Zbiór zadań C++

ZADANIA C++: podstawy

Zad. 1:
Zadeklaruj kilka zmiennych: `float`, `double`, `int`. Na stałe w kodzie programu wpisz ich wartości. Wypisz je na ekran (polecenie `cout`).

Zad. 2:
Jaka jest największa liczba całkowita którą można wpisać do programu?

Zad. 3:
Zadeklaruj stałą i spróbuj ją zmienić. Co się stanie?

Zad. 4:
Napisać program wyznaczający dziedziny typów całkowitoliczbowych: `char`, `short`, `int`, `long` oraz `long long`, zarówno `signed` i `unsigned`.

Zad. 5:
Napisać funkcję o deklaracji `unsigned doPot2 (unsigned x);` zwracającą najmniejszą potęgę dwójki większą lub równą `x`.

Zad. 6:
Mamy daną tablicę:

```
#define ROZMIAR 1000

struct ksiazka {
     char      autor[32], tytul[64];
     unsigned  ilosc;
     float     cena;
};

struct ksiazka magazyn [ROZMIAR];
```

Napisać funkcję obliczającą należny VAT 7% za książki podanego autora.

Zad. 7:
Napisać program, który porównuje dwa pliki tekstowe w poszukiwaniu pierwszej pary identycznych wierszy.

Zad. 8:
Stworzyć typ danych reprezentujący wielomian (dowolnego stopnia). Napisać funkcje obliczające: pochodną wielomianu oraz sumę i iloczyn dwóch wielomianów.

Zad. 9:
Napisać funkcję znajdującą medianę tablicy:

```
#define ROZMIAR 500

typedef struct probka_   {
     char           nazwa[32];
     unsigned long  id;
     double         wartosc;
} probka;

probka tablica [ROZMIAR];
```

Zad. 10:
Załóżmy, że w programie reprezentujemy wielokąty za pomocą typu `wielokat` określonego następująco:

```
typedef struct {
     double x, y;
} punkt;

typedef struct {
     // liczba wierzchołków wielokąta
     int       ileW;
     // tablica kolejnych wierzchołków
     punkt     *tabW;
} wielokat;
```

Napisać funkcje obliczające: obwód i pole powierzchni dowolnego wielokąta. *Wskazówka*: wykorzystać własności iloczynu wektorowego.

Zad. 11:
W opowiadaniu Artura Clarke'a *„Dziewięć miliardów imion Boga”* z roku 1955 specjalnie przystosowany komputer miał za zadanie wygenerować wszystkie słowa złożone z nie więcej niż dziewięciu liter z wyłączeniem słów zawierających trzy (lub więcej) kolejne takie same litery. Słowa te miały jakoby zawierać wszystkie prawdziwe imiona Boga. W opowiadaniu zatrudniony do tego celu komputer wykonał pracę w ciągu trzech miesięcy. Zaimplementować program realizujący to samo zadanie, zmierzyć jaki czas jest na to potrzebny współczesnym komputerom. Sprawdzić czy słów takich jest istotnie dziewięć miliardów.
(W opowiadaniu wypisanie wszystkich imion Boga okazało się być celem istnienia ludzkości. Zakończenie działania programu spowodowało koniec świata. Zadanie to zatem czytelnik wykonuje na własną odpowiedzialność.)

Zad. 12:
Napisać program, który wykonuje zadaną przez użytkownika operację arytmetyczną w postaci „$x \square y$”, gdzie x, y są liczbami, zaś $\square$ jest jednym ze znaków +, −, *, /.

Zad. 13:
Napisać program wyświetlający na ekranie pierwsze szesnaście potęg dwójki.

Zad. 14:
Napisać program wyświetlający na ekranie tabelkę funkcji: $x \rightarrow 3x^7 + (x-4)^2 - x$ dla $x \in \{0, \ldots, 10\}$.
Wskazówka: wykorzystać schemat Hornera.

Zad. 15:
Napisać program, który czyta z wejścia liczby całkowite aż do napotkania liczby ujemnej, a następnie zwraca największy/najmniejszy element z wczytanych liczb (z pominięciem ostatniej, ujemnej liczby).

Zad. 16:
Napisać program obliczający procentową częstotliwość występowania zadanego przez użytkownika znaku w linii tekstu.

Zad. 17:
Napisać program, który odgaduje liczbę całkowitą z zakresu 1, …, 100 pomyślaną przez użytkownika. Użytkownik odpowiada jedynie, czy zgadywana liczba jest za mała / za duża.

Zad. 18:
Napisz program drukujący na ekranie liczby. Wysokość wczytujemy z klawiatury. Oto wydruk dla wysokości `h`=5:
```
1
2 4
3 6 9
4 8 12 14
5 10 15 20 25
```
Wynik zapisz również do pliku.

Zad. 19:
Napisz program drukujący na ekranie prostokąt z literek `X`. Wysokość i szerokość prostokąta wczytujemy z klawiatury:
pusty
```
XXXXXXXXXX
X        X
X        X
XXXXXXXXXX
```
pełny
```
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```
Wynik zapisz również do pliku.

Zad. 20:
Program rysujący na ekranie poziomą kreskę (ze znaków minus) o długości zadawanej z klawiatury. Wynik zapisz również do pliku.

Zad. 21:
Napisz program drukujący na ekranie tabliczkę mnożenia do `n`, gdzie `n` jest podane przez użytkownika i jest z przedziału 1-10. Przykład wyjścia dla `n`=5:

```
   | 1  2  3  4  5
======================
1  | 1  2  3  4  5
2  | 2  4  6  8  10
3  | 3  6  9  12 15
4  | 4  8  12 14 20
5  | 5  10 15 20 25
```

Wynik zapisz również do pliku.

Zad. 22:
Napisz program wczytujący z klawiatury 10 liczb całkowitych. Wczytane liczby należy wydrukować na ekranie w odwrotnej kolejności. Użyj do tego tablicy.
Wynik zapisz również do pliku.

Zad. 23:
Program wyświetlający na ekranie kolejne liczby całkowite typu `int` (do 100), które są podzielne bez reszty przez `n` (gdzie `n` jest zadawane z klawiatury).
Wynik zapisz również do pliku.

Zad. 24:
Program wyświetlający na ekranie kolejne 100 liczb parzystych od zadanej liczby.
Wynik zapisz również do pliku.

Zad. 25:
Napisz program, który z gwiazdek `*` narysuje na ekranie choinkę o zadanej wysokości, z przedziału 4-14. Np. dla wysokości 6 („gałęzie” 5 + podstawa 1):

```
     *
    ***
   *****
  *******
*********
    ***
```

Wynik zapisz również do pliku.

Zad. 26:
Program wczytujący kolejne liczby z klawiatury i kończący się gdy:
- suma tych liczb przekroczy 100,
- łącznie podanych liczb ujemnych będzie więcej niż 10,
- dwie kolejne podane liczby będą miały identyczną wartość.

Zapisz również do pliku:
- ostatnią podaną liczbę,
- ile podano liczb w sumie oraz ile liczb ujemnych, dodatnich, parzystych, nieparzystych.

Zad. 27:
Program wczytujący znaki z klawiatury aż do momentu naciśnięcia klawisza `'y'`. W trakcie wczytywania należy zliczać liczbę podanych znaków. Na zakończenie należy wyświetlić informacje:
- ile było podanych znaków (przed naciśnięciem `'y'`)
- ile z tych znaków było dużymi literami.

Wynik zapisz również do pliku.

Zad. 28:
Program wczytujący liczby z klawiatury, z jednoczesnym zadawaniem pytania *„Czy koniec wprowadzania T/N?”*. Na zakończenie program powinien wyświetlić wartości: średnią, maksymalną, minimalną z podanych liczb.
Wynik zapisz również do pliku.

Zad. 29:
Napisz program, który po wpisaniu przez użytkownika liczby całkowitej wyświetli odpowiedni napis: *„parzysta”* lub *„nieparzysta”* oraz *„ujemna”* lub *„dodatnia”*.
Wynik zapisz również do pliku.

Zad. 30:
Zaimplementuj proste szyfry harcerskie: GA-DE-RY-PO-LUKI, PO-LI-TY-KA-RE-NU, KA-CE-MI-NU-TO-WY. Wyświetla się menu, gdzie użytkownik wybiera szyfr. Następnie komunikat *„Podaj ciąg wejściowy”*, który jest odczytywany przez program a następnie zmieniane litery zgodnie ze wzorcem szyfru.

Zad. 31:
Przeczytaj od użytkownika ciąg tekstu zakończony znakiem kropki (.) Następnie podziel tenże tekst na słowa i wszystkie dłuższe od 4 znaków zapisz do oddzielnej tablicy stringów. Na końcu wypisz na ekran napis złożony z 2 i 3 znaku każdego elementu w tej nowej tablicy.

Zad. 32:
Napisz funkcję sprawdzającą, czy wyraz jest palindromem. Tu konkurs na najkrótszy program :-)

Zad. 33:
Polecenie: napisz funkcję, która generuje informacje o porach, w których powinien rozlegać się dźwięk dzwonka.
Po pierwsze, godzina rozpoczęcia pierwszej lekcji może być zmienna i niekoniecznie musi być nią 8:00. Wynika z tego, że informację tę program musi przyjąć w postaci pierwszej danej wejściowej. Również długości przerw są zmienne, co akurat jest zupełnie zrozumiałe. Jedno tylko pozostaje niezmienne – długość lekcji, która zawsze trwa równo 45 minut.
Dane wejściowe: `string` określający godzinę rozpoczęcia zajęć, tablica `int` zawierająca przerwy, wyrażonych w minutach (całkowite `t`: `t` >0 i `t`<=10080).
Dane wyjściowe: jeden wiersz tekstu, a w nim lista rozdzielonych przecinkami pór uruchomienia dzwonka w formacie HH:MM
Przykład:
Wejście: `Tablica przerwy[] = {15, 15, 15}`
`cout << dzwonki(„08:00”, przerwy);`
Wyjście: `08:00,08:45,09:00,09:45,10:00,10:45,11:00,11:45`

Zad. 34:
Poczytaj o instrukcja preprocesora (`#define`, `#include`, `#error`, makra). Za pomocą tej wiedzy napisz kilka szybkich makr automatyzujących liczenie np. sinusa lub cosinusa.

Zad. 35:
Zaimplementuj Szyfr Cezara: kodowanie i dekodowanie (http://pl.wikipedia.org/wiki/Szyfr_Cezara) wraz z obsługą plików (szyfrogram, deszyfrogram).

Zad. 36:
Odczytaj tekst wpisywany przez użytkownika, a następnie wypisz statystykę analizy częstości występowania poszczególnych liter (od ‘a’ do ‘z’) zamieniając wielkie litery na małe. Zapisz statystykę do pliku w czytelnym formacie.

Zad. 37:
Wypełnij tablicę 10 x 10 tabliczką mnożenia… ale w kodzie trójkowym.

Zad. 38:
Program „zgadnij moją liczbę”. Program losuje liczbę z zakresu 1…100, a naszym zadaniem jest zgadnąć tę liczbę na podstawie „za dużo”, „za mało”. Po zgadnięciu program wyświetla liczbę prób.

Zad. 39:
Napisz program odwrotny do „zgadnij moją liczbę”: człowiek wymyśla liczbę, program ma ją znaleźć. Ale jest haczyk – program ma maksymalnie 13 prób odgadnięcia.

Zad. 40:
Zaprojektuj funkcję, która odczytuje liczby od użytkownika aż do podania 0. Funkcja ma zwrócić 0 lub 1 – w zależności od tego czy tych liczb była parzysta czy nieparzysta liczba. Co więcej – funkcja ma jednocześnie zwrócić sumę oraz iloczyn wszystkich podanych liczb.

Zad. 41:
Zadeklaruj tablicę o rozmiarze 100. Wypełnij tablicę zgodnie z regułami Ciągu Fibbonacciego (pierwszy i drugi element = 1, każdy następny to suma dwóch poprzednich: 1,1,2,3,5,8, itd.).

Zad. 42:
Napisz program, który zamienia liczbę dziesiętną podaną przez użytkownika na liczbę binarną i szesnastkową.

Zad. 43:
Wypisz całą tablicę ASCII na ekran (każdy znak w nowej linijce opatrzony „numerkiem”) i zapisz ją do tablicy o nazwie `tab_ASCII` w programie.

Zad. 44:
Zadeklaruj napis c-string o długości 10 i przypisz mu 11 znaków (najpierw w deklaracji, a potem za pomocą konkatenacji).

Zad. 45:
Przeczytaj z klawiatury kilka napisów i dodawaj je do jednej zmiennej c-string. Jeśli będzie za dużo znaków, poinformuj o tym użytkownika.

Zad. 46:
Napisz funkcję `HTML2RGB` zwracającą tablicę liczb całkowitych, a przyjmującą jeden napis. Ów napis to sześcioznakowa liczba zapisana w kodzie szesnastkowym (np. `0A45F3`). Twoim zadaniem jest zwrócenie tablicy trzech liczb całkowitych w systemie dziesiętnym odpowiadających kolejno składowej czerwonej (pierwsze dwie cyfry napisu), zielonej (kolejne dwie) i niebieskiej (ostatnie dwie).
Przykładowo:

