# ZADANIA

# Z

# C/C++

C++11/C++14/C++17

Kurs 1

https://www.youtube.com/watch?v=Z_DPVKlk2nA&list=PLKmH7u1gA9hpkNGJJFA5fB8MfPXWDQ1Fy

Kurs 2

https://www.youtube.com/watch?v=fuXnZgmT9yg&list=PLKmH7u1gA9hrP_5F8LEtU1T61YjN_M5QN

*Przygotował:*

Tomasz Jaśniewski

## *UMIEJĘTNOŚCI PIERWOTNE*

Przybliżona lista umiejętności:

- znajomość podstawowego schematu pliku .cpp
- instrukcja **`if-else`**, warunek
- pętle **`for`**,**`while`**,**`do..while`**
- znajomość podstawowych typów liczbowych, tworzenie zmiennych tego typu
- znajomość typu **`char`**, **`bool`**
- operacje na zmiennych liczbowych (dodawania, mnożenia, reszty z dzielenia itp.),
- operacje logiczne (operatory porównania)
- tablice klasyczne (język C)
- **`vector`**<>, **`array`**<>
- **`cout`**, **`cin`**

***UWAGA! Jeżeli zadanie czegoś nie określa jasno, to znaczy, że masz pełną dowolność interpretacji zgodnie z logiką i nie wbrew treści zadania. Jeżeli masz kilka pomysłów na zadanie - zrób je na wiele sposobów. Jeżeli zadanie zrodziło pomysł na jakiś wspaniały program - napisz go!***
***Liczba w nawiasach na koniec zadania do liczba punktów za zadanie a równocześnie przybliżona informacja o trudności zadania w danej grupie umiejętności.***

1. Pobierz 3 liczby z klawiatury. Znajdź największą z nich. Wyświetl sumę pozostałych liczb (mniejszych od największej) tyle razy, ile wynosi wartość największej. [1]
2. Pobraną z klawiatury liczbę całkowitą, zweryfikuj pod kątem parzystości. Wyświetl "tak" lub "nie", gdy jest lub nie jest parzysta. [1]
3. Pobierz liczbę całkowitą z klawiatury i sprawdź, czy jest podzielna przez 3. Jeżeli nie jest, to sprawdź, czy jest podzielna przez 5. Dla każdej z odpowiedzi wyświetl informujący o tym komunikat, czyli: podzielna przez 3; lub przez 5; albo ani przez 3 ani przez 5. [1]
4. Pobierz znak z klawiatury. Jeżeli jest to samogłoska, poinformuj o tym, a jeżeli nie, to sprawdź czy jest to spółgłoska, jeżeli tak, to poinformuj o tym, a jeżeli nie, to sprawdź czy jest to znak interpunkcyjny i poinformuj o tym. Poinformuj też o fakcie niespełnienia żadnego z warunków. Uwzględnij alfabet angielski. [1]
5. Masz funkcję: **`(((a1 + a2)*a3)-a4)/a5`**. Pobierz z klawiatury każdą ze zmiennych a1 do a5 i oblicz wartość. Uwaga, zaprojektuj program tak, aby w przypadku dzielenia przez zero informował o tym i nie wykonywał działania.[1]
6. Pobierz 5 liczb i wyświetl, ile było parzystych, a ile nieparzystych z tych liczb. [1]
7. Pobierz 5 znaków i policz, ile było wśród nich samogłosek. Pobieranie zrób w pętli. [1]
8. Pobierz z klawiatury 2 liczby wymierne (zmiennoprzecinkowe) duże (**`double`**) i znak działania (jeden z tych: *,+,-,/). W zależności od znaku wykonaj na pobranych dwóch liczbach odpowiednie działanie informując o wyniku. Zwróć uwagę na dzielenie przez 0 i uniemożliw wykonanie takiego dzielenia np. poprzez wyświetlenie odpowiedniego komunikatu.
   Przykład: dla pobranej liczby 3 i 9.5 oraz znaku '+' zwróć sumę 12.5!
   Uwaga! Sprawdź też operację przy zamianie liczb miejscami. Jeżeli wynik różni się od wcześniejszego, wyświetl oba. Jeżeli jest taki sam, wyświetl go tylko jeden raz. Np. dla liczb 5 i 2 oraz znaku '/' (dzielenie) trzeba wyświetlić 5/2 i 2/5, ponieważ dają różny wynik! [2]
9. Pobierz liczbę z klawiatury i wykonaj na niej poniższe operacje:
   jeżeli liczba była ujemna, zmniejsz ją o 1;
   jeżeli liczba była dodatnia, zwiększ ją o 1;

jeżeli była zerem, pozostaw bez zmian;
Następnie określ czy uzyskana po operacjach liczba jest parzysta? (tak/nie) [1]

10. Wyświetl liczby całkowite od 1 do 100. [1]
11. Wyświetl liczby całkowite od 100 do 10, pomijając podzielne przez 7. [1]
12. Wyświetl liczby całkowite od -25 do 25 z pominięciem 0. [1]
13. Wyświetl liczby całkowite od 1 do 1000 z pominięciem liczb podzielnych przez 11 oraz podzielnych przez 5. [1]
14. Odgadnij wzór ciągu a następnie wyświetl jego 100 kolejnych elementów. Początek ciągu: 3,1,2,1,3,1,2,1, ... [1]
15. Odgadnij wzór ciągu a następnie wyświetl jego 100 kolejnych elementów. Początek ciągu: 1,2,2,3,3,3,4,4,4,4, ... [1]
16. Odgadnij wzór ciągu a następnie wyświetl jego 100 kolejnych elementów. Początek ciągu: 100,99,97,94,90,85, ... [1]
17. Odgadnij wzór ciągu a następnie wyświetl jego 100 kolejnych elementów. Początek ciągu: 6,2,8,3,10,4,12,5,14,6, ... [1]
18. Pobieraj liczbę z klawiatury i wyświetlaj jej dwukrotność tak długo aż zostanie wpisana wartość pomiędzy 1 a 10 włącznie. [1]
19. Wyświetl wszystkie liczby podzielne przez 6 ze zbioru od 0 do 1000. [1]
20. Wyświetl liczby od 1 do 100 z wyjątkiem podzielnych przez 5 za pomocą każdej z poznanych pętli. [1]

