowania program), które obliczają: sumę, średnia arytmetyczną i geometryczną liczb z tablicy (do której liczby wczytasz wcześniej z pliku tekstowego jak w poprzednim zadaniu).
3. Napisz następujące funkcje: (a) obliczającą pole prostokąta, (b) obliczająca pole koła, (c) obliczająca pole trójkąta oraz program, który posiada menu umożliwiające wariantowe i wielokrotne wykonywania obliczeń.
4. Napisz funkcję, która wylosuje liczbę naturalną z założonego zakresu (np. 0÷2000), a następnie zgadnie tę liczbę metodą wyszukiwania binarnego. Funkcja powinna wypisywać wyniki kolejnych prób zgadywania.

7. Sortowanie, struktury

1. Napisz program, który wczyta do tablicy ciąg liczb z utworzonego wcześniej pliku, wypisze tablicę na ekranie, posortuje elementy tablicy malejąco i ponownie wypisze tablicę na ekranie a następnie posortuje elementy tablicy rosnąco i ponownie wypisze tablicę na ekranie.
2. Napisz program - kalkulatorek wykonujący następujące działania: dodawanie, odejmowanie, mnożenie, dzielenie, i ewentualnie jeszcze jakieś inne działania. Dane do programu (liczby i symbol operacji, np. +, -, x, / itp.) powinny być wczytywane z linii komend. (Pyt. dodatkowe: dlaczego nie można zastosować znaku * ?).
3. Napisz strukturę reprezentującą punkt w przestrzeni dwuwymiarowej. Wykorzystaj ją w funkcji obliczającej odległość między dwoma punktami.
4. Napisz strukturę reprezentującą w przestrzeni dwuwymiarowej okrąg. Wykorzystaj ją w funkcji rozstrzygającej problem czy podany punkt należy do wnętrza, brzegu lub zewnętrza okręgu.

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

Programowanie w C++

8. Wprow. do C++ – iostream, typy string i bool

1. Napisz program wypisujący tabliczkę mnożenia o *m* kolumnach i *n* wierszach (zwróć uwagę na wyrównanie wypisywanych wartości w kolumnach).
2. Napisz program rozwiązujący równania kwadratowe, współczynniki powinny być wczytywane z klawiatury, a program po wypisaniu wyników powinien pytać czy liczyć jeszcze raz (odp. tekstowe: „tak” lub „Nie” – program powinien ignorować wielkość liter).
3. Napisz program, który będzie generował losowe *n*-literowe hasło zawierające co najmniej jedną literę wielką, cyfrę i znak specjalny. W wygenerowanym haśle nie powinny sąsiadować dwie spółgłoski lub samogłoski.

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

9. Strumienie, strumienie plikowe i stringowe

1. Napisz program rozwiązujący kilka równań kwadratowych, dane powinny być wczytywane z pliku, a wyprowadzane na ekran i do pliku.
2. Napisz program, który we wskazanym pliku zamieni pierwsze litery wyrazów na wielkie, a wynik umieści w nowym pliku.
3. Napisz program, który w wielowierszowym tekście wczytanym z klawiatury policzy wiersze, słowa, znaki „czarne” i spacje (wykorzystuj strumień stringstream).

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

10. Funkcje (przeładowanie), tablice, wskaźniki i referencje

1. Napisz dwie funkcje o tej samej nazwie (`pierw`), które obliczą:
   (a) pierwiastek kwadratowy liczby x,
   (b) pierwiastek n-tego stopnia z liczby x.
   (Zastosuj te funkcje w prostym programie, np. wypisującym pierwiastki kwadratowe i 3-go stopnia liczb od 1 do 5).
2. Napisz program (z użyciem funkcji) który wczyta (np. z pliku) elementy macierzy kwadratowej (o rozmiarze 3) oraz policzy:
   (a) sumę elementów na diagonali,
   (b) iloczyn tej macierzy przez liczbę skalarną,
   (c) iloczyn tej macierz przez samą siebie.
   Nie zapomnij o wypisaniu macierzy wynikowej.
   Dla bardziej ambitnych: wczytaj z pliku elementy macierzy bez założenia jej rozmiaru (kolejne wiersze macierzy powinny być w kolejnych wierszach pliku).
3. Posługując się wskaźnikami do funkcji napisz program, który z tablicy wczytanych z pliku liczb obliczy:
   (a) sumę elementów,
   (b) średnią arytmetyczną,
   (c) średnią geometryczną.
   Każdy z tych podpunktów powinna realizować osobna funkcja, wywoływana za pomocą wskaźnika do funkcji.
   Dla bardziej ambitnych: rozszerz zadanie o szukanie wartości środkowej (mediany).