```
System.out.println(HTML2RGB(„FFFFF0”)[0])
```

powinno wyświetlić na ekranie wartość 255, 255, 240.

Zad. 47:
Polecenie: napisz program, który skróci dowolną nazwę zmiennej do maksymalnej długości `n` w sposób opisany poniższym algorytmem. Napisz nazwę zmiennej w postaci, w której życzyłbyś sobie ją widzieć – używaj tylko liter, cyfr oraz znaków '_' (podkreślenie) i '$' (dolar);
- jeśli zmienna zawiera znaki niedopuszczalne – zwróć w wyniku „0”;
- jeśli długość nazwy jest mniejsza lub równa `n`, możesz jej użyć i nie musisz robić nic więcej;
- w przeciwnym wypadku usuwaj z nazwy, począwszy od końca, wszystkie znaki, które nie są literami i cyframi – w chwili, w której długość nazwy osiągnie `n`, możesz zakończyć pracę i użyć nazwy zmiennej;
- jeśli długość nazwy nadal jest większa od `n`, usuwaj z niej, począwszy od końca, kolejne cyfry – w chwili, w której długość nazwy osiągnie `n`, możesz zakończyć pracę i użyć nazwy zmiennej;
- jeśli długość nazwy nadal jest większa od `n`, usuwaj z niej, począwszy od początku, kolejne samogłoski z wyjątkiem pierwszej (chodzi o to, by w nazwie została chociaż jedna samogłoska, o ile jakakolwiek została użyta) – w chwili, w której długość nazwy osiągnie `n`, możesz zakończyć pracę i użyć nazwy zmiennej;
- jeśli długość nazwy nadal jest większa od `n`, usuwaj z niej znaki od końca, począwszy od przedostatniego – w chwili, w której długość nazwy osiągnie `n`, możesz zakończyć pracę i użyć nazwy zmiennej.

Dane wejściowe: plik tekstowy zawierający w kolejnych porcjach danych: 2 wiersze zawierające kolejno: maksymalną dopuszczalną długość zmiennej (`n`: `n` >= 1 i `n` <= 65535) nazwę zmiennej (o długości `l`), która będzie podlegać skracaniu (`l`: `l` >= 1 i `l` <= 65636)
Dane wyjściowe:
Po jednym wierszu na każdy przypadek testowy, zawierający nazwę zmiennej poddaną algorytmowi skracania.
Przykład:
Wejście:
- 7
- ALA_MA_KOTA_I_2_PSY
- 10
- SUPER_ZMIENNA(

Wyjście:
- ALMKTPS
- 0

Zad. 48:
Zmień małe na wielkie litery i odwrotnie w napisie przeczytanym z klawiatury.

Zad. 49:
Napisać program, który „rysuje” w trybie tekstowym choinkę o liczbie segmentów zadanej przez użytkownika. Przykładowo dla dwóch segmentów winno się uzyskać:

```
  *
 ***
  *
 ***
*****
```

Zad. 50:
Zaimplementować algorytm Euklidesa dla liczb naturalnych.

Zad. 51:
Napisać program wyznaczający największy wspólny dzielnik i najmniejszą wspólną wielokrotność dwóch liczb naturalnych podanych przez użytkownika.

Zad. 52:
Zadeklaruj zmienną wiek typu `int`. Wczytaj z klawiatury swój wiek i go wyświetl. Spróbuj wpisać coś niepoprawnego na wejściu (np. *„x”* lub *„Ala ma kota”*). Co się dzieje?

Zad. 53:
Jaki będzie wynik: `float a = 7 / 2; cout << a;` ?
Co zrobić, aby poprzednie wyrażenie dało prawidłowy wynik?

Zad. 54:
Napisz program, który pobiera od użytkownika ogniskową aparatu (f) w milimetrach, odległość od celu (Z) w metrach oraz wielkość celu (X) w metrach oraz wylicza wielkość celu na migawce (x) w milimetrach aparatu zgodnie ze wzorem $x=X \cdot f/Z$. Pamiętaj o odpowiedniej zamianie jednostek.

Zad. 55:
Co będzie wynikiem działania takiego kodu:

```
// 1)
        int a = 1;
        a = a++;
        a = ++a;
        a = a++ + ++a;
        printf("%d %d\n", ++a, ++a);
        printf("%d %d\n", a++, a++);
// 2)
        int a = 2147483647;
        cout << a++ << endl;
        cout << a << endl;
```

?

Zad. 56:
Napisz program liczący konkretny wyraz ciągu geometrycznego. Użytkownik podaje a_1 oraz q. Program ma podać pierwsze dziesięć wyrazów. Wyrazy mogą być zmiennoprzecinkowe. Wzór na element szeregu geometrycznego: $a_n=a_1 \cdot q^{(n-1)}$

Zad. 57:
Co będzie wynikiem:
```
int a = 1, b = 2;
if(++a = b) cout << "Równe";
else cout << "Różne";
cout << a << " " << b << endl;
```
?

Zad. 58:
Zadeklaruj trzy zmienne o nazwach A, B, C, które będą współczynnikami równania kwadratowego zapisanego w formie: $Ax^2 + Bx + C = 0$. Napisz program rozwiązujący te równanie kwadratowe dla współczynników podanych z klawiatury. Zadbaj o idiotoodporność programu.
Pierwiastek:
```
#include <cmath>
sqrt()
```

Zad. 59:
Jaki będzie wynik działania kodu:
```
int i;
for (i = 1; i <= 5; ++i) cout << i;
for ( ; i >= 1; i --) cout << i;
```
?

Zad. 60:
Co będą robić następujące pętle?
```
for (;;) { /* ... */ }
for (;1;) { /* ... */ }
for (a;a;a) { /* ... */ } //a należy do N
while (1) { /* ... */ }
do { /* ... */ } while (1);
```
Jak je zatrzymać?

Zad. 61:
Napisz program, który wczytuje liczby naturalne, aż do podania przez użytkownika 0. Następnie wypisuje minimalną, maksymalną liczbę spośród podanych oraz ich średnią.

Zad. 62:
Wykorzystując pętlę `for` napisz program, który wyświetli parzyste liczby całkowite z zakresu od 31 do 52.

Zad. 63:
Napisz program, który wyświetli na ekranie liczby z zakresu od 1 do 100 podzielne przez 4, ale niepodzielne przez 8 i niepodzielne przez 10. Wykorzystaj w tym celu instrukcję `continue`.

Zad. 64:
Napisz program, określający ile lat trzeba oszczędzać w banku na lokacie 5%, aby przy zarobkach rzędu 12 000 zł rocznie netto, mieć na koncie sumę co najmniej 200 000 zł. Załóż, że od odsetek ani dochodu nie jest pobierany żaden podatek.

Zad. 65:
Napisz program generujący tabliczkę mnożenia 10 x 10 i wyświetlający ją na ekranie.

Zad. 66:
Napisz program wyznaczający silnię podanej przez użytkownika liczby.

Zad. 67:
Napisz program *kalkulator*, który będzie realizował następujące operacje:
- Dodawanie dwóch liczb.
- Odejmowanie dwóch liczb.
- Dzielenie dwóch liczb.
- Mnożenie dwóch liczb.
- Wyznaczanie pierwiastka kwadratowego z liczby.
- Wyznaczanie procent liczby.
- Wyznaczanie reszty z dzielenia dwóch liczb.
- Wyznaczanie dowolnej potęgi danej liczby.

Kalkulator powinien umożliwiać wybór operacji tak długo jak tego chce użytkownik.
Do realizacji tego programu przydatne mogą być instrukcje: `cin.good()` oraz `cin.fail()`.

Zad. 68:
Sprawdź ile dni/godzin/sekund zostało do końca roku (podpowiedź: `time_t`).

Zad. 69:
Napisz aplikację, która na wejściu dostaje napis postaci *„W Roku Pańskim 1345, władca Henryk 12, na rzecz swoich 143209 poddanych uchwalił dekret o 20 procentowej zniżce podatków”*. Twoim zadaniem jest wyłuskać wszystkie liczby i wyświetlić ich sumę.

Zad. 70:
Napisz program do obsługi książki adresowej. Każdy kontakt to oddzielna struktura (imię, nazwisko, gg). Program ma umożliwiać zapisanie do 100 kontaktów, odczyt całej książki adresowej, szukanie konkretnej osoby po nazwisku, zapis i odczyt całej struktury do pliku tekstowego. Wykorzystaj funkcje!

Zad. 71:
Napisz program wykonujący działania na dwóch liczbach zespolonych podawanych z klawiatury (każda liczba to struktura): dodawanie, odejmowanie, mnożenie.

Zad. 72:
Wyświetl ładnie tabliczkę mnożenia (10x10) na ekran. Ładnie, to znaczy tak, aby poszczególne elementy były wyrównane do prawej i zajmowały tyle samo miejsca (użyj `prinf`, potem spróbuj z `cout`).

Zad. 73:

Dokonaj normalizacji danych w pliku. Załóżmy, że plik `input.txt` wygląda tak:

```
Case1     1    0,12
Case2     1    1,90
Case1     2    100,89
Case3     1    645,1
Case2     2    44,9
Case4     1    0,33333333
Case1     3    1,12
Case1     4    0,07
```

1. Dla każdego przypadku testowego wyświetl średnią z trzeciej kolumny.
2. Znormalizuj wszystkie dane w trzeciej kolumnie do przedziału [0…1].
3. Zapisz tak przetworzony plik do nowego pliku tekstowego.

$y_{min} = 0$
$y_{max} = 1$

Zad. 74:

Napisz funkcję symulującą grę w statki przeciwko komputerowi. Zakładamy, że tylko człowiek „strzela". Na początku komputer ma wygenerować tablicę 10x10 losowo wypełnioną maksymalnie 10 jednomasztowcami (nie musisz przejmować się, jeśli zdarzy się, że wylosowane będzie już zajęte miejsce). Tablica ta jest nieznana użytkownikowi. W kolejnym kroku program pyta użytkownika o koordynaty strzału (numer wiersza i kolumny). Po oddanym strzale następuje wypisanie komunikatu: *„Pudło!"*, *„Już strzelałeś/łaś w to pole!"*, *„Trafiony, zatopiony!"*. Gra trwa dopóki użytkownik nie strąci wszystkich statków. Po zakończonej grze należy wypisać komunikat, ile strzałów zużył użytkownik.

Dodatkowo:
program ma umożliwiać zapis do pliku tekstowego wybranej liczby (podanej przez użytkownika) plansz do gry w statki (czyli tego, co generuje w pierwszym kroku).

Uwaga: generator liczb losowych w C++:
Inicjalizacja:

```
srand ( time(NULL) ); //gdzieś w main(), na początku, wystarczy raz
```

Użycie:

```
int liczba_losowa = rand()%10; //zwraca liczbę pseudolosową z zakresu 0…RAND_MAX (co najmniej do 32267)
```

Zad. 75:

Napisz program, który zlicza liczbę podanych przez użytkownika znaków w pliku tekstowym o nazwie podawanej przez użytkownika. Przykładowe wyjście i wejście programu:

```
//w ala.txt mamy zawarte:    Ala ma kota
Podaj nazwę pliku wejściowego: ala.txt
Podaj znak do sprawdzenia: a
Liczba znaków „a" w podanym ciągu wynosi: 3
```

Zad. 76:
Zapisz funkcję obliczającą sumę elementów tablicy jednowymiarowej (`int`) od wskazanego indeksu początkowego do wskazanego indeksu końcowego. Jeżeli użytkownik poda w pierwszej kolejności indeks większy należy zamienić parametry tak, aby otrzymać prawidłowy zakres. Program ma również zapisywać do pliku tekstowego *suma.txt* sumę elementów o parzystych indeksach.

Zad. 77:
Napisz konwerter temperatur. W parametrze funkcja dostaje tablicę oraz dwa parametry: z czego ma zamienić i na co (np. `zamiana(tablica[], F, K)`). Ma zamienić wewnątrz tablicy i zwrócić wartość 0 lub 1 w zależności od sukcesu (zwróci 1 jeśli np. temperatura w Kelvinach była < 0).
Wzory:

Celsjusz na Fahrenheit	°F = (°C × 1.8) + 32
Celsjusz na Kelvin	°K = °C + 273.15
Kelvin na Fahrenheit	°F = (°K × 1.8) - 459.67
Kelvin na Celsjusz	°C = K – 273.15
Fahrenheit na Celsjusz	°C = (°F – 32) /1.8
Fahrenheit na Kelvin	°K = (°F + 459.67) × 5/9

Zad. 78:
Każda z poniższych funkcjonalności powinna być oddzielną funkcją:

1. Wypisz całą tablicę ASCII (`char`-y) na ekran (bezparametrowa)
2. Zapisz ją do tablicy o nazwie `tab_ASCII` w programie. (dostaje w parametrze nazwę tablicy do której ma zapisywać)
3. Zapisz co drugi element tablicy do pliku tekstowego (dostaje w parametrze nazwę pliku wyjściowego)

W głównym programie wywołaj przedstawione powyżej funkcje w odpowiedniej kolejności.

Zad. 79:
Napisać program, który wczytuje od użytkownika liczbę oznaczającą, ile należy wylosować liczb z zakresu od -100 do 100. Następnie dokonuje obliczeń stosunku wylosowanych liczb dodatnich (bez zera) do liczb ujemnych oraz najmniejszej i największej znalezionej liczby. Wynikiem działania programu powinna być informacja na temat wszystkich trzech wartości. Ponadto, program powinien zapisać do tablicy `liczbyParzyste` (rozmiar tablicy na stałe 201) wszystkie liczby parzyste wylosowane, a pozostałe puste miejsce uzupełnić zerami.
W programie nie należy używać tablic.

Przykład dla liczby podanej przez użytkownika 3, następuje losowanie:
`11 -5 93`
W wyniku na ekran zostaje wyświetlona informacja:
`2 -5 93`
// bo stosunek 2/1 min: -5, max: 93

Zad. 80:
Sprawdź, czy zadana liczba jest podzielna przez 3.

Zad. 81:
Sprawdź, czy zadana liczba jest parzysta, dwucyfrowa i podzielna przez 3.

Zad. 82:
Napisać funkcję `SprawdzPlik()`, która jako parametr otrzymuje nazwy dwóch plików tekstowych. W pierwszym pliku są zapisane ciągi zero-jedynkowe. W wyniku działania funkcji powinna zostać zwrócona wartość `true` (jeśli we wszystkich ciągach było tyle samo zer co jedynek) lub `false` (w przeciwnym przypadku). Ponadto funkcja powinna przepisać do drugiego pliku ciągi z równą liczbą zer i jedynek.

Zad. 83:
Dany jest ciąg liczb całkowitych zapisany w tablicy. Napisz funkcję, której parametrem będzie owa tablica (wielkość tablicy może się zmieniać w poszczególnych wywołaniach funkcji). Funkcja ta ma wyświetlić następujące informacje:
- informację, czy ciąg jest rosnący, malejący lub inny (ani rosnący, ani malejący),
- jaka jest suma pierwszego i ostatniego elementu
- ile elementów jest jednocześnie parzystych, dwucyfrowych oraz podzielnych przez 3.

Funkcja nie zwraca żadnej wartości.

Przykład dla: `Tablica[] = {2, 5, 7, 19, 27, 40, 42, 53, 60, 64}`