## *UMIEJĘTNOŚCI FUNDAMENTALNE*

Um. Fundamentalne zawierają um. pierwotne. Przybliżona lista umiejętności fundamentalnych:

- tablica dwuwymiarowa (lub np. **vector<vector<>>** )
- funkcje (różne przekazywanie argumentów do funkcji, zwracanie czegoś przez funkcję, przekazanie tablicy [] do funkcji, referencja)
- typ **void**
- przeciążenie funkcji
- losowość (biblioteka **<random>** oraz funkcja **rand**())
- napisy / typ **string**
- operacja na napisach, w tym funkcje obsługujące napisy (takie jak **find**, **size**() itp. )
- znajomość funkcji konwertujących (jak **stoi()** itp.)
- operacje na plikach (podstawowy odczyt i zapis) ( **>> , <<** , **getline()** )
- wstawianie pobranych z pliku danych do różnych zmiennych/tablic/kontenerów
- tworzenie struktury (**struct**) jako własnego typu złożonego
- proste rzutowanie: (typ)zmienna
- znajomość struktury danych (teoretyczna) : stos, kolejka, lista jedno i dwukierunkowa, drzewo binarne itp.

***UWAGA! Jeżeli zadanie czegoś nie określa jasno, to znaczy, że masz pełną dowolność interpretacji zgodnie z logiką i nie wbrew treści zadania. Jeżeli masz kilka pomysłów na zadanie - zrób je na wiele sposobów. Jeżeli zadanie zrodziło pomysł na jakiś wspaniały program - napisz go!***
***Liczba w nawiasach na koniec zadania do liczba punktów za zadanie a równocześnie przybliżona informacja o trudności zadania w danej grupie umiejętności.***

1. Wyświetl swoje imię w pętli tyle razy, ile jest w tym imieniu samogłosek. [1]
2. Wylosuj trzy liczby. Znajdź największą liczbę. Wyświetl sumę pozostałych 2 liczb tyle razy, ile wynosi ta maksymalna. Np. dla 1,2,3. wyświetlasz 3 razy sumę 1 i 2. [1]
3. Losuj w pętli dowolną literkę małą lub dużą tak długo, aż nie zostanie wylosowana mała literka 'z' lub duża 'A'. Podaj ilość losowań po zakończeniu działania pętli. [1]
4. Wyświetl 10 losowych liczb całkowitych od 0 do 20. [1]
5. Wyświetlaj losowe liczby całkowite od 0 do 100 tak długo aż wypadnie 100. Wyświetl informację, ile losowań nastąpiło, zanim przerwała się pętla przy wartości 100. [1]
6. Wypełnij 10 elementową tablicę losowymi liczbami całkowitymi z zakresu od -10 do 10. Ile jest liczb ujemnych w tak wylosowanej tablicy, ile jest dodatnich, ile parzystych a ile nieparzystych? [1]
7. Wylosuj 20 liczb z zakresu od 0 do 1000 każda i wyświetl największą z nich. [1]
8. Wylosuj i wyświetl liczbę ułamkową (typ **double**) w zakresie 0 do 1 z maksymalnie 3 miejscami po przecinku. [1]
9. Stwórz tablicę 20 losowych liczb typu **double** z zakresu -1 do 1 i precyzją do 3 miejsc po przecinku. Znajdź najmniejszą i największą wylosowaną liczbę w tej tablicy. [1]
10. Wylosuj liczbę całkowitą L (między 20 a 30 włącznie), oraz pobierz z klawiatury znak Z. Wyświetl L-krotnie znak Z. [1]
11. Wylosuj małą literę alfabetu angielskiego i wyświetl ją. Przy losowaniu wykorzystaj fakt, iż typ **char** można traktować jak typ **int**. (przypominam, że **'a'** to **97, b to 98 itd.** Sprawdź wynik działania instrukcji i wykorzystaj ten fakt w zadaniu.
    ```
    cout << (int)'a';
    cout << (char)97;
    ```
    [1]
12. Dla tablicy 20 elementowej liczb typu **double** wylosuj liczby losowe z zakresu <1;10> z dokładnością do 1-ego miejsca po przecinku (np. 1.4 itd.) [1]

13. Pobierz znak z klawiatury, a następnie wylosuj 2 liczby całkowite **a**, **b** w zakresie od 5 do 10 każda. Narysuj **a** wierszy, w których będzie **b** znaków, które pobrałeś z klawiatury.

    Np. dla znaku # i liczb 3 i 8 narysuj 3 wiersze w każdym po 8 znaków #. [1]

14. Z tablicy 1000 losowych liczb całkowitych wyświetl te podzielne prze 4, ale nie podzielne przez 5. (masz dowolność tworzenia tablicy) [1]

15. Stwórz pętle, która losuje liczbę z zakresu od 1 do 1000, ale po każdym kroku zmienia zakres losowania od ostatnio wylosowanej liczby do 1000. Pętla przerywa się, gdy wylosujesz 1000.

    Np. pierwsze losowanie jest z zakresu <1;1000> i wypada 200. Zatem drugie losowanie jest z zakresu <200;1000> i np. wypada 254. Zatem trzecie losowanie ma być dokonane z zakresu <254;1000>.