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

11. Klasy, obiekty

1. Utwórz klasę (o nazwie *punkt*) opisującą punkt w układzie kartezjańskim. Klasa ta powinna mieć funkcje składowe, które umożliwiają:
   (a) przypisanie współrzędnych punktu,
   (b) wypisanie współrzędnych punktu,
   (c) obliczenie odległości punktu od innego punktu (tej samej klasy),
   (d) przesunięcie punktu o wektor [x,y]
   Dla bardziej ambitnych: rozszerz zadanie o obliczenie współrzędnych punktu obróconego względem początku układu wsp. o kąt alfa
2. Utwórz klasę opisującą okrąg w układzie kartezjańskim.
   Klasa powinna umieć zainicjować wartości opisujące okrąg (zarówno postaci punktu środka i dł. promienia, jak i wsp. środka i dł. promienia),
   wypisać informacje o sobie (środek, długość promienia),
   policzyć swoje pole,
   określić liczbę punktów wspólnych z obiektem takiej samej klasy.
   Skorzystaj z klasy *punkt* utworzonej w poprzednim zadaniu.

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

12. Dziedziczenie, polimorfizm

1. (a) Utwórz klasę opisującą ogólne właściwości płaskiej figury geometrycznej (zdefiniuj wirtualne funkcje składowe liczące: pole, obwód oraz wypisujące informacje o figurze).
   (b) Utwórz klasy pochodne z niej (np. prostokąt, kwadrat) i zredefiniuj wirtualne funkcje składowe (pole, obwód).
   (c) Utwórz na podstawie (a) klasę opisującą figurę przestrzenną (objętość).
   Dla bardziej ambitnych: (d) Zdefiniuj klasę potomną opisującą figury przestrzenne (stożki, graniastosłupy) dziedzicząc po klasie z (c) oraz (b)
2. Utwórz klasę do podawania danych z walidacją wartości (o nazwie np. `dana`).
   Klasa powinna zawierać funkcje: wypisującą przechowywaną wartość, wypisującą komunikat zachęty i wczytującą wartość z klawiatury, sprawdzającą, czy wartość jest poprawna (np. o nazwie `isvalid`()), a także podającą przechowywaną wartość `get`().
   Klasa powinna zawierać konstruktor kopiujący, konstruktor z komunikatem zachęty.
   Na podstawie tej klasy (`dana`) utwórz klasy potomne do przechowywania wartości długości boków (nie mogą być ujemne), a także kątów (muszą być z zakresu 0-360)

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

13. Operatory, struktury danych, kontenery

1. Utwórz klasę opisującą wektor w układzie kartezjańskim, zaimplementuj operatory liczące: długość wektora (moduł), iloczyn wektora i liczby skalarnej, iloczyn skalarny z innym wektorem.

2. Wykorzystując szablon *vector* z STL utwórz własną klasę przechowujący stringi. Wczytaj do niej zawartość wybranego pliku tekstowego wiersz po wierszu, wypisz jego elementy, wysortuj je i znajdź wiersz najkrótszy.

Dla bardziej ambitnych:

3. Wczytaj plik tekstowy zawierający imiona. Policz powtarzające się słowa i wypisz ich statystykę. W programie posłuż się szablonem STL o nazwie *map* (tablicą asocjacyjną).

Aby zaliczyć ćwiczenie, wyślij list z załącznikami, zawierającymi postać źródłową (a nie skompilowaną!) napisanych dzisiaj programów do prowadzącego zajęcia (kopia do wykładowcy).

14. Zaliczenie