```
ciąg jest: rosnący
suma pierwszego i ostatniego elementu: 66                              // 2 + 64
liczba elementów parzystych, dwucyfrowych i podzielnych przez 3: 2     // 42, 60
```

Zad. 84:
Przykład ciągu:
7 3 5 78 23 14 3 7
Pierwsza liczba: 7 - oznacza liczbę danych wejściowych.
Pozostałe wyrazy ciągu oznaczają dane wejściowe: 3 5 78 23 14 3 7, których dokładnie jest 7.
Wynika stąd, że ciąg pusty wygląda następująco:
0

Zadania:
Dany jest ciąg liczb całkowitych, poprzedzony ich liczbą (definicja, jak powyżej). Napisz program:
1. Ile elementów tego ciągu stanowi wielokrotność pierwszego elementu ciągu?
2. Ile elementów tego ciągu jest podzielnych przez swoją ostatnią cyfrę?
3. Sprawdź, czy ciąg ten jest rosnący.
4. Ile elementów tego ciągu jest sumą dwóch swoich bezpośrednich poprzedników?
5. Oblicz sumę pierwszego i ostatniego elementu tego ciągu.

Uwaga: wszystkie zadania można (i należy) wykonać bez użycia tablic.

Zad. 85:
Napisz funkcję, która otrzymuje w parametrze tablicę kwadratową dwuwymiarową `int`ów o rozmiarach podawanych w parametrze do funkcji. Na wyjściu ma zwracać posortowaną rosnąco tablicę jednowymiarową złożoną z elementów pierwszej tablicy.

Zad. 86:

Napisz program „generator losowych numerów telefonów”. Program ma umożliwiać generowanie podanej przez użytkownika liczby numerów telefonów z konkretnej strefy numeracyjnej. Przykładowe zastosowanie:

```
for(int i=11;i++;i<33) {
setWojewództwo(i); //ustawienie strefy numeracji na i oraz
     wyczyszczenie tablicy wewnętrznej z numerami telefonów
generate(1000); //wygenerowanie 1000 numerów, zapisywanych w
     tablicy (odpowiednio zadeklarowanej wcześniej).

save(i+„plik.tel”) //zapis do pliku tekstowego – ścieżka
     podawana w parametrze

show(); //wypisuje na ekran numery zapisane w tablicy
}
```

Zad. 87:

Napisz program, który sprawdza czy numer PESEL jest poprawny. Numery PESEL mają być czytane z pliku `pesele.txt`. Każdy numer jest w oddzielnej linii. W wyniku program ma zwrócić informację, ile było poprawnych numerów PESEL w pliku. Algorytm sprawdzania jest następujący:

4	4	0	5	1	4	0	1	3	5	8
a	b	c	d	e	f	g	h	i	j	k

PESEL jest poprawny, jeśli:

*(10-K) = (1*a + 3*b + 7*c + 9*d + 1*e + 3*f + 7*g + 9*h + 1*i + 3*j) (mod 10)*

Przykład dla numeru PESEL 44051401358:

1*4 + 3*4 + 7*0 + 9*5 + 1*1 + 3*4 + 7*0 + 9*1 + 1*3 + 3*5 = 101

Wyznaczamy resztę z dzielenia sumy przez 10: 101/10 = 10 reszta = 1

Jeżeli reszta = 0, to cyfra kontrolna wynosi 0. Jeżeli reszta ≠ 0, to cyfra kontrolna będzie uzupełnieniem reszty do 10, czyli w podanym przykładzie jest to cyfra 9.

10 - 1 = 9

Wynik 9 nie jest równy ostatniej cyfrze numeru PESEL, czyli 8, więc numer zawiera błąd.

Zad. 88:

Zaprojektuj program realizujący następujące funkcjonalności:

a) Zaprojektuj strukturę opisującą samochód.

b) Jeśli auto będzie miało więcej niż 2 litry pojemności, zwiększ opłatę za ubezpieczenie o 5%.

c) Dla głównego użytkownika auta (kobieta) zastosuj bonus ubezpieczeniowy 10%.

d) Napisz program, który zapisuje do 100 użytkowników w tablicy struktur.

e) Napisz funkcję `save(string fileName, string color)` zapisującą do pliku `fileName` wszystkie informacje o samochodach koloru `color`.

Struktura: (imię, nazwisko posiadacza samochodu, płeć, rok produkcji samochodu, pojemność, kolor, stawka ubezpieczenia).

Zad. 89:

Dana jest następująca tablica struktur:

```
struct student {
     double ocena_dyplom;
```

```
    int czyWnioslOplate;
    string nazwisko;
    }
student kartoteka[100];
```

Napisz funkcję `doObrony(string nazwaPliku)`:

a) Funkcja zwraca liczbę całkowitą studentów dopuszczonych do obrony (takich, którzy mają ocenę na dyplomie > 3 oraz wniesioną opłatę (`czyWnioslOplate` równe 1).
b) Funkcja powinna zapisywać do pliku tekstowego `<nazwaPliku>` w kolejności alfabetycznej nazwiska studentów dopuszczonych do obrony.

Zad. 90:

Napisz funkcję `liczbaRozdzialow(string nazwaPliku, bool & ok)`, która przyjmuje nazwę pliku tekstowego oraz:

1. Zwraca liczbę całkowitą wystąpień słowa *„rozdział”* w tekście
2. W zmiennej `ok` ustawia wartość, czy numeracja rozdziałów jest ciągła (czyli nie ma „dziur”)
3. Uwzględnij, iż słówko *rozdział* może być pisane wielkimi lub małymi literami (lub mieszanie).

Zad. 91:

Napisz funkcję `generujTozsamosc(int ile, string[] imiona, string[] nazwiska, int[] numery)`, która:

a) Generuje *ile* danych osób (imię, nazwisko, nr telefonu) i zapisującą je do pliku tekstowego *daneXXX.txt*, gdzie XXX – to liczba *ile* (zapisana na **trzech** pozycjach).
b) Dane osób są losowane z tablic `imiona`, `nazwiska` oraz `numery`, przy czym każdą z danych można wykorzystać tylko **raz**.
c) Jeśli braknie danych służących do generowania, funkcja powinna zwrócić **false**.

Zad. 92:

Załóżmy, że w pliku *eksperymenty.txt* trzymane są wyniki pewnych obserwacji w następującym formacie:

```
False    0    4
False    1    12
True     2    102
False    3    128
True     4    210
True     5    1
False    7    1200
```

Twoim zadaniem jest sprawdzenie, czy plik jest poprawnie zbudowany. Poprawny plik charakteryzuje się:

a) Taką samą liczbą eksperymentów oznaczonych jako „*false*” i jako „*true*”.
b) Kolejnymi, bez przerw liczbami w drugiej kolumnie (tj. numeracja ciągła, bez „dziur”) (uwaga: numeracja niekoniecznie rozpoczyna się od 0).

Dane w pliku oddzielone są znakami tabulacji. W wyniku program ma wyświetlić na ekran informację *„plik poprawny”* lub *„plik niepoprawny”*.

Zad. 93:

Wyznaczyć wszystkie liczby pierwsze od 0 do 100.

Zad. 94:
Zadeklaruj trzy zmienne typu `int`: `x`, `y`, `z`. Przypisz im dowolne wartości całkowite przeczytane z klawiatury, po czym wykonaj działanie `x % y % z`. Wynik tego działania przypisz czwartej zmiennej nazwanej `rezultat` i wyświetl jej zawartość na ekranie.
Spróbuj tak dobrać wartości zmiennych z poprzedniego ćwiczenia, by wynikiem nie było zero.

Zad. 95:
Zadeklaruj zmienną typu `int` o dowolnej nazwie i zainicjuj ją wartością 256. Wykonaj dwa działania (na tej samej zmiennej):
◦ Przesunięcie bitowe w prawo o 2 miejsca.
◦ Przesunięcie bitowe w lewo o 2 miejsca.
Wynik wszystkich trzech działań wyświetl na ekranie.

Zad. 96:
Napisać program wyliczający pole trójkąta – program wczytuje wysokość *h*, podstawę *a* (są to dowolne liczby rzeczywiste), oblicza pole ($P=(a*h)/2$) i wyświetla wynik. Zakładamy, że długości *a* i *h* są wyrażone w **centymetrach**, wynik ma być wyrażony w **metrach kw**. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 97:
Napisać program wyliczający pole koła oraz kwadratu na nim opisanego – program wczytuje promień *r* (to dowolna liczba rzeczywista), oblicza pole ($P=\pi*r^2$), długość boku *a* kwadratu opisanego na takim okręgu, oraz jego pole ($P=a^2$) i wyświetla te wyniki. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 98:
Prędkość w ruchu jednostajnym prostoliniowym może być określona uproszczonym wzorem $v=s/t$, gdzie *s* to droga przebyta w czasie *t*. Napisać program wyliczający prędkość *v* – program wczytuje drogę *s* i czas jej przebycia *t* (są to dowolne liczby rzeczywiste) i wyświetla wynik. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 99:
Cena brutto to cena netto powiększona o pewien podatek, wyrażony procentowo. Jeżeli coś kosztuje netto 100zł, a kwota podatku to 23%, cena brutto wynosi 123zł. Napisać program, który wyznaczy cenę brutto na podstawie ceny netto oraz podatku wyrażonego procentowo – program wczytuje cenę netto, podatek wyrażony procentowo (są to dowolne liczby rzeczywiste) i wyświetla wynik. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 100:
Korzystając z funkcji `printf` wyświetl na ekranie tabliczkę mnożenia 15 x 15, tak aby wszystkie jej elementy były równo oddalone od siebie.

Zad. 101:
Wynagrodzenie pewnego pracownika to liczba przepracowanych godzin przemnożona przez stawkę godzinową. Napisać program, który wyznaczy wynagrodzenie pracownika po wczytaniu liczby przepracowanych godzin oraz stawki (są to dowolne liczby rzeczywiste). Dodatkowo program ma wyznaczyć, ile pracownik zarobił na dniówkę, zakładając, że pracuje zawsze równo 8 godzin. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyników – do czasu naciśnięcia klawisza Enter.

Zad. 102:
Szybkostrzelność teoretyczna karabinka automatycznego AK (Automat Kałasznikowa) wynosi 600 strzałów/minutę. Magazynek karabinka mieści 30 naboi. Napisać program, który wczyta wyrażony w sekundach czas (dowolna liczba całkowita) trwania ognia ciągłego, prowadzonego z takiego karabinka. Ćwiczeniem programu jest wyznaczyć liczbę magazynków, które trzeba by wymienić, aby strzelać ogniem ciągłym przez wprowadzony czas. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 103:
Sportowiec w trakcie jednego treningu spala średnio 1500 kalorii. Napisać program, który wczyta: ile razy sportowiec trenuje w tygodniu i ile planuje tygodni trenować (dowolne liczby całkowite). Na tej podstawie program ma wyliczyć, ile kilokalorii sportowiec spali w tym okresie czasu. Program powinien na samym początku wyświetlić krótką informację o jego przeznaczeniu oraz zatrzymać swoje wykonanie po wyświetleniu wyniku – do czasu naciśnięcia klawisza Enter.

Zad. 104:
Mając dany następujący plik wejściowy rozdzielany przecinkami:

```
Imie, Nazwisko, Wzrost
Jan, Nowak, 1.80
Zbigniew, Kowalski, 2.40
Tomasz, Kozak, 1.64
Klara, Rydz, 1.70
```

Wczytaj z niego danego korzystając **uprzednio zadeklarowanej** przez siebie **struktury** (`struct`). Następnie posortuj te dane rosnąco (dowolnym algorytmem sortowania) według nazwiska i wypisz je na ekran ze stałymi odstępami. Na końcu zapisz dane posortowane do pliku o nazwie *„test2.txt”*.

Zad. 105:
Zaprojektuj strukturę składająca się z dwóch pól: `liczba` oraz `parzystość`. Następnie stwórz 50-elementową tablicę takich struktur. Po stworzeniu tablicy wypełnij polę `liczba` dla każdej struktury, losowo wygenerowaną liczbą całkowitą z zakresu od 5-40. Następnie dla każdego elementu tablicy określ, czy wartość w polu liczba jest parzysta bądź nie i wpisz tę informację do pola `parzystość`. Poprawnie wypełniona struktura może wyglądać następująco (przy założeniu, że zmienna `a` jest strukturą):

```
a.liczba = 30;
a.parzystosc = true;
```

Następnie korzystając z funkcji `printf` wyświetl zawartość całej tablicy (czyli wylosowaną liczbę i informacje o tym, czy jest ona parzysta czy nie) na ekranie.

Zad. 106:

Napisz program, który sprawdza, **czy podana liczba jest liczbą pierwszą** oraz jeśli nie, wyznacza najbliższą do niej liczbę pierwszą. Zadbaj o walidację wejścia.

Przykład: *Podaj liczbę do sprawdzenia: 14.*
Podana liczba nie jest liczbą pierwszą. Najbliższa jej liczba pierwsza to 13.

Program powinien być odporny na próby wpisania jakichkolwiek błędnych znaków (walidacja wejścia). Proszę nie korzystać z bibliotek matematycznych.

Zad. 107:

Napisać procedurę `void emerytura(char nazwaPliku[])`, która wczyta z pliku o podanej nazwie dane pracowników zapisane w kolejnych wierszach w następujący sposób:

```
Imię Nazwisko Płeć Wiek
```

oraz zapisze je do zmiennej typu strukturalnego (`struct`) zawierającej pola `imie`, `nazwisko`, `plec`, `wiek`.

Pola imię i nazwisko to ciągi znaków, płeć to jeden znak, natomiast wiek to liczba całkowita. Następnie procedura dla każdego pracownika powinna wyznaczyć, ile lat zostało do jego emerytury (zakładając, że wiek emerytalny mężczyzn to 65, a kobiet 60 lat). Wyniki należy zapisać w pliku *mezczyzni.txt* lub *kobiety.txt* (w zależności od płci pracownika) w następujący sposób:
`Nazwisko Imię „Lata do emerytury”`

Przykład:

```
Tomasz Nowak M 45
Marta Ziobro K 42
Jan Kowalski M 27
Ewelina Tusk K 59
```

Plik mezczyzni.txt:

```
Nowak Tomasz 20
Kowalski Jan 38
```

Plik kobiety.txt

```
Tusk Ewelina 1
Ziobro Marta 18
```

Zad. 108:

Napisz program, który pobiera od użytkownika liczby (do maksymalnie 10 liczb) z zakresu od -100 do 100 wprowadzając je do tablicy oraz sortuje je metodą sortowania bąbelkowego w kolejności rosnącej. Zakończenie wprowadzania liczb symbolizuje liczba 0 (niebrana pod uwagę). Wyświetl na ekran tablicę przed jak i po sortowaniu. Algorytm sortowania bąbelkowego jest następujący:
Ciąg wejściowy to [4,2,5,1,7]. Każdy wiersz symbolizuje wypchnięcie kolejnego największego elementu na koniec („wypłynięcie największego bąbelka“). W każdym przebiegu porównywane są ze sobą dwa kolejne elementy i zamieniane miejscami, jeśli ich kolejność się nie zgadza. Jeśli w danym przebiegu nie dokonano żadnej zmiany, algorytm kończy działanie.

$$\underbrace{[4,2}_{4>2},5,1,7] \rightarrow [2,\underbrace{4,5}_{4<5},1,7] \rightarrow [2,4,\underbrace{5,1}_{5>1},7] \rightarrow [2,4,1,\underbrace{5,7}_{5<7}]$$

$$\underbrace{[2,4}_{2<4},1,5,7] \rightarrow [2,\underbrace{4,1}_{4>1},5,7] \rightarrow [2,1,\underbrace{4,5}_{4<5},7]$$

$$\underbrace{[2,1}_{2>1},4,5,7] \rightarrow [1,\underbrace{2,4}_{2<4},5,7]$$

$$\underbrace{[1,2}_{1<2},4,5,7]$$

Zad. 109:
Napisz program, który:
- stworzy tablicę 5 x 5 liczb rzeczywistych i wypełni ją losowymi wartościami z zakresu [0, 10].
- dla każdej kolumny wyznaczy jej minimum i maksimum.
- wyznaczy trzy największe liczby na lewej przekątnej.
- wyznaczy trzy największe liczby na prawej przekątnej.

Wszystkie wyznaczone wartości wraz z wygenerowaną tablicą mają zostać wyświetlone na ekranie i zapisane do pliku.

Zad. 110:
Napisz program, który wyświetla tabliczkę mnożenia, której rozmiar podaje użytkownik (od 2 x 2 do 5 x 5), ale zapisaną w kodzie binarnym. Zadbaj o poprawność rozmiaru tabliczki mnożenia (użytkownik podaje zawsze dowolne liczby całkowite jako jej początkowy rozmiar). Załóż, że tabliczka mnożenia zawsze będzie tablicą kwadratową.

Przykład:
```
Podaj rozmiar tabliczki: 2
001 010
010 100
```

Zad. 111:
Napisz program, który pobiera od użytkownika liczby całkowite. Zakończenie wprowadzania liczb określa liczba 0 (nie brana pod uwagę w wyliczeniach). W wyniku jego działania powinny zostać wyświetlone na ekranie:
- największa i najmniejsza z wprowadzonych liczb oraz iloczyn liczb nieparzystych dodatnich;
- ciąg liczb Fibonacciego o wartościach mniejszych niż iloczyn wartości bezwzględnych największej i najmniejszej liczby wprowadzonej w pierwszym kroku.

Program powinien być odporny na próby wpisania jakichkolwiek błędnych znaków (walidacja wejścia). Proszę nie korzystać z bibliotek matematycznych.
Przykład: (użytkownik w kolejnych krokach podał liczby -5, 10, 2, 3, 7, 0)
```
Największa liczba: 10.
Najmniejsza liczba: -5.
Iloczyn liczb nieparzystych dodatnich: 21.
Ciąg Fibonacciego: 1,1,2,3,5,8,13,21,34.
```

Zad. 112:
Napisz program, który pobiera od użytkownika łańcuch znaków o maksymalnej długości 40 znaków i sprawdza, czy jest on palindromem. W przypadku, gdy podany tekst nie jest palindromem, program powinien wyświetlić minimalną liczbę zmian znaków potrzebną, by zmienić podany ciąg w palindrom. Uwaga: zakłada się, że wielkość liter nie ma znaczenia.

Przykład1:
```
Podaj tekst: robot
Najmniejsza liczba zmian liter konieczna, by przekształcić słowo
„robot” w palindrom wynosi 1.
```

Przykład2:
```
Podaj tekst: AlaAlA
Podany ciąg jest palindromem.
```

Zad. 113:
Napisz program, który wyświetla tabliczkę mnożenia, której rozmiar podaje użytkownik (od 2 x 2 do 10 x 10), ale zapisaną w kodzie binarnym lub szesnastkowym (w zależności od wyboru użytkownika). Zadbaj o poprawność rozmiaru tabliczki mnożenia – użytkownik może podać dowolny ciąg znaków, należy zadbać by był on prawidłową liczbą (zgodnie z treścią zadania) jak również pozostałych danych wprowadzanych przez użytkownika.

Przykład:
```
Podaj rozmiar tabliczki: 2
Podaj system liczenia (b – binarny, s – szesnastkowy): b
001 010
010 100
```
Proszę nie korzystać z bibliotek matematycznych.

Zad. 114:
Napisz program, który wczytuje od użytkownika ciąg znaków zakończony wciśnięciem klawisza Enter (dopuszczalne są znaki tylko z alfabetu angielskiego). Program powinien wyświetlić:

1. Liczbę białych znaków w tekście (rozumianych jako spacje i tabulator poziomy).
2. Liczbę liter w tekście.
3. Liczbę samogłosek w tekście (tylko z alfabetu angielskiego).
4. Sumę wszystkich cyfr podanych w tekście.

Przykład:
```
Podaj tekst: Ala ma          2 koty i 15 rybek.
Białe znaki: 7
Litery: 15
Samogłoski: 8
Suma cyfr: 8
```

Zad. 115:
Napisz program, który utworzy tablicę 20 liczb całkowitych, generowanych losowo z przedziału [-10 , 10], a następnie wypisze na ekranie, ile razy każda z liczb z tego przedziału powtarza się w tablicy. Przykład:
```
Wylosowane liczby: -4 -4 1 2 3 -5 -5 -5 -5 2 1 1 4 4 4 4 4 4 4 4
Wystąpienia:
-5 - 4
-4 - 2
 1 - 3
 2 - 2
 3 - 1
 4 - 8
```

Do wygenerowania liczb losowych użyj funkcji `rand()`.
```
srand ( time(NULL) ); - inicjalizacja generatora liczb pseudolosowych
rand()          - wygenerowanie liczby całkowitej z zakresu [0, 32767]
```

Zad. 116:
Napisz program, który zamienia liczbę dziesiętną podaną przez użytkownika na liczbę binarną i szesnastkową.

Zad. 117:
Napisz program, który pobiera od użytkownika *N* liczb całkowitych z zakresu [0, 100]. Liczba *N* również jest podawana przez użytkownika. Po wczytaniu program wyświetla sumę największej i najmniejszej z podanych liczb, a następnie sumę drugiej w kolejności największej i drugiej w kolejności najmniejszej liczby itp. Łącznie powinno zostać wyświetlone dokładnie *N/2* sum.

Przykład dla `N = 10`:
```
podaj 1. liczbę: 2
podaj 2. liczbę: 3
podaj 3. liczbę: 10
podaj 4. liczbę: 7
podaj 5. liczbę: 20
podaj 6. liczbę: 5
podaj 7. liczbę: 4
podaj 8. liczbę: 1
podaj 9. liczbę: 2
podaj 10. liczbę: 8

Wynik programu:
1 + 20 = 21
2 + 10 = 12
2 + 8 = 10
3 + 7 = 10
4 + 5 = 9
```

Zad. 118:
Napisać funkcję `void diff(string plikWe1, string plikWe2)`, której zadaniem jest sprawdzenie różnic w dwóch plikach, których nazwy podane są jako parametry funkcji.
W pierwszej wersji programu powinny zostać wyświetlone linie w plikach, które są w jakiś sposób rożne, z zaznaczeniem, która to jest linia i w którym pliku. Jako różnicę należy potraktować dodatkowe linie (nie występujące w drugim pliku) lub zmiany w konkretniej linii, np.

```
Plik 1:
To jest pierwsza linia
w przykładowym pliku.
Są też jakieś kolejne.
```

```
Plik 2:
To jest pierwsza linia
w innym pliku.
Bo przecież są dwa
podane pliki z liniami.
Są też jakieś kolejne.
```

W wyniku powinno zostać wyświetlone:
```
Różnice w pliku:
p1 l.2: w przykładowym pliku.
p2 l.2: w innym pliku.

Dodatkowe linie:
p2 l.3: Bo przecież są dwa
p2 l.4: podane pliki z liniami.
```

Zad. 119:
Czym różni się operator `[]` od funkcji `at`? Sprawdź to praktycznie.

Zad. 120:

Zaimplementować funkcję `int sum(string s)`, która zlicza sumę wszystkich cyfr całkowitych występujących w podanym łańcuchu znaków. Funkcja zwraca wartość wyliczonej sumy.
Dla następujących wywołań funkcji:

```
sum(”0123”);
sum(”01010101”);
sum(”a 1 b 2 c 3 d 4 g 5 i 6 k 7 m”);
```

funkcja powinna zwrócić wartości:

```
6
4
28
```

Zad. 121:

Napisać procedurę `void szukaj(const char plikWe[], const char plikWy[], const char slowo[])`, której zadaniem jest znalezienie wszystkich wierszy w pliku, które zawierają szukane słowo. Wszystkie wiersze, które zawierają szukane słowo powinny być zapisane w pliku wynikowym wraz z nr wiersza (z pierwszego pliku). Nazwa pierwszego pliku zapamiętywana jest w parametrze `plikWe`, nazwa pliku wynikowego podana jest w parametrze `plikWy`, natomiast szukane słowo w parametrze `slowo`.

Przykład:

```
Plik wejściowy:
Ala ma jutro egzamin z biologii.
Jan myje auto.
Eh, jutro kolejny egzamin.
Nie lubie polityki.
```

Jeżeli szukanym słowem byłoby *„egzamin”* to plik wyjściowy powinien wyglądać następująco:

```
1: Ala ma jutro egzamin z biologii.
3: Eh, jutro kolejny egzamin.
```

Zad. 122:

Napisać funkcję `przepisz`, która jako pierwszy parametr otrzymuje nazwę pliku tekstowego, w którym każda linia wygląda następująco:

```
imię*nazwisko*plec*wiek*pensja
```

gdzie imię i nazwisko zapisane są literami alfabetu angielskiego, płeć to litera *'K'* lub *'M'*, a wiek i pensja, to liczby całkowite dodatnie. Liczba osób zapisanych w pliku jest nieokreślona. W rezultacie jej działania powinny powstać dwa pliki wyjściowe, gdzie jeden będzie zawierał jedynie kobiety (nazwa taka sama jak pliku w przypadku wejściowego, poprzedzona literą *'k'*), a drugi mężczyzn (nazwa poprzedzona literą *'m'*).
Jednocześnie do nowych plików należy:
• nie przepisywać oznaczenia płci,
• kobietom o wieku większym niż podany jako drugi parametr (wiek) podnieść pensję o 15%,
• mężczyznom podnieść pensję o tyle, ile mają lat.
W wyniku działania funkcji powinien zostać zwrócony średni wiek wszystkich mężczyzn z pliku wejściowego.
Uwaga, nie należy używać (importować) żadnych dodatkowych plików nagłówkowych, poza klasami niezbędnymi do obsługi plików tekstowych i ciągów znaków.

Zad. 123:

Informatyk z firmy „KompOK” zapisał w pliku *hasla.txt* 200 haseł. Każde hasło umieszczone jest w osobnym wierszu pliku. Hasło składa się tylko z małych liter alfabetu angielskiego, zaś jego długość wynosi od 3 do 10 znaków. Wykorzystując dane zawarte w tym pliku, wykonaj poniższe polecenia. Odpowiedzi do poszczególnych podpunktów zapisz w pliku tekstowym *odpowiedzi1.txt* opatrując je numerem podpunktu:
1. Podaj, ile haseł ma parzystą, a ile nieparzystą liczbę znaków.
2. Utwórz zestawienie haseł (po jednym w wierszu), które są palindromami. Palindrom to wyraz brzmiący tak samo przy czytaniu z lewej strony do prawej, jak i odwrotnie, np. kajak, potop.
3. Utwórz zestawienie haseł (po jednym w wierszu) zawierających w sobie dwa kolejne znaki, których suma kodów ASCII wynosi 220.

Przykłady:
* Hasło *krzysio* zawiera dwa kolejne znaki *si*, których suma kodów ASCII wynosi 220. Kod ASCII znaku *s* to 115, kod znaku *i* to 105; suma kodów wynosi 115+105 = 220.
* Hasło *cyrk* zawiera również takie dwa kolejne znaki. Kod ASCII znaku *c* to 99, kod ASCII znaku *y* to 121; suma kodów wynosi 99+121=220.

Zad. 124:

W pliku *liczby.txt* w oddzielnych wierszach, znajduje się 1000 liczb zapisanych w systemie dwójkowym o długościach zapisów od 2 do 16 cyfr (0 lub 1). Napisz program, którego wykonanie da odpowiedzi do poniższych podpunktów. Odpowiedzi zapisz w pliku *odpowiedzi2.txt*, a każdą odpowiedź poprzedź literą oznaczającą ten podpunkt.
1. Ile jest liczb parzystych w całym pliku?
2. Jaka jest największa liczba w tym pliku? Podaj jej wartość w dwóch systemach: dwójkowym i dziesiętnym.
3. Ile liczb w całym pliku ma dokładnie 9 cyfr? Podaj sumę tych liczb w systemie dwójkowym.

Zad. 125:

Napisać program, którego zadaniem jest odczytanie danych tabelarycznych w pliku tekstowym, a następnie zapisanie ich do nowego pliku w postaci kodu HTML. Dane rozdzielone spacją.

Przykład:

Wejście:

```
"Waga" "Wzrost" "BMI" "Nadwaga"
70 1,8 21,6 "NIE"
67 1,77 21,39 "NIE"
85 1,7 29,41 "TAK"
100 1,92 27,13 "TAK"
```

Wynik:

```
<html><body>
<table>
<tr><td>"Waga"</td><td>"Wzrost"</td><td>"BMI"</td><td>"Nadwaga"</td></tr>
<tr><td>70</td><td>1,8</td><td>21,6</td><td>"NIE"</td></tr>
<tr><td>67</td><td>1,77</td><td>21,39</td><td>"NIE"</td></tr>
<tr><td>85</td><td>1,7</td><td>29,41</td><td>"TAK"</td></tr>
<tr><td>100</td><td>1,92</td><td>27,13</td><td>"TAK"</td></tr>
</table>
</body></html>
```

Zad. 126:

Co roku w Megabajtolandii odbywa się Zlot Obżartuchów. Podczas każdego zlotu tradycją jest, że pierwszego dnia wszyscy uczestnicy obżerają się ciastkami przez całą dobę non-stop, nie mając ani ułamka sekundy przerwy. Jak tylko obżartuch skończy jeść jedno ciastko to od razu musi zabrać się za następne (nie dotyczy to sytuacji pod koniec doby, kiedy to uczestnikowi nie wolno napocząć ciastka jeśli wie, że nie zdąży go zjeść przed końcem doby). Kolejnym ważnym elementem tradycji jest to, że każdy obżartuch je każde swoje ciastko w niezmiennym przez całą dobę, charakterystycznym dla siebie tempie.
Na najbliższy Zlot zostały zaproszone tylko te obżartuchy, które już uczestniczyły w poprzednich Zlotach. Dzięki temu wiadomo z góry, w jakim tempie każdy z nich je ciastka (obżartuchy nie lubią zmieniać swojego wyuczonego tempa). Na podstawie tych danych organizatorzy Zlotu chcą określić, ile należy kupić ciastek. Sytuację utrudnia fakt, że ciastka w sklepie sprzedawane są w pudełkach o stałej wielkości, a nie na sztuki - przez to być może trzeba będzie kupić trochę więcej ciastek niż zostanie zjedzonych.
Zadanie:
* Mając daną liczbę obżartuchów zaproszonych na Zlot oraz czas jedzenia pojedynczego ciastka (podany w sekundach) przez każdego z nich Twój program powinien policzyć, ile należy kupić pudełek z ciastkami.
Specyfikacja wejścia:
* W pierwszej linii znajdują się dwie liczby całkowite N i M oddzielone pojedynczą spacją ($1 \leq N \leq 10.000$, $1 \leq M \leq 1.000.000.000$). Oznaczają one odpowiednio liczbę zaproszonych obżartuchów na Zlot oraz liczbę ciastek w jednym pudełku. Kolejne N wierszy zestawu zawiera po jednej liczbie całkowitej dodatniej niewiększej niż 100.000. Są to czasy (w sekundach) jedzenia pojedynczego ciastka przez kolejnych obżartuchów.
Specyfikacja wyjścia:
* Dla każdego zestawu danych pojawiającego się na wejściu należy wypisać dokładnie jedną liczbę całkowitą (każdą w osobnej linii) oznaczającą liczbę pudełek z ciastkami, jaką organizatorzy muszą kupić na Zlot.
Przykład:
Wejście:
a)
2 10
3600
1800
b)
3 356
123
32999
10101
Wyjście:
8
2

Zad. 127:

Napisz program deszyfrujący podany ciąg (zaszyfrowany prostym szyfrem podstawieniowym:
`DOXQRIRGB MFBOTPWX LPLYX ALPQXGB MIRPFHX`

Zad. 128:
Zadeklaruj dłuższy tekst zawierający słowa zakazane (np. *„matematyka”*) i pozamieniaj wszystkie wystąpienia na coś neutralnego (np. *„kwiatek”*).

Zad. 129:
Napisz funkcję `strfind`, która szuka w tekście (pierwszy parametr) podanej frazy (drugi parametr). Wynikiem funkcji ma być indeks znaku, od którego podana fraza zaczyna się w tekście lub -1, jeżeli tekst nie zawiera szukanej frazy. Wielkość liter w podanych ciągach nie ma znaczenia.
Przykład:
Dla podanego fragmentu programu:

```
char zdanie[] = "Jutro jest egzamin z programowania.";
char fraza[] = "Program";
cout << "Szukam w \"" << zdanie << "\"" << endl;
cout << "\t\"" << fraza << "\" : " << strfind(zdanie,
fraza) << endl;
cout << "\t\"" << fraza << "\" : " << strfind(zdanie,
fraza) << endl;
cout << "\t\"jutro JEST\" : " << strfind(zdanie, "jutro
JEST") << endl;
```

na ekranie powinno zostać wyświetlone:

```
Szukam w "Jutro jest egzamin z programowania."
"Program" : 21
"jutro JEST" : 0
```

Zad. 130:
Zadeklaruj zmienną typu `enum`, w której zapiszesz stałe określające z ilu jednostek czasu składa się rok (tj. np. 4 kwartały, 12 miesięcy, 365 dni, `x` godzin, `y` minut, `z` sekund). Następnie sprawdzaj, ile dni/godzin/sekund zostało do końca roku.

Zad. 131:
Napisz program „Jednoręki bandyta”. Na początku gracz posiada 100 żetonów. Każde pociągnięcie dźwigni jednorękiego bandyty odejmuje mu z konta 5 żetonów. Po pociągnięciu losowanych jest 9 liczb w następującej konfiguracji:

L1	L2	L3
L4	L5	L6
L7	L8	L9

Gracz wygrywa, jeśli w którymkolwiek poziomie, pionie lub ukosie znajdą się te same liczby. Wygrana jest uzależniona od rodzaju liczb: stawka (5 żetonów) * 2 * liczba.
Program ma umożliwiać zakończenie gry w dowolnym momencie.

Zad. 132:
Napisać funkcję `SprawdzCiag()`, która jako parametr otrzymuje ciąg znaków zawierających nawiasy. W wyniku działania funkcji powinna zostać zwrócona wartość `true` (jeśli we wszystkich ciągach było tyle samo nawiasów otwierających i zamykających) lub `false` (w przeciwnym przypadku).

Zad. 133:
Proszę napisać program do symulowania rzutów kostką do gry. Żeby nie było tak łatwo, to:
a) Program ma się kończyć po tym jak użytkownik powie (czyli rzucamy kostką dopóki użytkownik sobie tego życzy; program ma się nie zamykać po jednokrotnym rzucie).
b) Użytkownik wybiera liczbę kostek do rzutów (od 1 do 10 kostek) oraz liczbę ścian kostek (zwykła kostka ma 6 ścian, a w programie można wybrać od 2 do 100 ścian).

Zad. 134:
Zaprogramuj grę w „papier-kamień-nożyce”. W pierwszym kroku,program pyta użytkownika, do ilu wygranych chce grać. Kolejno potem następują rundy, w których może wygrać komputer, użytkownik lub może być remis. Po osiągnięciu limitu wygranych przez któregokolwiek z graczy – następuje zakończenie programu i wyświetlenie wyników.

Zad. 135:
Napisać funkcję sprawdzającą podzielność podanej jako parametr liczby (całkowitej dodatniej `long`) przez 3 metodą sumowania cyfr. Jeżeli po pierwszym zsumowaniu cyfr liczby wartość sumy jest większa niż 9, sumowanie należy przeprowadzić powtórnie, itd. Gdy końcowa suma cyfr jest mniejsza od 10, to funkcja kończy obliczenia i zwraca `true` (suma jest cyfrą 0, 3, 6, 9) lub `false` (suma jest jedną z pozostałych cyfr).
Przykład:
liczba: 123454, kolejne sumy: 19-10-1, wynik: `false`
liczba: 123453, kolejne sumy: 18-9, wynik: `true`

Zad. 136:
Dany jest plik, w którym zapisano informacje o nieruchomościach w postaci kolejnych trójek:
- `String nazwDom;`
- `int lKondygn;`
- `BigDecimal cena;`

Napisać funkcję zwiększającą o 15% cenę domów 2-piętrowych i wpisującą dane tych domów do pliku tekstowego (dane o jednym domu w jednym wierszu; cena po zmianie) i zwracającą jako swoją wartość liczbę domów, których cen nie zmieniono. Nazwy plików są przekazywane przez parametry funkcji.

Zad. 137:
Napisz program, który wyświetla na ekranie tablicę 15x15 liczb całkowitych losowych (z zakresu [0…100)) i wypisuje sumę liczb znajdujących się na przekątnej tablicy. Ponadto, jeśli suma wszystkich liczb w tablicy jest podzielna przez 11, program ma czyścić co drugi rząd w tablicy (czyli wpisywać wartości 0 we wszystkich kolumnach parzystych wierszy).
Przykład (tablica 3x3):

0	66	17
18	99	12
3	4	6

`Suma przekątna: 105`

Zad. 138:
Napisz program „randomowy pokemonizator”. Program na wejściu dostaje od użytkownika zdanie, a następnie losowo wybrane litery zostają zamienione na wielkie. Dodatkowo, program ma liczyć liczbę przeprowadzonych zmian i jeśli ta była nieparzysta – zamienić wszystkie „e” na „3”.

Zad. 139:
Napisz symulator „Gry w życie” na tablicy dwuwymiarowej 10x10. Gra ma wypisywać na ekran w oddzielnych linijkach informacje o liczbie żywych komórek w poszczególnych pokoleniach. Przyjmij maksymalną liczbę pokoleń na 1000. Gra polega na losowym umieszczeniu 30 bakterii na planszy. W każdym pokoleniu analizowane jest bezpośrednie otoczenie każdej bakterii. Jeśli w jej obrębie jest mniej niż 3 lub więcej niż 6 żywych sąsiadów – bakteria ginie. W przeciwnym wypadku – przeżywa lub odradza się.

Zad. 140:
Napisać program, który sprawdza, czy dana osoba ma nadwagę. Dane mają być odczytywane z pliku tekstowego o następującym formacie: `"Imię" "Waga" "Wzrost"`.
Przykład:
```
Jan 70 1,8
Beata 67 1,77
Donald 85 1,7
Jarosław 100 1,92
```
Za nadwagę uznajemy sytuację, gdy czyjeś BMI jest powyżej 25. BMI liczymy zgodnie ze wzorem:

$$BMI = masa / wzrost^2$$

Wynikiem działania programu ma być wypisanie na ekran imion osób z nadwagą wraz z ich wskaźnikiem BMI.

Zad. 141:
Napisz funkcję `szyfruj(String plikWejściowy, String plikWyjściowy, String tajneSłowo, String klucz)`. Zadaniem funkcji będzie zamiana wszystkich wystąpień słów `tajneSłowo` na słowo `klucz` w `plikuWejściowym`. Wynik należy zapisać do `plikuWyjściowego`.

Zad. 142:
Napisz funkcję odwracającą plik tekstowy. Funkcja ma mieć dwa parametry: plik wejściowy i plik wyjściowy. Ma odwracać plik tekstowy znak po znaku. Dodatkowo, funkcja ma zliczać liczbę wystąpień spacji w pliku.
UWAGA: nie wolno korzystać z metody `reverse` klasy `StringBuffer`.

Zad. 143:
Napisz program, który odczytuje linijka po linijce plik tekstowy zawierający hasła (jedno hasło = jedna linijka). Program ma obliczać czas potrzebny do złamania tego hasła metodą *brute force* zakładając, że w ciągu 1s jesteśmy w stanie sprawdzić 1000 kombinacji. Twoim zadaniem jest obliczenie liczby możliwych kombinacji dla każdego hasła i sprawdzenie, jaki czas będzie potrzebny do złamania hasła. Zakładamy, że hasła mogą zawierać 62 możliwe znaki. Stąd złożoność hasła *„ala”* wynosi $62^3 = 238328$ (trzy znaki, 62 kombinacje każdy). Otrzymany wynik wyraź w „normalnych jednostkach”, czyli w przykładzie powinno to być: 6 min 37s. Załóż, że hasło możemy łamać maksymalnie 24h.

Zad. 144:
Zakładając, że w zmiennej typu `int` o nazwie `check` zapisana jest liczba zmiennoprzecinkowa, za pomocą jednej instrukcji przypisz do zmiennej typu prawda/fałsz informację, czy liczba ta jest dodatnia (prawda) czy ujemna (fałsz). Uwaga: nie korzystaj z konstrukcji `IF`.

Zad. 145:
Napisz funkcję zapisującą losowe liczby do pliku tekstowego. Funkcja ma przyjmować cztery parametry: nazwę pliku, zakres „od”, zakres „do”, liczbę liczb. Wywołanie:
```
zapiszLosowe(”plik.txt”,-5,10,100);
```
spowoduje zapisanie do *„plik.txt”* 100 liczb z zakresu od -5 włącznie do 10 włącznie. Jeśli użytkownik poda złe parametry (np. zakres „od” większy od zakresu „do” lub ujemną liczbę liczb) należy wyświetlić komunikat na ekran i nie zapisywać niczego do pliku. W każdym przypadku: udanego lub nieudanego zapisu – program ma kończyć się wypisując na ekran „Skończyłem”.

Zad. 146:
Napisz kompletny (ze wszystkimi importami, nazwami funkcji `main`, pakietów, itp.) program, który sprawdza, czy liczba podana przez użytkownika jest podzielna przez 13. Dodatkowo niech program sprawdza, czy liczba jest mniejsza czy większa od zera i wyświetla odpowiednie komunikaty na ekran (np. dla wejścia `-26`: *„Podana liczba to -26 (mniejsza od zera) i jest ona podzielna przez 13 bez reszty”*).

Zad. 147:
Napisz program, który odczytuje z klawiatury znak podany przez użytkownika. Dla liter „a”, „c”, „e”, „f”, „g”, „k”, „l” wyświetla napis „`..//| MAGICZNA LITERKA |\..`”. Dla wszystkich liczb:
```
-
"To
jest
liczba"
```
(każde słowo w oddzielnej linijce).
Uwaga: możesz w całym programie tylko **dwukrotnie** skorzystać z funkcji `printf/cout`.

Zad. 148:
Jaki będzie wynik działania programu?
```
int a=3, b=2, c=3;
cout << ++a;
cout << b++ + ++c;
boolean decyzja = (a<=b) ? true : false;
if (decyzja)   cout << "Dostanę 5.0 z kolokwium.");
else cout << "Nie zaliczę kolokwium.");
```

Zad. 149:
Który z podanych programów skompiluje się i wykona poprawnie? Pierwszy, drugi czy oba? Odpowiedź uzasadnij.