    Losowania kończą się, gdy padnie 1000. [1]

16. Wypełnij tablicę losowymi liczbami całkowitymi, następnie wszystkie parzyste wyzeruj, a nieparzystym zmień znak, następnie wyświetl tę tablicę od tyłu. (od elementu ostatniego do pierwszego). [1]

17. Pobierz lub wylosuj liczbę całkowitą n z zakresu <5;12>. Następnie dla tego n narysuj 'kwadrat' złożony ze znaków '#', który ma n wierszy i n kolumn (czyli n znaków w wierszu). Np.:
    #####
    #####
    #####
    #####
    #####
    [1]

18. Dla losowego n z zakresu <5;12> narysuj szachownicę ze znaku '#' oraz spacji, np.: dla n=5:

    # # #
     # #
    # # #
     # #
    # # #
    [1]

19. Dla losowego n z zakresu <5;12> narysuj 'kwadrat' n-wierszy i n-kolumn, którego krawędź to znak '#' a wnętrze jest puste (spacje). Np.:
    #####
    #   #
    #   #
    #   #
    #####
    [1]

20. Dla losowego n z zakresu <5;12> narysuj 'trójkąt prostokątny'. Przykładowo dla n=5 ma to wyglądać tak:

    #
    ##
    ###
    ####
    #####

    W ostatnim wierszu ma być 5 (ogólnie n) znaków '#', a w pierwszym jeden znak '#'.

    Następnie narysuj podobny trójkąt, ale tak, by najdłuższy bok nie był z lewej, ale z prawej strony.

        #
       ##

```
  ###
 ####
#####
```
[2]

21. Wylosuj nieparzyste n z zakresu <7;21> (jak będzie parzyste losuj ponownie) a następnie narysuj 'piramidę', gdzie w pierwszym wierszu jest jeden znak '#', w drugim 3-y znaki '#' itd. a w ostatnim wierszu ma być n znaków.

    (Uwaga! Piramida ma być symetryczna, czyli środkowy '#' w każdej linii ma być w tej samej pionowej linii). Np. dla n=3:

```
  #
 ###
#####
```

    Następnie narysuj 'diament' czyli to samo, ale po osiągnięciu najdłuższej linijki znaków '#' niech się kolejne linie skracają aż do ostatniej linijki z jednym znakiem '#'.

```
  #
 ###
#####
 ###
  #
```
[2]

22. Narysuj 'kwadrat' złożony ze znaków '#' dla dowolnego n z przedziału <5;12>. Jedno z wewnętrznych pól (wylosuj, które) ma być znakiem '@' a nie znakiem '#'.

    Np.:

```
#####
##@##
#####
#####
#####
```

    Następnie zrób to samo, ale kwadrat ma być pusty w środku (spacje) a posiadać tylko krawędzi.

    [2]

23. Dla naturalnego N >= 3 narysuj 'choinkę' złożoną z piramid (jak w zadaniu 21), z tym, że każda kolejna część choinki ma mieć 'piramidę' o jeden wiersz dłuższą a łączna ilość 'piramid' tworzących 'choinkę' ma być równa N. Pierwsza 'piramida' tworząca czubek 'choinki' ma być złożona z 3 linijek:

```
  #
 ###
#####
```

    Dla N=3:

```
    #
   ###
  #####
    #
   ###
  #####
 #######
    #
   ###
  #####
 #######
#########
```

    Oczywiście 'choinka' ma być symetryczna. [2]

24. Stwórz funkcję pozwalającą wykonać działanie potęgowania liczby wymiernej do potęgi naturalnej i zwracającą wynik takiej potęgi. Stwórz przeciążenie funkcji dla potęgi całkowitej (może być potęga ujemna). [1]
25. Stwórz funkcję, która wyświetla przekazaną tablicę i zwraca jej największy element (tablicę liczb **double**). [1]
26. Stwórz funkcję, która zwraca zaokrąglenie (typu **long long int**) liczby wymiernej (**double**): zaokrąglenie do góry, gdy część ułamkowa >= .50 i do dołu w przeciwnym wypadku. [1]
27. Stwórz funkcję, która zmienia wartość zmiennej (niczego nie zwracając instrukcją return!) w taki sposób, by część ułamkowa przekazanej zmiennej podwoiła się. (3.22 -> 3.44; 3.6 -> 4.2 itd.) [1]
28. Stwórz funkcję, która zwraca wartość bezwzględną przekazanej doń liczby. [1]
29. Sprawdź w Internecie lub zapytaj sąsiadkę, co robią funkcje **abs() i pow()**. Czy wymagają dodatkowej biblioteki? Jeżeli wiesz bez sprawdzania, to trudno, daruj sobie to zadanie. [0]
30. Stwórz funkcję, która dla trzech podanych jej argumentów (**double**) zwraca podwojony największy z nich. [1]
31. Stwórz funkcję, która oblicza wartość funkcji liniowej **ax+b** w punkcie dla podanych niezbędnych argumentów (współczynniki a, b oraz wartość punktu x) i zwraca tę wartość. Korzystaj z **double**. [1]
32. Stwórz funkcję obliczającą wartość funkcji liniowej dla podanego x. (jak w zadaniu 31.). Następnie stwórz losową tablicę dwuwymiarową [100][2] (sto wierszy, dwie kolumny) i potraktuj wiersz jak punkt w układzie współrzędnych. Dla funkcji liniowej 2x+3 sprawdź, które z punktów są nad wykresem funkcji, które są pod wykresem funkcji, a które leżą na wykresie funkcji. Podaj statystykę, ile punktów leży pod, nad i na linii. [2]
33. Losuj liczby całkowite z przedziału <0,1000> tak długo, aż zostanie wylosowana liczba 1000. Każdą liczbę wylosowaną podzielną przez 10 dodaj do pliku liczby.txt z zachowaniem zasady: jednak liczba w wierszu. [1]
34. Stwórz program, który odczyta wszystkie liczby z pliku tekstowego i wyświetli je. Zasady pliku:

    - każda liczba jest typu **int**

    - w jednym wierszu znajduje się jedna liczba

    (Plik może mieć dowolną zawartość zgodną z zasadami pliku, po prostu utwórz go samodzielnie) [1]
35. Stwórz program, który odczyta wszystkie liczby z pliku tekstowego i wyświetli je. Zasady pliku:

    - każda liczba jest typu **int**

    - w jednym wierszu znajdują się dwie liczby oddzielone średnikiem

    (Plik może mieć dowolną zawartość zgodną z zasadami pliku, po prostu utwórz go samodzielnie) [1]
36. Stwórz program, który odczyta wszystkie liczby i teksty z pliku a następnie je wyświetli. Zasady pliku:

    - każda liczba jest typu **int**

    - w jednym wierszu znajdują się 3 liczby oddzielone średnikiem, następnie jest dowolnie długi napis, np:

    1;231;3423;dowolnie długi napis

    321;345;67;inny długi napis

    itd.

    Pobrane liczby umieść w tablicy/**vector**'ze. Pobrane napisy umieść w jednym długi napisie i wyświetl.

(Plik może mieć dowolną zawartość zgodną z zasadami pliku, po prostu utwórz go samodzielnie) [2]

37. Stwórz program, który odczyta wszystkie liczby i teksty z pliku a następnie je wyświetli. Zasady pliku:

    - każda liczba w pliku jest typu **double** (uwaga, np.: 123.21 to liczba typu **double**, ale sprawdź, czy twój program konwertuje taką liczbę pobierając z pliku ciąg z przecinkiem czy z kropką? ‘123.21’ albo ‘123,21’)

    - w jednym wierszu znajdują się liczby i napisy oddzielone średnikiem, jednak ich kolejność i ilość w wierszu nie jest znana! Np.:

    1;231;3423;dowolnie długi napis;345456;inny napis

    napis;321;345;67;inny długi napis;8893;tekst;123123;1212;12;tekst

    itd.

    Pobrane liczby umieść w **vector**'ze.

    Pobrane napisy umieść w innym **vector**'ze.

    (Plik może mieć dowolną zawartość zgodną z zasadami pliku, po prostu utwórz go samodzielnie) [3]

38. Utwórz własną dowolną strukturę złożoną z kilku typów, np.: **int**, **string**, **char**, **float** itp. Następnie zaprojektuj mechanizm, który:

    - zapisze w pliku zmienną utworzonego typu strukturalnego (jedną zmienną)

    - odczytaj z pliku dane, na podstawie których utworzysz jedną zmienną utworzonego typu strukturalnego

    - spróbuj zapisać dane tak, by można było zapisać je w pliku dla dowolnej ilości zmiennych utworzonego typu strukturalnego, a następnie je odczytywać

    do tablicy/vector'a zawierającego wartości naszego utworzonego typu strukturalnego.

    np.:

```
struct A {
        int a;
        float b;
        string c;
        char d;
        bool e;
}
A zmienna1; // zmienna1 powinna dać się zapisać/odczytać do/z pliku.
vector<A> va;
```

    A do w/w vector’a trzeba wprowadzić wartości pobrane z pliku pełnego informacji o obiektach typu strukturalnego A. [3]

39. Stwórz funkcję, która zwraca **true**, gdy otrzymany w argumencie napis (**string**) jest palindromem, i **false** w przeciwnym wypadku. (Palindrom to napis, który zapisany od tyłu i od przodu jest taki sam) [2]

40. Stwórz funkcję, która jako argument otrzymuje napis A i B. Zwraca ilość wystąpienia napisu A w napisie B. [2]

41. Napisz własną funkcję podobną do funkcji **stoi**(), która zamienia string na liczbę typu **long long int** i którą zwraca. [2]

42. Napisz własną funkcję, która liczbę całkowitą zamienia na napis i zwraca ten napis. [2]

43. Napisz własną funkcję konwertującą, która pobiera napis reprezentujący liczbę binarną, np. "**100101**" a następnie zwraca liczbę całkowitą, która tę liczbę reprezentuje. Stwórz funkcję, która wykonuje operację odwrotną:

    jako argument otrzymuje liczbę całkowitą a następnie zamienia ją na napis zawierający reprezentację w postaci liczby binarnej. [2]
    Pomyśl nad stworzeniem podobnej konwersji, ale dla liczb systemu ósemkowego i szesnastkowego. [3]

44. Ciąg tekstowy T o długości N składa się z losowych cyfr i liter angielskich. Znajdź przynajmniej jeden najdłuższy podciąg ciągu T, który składa się z samych samogłosek. Znajdź najdłuższy podciąg ciągu T, który

składa się z samych cyfr, ale te cyfry tworzą podciąg niemalejący, np. w ciągu T = "dfgnqeiut98tnav0w3r123334asdsh" istnieje najdłuższy podciąg niemalejący z samych cyfr "123334". [3]

45. Do tablicy dwuwymiarowej 100 na 100 wstaw losowe ciągi tekstowe dokładnie o długości 3 znaków każdy. Te 3 literowe ciągi powinny składać się z losowych znaków ze zbioru "**abcdef**".

    Znajdź najdłuższy podciąg kolejnych napisów w pionie lub poziomie, który składa się z dowolnego spośród napisów "**abc**" lub "**bcd**" lub "**cde**" lub "**def**". [3]

46. W pliku mapa.txt losowo umieść 20 wierszy tekstu, gdzie w wierszu możesz wstawić kropkę '.' lub kreskę (minus) '-'. Każdy wiersz powinien mieć 20 losowych znaków '.' lub '-', z tym, że każdy wiersz i kolumna musi zaczynać się i kończyć kropką, jednak w wierszu nie może być więcej jak 10 kropek.

    Stwórz funkcję, która czyta tak wygenerowany losowo plik i sprawdza, ile wierszy lub kolumn w pliku jest symetryczna. [3]
    *Przykład symetrycznego wiersza (posiadającego oś symetrii):*

    .-.-.----------.-.-.

47. Pobierz ciąg tekstowy z klawiatury. Następnie używając tylko liczb dziesiętnych, spacji oraz średnika wymyśl sposób zapisu tego napisu w pliku kod.txt. Stwórz mechanizm odczytu treści pliku kod.txt i odwracając swoją własną metodę zamień treść z pliku na tekst, który na początku pobrałeś z klawiatury. [3]

    *Np. pobierasz słowo "rower". Następnie zapisujesz go w jakiś sposób w pliku kod.txt, korzystając tylko z liczb dziesiętnych, spacji i średnika. Następnie odwróć ten proces: przeczytaj ten kod.txt i odwracając proces zamień go na słowo "rower".*

48. Stwórz funkcję, która dla podanego jako argument wektora V z liczbami całkowitymi oraz pewnej liczby X, szuka tej liczby X w wektorze V i zwraca ilość jej wystąpień. [1]

49. Stwórz funkcję, która w wektorze V przekazanym jako argument szuka wszystkich podciągów przynajmniej dwuelementowych złożonych z takich samych liczb. Funkcja powinna wyświetlać wszystkie takie podciągi, od najdłuższych po najkrótsze, podając indeks pierwszego i ostatniego elementu dla każdego podciągu. [2]

    *Np. {3,2,1,1,4,2,4,4,4} zawiera 4 podciągi przynajmniej dwuelementowe złożone z takich samych liczb:*

    *1,1 od pozycji 2 do 3*

    *4,4 od pozycji 6 do 7*

    *4,4 od pozycji 7 do 8*

    *4,4,4 od pozycji 6 do 8*

50. Stwórz funkcję, która otrzymuje dwa wektory z liczbami a zwraca wektor zawierający tylko te liczby, które występują w obu wektorach (część wspólna). Liczby występujące w wektorze zwracanym nie powinny się powtarzać. [2]

51. Wygeneruj losowy napis z dowolnych liter i cyfr jak również spacji. Napis nie powinien być dłuższy niż 1000 znaków, ale nie krótszy niż 50. Przetwórz napis tak, aby zniknęły z niego wszelkie podciągi znaków powtarzających się. Np.

    z fragmentu "**skfuu12m2111flx**" powinno zniknąć "**uu**" oraz "**111**". [3]

52. Stwórz funkcję, która dla podanej liczby całkowitej zwraca **true**, gdy liczba jest pierwsza i **false** w przeciwnym wypadku. [2]

53. Stwórz funkcję, która przekazane dwa wektory z liczbami łączy w jeden wektor i zwraca go. Liczby w wektorze wynikowym nie mogą się powtarzać i powinny być posortowane w następujący sposób:
    - najpierw powinny wystąpić liczby dodatnie z zerem posortowane malejąco, a potem powinny wystąpić liczby ujemne posortowane rosnąco:
    8,6,4,2,1,0,-20,-12,-8,-1. [3]

54. Sprawdź poprawność wyrażenia algebraicznego podanego jako tekst. Nie chodzi o jego obliczenie, ale poprawność. Wyrażenie algebraiczne może składać się z: liczb, liter angielskich małych, działań +,-,*,/,^,%, porównania = oraz z nawiasów (). (uznajmy, że w wyrażeniu może być wiele razy porównanie np. a=b=c)
    Przykładowe poprawne wyrażenia:
    **(12*x+3*(3*d-a)^3)/11=z**
    albo
    **-(-12^a*b*c/12-(-5-a*b))**
    Takie wyrażenie jest poprawnie zapisane i to właśnie masz zweryfikować. Przykład niepoprawnego:
    **---a**
    **a+b=c=d=** [5]
55. Korzystając ze zbioru imion i nazwisk (plik imiona.txt, oraz nazwiska.txt – utwórz je sobie sam) napisz generator w postaci funkcji, która zwraca losowe imię i nazwisko. [1]
56. W pewnym kraju wszystkie imiona są zlepkiem sylab, w których każda sylaba w środku ma ‘e’ lub ‘o’, kończy się spółgłoską a zaczyna się jedną z liter ‘s’,’f’,’r’ lub ‘o’. Jeżeli imię składa się z parzystej ilości sylab, wszystkie sylaby mają w środku tę samą samogłoskę (‘e’ lub ‘o’) a jeżeli imię składa się z nieparzystej ilości sylab, to każda następna sylaba musi mieć inną samogłoskę w środku od poprzedniej sylaby. Na podstawie informacji stwórz generator imion w postaci funkcji, która imię zwraca. Wygeneruj 15 imion i przeczytaj je na głos pierwszej osobie, jaką zobaczysz. [2]
57. Wygeneruj losową liczbę składającą się z 10 cyfr. Liczba ta nie może się zaczynać od 0, suma jej cyfr nie może być mniejsza niż 30, a żadne bezpośrednio sąsiadujące cyfry nie mogą być identyczne, chyba, że są to dwie ostatnie cyfry – te mogą być identyczne. Trzecia cyfra licząc od lewej nie może być liczbą parzystą. [2]
58. Stwórz losowy symulator/generator pomiaru wartości temperatury, zakładając:
    * najmniejsza mierzalna temperatura to 0.1 stopnia C.
    * temperatura jest mierzona co godzinę od 01.01.2020 od godziny 6:00 rano włącznie. Pierwszy pomiar nie jest losowy i wynosi -5 stopni C.
    * temperatura nie może przez godzinę zmienić się o więcej niż 1.5 stopnia C.
    * temperatura w trakcie doby nie może zmienić się o więcej niż 12 stopni C. (zwróć uwagę, że doba to 24 godziny wstecz od obecnego pomiaru, każdy pomiar uwzględniać ma sobie właściwą dobę a nie dobę sztywną od 00:00 do 23:59)
    * przy założeniu, że pierwsze 3 miesiące to zima, drugie trzy to wiosna, następnie 3 kolejne to lato a ostatnie 3 to jesień, średnia wygenerowana temperatura w tych okresach musi spełniać warunek, iż zima < wiosna < jesień < lato (zima najchłodniejsza, lato najcieplejsze).
    Gdy skończysz generator, wylosuj pomiar temperatur co godzinę przez okres jednego roku. Dla tak powstałych danych podaj:
    - najwyższą i najniższą temperaturę w roku
    - najchłodniejszy i najcieplejszy miesiąc w roku
    - tydzień w roku, w którym różnica najwyższej i najniższej temperatury jest najmniejsza. [4]
59. Struktura Punkt zawiera wartość x i y typu **int** (nie **double**!). Stwórz funkcję, która otrzymuje jako argumenty 3 zmienne typu Punkt, i sprawdza, czy mogą one leżeć na jednej linii prostej? [1]
60. W wektorze znajdują się liczby całkowite. Sprawdź, czy tworzą one ciąg: rosnący, malejący, nierosnący, niemalejący, stały lub nie tworzą żadnego monotonicznego ciągu. [1]
61. Zaprojektuj mechanizm, który przechowuje liczby całkowite w „prostokącie” (macierzy) o N wierszach i M kolumnach (N i M >=10 i <=50). Liczby powinny być losowe, z przedziału <0;100>. Stwórz funkcję, która sprawdza, czy istnieje taki wiersz, że istnieje taka kolumna o identycznych wartościach (kolejność wartości nie i ich ilość musi być zachowana). Ile jest takich wierszy?
    *Dla N=2, M=3 jeżeli w wierszu jest {1,5} a w kolumnie {5,1,1} to wiersz spełnia*

*warunki. Ale wiersz {1,5} dla kolumny {1,5,2} nie spełnia, gdyż kolumna ma więcej wartości.* [2]

62. Zaprojektuj funkcję, która dla podanej liczby naturalnej N reprezentującej ilość dni, obliczy datę, która nastąpi od dnia 2020-01-01 po upływie N dni. *Np.: dla N=3 obliczona data to 2020-01-04.* Podaj również, jaki to dzień tygodnia. *(Nie zapomnij o latach przestępnych!)* [2]
63. Stwórz program, który dla napisu (**string**) reprezentującego datę i czas, zachowującego następujący format: „RRRR-MM-DD mm:gg:ss”, np.: „2020-05-02 11:05:08”, obliczy ilość minut lub godzin od daty i godziny „2020-01-01 12:00:00”. [3]
64. Stwórz dowolny program, który z napisu (string) reprezentującego liczbę szesnastkową, np.: „FFC8”, dokona konwersji na napisy (również string) reprezentujące liczbę w systemie binarnym oraz dziesiętnym. Możesz wymyślić własny sposób lub skorzystać z istniejących algorytmów. [4]
65. Stwórz funkcję, która porównuje dwa napisy, zbudowane z liter polskiego alfabetu, a następnie zwraca je w kolejności alfabetycznej. [2]
66. Stwórz funkcję, która sprawdza, czy z podanego słowa (napisu) zawierającego małe litery alfabetu angielskiego można ułożyć palindrom? *Palindrom ma zawierać tyle samo znaków co oryginalne słowo, ale można dowolnie przestawiać znaki w słowie.* [4]
67. A
68. A
69. A
70. A
71. A
72. A

## *UMIEJĘTNOŚCI PODSTAWOWE*

Umiejętności podstawowe zawierają umiejętności fundamentalne. Oto przybliżona lista umiejętności podstawowych:

- utworzenie prostej klasy z konstruktorem, destruktorem, przykładową metodą
- wskaźniki, w tym wskaźniki do struktury, wskaźniki do klasy, przekazywanie wskaźnika do funkcji, zwracanie wskaźnika itp.
- teoria algorytmów (znajomość teoretyczna wszystkich algorytmów jakie podajemy w szkole, teoria złożoność algorytmów)
- **implementacja prostych algorytmów** (proste sortowania, wyszukiwania, proste algorytmy zachłanne itp.)
- umiejętność korzystania **z podstawowych struktur danych** dostarczonych przez STL (**`stack`, `forward_list`, `list`, `queue`, `dequeue`** itp.)
- umiejętność korzystania z **kontenerów** dostarczonych przez STL (**`pair`, `map`, `set`, `unordered_map`, `unordered_set`, `tuple`, `multimap`, `multiset`, `unordered_multimap`, `unordered_multiset`** itp.)
- sprawne posługiwanie się typem **`auto`**

***UWAGA! Jeżeli zadanie czegoś nie określa jasno, to znaczy, że masz pełną dowolność interpretacji zgodnie z logiką i nie wbrew treści zadania. Jeżeli masz kilka pomysłów na zadanie - zrób je na wiele sposobów. Jeżeli zadanie zrodziło pomysł na jakiś wspaniały program - napisz go!***
***Liczba w nawiasach na koniec zadania do liczba punktów za zadanie a równocześnie przybliżona informacja o trudności zadania w danej grupie umiejętności.***

1. Stwórz klasę reprezentującą punkt (**`Point`**) na płaszczyźnie. Wygeneruj losowo 100 punktów i umieść je w dowolnym kontenerze. Współrzędne punktów powinny być w zakresie <-100;100> i mieć dokładność do 2 miejsc po przecinku.
    a. Stwórz metodą klasy **`Point`**, która wyświetla, do jakiej ćwiartki należy punkt, lub na jakiej osi leży. [1]
    b. Stwórz metodę klasy **`Point`**, która jako argument otrzymuje wskaźnik na inny obiekt Point i sprawdza, czy te punkty tworzą odcinek prostopadły do osi X (metoda zwraca **`true`**/**`false`**). [1]
    c. Stwórz metodę klasy **`Point`**, która jako argument otrzymuje liczbę typu **`double`**. Przyjmij tę liczbę jako promień koła o środku w punkcie. Zwróć **`true`**, gdy to koło nie ma części wspólnej z żadną osią współrzędnych i **`false`** w przeciwnym razie. [1]
    d. Stwórz metodę klasy **`Point`**, która jako argument otrzymuje wskaźnik na inny obiekt **`Point`** i sprawdza, czy odcinek, który tworzy punkt (**`this`**) i punkt przekazany przecina oś X? (metoda zwraca **`true`**/**`fale`**) [1]
    e. Stwórz funkcję globalną, która jako argumenty otrzymuje 3 punkty (**`Point`**). Funkcja ma zwracać **`true`**, jeżeli z punktów da się utworzyć trójkąt prostokątny i **`false`** w przeciwnym wypadku. [2]
2. Stwórz klasę reprezentującą Trójkąt, składający się z 3 punktów. Stwórz również klasę reprezentującą Koło. Dla klasy Trójkąt utwórz metodę sprawdzającą, czy obiekt Koło może zmieścić się wewnątrz trójkąta. Dla klasy Koło utwórz metodę sprawdzającą, czy obiekt Trójkąt może zmieścić się wewnątrz koła. [2]
3. Umieść w wektorze **`vector<int*>`** wskaźniki, które wskazują na losowe liczby z zakresu <10;10>. Zaprojektuj funkcję, która usuwa z przekazanego wektora element na wskazanej pozycji i dba o właściwe zwalnianie pamięci! [1]
4. Stwórz klasę reprezentującą punkt na płaszczyźnie. Przyjmij pewną zasadę, iż punkt A jest większy od punktu B, wtedy, kiedy suma wartości bezwzględnej współczynników A jest większa od takiej samej sumy współczynników punktu B. Następnie utwórz dowolny kontener, w którym umieść 100 losowych punktów utworzonej klasy. Zaprojektuj funkcję, która metodą bąbelkową posortuje ten kontener. [3]

5. Stwórz klasę reprezentującą Ucznia. Powinna zawierać pola: imię, nazwisko, rok urodzenia, płeć, obecną klasę i numer w dzienniku. Utwórz następnie klasę MojaKlasa, która będzie w dowolnym kontenerze przechowywać uczniów. Zaprojektuj mechanizm ładowania całej twojej klasy z pliku klasa.txt w taki sposób, aby każdy uczeń z Twojej klasy był obiektem Uczeń, a wszyscy uczniowie byli przechowywaniu w klasie MojaKlasa. [4]
   Następnie:
   * Stwórz metodę w MojaKlasa, która sortuje uczniów wg ich numeru z dziennika. Nie pozwól, aby w klasie numer z dziennika się powtarzał! [3]
   * Stwórz metodę w MojaKlasa, która zwraca Ucznia na podstawie podanego numeru z dziennika. [1]
   * Stwórz metodę w MojaKlasa, która wyświetla ładnie listę uczniów. Powinna być dodatkowa możliwość wyświetlenia tylko chłopców lub tylko dziewczynek. [1]
   * *PROPOZYCJA! Pomyśl nad szerszym programem, który dla każdego ucznia w klasie zapamiętuje jego oceny z dowolnego przedmiotu. Przemyśl, jak mogłaby wyglądać taka aplikacja? A gdyby powiększyć ją o informację, jak nazywa się nauczyciel danego przedmiotu? A może masz jeszcze jakiś ciekawy pomysł? Librus 2.0?*
6. Zaprojektuj klasę Ork. Powinna zawierać przynajmniej imię orka, jego poziom zdrowia i siłę. Możesz jednak dodać inne atrybuty, które jednak potem postaraj się wykorzystać. Następnie stwórz kontener z 10-ma orkami, których statystyki i imiona są losowe. Wymyśl metodę dla klasy Ork, która jako argument przyjmuje drugiego Orka a następnie doprowadza do „bitwy” dwóch orków, z której jeden tylko może przeżyć. Następnie zaprojektuj dla wcześniej utworzonego kontenera mechanizm walki orków tak, by przeżył tylko jeden ork. Po każdej bitwie orka z orkiem osobnik, który przeżył ma mieć odnowione statystyki. Może też zyskiwać jakieś atrybuty za zwycięstwo. Zaprojektuj system komunikatów o tym, jakie orki właśnie się biją. Przeprowadź bitwę do końca i podaj imię zwycięzcy. [5]
7. Zaprojektuj klasę SZACHOWNICA oraz odpowiednie struktury dla każdej figurki. Wykorzystuj wskaźniki. Klasa ma być tak zaprojektowana, by pamiętać obecny układ pionków/figur na szachownicy. Zaprojektuj metody, które ruszają daną figurę/pionek po szachownicy z miejsca na miejsce. Klasa powinna „zapamiętać” całą rozgrywkę dwóch graczy – od pierwszego do ostatniego ruchu. Przemyśl dobrze, jak to zrobić. Zadanie wymaga dobrego planowania. Zły pomysł utrudni pisanie kodu, niepotrzebnie go skomplikuje. [7] Stwórz również metodę, która obecną sytuację na szachownicy może zapisać do pliku, lub ją z pliku pobrać. [+5]
8. Zaprojektuj klasę reprezentującą kartę. (papierowa karta. Np. walet pik itd.) Następnie zaprojektuj klasę reprezentującą całą talię 52 kart. Stwórz mechanizm losowania 5 kart z talii i ich wyświetlania. Sprawdź, jakie karty w grze POKER mają znaczenie. Jeżeli wśród 5 kart, które wylosowałeś, znajdują się jakieś istotne i punktowane kombinacje: np. para, ful, czwórka itp. poinformuj o tym. [4]
9. Dodawaj do listy losowe liczby całkowite z zakresu <1;6> na losowej pozycji. Informuj o tym, gdzie dodano liczbę i jaka to liczba. Gdy wylosujesz 3 razy 1-kę przerwij losowanie. Usuń z listy wszystkie 6-ki. [1]
10. Stwórz prostą klasę reprezentującą Osobę (wystarczy, że będzie zawierać tylko imię). Następnie umieszczaj w kolejce o dwóch końcach **deque<Osoba*>** osoby, symulując mechanizm „ktoś do kolejki doszedł, ktoś ją opuścił”. Zrealizuj następujący algorytm:
    * wciskam **Enter** (sam **Enter**, lub po jakiejś sekwencji znaków, poleceń) i losuję, czy ktoś ma się dodać do kolejki, czy z niej wyjść
    * jeżeli ma się dodać, tworzy się wygenerowana osoba o losowym imieniu i dodaje do kolejki na końcu
    * jeżeli ma nastąpić odejście z kolejki, pierwszą osobę z listy usuwam (nie zapomnij zwalniać pamięci)
    * Po każdej operacji dopisania/wyjścia powinna być informacja co się stało.
    * wpisanie **‘q’** i **Enter** kończy algorytm. [3]

11. Utwórz mapę, w której znajduje się para **<int,string>** i umieść w niej imiona osób z Twojej grupy na informatyce. Stwórz funkcję, która losuje jeden spośród kluczy i zwraca imię. [1]
12. Wykorzystaj **<stack>** i zaimplementuj operację obliczenia działań ONP (Odwrotnej Notacji Polskiej). [4]
13. Stwórz funkcję, która przyjmuje **vector<int>** i zwraca **set<int>** utworzony na podstawie przekazanego vectora. [1]
14. Przyjmij, że na płaszczyźnie narysowany jest kwadrat, o środku w punkcie (0,0) i boku 20. W ten kwadrat wpisane jest koło. Stosując dokładność 4 miejsc po przecinku „strzelaj losowo" w punkty wewnątrz kwadratu. Strzel 25 tysięcy razy. Policz, ile punktów strzeliło do wnętrza koła (krawędź koła uznajemy za wnętrze koła), a ile jest poza kołem (w narożnikach kwadratu, ale poza kołem). Sprawdź stosunek pola kwadratu do pola koła. Sprawdź stosunek wszystkich punktów w kwadracie do punktów w kole. Wnioski?
    Uwaga! Korzystaj z losowania gwarantującego równomierny rozkład. Unikaj funkcji **rand()**. [2]
15. Wykorzystaj poznany na lekcji/kursie algorytm `Wieży Hanoi` i jego prostą implementację (z wykorzystaniem **string**) i przerób implementację w taki sposób, aby odpowiednie paliki były stosem **<stack>**, a elementy przerzucane z palika na palik były obiektem własnej klasy Roller (krążek).
    *(użyj* **stack<Roller>** *aby realizować ruch krążków pomiędzy palikami (stosami))* [2]
16. Stwórz mechanizm symulacji pracy kolejek w sklepie. Proces, który symulujesz musi być czytelnie opisany. Skup się. Wyjaśniam o co chodzi, a Ty to zaimplementujesz wykorzystując **dequeue**:
    - W sklepie znajdują się 3 kasy (3 kolejki);
    - Algorytm realizowany jest w pętli (dokładnie 200 kroków pętli);
    - W każdym kroku mogą zajść akcje:
    a) 55% szansy, że ktoś podejdzie do kasy (doda się do kolejki przy kasie)
    b) 50% szansy, że kasjer obsłuży klienta. Jeżeli obsłuży, klient w tym kroku pętli wychodzi z kolejki, a jak nie, to zostaje do następnego kroku pętli. [3]
    Po realizacji w/w symulacji uwzględnij następujące modyfikacje:
    - Kasy, oznaczmy je kasą A, B i C, pracują w innym tempie, ponieważ są tam różni kasjerzy, dlatego szansa obsługi przez kasjera klienta jest inna. A – 65%, B - 40%, C – 50%.
    - Jeżeli na koniec kroku w pętli istnieje sytuacja, że jakaś kasa nie ma klientów, a w którejś z pozostałych jest co najmniej 3 klientów, niech ostatni klient z tej zapełnionej kolejki do kasy przejdzie do pustej kasy;
    *Prawdopodobnie powinieneś zaobserwować proces, w którym osoby z kolejki do kasy B/C chociaż raz przejdą do kolejki do kasy A. Postaraj się ładnie opisywać kolejne kroki pętli, co się dzieje, kto wchodzi kto wychodzi.*
    - Każdy klient w kolejce płaci losową kwotę od 5zł do 1500zł (z dwoma miejscami po przecinku). Oblicz w procesie symulacji, ile pieniędzy wpłynęło do kas.
    - Każdy klient płaci w/w losową kwotę za pomocą tylko banknotów o nominale 100zł (czyli jak płaci 123zł to daje do kasy 200zł). Kasa musi więc wydawać resztę. Wydawaj resztę korzystając z algorytmu zachłannego wydawania reszty przy założeniu, że kasa ma nieskończoną ilość monet i banknotów. Każdą operację wydawania reszty zapisz do pliku: „historia.txt", w taki sposób, aby było widać która kasa wydała resztę, z jakiej kwoty, i co wydała. [4]
17. Zaprojektuj program, który dla podanego pliku z tekstem źródłowym oblicza ilość wystąpienia każdego ze znaków w tym pliku. Podaj stosunek znaków z alfabetu do znaków pozostałych. Podaj stosunek samogłosek do spółgłosek. Uporządkuj znaki wg ilości wystąpień i podaj tak uporządkowaną listę znaków. [2]
18. 
19. Kolejne zadanie, oczekuj.
20. Kolejne zadanie, oczekuj.
21. Kolejne zadanie, oczekuj.

22. Kolejne zadanie, oczekuj.