```
inx x = 2147483648;      // MAX_INT = 2147483647
cout << "Zmienna x ma wartość = " << x;

inx x = 2147483647;      // MAX_INT = 2147483647
x++;
cout << "Zmienna x ma wartość = " << x;
```

Zad. 150:
Niech użytkownik podaje z klawiatury 4 liczby zmiennoprzecinkowe. Następnie wypisz prostą statystykę tych liczb: średnią, wariancję i odchylenie standardowe. Wypisz wszystkie wyliczone wartości na ekran.

Zad. 151:
Napisz program, który wyznacza największą i najmniejszą wprowadzoną przez użytkownika liczbę zmiennoprzecinkową. Zakończenie wprowadzania liczb określa liczba 0 (nie brana pod uwagę). Zadbaj o sprawdzenie, czy użytkownik wprowadza liczby. Jeśli wprowadzi coś innego niż liczbę – zakończ program i wyświetl komunikat *„Błędne dane wejściowe”*.
Alternatywnie: zakończenie wprowadzania liczb określa ciąg *„koniec”*.

Zad. 152:
Napisz program, który czyta od użytkownika kolejne liczby całkowite (załóż, że użytkownik nie jest złośliwy i podaje tylko poprawne liczby) aż do podania liczby 0. Maksymalnie może podać 50 liczb.
Po zakończeniu podawania, wyświetl na ekranie:

a. Sumę parzystych liczb
b. **Liczbę** nieparzystych liczb
c. Średnią wszystkich liczb

Zad. 153:
Napisz program, który określi, jaki dzień tygodnia mamy dzisiaj. Zakładamy, że rok zawsze zaczyna się od poniedziałku, a użytkownik podaje nr dnia z roku kalendarzowego. Przykładowo dla liczby 157 będzie to środa, dla liczby 23 – wtorek, a dla liczby 7 – niedziela.

Zad. 154:
Napisz program losujący liczby całkowite z zakresu [-100…100] do dwóch tablic o rozmiarach 10x10 każda. Program ma również w trzeciej tablicy umieszczać odpowiednie maksymalne wartości z dwóch tablic (tj. jeśli w *tab1* w kolumnie *i,j* jest większa wartość niż w *tab2* w tej samej komórce, to do *tab3* przepisywana jest wartość z *tab1*).

Zad. 155:
Napisz program, która w podanym wyrazie zamienia wszystkie litery "a" na "A" oraz "b" na "B", a także podaje liczbę wszystkich dokonanych zmian. Ponadto, jeżeli liczba znaków wyrazu jest nieparzysta to środkowa litera również musi być zamieniona na dużą.
Przykład działania:
```
Podaj napis:   abcdCBA
Zmieniony napis:    ABcDCBA
Liczba zmian: 3
```
liczba znaków jest nieparzysta więc środkowe "d" → "D".

Zad. 156:
Napisz funkcję która otrzymuje w parametrze tablicę kwadratową dwuwymiarową `int`ów o nieznanych rozmiarach. Na wyjściu ma zwracać posortowaną rosnąco tablicę jednowymiarową złożoną z elementów pierwszej tablicy.

Zad. 157:
Napisz funkcję przyjmującą trzy parametry: `double min`, `double max`, `double[] tab_wej`. Funkcja ma zwracać tablicę `int`-ów będących indeksami elementów z zakresu `min` i `max` tablicy `tab_wej`.

Zad. 158:
Szkoła Zdolnych Programistów boryka się z niebagatelnym problemem dzwonków. Po pierwsze, godzina rozpoczęcia pierwszej lekcji może być zmienna i niekoniecznie musi być nią 8:00. Również długości przerw są zmienne, co akurat jest zupełnie zrozumiałe. Jedno tylko pozostaje niezmienne – długość lekcji, która zawsze trwa równo 45 minut. Polecenie: napisz program, który generuje informacje o porach, w których w SZP powinien rozlegać się dźwięk dzwonka.

Dane wejściowe: nieznana z góry liczba wierszy tekstu, zawierających kolejno: godzinę rozpoczęcia pierwszej lekcji w formacie HHMM (np. 0800) dowolną liczbę długości kolejnych przerw, wyrażonych w minutach (całkowite t: t >0 i t<=10080) zakończone symbolem 0.

Dane wyjściowe: jeden wiersz tekstu, a w nim lista rozdzielonych przecinkami pór uruchomienia dzwonka w formacie HH:MM.

Przykład:
Wejście podawane z klawiatury:
```
0800
15
15
15
0
```
Wyjście:
```
08:00,08:45,09:00,09:45,10:00,10:45,11:00,11:45
```

Zad. 159:
Napisz prosty kalkulator wyrażeń arytmetycznych. Dane podawane są na standardowe wejście (klawiatura). W pierwszym wierszu podana jest liczba *N* (1<=*N*<=20) zestawów danych. Dalej podawane są zestawy danych zgodnie z poniższym opisem:
W pierwszym i jedynym wierszu zestawu danych podane jest wyrażenie arytmetyczne. Wyrażenie to składa się na przemian z pojedynczych cyfr i znaków + bądź - . Długość wyrażenia jest nie mniejsza niż 3 znaki i nie większa niż 99 znaków.
Wyniki programu powinny być wypisywane na standardowe wyjście (ekran). W kolejnych wierszach należy podać odpowiedzi obliczone dla kolejnych zestawów danych. Wynikiem dla jednego zestawu jest wartość wyrażenia podanego na wejściu.
Przykład:
dane wejściowe:
```
2
1+2+3-4
0-5+3
```
wynik:
```
2
-2
```

Zad. 160:
Napisz funkcję, która zlicza występowanie znaku podanego w parametrze funkcji w pliku (nazwa pliku również podawana jako parametr). Przykładowo:

```
liczCzęstość('a',"plik.txt")
/* plik.txt:
Pchnąć w tę łódź jeża lub ośm skrzyń fig
*/
```

zwróci wartość: 1

Zad. 161:
Napisz program do generowania zestawów liczb, które na pewno padną w najbliższych losowaniach Dużego Lotka. Użytkownik ma podać liczbę zestawów, która go interesuje, a Twoim zadaniem jest wylosowanie tylu zestawów. Przypominam, że duży lotek losuje 6 liczb z zakresu [1…49]. Przykładowo, dla wejścia równego 4 program może wypisać:

```
Zestaw 1:      32   8    38   4    8    14
Zestaw 2:           24   23   19   49   25   11
Zestaw 3:           3    21   8    49   28   2
Zestaw 4:           20   6    45   11   44   38
```

Zad. 162:
W pewnym akwarium żyją rybki. Owe rybki co 2 miesiące rozmnażają się w taki sposób, że ich liczba ulega podwojeniu. Niestety ilość tlenu w akwarium jest ograniczona – jeśli liczba rybek przekroczy 100 – 90% z nich ginie (czyli na przykład, jeśli po rozmnożeniu miało być 150 rybek, to zostaje 15).

a. Napisz program, który zakładając, że początkowo mam 10 rybek, liczy, ile ich będę miał po 24 miesiącach hodowli?
b. W którym z miesięcy (od 1 do 24) będzie najwięcej rybek w akwarium? Zakoduj odpowiednie funkcjonalności w programie, aby ten wyświetlił numer miesiąca oraz liczbę rybek w tym miesiącu.

Zad. 163:
Napisz program do przybliżonego wyliczania minimum i maksimum funkcji kwadratowej. Użytkownik podaje z klawiatury trzy współczynniki: *A, B, C*. Twoim zadaniem jest wypisanie wartości minimalnej i maksymalnej funkcji $Ax^2+Bx+C=0$ w przedziale [-1000 … 1000] z krokiem co 0,1. Przykładowo, dla $A=1$, $B=2$, $C=1$ program wypisze $max=1002001$, $min=0$, a dla $A=-3$, $B=2$, $C=3$ $max=3{,}33$, $min=-3001997$.

Zad. 164:
Napisz funkcję przyjmującą w parametrze nazwę pliku tekstowego. Funkcja ma **zwracać** w wyniku liczbę samogłosek wewnątrz tego pliku.
Przykładowo:

```
System.out.println(liczSamogłoski("plik.txt"));
/* plik.txt:
Ala ma kota, czipsy i krakersy.
Ola ma krzywe zęby.*/
```

Wypisze na ekran wartość: 17

Zad. 165:

Napisz funkcję przyjmującą w parametrze nazwę pliku tekstowego zawierającego ciągi wyrażeń arytmetycznych. W każdej linijce znajduje się wyrażenie typu:

```
(a+b)*((c+d)*e)
```

Twoim zadaniem jest napisanie funkcji, która zwróci wartość **true,** jeśli **wszystkie** wyrażenia w pliku są zbudowane poprawnie, tj. mają poprawnie umieszczone nawiasy. Liczba nawiasów otwierających musi być zgodna z liczbą nawiasów zamykających, co więcej – nie można zamknąć nawiasu nieotwartego.

Przykładowo:

```
System.out.println(czyPoprawneNawiasy("plik.txt"));
/* plik.txt:   */
   (a+b)*((c+d)*e)
   (((a)))
   ((a+b+c+d)))
   ((a*c)*d)
```

Zwróci wartość **false**, bo w trzeciej linijce jest niepoprawne wyrażenie. Jeśli funkcja znajdzie niepoprawne wyrażenie, ma nie sprawdzać do końca pliku, ale od razu dać wynik.

Zad. 166:

Często się zdarza, że pisząc stronę internetową piszemy tagi htmlowe w postaci dużych, a czasami małych liter. Powoduje to, że kod strony wygląda nieestetycznie. Twoim zadanie jest napisanie funkcji, która przerobi wszystkie tagi htmlowe na duże litery, tzn. wszystkie litery pomiędzy znakami "<" a ">" zamieni na duże litery.

Input

W pliku wejściowym znajduje się fragment kodu htmlowego zawierającego znaczniki HTML pisane w postaci dużych i małych liter.

Output

Na wyjściu znajduje się kod HTML z wejścia, tyle tylko, że wszystkie tagi htmlowe składają się z dużych liter.

Prototyp

```
htmlTagsCaps(String inputFileName, String outputFileName, boolean verbose);
```

`inputFileName` – ścieżka do pliku wejściowego

`outputFileName` – ścieżka do pliku wyjściowego

`verbose` – jeśli parametr ma wartość `true`, to oprócz wpisania znaczników do pliku należy wyświetlić zawartość pliku wyjściowego na konsolę

Zad. 167:

Napisz program, który wypełni losowo tablicę 10 x 10 wartościami typu `double` z zakresu [-5;5], a następnie:

a. Wyznaczy wiersz, w którym suma liczb jest maksymalna,
b. Wyznaczy kolumnę, w której suma jest minimalna,
c. Wyznaczy iloczyn wartości na przekątnych,
d. Wyznaczy **dwie** największe liczby z tablicy

Program powinien wypisywać wyznaczone wartości na ekran.

Zad. 168:

Napisz funkcję `String timeDifference(String one, String two)` wyznaczającą różnicę czasu pomiędzy dwoma lokalizacjami o różnych długościach geograficznych. W zależności od długości geograficznej, można obliczać matematycznie czas słoneczny miejscowy. Do obliczeń czasu:

1. 15° to jedna godzina (360°/24h = 15°)
2. 1° to 4 minuty.

Funkcja ma działać w następujący sposób:

```
System.out.println(timeDifference(„000E”,”090E”); //wyświetli napis „6h 00 min”
System.out.println(timeDifference(„090E”,”090W”); //wyświetli napis „12h 00 min”
System.out.println(timeDifference(„001E”,”012E”); //wyświetli napis „0h 44 min”
```

T1:

213	174	251
217	198	155
134	227	198

T2:

134	217	213
227	198	174
198	155	251

Przyjmij, że napis ma 4 znaki, trzy pierwsze to liczba, trzeci to określenie półkuli.
Po jej napisaniu, użyj jej do odczytu pliku tekstowego, w którym w każdej linijce rozdzielone znakiem tabulacji są dwie lokalizacje geograficzne (przyjmij, że jest ich dokładnie 100). Po odczytaniu i policzeniu różnicy w każdym przypadku, wyświetl na ekran sumaryczną różnicę czasu pomiędzy wszystkimi lokalizacjami w godzinach i minutach.

Zad. 169:

Napisać program, który generuje zmiennoprzecinkowe liczby pseudolosowe z zakresu od [-31; 57) tak długo, aż zostanie wylosowana liczba z zakresu [0;1]. W wyniku wykonania programu (bez wykorzystania tablic), powinny na konsoli zostać wyświetlone:

a) największa i najmniejsza z wylosowanych liczb oraz iloczyn liczb ujemnych.
b) ciąg całkowitych liczb pseudolosowych z zakresu [13;52] o długości równej największej wartości bezwzględnej zaokrąglonej do dołu liczby wylosowanej w pierwszym kroku.

Uwaga 1: [oraz] – oznacza **włącznie (zamknięty przedział)**; (oraz) – przedział otwarty.
Uwaga 2: największa wartość bezwzględna nie zawsze jest równa największej liczbie.

ZADANIA C++: wskaźniki i dynamiczny przydział pamięci

Zad. 1:

Napisać program wypełniający kwadratową tablicę kolejnymi liczbami naturalnymi „po spirali”, przykładowo dla wymiaru 5 winniśmy uzyskać:

```
 1 -  2 -  3 -  4 -  5
                    |
16 - 17 - 18 - 19    6
|                 |  |
15   24 - 25   20    7
|    |         |  |
14   23 - 22 - 21    8
|                    |
13 - 12 - 11 - 10 -  9
```

Zad. 2:

Opracować funkcje, które obliczają sumę dwóch liczb rzeczywistych `x` i `y` typu `double`. Prototypy funkcji mają następującą postać:

a) `double sum1 (double x, double y);` // suma zwracana przez wartość funkcji
b) `void sum2(double x, double y, double * z);` // suma zwracana przez wskaźnik
c) `void sum3(double x, double y, double& z);` // suma zwracana przez referencję

Zad. 3:

Zdefiniować wskaźnik na zmienną typu `int`. Zainicjować wskaźnik adresem zmiennej całkowitej `int a = 5`. Wyprowadzić na ekran zawartość zmiennej za pomocą wskaźnika oraz za pomocą identyfikatora zmiennej (wartości powinny być identyczne).

Zad. 4:

Opracować funkcje, które obliczają `max(x,y)` zgodnie z następującymi prototypami:

a) `double max1(double, double);` // max zwracane przez wartość funkcji
b) `void max2(double, double, double *);` // max zwracane przez wskaźnik
c) `void max3(double, double, double&);` // max zwracane przez referencję

Zad. 5:

Opracować funkcje, które zamieniają wartości zmiennych `x` i `y` typu `double` (po wyjściu z funkcji `x` zawiera wartość `y` i odwrotnie). Dane są przekazywane do funkcji o następujących prototypach:

a) `void zamien1(double*, double*);` // dwa wskaźniki
b) `void zamien2(double&, double&);` // dwie referencje

Zad. 6:

Napisz funkcje, które zamienią temperaturę w stopniach Fahrenheita na temperaturę w stopniach Celsjusza *c = (f - 32) * 5 / 9*. Prototypy funkcji mają następującą postać:

a) `double far2cel1 (double f);` // zwracana przez wartość funkcji
b) `void far2cel2 (double f, double * c);` // zwracana przez wskaźnik
c) `void far2cel3 (double f, double& c);` // zwracana przez referencję

Zad. 7:
Napisz funkcję, która przyjmuje liczbę całkowitą *a* z przedziału 1-30. Tę liczbę przekazuje do funkcji, a funkcja działa następująco: jeżeli wartość liczby będzie podzielna przez 3 to wynik powinien wynosić 7, w przeciwnym przypadku wynik powinien zostać obliczony ze wzoru: 5**a* – 7. Jeśli liczba *a* jest spoza przedziału, funkcja zwraca 0. Prototypy funkcji mają następującą postać:
a) `int funkcja1 (int a);` // zwracana przez wartość funkcji
b) `void funkcja2 (int a, int * f);` // zwracana przez wskaźnik
c) `void funkcja3 (int a, int & f);` // zwracana przez referencję

Zad. 8:
Napisać funkcję przyjmującą trzy parametry: `op`, `a`, `b`. Funkcja ma zwracać następujące wyniki w zależności od wartości parametru `op`:
op = 1: a+b
op = 2: a-b
op = 3: a*b
op = 4: a/b
Prototypy funkcji mają następującą postać:
a) `int funkcja1 (int op, int a, int b);` // zwracana przez wartość funkcji
b) `void funkcja2 (int op, int a, int b, int * f);` // zwracana przez wskaźnik
c) `void funkcja3 (int op, int a, int b, int & f);` // zwracana przez referencję

Zad. 9:
Napisać własną funkcję dodawania trzech liczb `add3` zwracającą dla parametrów `a,b,c` wartość a+b+c. Prototypy funkcji mają następującą postać:
a) `int add31 (int a, int b, int c);` // zwracana przez wartość funkcji
b) `void add32 (int a, int b, int c, int * suma);` // zwracana przez wskaźnik
c) `void add33 (int a, int b, int c, int & suma);` // zwracana przez referencję

Zad. 10:
Napisać funkcję, która przyjmuje jako parametr liczbę rzeczywistą `x` oraz całkowitą `n`, a na wyjściu zwróci funkcję `x^n`. Prototypy funkcji mają następującą postać:
a) `double potega1 (double x, int n);` // zwracana przez wartość funkcji
b) `void potega2 (double x, int n, double * potega);` // zwracana przez wskaźnik
c) `void potega3 (double x, int n, double & potega);` // zwracana przez referencję

Zad. 11:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 25. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Po podaniu wszystkich liczb:
- liczby na indeksach parzystych zamienia na dodatnie,
- liczby na indeksach nieparzystych zamienia na ujemne,

Następnie należy wydrukować wszystkie liczby na ekranie. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 12:
Napisać funkcje, które przyjmują jako parametr długość dwóch boków prostokąta i liczą jego pole. Prototypy funkcji mają następującą postać:
a) `int pole1 (int x, int y);` // zwracana przez wartość funkcji
b) `void pole2 (int x, int y, int * pole);` // zwracana przez wskaźnik
c) `void pole3 (int x, int y, int & pole);` // zwracana przez referencję

Zad. 13:
Napisać funkcje, które przyjmują jako parametr dwie liczby całkowite, a na wyjściu zwrócą resztę z dzielenia pierwszej przez drugą. Prototypy funkcji mają następującą postać:
a) `int reszta1 (int m, int n);` // zwracana przez wartość funkcji
b) `void reszta2 (int m, int n, int * reszta);` // zwracana przez wskaźnik
c) `void reszta3 (int m, int n, int & reszta);` // zwracana przez referencję

Zad. 14:
Napisać funkcje, które przyjmują jako podstawę *a* i wysokość *h* trójkąta, a na wyjściu zwrócą jego pole. Prototypy funkcji mają następującą postać:
a) `int pole1 (int a, int h);` // zwracana przez wartość funkcji
b) `void pole2 (int a, int h, int * pole);` // zwracana przez wskaźnik
c) `void pole3 (int a, int h, int & pole);` // zwracana przez referencję

Zad. 15:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 21. Na tej podstawie tworzy tablicę tej wielkości i wypełnia ją kolejnymi potęgami dwójki, zaczynając od 2^0. Wydrukuj zawartość tablicy na ekran. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 16:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 22. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Po podaniu wszystkich liczb do tablicy liczby ujemne należy zastąpić zerami, a następnie wydrukować wszystkie liczby na ekranie. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 17:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 25. Na tej podstawie tworzy tablicę tej wielkości i wypełnia ją wartościami od 7 do `n`+6 (`n` to wielkość tablicy). Wydrukuj zawartość tablicy na ekran. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 18:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 24. Na tej podstawie tworzy tablicę tej wielkości i wypełnia ją wartościami od 0 do `n`-1 (`n` to wielkość tablicy). Wydrukuj zawartość tablicy na ekran. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 19:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 26. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Po podaniu wszystkich liczb do liczb parzystych należy dodać 100, a następnie wydrukować wszystkie liczby na ekranie. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 20:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 27. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Na koniec te wartości sumuje, a wynik wyświetla na ekranie. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 21:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 28. Na tej podstawie tworzy dwie tablice tej wielkości (lub jedną tablicę dwuwymiarową). Do tablicy pierwszej wczytuje kolejno liczby podane przez użytkownika. Następnie kopiuje zawartość tablicy nr 1 do tablicy nr 2 w odwróconej kolejności, po czym wyświetla zawartość tablicy numer 2. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 22:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 29. Na tej podstawie tworzy dwie tablice tej wielkości (lub jedną tablicę dwuwymiarową). Do pierwszej wpisuje liczby podane przez użytkownika. Do drugiej wpisuje kolejno resztę z dzielenia tych liczb przez 5. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 23:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 30. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Każdą liczbę podnieś do kwadratu, a następnie całą tablicę wyświetl na ekranie. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 24:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 31. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Po podaniu wszystkich liczb należy wyświetlić tylko liczby nieparzyste oraz ich indeksy. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 25:
Napisz program, który pyta użytkownika o liczbę naturalną, maksymalnie 32. Na tej podstawie tworzy tablicę tej wielkości. Do tablicy wczytuje kolejno liczby podane przez użytkownika. Wydrukuj na ekranie te spośród liczb, których indeks jest mniejszy niż liczba tam zapisana, oraz indeksy tych liczb. Pamięć dla tablicy ma być przydzielona dynamicznie.

Zad. 26:
Co należy zrobić zawsze po zadeklarowaniu wskaźnika?
A) Nic szczególnego
B) Zainicjować go adresem 0
C) Zastanowić się nad użyciem operatora `delete`

Zad. 27:
Napisz program definiujący zmienną typu `int` oraz wskaźnik do zmiennej typu `int`. Program powinien wczytać z klawiatury wartość i podstawić ją do zmiennej stosując wskaźnik i operator adresu (nie stosuj operatora `new`).

Zad. 28:
Co należy zawsze zrobić po użyciu operatora `new`?
A) Nic szczególnego
B) Zaplanować późniejsze użycie operatora `delete`
C) Ustawić wskaźnik na 0

Zad. 29:
a) Zadeklaruj strukturę składającą się z trzech pól: `double`, `int` oraz `char`.
b) Napisz funkcję przyjmującą jeden parametr (wskaźnik na utworzoną przez siebie strukturę) i pozwalającą na ustawienie wszystkich trzech pól wartościami przeczytanymi od użytkownika z klawiatury.
c) Napisz funkcję wypisującą wartości elementów struktury (przykładowe wyjście „2.5 10 A").
a) W funkcji `main()`:
    a. Zainicjuj generator liczb pseudolosowych
    b. Pobierz od użytkownika wartość typu int
    c. Zadeklaruj **dynamicznie alokowaną** tablicę o rozmiarze pobranym wcześniej od użytkownika w punkcie b.
    d. Wypełnij wszystkie pola wszystkich struktur w tablicy wartościami losowymi.
    e. Wypisz całą tablicę struktur na ekran. W tym celu użyj funkcji napisanej w punkcie 3.

Zad. 30:
Napisz funkcję przekształcającą jednowymiarową tablicę liczb typu `double` w listę jednokierunkową taką, że liczby zapisane w niej są uporządkowane rosnąco. Przykładowo:

```
struct elem
{
    double dane;
    elem *next;
};

double tablica[] = {1.0, -3.5, 4.3, 5.0};
elem * lista = tabToList(tablica); //spowoduje utworzenie uporządkowanej listy jednokierunkowej z tablicy tablica i zwrócenie wskaźnika na tą strukturę oraz przypisanie go do zmiennej lista
```

Zad. 31:
Napisz funkcję `int checkForDoubles(string file1, string file2)`.
Funkcja w parametrze otrzymuje nazwy dwóch plików (typ `string`), w których zapisane są liczby całkowite (jedna liczba w jednej linijce, nieokreślona liczba liczb). Zadaniem funkcji jest zliczyć jak dużo liczb znajduje się w pierwszym pliku i **NIE** znajduje w pliku drugim. Przykładowo:

```
file1:                                  file2:
1                                       1
2                                       7
3                                       5
4                                       6
                                        3
cout << checkForDoubles(file1, file2) << endl;
// wypisze na ekran 2, bo liczby "2" oraz "4" są w pierwszym pliku, a nie ma ich w drugim
```

Zad. 32:
Napisz program, który:
- Stworzy tablicę (macierz) o rozmiarze podanym przez użytkownika, z zakresu od 2 do 10, a następnie wypełni ją losowymi liczbami całkowitymi z przedziału od -10 do 10. Tablice proszę tworzyć na stercie (dbając o prawidłową alokację i zwalnianie pamięci).
- Dla każdej kolumny wyznaczy wartość minimalną.
- Wyznaczy największą wartość na obu przekątnych macierzy (po jednej wartości dla każdej z dwóch przekątnych).

Program na wyświetlać tablicę wypełnioną liczbami, tablicę z minimami oraz maksymami – zatem w programie ostatecznie tworzone są 3 tablice. Program powinien być odporny na próby wpisania jakichkolwiek błędnych znaków (walidacja wejścia).

Zad. 33:
Jak wyświetlić adres zmiennej łańcuchowej nazwa?
A) `cout << *nazwa << endl;`
B) `cout << &nazwa << endl;`
C) `cout << nazwa << endl;`
D) `cout << %nazwa << endl;`

Zad. 34:
Co wyświetli poniższy kod?
```
int main()
{
    int a(4);
    int* p(0);
    p = &a;

    (*p) += 2;

    a += 3;

    cout << a << " " << *p << endl;

    return 0;
}
```

A) 7 2
B) 9 0
C) 4 2
D) 9 9
E) 4 0

Zad. 35:
Napisz program definiujący zmienną typu `TData` (struktura) oraz wskaźnik do zmiennej typu `TData`. Program powinien wczytać z klawiatury wartości pól zmiennej. Zastosuj wskaźnik i operator adresu (nie stosuj operatora `new`).

Zad. 36:

Spójrz na poniższy fragment programu:

```
double const pi(3.14);
double *ptr(0);
ptr = π
```

Jeśli przyjmiemy, że zmienna `pi` znajduje się pod adresem 54673, a sam wskaźnik pod adresem 92415, to co wyświetli poniższy kod?

```
cout << *ptr << endl;
```

A) 54673
B) 92415
C) 3.14

Zad. 37:

Używając dynamicznego przydziału pamięci, stwórz dwuwymiarową tablicę, której rozmiar wierszy będzie zależy od ilości liczb wprowadzonych przez użytkownika. Następnie wyświetl tablicę wylicz sumę liczb w pierwszej kolumnie tablicy. Przykład:

```
Podaj liczbę wierszy: 3
Podaj liczby do zapisania w 1 wierszu tablicy: 4 -10 2
Podaj liczby do zapisania w 2 wierszu tablicy: 2 8 1 4 1
Podaj liczby do zapisania w 3 wierszu tablicy: 1
Tablica:
4 -10  2
2   8  1  4  1
1
Suma liczb w pierwszej kolumnie wynosi 7.
```

Zad. 38:

Korzystając z następującej struktury (symbolizującej element stosu):

```
struct elem    {
     int dane;
     elem * nast;
}
```

Zaimplementuj podstawowe operacje stosowe:

- Położenie elementu na wierzchołku stosu

```
void push (elem* &stos, int x)
```

- Położenie ostatnio odłożonego elementu i zwrócenie go jako wartości funkcji

```
int pop(elem* &stos)
```

- Zwrócenie elementu znajdującego się na wierzchołku stosu bez jego usuwania

```
int topEl(elem *stos)
```

- Sprawdzenie, czy stos jest pusty

```
bool isEmpty(elem *stos)
```

Zad. 39:

Zadeklaruj tablicę o rozmiarze 100. Wypełnij tablicę zgodnie z regułami Ciągu Fibbonacciego (pierwszy i drugi element = 1, każdy następny to suma dwóch poprzednich: 1,1,2,3,5,8, itd.). Użyj dynamicznego przydziału pamięci oraz operacji na wskaźnikach.

Zad. 40:
Stwórz trzywymiarową tablicę dynamiczną.

Zad. 41:
1. Poczytaj o listach jednokierunkowych, dwukierunkowych, cyklicznych, itp.
2. Jak deklarujemy i jak je używamy w C++?
3. Zaimplementuj użycie listy jednokierunkowej liczb całkowitych:
- Dodawania do listy.
- Usuwania elementu (o danej wartości).
- Wyszukiwania, czy dany element znajduje się w liście.
- Zliczania liczby elementów.

Zad. 42:
1. Zadeklaruj wskaźnik na typ całkowity. Spróbuj go przypisać tak, aby wskazywał na typ `double`.
2. Zadeklaruj poprawny wskaźnik, sprawdź czy wyłuskuje wskazywaną wartość. Sprawdź jakie operatory działają na wskaźnikach (dodaj wskaźniki, odejmij, dodaj/odejmij stałą wartość liczbową, pomnóż, itp.)
3. Czy jest możliwe zadeklarowanie takiego uniwersalnego wskaźnika na cokolwiek?

Zad. 43:
Napisz program do mnożenia dwóch macierzy o dowolnym rozmiarze. Sprawdzaj, czy mnożenie jest możliwe! Użyj dynamicznego przydziału pamięci oraz operacji na wskaźnikach.

Zad. 44:
Wypisz całą tablicę ASCII na ekran (każdy znak w nowej linijce opatrzony „numerkiem”) i zapisz ją do tablicy o nazwie tab_ASCII w programie. Użyj dynamicznego przydziału pamięci oraz operacji na wskaźnikach.

Zad. 45:
Zrealizować grę w statki na planszy generowanej dynamicznie (do 10 x 10). Raz strzela użytkownik, raz komputer.
Dozwolone okręty: 4 x jednomasztowiec.
Zaprogramować sytuacje:
„Trafiony – zatopiony”,
„Pudło”,
*„W plansze id****”.*

Zad. 46:
Ułożyć program obliczający sumę *S* elementów o indeksach parzystych tabeli dwuwymiarowej. Dla tabeli T[4][5] będzie to suma *S* = T[0][0] + T[0][2] + T[0][4] + T[2][0] + T[2][2] + T[2][4]. Wielkość tabeli jest pobierana od użytkownika i wypełniania losowymi liczbami całkowitymi.

Zad. 47:
Napisz program, który tworzy tabelę na podstawie rozmiaru podanego przez użytkownika (co najmniej 30) i wypełnia ją losowymi wartościami z przedziału 0-9. Napisz funkcję, która w parametrach przyjmie tabelę oraz jej rozmiar i będzie zwracać cyfrę, która wystąpiła w tabeli największą liczbę razy. Użyj dynamicznego przydziału pamięci.

Zad. 48:

Napisz program, który tworzy tabelę dwuwymiarową liczb całkowitych, o rozmiarach pobranych od użytkownika. Tabelę wypełnij losowymi liczbami z przedziału [-100, 100]. Następnie w zależności od wyboru użytkownika oblicz sumę wybranego wiersza lub kolumny (użytkownik podaje *'w'* – wiersz lub *'k'* – kolumna oraz numer). Obliczanie sum mają realizować funkcje o następujących prototypach:

```
sumaWiersza(int numer);
```

```
sumaKolumny(int numer, int *suma);
```

Wynik podaj na standardowym wyjściu. Zadbaj o idiotoodporność programu.

Zad. 49:

Napisz program, który tworzy dwie kwadratowe tabele dwuwymiarowe o rozmiarze pobranym od użytkownika. Pierwszą tabelę wypełnij wartościami losowymi z przedziału [129,256]. Przekopiuj pierwszą tabelę do drugiej obracając ją o 90 stopni. Przykład dla tablicy 3x3:

Zad. 50:

a) Zadeklaruj strukturę postaci: imię, nazwisko, wzrost, waga (wzrost jest liczbą całkowitą i jest podany w centymetrach; waga jest liczbą rzeczywistą i podana jest w kilogramach).
b) Napisz funkcję przyjmującą jeden parametr (wskaźnik na utworzoną przez siebie strukturę) i pozwalającą na ustawienie wszystkich trzech pól wartościami przeczytanymi od użytkownika z klawiatury.
c) Napisz funkcję wypisującą wartości elementów struktury (przykładowe wyjście *„Jan Kowal 175cm 73,5kg”*).
d) W funkcji `main()`:
    a. Pobierz od użytkownika wartość typu `int`.
    b. Zadeklaruj **dynamicznie alokowaną** tablicę o rozmiarze pobranym wcześniej od użytkownika w punkcie a.
    c. Pobierz od użytkownika dane osób do wypełnienia tej tablicy.
    d. Wypisz całą tablicę struktur na ekran. W tym celu użyj funkcji napisanej w punkcie 3.

Zad. 51:

a) Zadeklaruj strukturę `Student` postaci: imię, nazwisko, ocena.
b) Napisz funkcję przyjmującą jeden parametr (wskaźnik na utworzoną przez siebie strukturę) i pozwalającą na ustawienie wszystkich trzech pól wartościami przeczytanymi od użytkownika z klawiatury.
c) Napisz funkcję wypisującą wartości elementów struktury (przykładowe wyjście *„Zbigniew Kotnicki 3.5”*)
d) W funkcji `main()`:
    a. Pobierz od użytkownika wartość typu `int`.
    b. Zadeklaruj **dynamicznie alokowaną** tablicę o rozmiarze pobranym wcześniej od użytkownika w punkcie a.
    c. Pobierz od użytkownika dane osób do wypełnienia tej tablicy.
    d. Wypisz całą tablicę struktur na ekran. W tym celu użyj funkcji napisanej w punkcie 3.
    e. Wypisz podsumowanie: średnią ocen.
    f. Wypisz dane tych studentów, których oceny były wyższe niż średnia.

Zad. 52:
Napisz program, który tworzy tabelę dwuwymiarową liczb całkowitych, o rozmiarach pobranych od użytkownika. Tabelę wypełnij losowymi liczbami z przedziału [1900,1999]. Następnie na standardowe wyjście wypisz liczbę maksymalną z całej tabeli, wraz z indeksami, gdzie się znajduje.

Zad. 53:
Napisz program, który tworzy trzy kwadratowe tabele dwuwymiarowe o rozmiarze 10x10. Dwie pierwsze tabele wypełnij losowymi wartościami z przedziału [-400,400]. Do trzeciej tabeli wpisz sumę dwóch pierwszych *T3[a][b] = T1[a][b] + T2[a][b]*. Wszystkie tabele po kolei zapisz do pliku.

Zad. 54:
Pobierz od użytkownika dodatnią liczbę całkowitą. Będzie to liczba wierszy tabeli dwuwymiarowej. Dla każdego wiersza pobierz od użytkownika dodatnią liczbę całkowitą, która będzie określać długość kolejnych wierszy. W dowolnym momencie tabelę zainicjuj wartościami losowymi z przedziału [1,1024]. Na koniec wyświetl całą tabelę na ekranie.

Przykład:

```
Podaj liczbę: 4
Podaj długość wiersza #1: 5
Podaj długość wiersza #2: 2
Podaj długość wiersza #3: 3
Podaj długość wiersza #4: 6
Tabela:
547  446  1000 681  532
639  47
375  2    233
906  1015 39   9    409  142
```

Zad. 55:
Zaimplementuj listę jednokierunkową liczb rzeczywistych. Użyj w tym celu struktury:

```
struct elem
{
    double dane;
    elem *next;
};
```

Zaimplementuj następujące funkcjonalności: dodawanie do listy; usuwanie elementu (o danej wartości); wyszukiwanie, czy dany element znajduje się na liście; zliczanie liczby elementów.

Zad. 56:
Napisz program, który transponuje podaną macierz.

http://pl.wikipedia.org/wiki/Macierz_transponowana

Zad. 57:

Zaimplementuj listę dwukierunkową liczb rzeczywistych. Użyj w tym celu struktury:

```
struct elem
{
    elem *prev;
    double dane;
    elem *next;
};
```

Zaimplementuj następujące funkcjonalności: dodawanie do listy; usuwanie elementu (o danej wartości); wyszukiwanie, czy dany element znajduje się na liście; zliczanie liczby elementów.

Zad. 58:

Zaimplementuj listę cykliczną liczb rzeczywistych. Użyj w tym celu struktury:

```
struct elem
{
    double dane;
    elem *next;
};
```

Zaimplementuj następujące funkcjonalności: dodawanie do listy; usuwanie elementu (o danej wartości); wyszukiwanie, czy dany element znajduje się na liście; zliczanie liczby elementów.

Zad. 59:

a) Zadeklaruj strukturę `Data` składającą się z trzech pól całkowitoliczbowych `Dzien`, `Miesiac`, `Rok`.
b) Napisz funkcję przyjmującą jeden parametr (wskaźnik na utworzoną przez siebie strukturę) i pozwalającą na ustawienie wszystkich trzech pól wartościami przeczytanymi od użytkownika z klawiatury.
c) Napisz funkcję wypisującą wartości elementów struktury (przykładowe wyjście: *„24-12-2013”*)
d) W funkcji `main()`:
    a. Zainicjuj generator liczb pseudolosowych.
    b. Pobierz od użytkownika wartość typu `int`.
    c. Zadeklaruj **dynamicznie alokowaną** tablicę o rozmiarze pobranym wcześniej od użytkownika w punkcie b.
    d. Wypełnij wszystkie pola wszystkich struktur w tablicy wartościami losowymi, pamiętając o odpowiednich zakresach (rok może być maksymalnie obecny).
    e. Wypisz całą tablicę struktur na ekran. W tym celu użyj funkcji napisanej w punkcie 3.
    f. Wypisz podsumowanie – najstarszą i najbardziej aktualną datę.

Zad. 60:

Napisz funkcję mnożenia macierzy przez skalar, która przyjmuje dwa argumenty: macierz i skalar. Wynik musi być dostępny w programie po wyjściu z funkcji.

http://pl.wikipedia.org/wiki/Mnożenie_macierzy#Mnożenie_przez_skalar

Zad. 61:

a) Zadeklaruj strukturę `Data` składającą się z trzech pól całkowitoliczbowych `Dzien`, `Miesiac`, `Rok`.
b) Napisz funkcję przyjmującą jeden parametr (wskaźnik na utworzoną przez siebie strukturę) i pozwalającą na ustawienie wszystkich trzech pól wartościami przeczytanymi od użytkownika z klawiatury.
c) Napisz funkcję wypisującą wartości elementów struktury (przykładowe wyjście: *„24-12-2013”*).
d) Napisz funkcję, która oblicza, ile dni dzieli te dwie daty.
e) W funkcji `main()` pobierz od użytkownika dwie daty (wykorzystaj funkcję z pkt 2), wywołaj funkcję z pkt 4 i wyświetl wynik. Nie pozwól użytkownikowi wpisać niepoprawnych dat (rok może być maksymalnie obecny).

Zad. 62:

Napisz dwie funkcje: dodawania macierzy oraz odejmowania macierzy. Sprawdź poprawność danych wejściowych. Jeśli macierze nie są tego samego rozmiaru, funkcja zwraca 0. Wynik musi być dostępny w programie po wyjściu z funkcji.

http://pl.wikipedia.org/wiki/Dodawanie_macierzy

Zad. 63:

Napisz funkcję generującą macierz jednostkową dowolnego rozmiaru (np. podanego przez użytkownika). Sprawdź poprawność danych wejściowych. Jeśli macierz nie jest kwadratowa, funkcja zwraca 0. Macierz musi być dostępna w programie po wyjściu z funkcji.

http://pl.wikipedia.org/wiki/Macierz_jednostkowa

Zad. 64:

Zaimplementuj funkcję, która będzie rozwiązywać układ dwóch równań liniowych z dwiema niewiadomymi. Rozwiązanie powinno zostać zwrócone przez wskaźniki lub referencje. W funkcji `main()` użytkownik podaje 6 współczynników (*a, b, c, d, e, f*), następnie jest wywoływana funkcja rozwiązująca układ równań, a po policzeniu wyniku w funkcji `main()` jest on wyświetlany.

$a \cdot x + b \cdot y = c$

$d \cdot x + e \cdot y = f$

Zad. 65:

Zaimplementuj funkcję, która będzie rozwiązywała równanie kwadratowe. Funkcja zwraca liczbę rozwiązań oraz te rozwiązania, jeśli istnieją. W funkcji `main()` użytkownik podaje 3 współczynniki (*a, b, c*), następnie jest wywoływana funkcja rozwiązująca równanie kwadratowe, a po policzeniu wyniku w funkcji `main()` jest on wyświetlany.

$a \cdot x^2 + b \cdot x + c = 0$

Zad. 66:

Napisz program, który czyta z wejścia ciąg liczb całkowitych aż do napotkania zera, a następnie wypisuje je na wyjściu w odwrotnej kolejności. Skorzystaj z tablicy alokowanej dynamicznie lub listy jednokierunkowej (wskaźników).

Zad. 67:
Zadeklaruj zmienne `x` i `y` typu `int` oraz wskaźniki `p` i `q` na te zmienne. Przypisz adres `x` do `p` oraz `y` do `q`. Korzystając z przypisania bezpośrednio do zmiennej ustaw wartość `x` na 2. Korzystając ze wskaźnika ustaw wartość `y` na 8. Wyświetl następujące informacje:

1) adres `x` i wartość `x`,
2) wartość `p` i wartość `*p`
3) adres `y` i wartość `y`,
4) wartość `q` i wartość `*q`,
5) adres `p` (nie zawartość!),
6) adres `q` (nie zawartość!).

Zad. 68:
Zadeklaruj trzy zmienne `x`, `y`, `z` typu `int` oraz trzy wskaźniki `p`, `q`, `r` na te zmienne. Przypisz adres `x`, `y`, `z` odpowiednio do `p`, `q`, `r`. Korzystając z przypisania bezpośrednio do zmiennej ustaw wartość `x` na 67. Korzystając ze wskaźnika ustaw wartość `y` na 9 oraz `z` na 45.

1) Wyświetl wraz z opisem adres i wartość `x`.
2) Wyświetl wraz z opisem wartości `x`, `y`, `z`, `p`, `q`, `r`, `*p`, `*q`, `*r`.
3) Wyświetl napis: „Zamieniam wartości”.
4) Wykonaj kod zamiany: `z = x; x = y; y = z;`
5) Wyświetl wraz z opisem wartości `x`, `y`, `z`, `p`, `q`, `r`, `*p`, `*q`, `*r`.
6) Wyświetl napis: „Zamieniam wskaźniki”.
7) Wykonaj kod zamiany: `r = p; p = q; q = r;`
8) Wyświetl wraz z opisem wartości `x, y, z, p, q, r, *p, *q, *r.`

Zad. 69:
Zadeklaruj wskaźnik `a` typu całkowitoliczbowego. Następnie:

1) Użyj słowa kluczowego `new`, aby przydzielić pamięć i stworzyć dynamiczną tablicę o rozmiarze 5, na którą będzie wskazywać `a`.
2) Użyj pętli, aby wypełnić tablicę wartościami: 3, 7, 11, 15, 19.
3) Wyświetl adres przechowywany w `a`.
4) Użyj pętli, aby wyświetlić wartości z `a`, każda komórka w osobnym wierszu.
5) Usuń dynamicznie przydzieloną pamięć przy pomocy słowa kluczowego `delete`.

Zad. 70:
Napisz funkcję, która otrzymuje w parametrze tablicę kwadratową dwuwymiarową `int`ów o rozmiarach podawanych w parametrze do funkcji. Na wyjściu ma zwracać posortowaną rosnąco tablicę jednowymiarową złożoną z elementów pierwszej tablicy. Wykorzystaj dynamiczne tworzenie tablic za pomocą wskaźników.

Zad. 71:
Zadeklaruj wskaźnik `a` typu całkowitoliczbowego. Następnie:

1) Pobierz od użytkownika liczbę od 1 do 20.
2) Użyj słowa kluczowego `new`, aby przydzielić pamięć i stworzyć dynamiczną tablicę, na którą będzie wskazywać `a`. Rozmiarem tablicy jest wartość podana przez użytkownika w punkcie 1).
3) Wypełnij tablicę losowymi wartościami z przedziału [-50,50].
4) Wyświetl adres przechowywany w `a`.
5) Użyj pętli, aby wyświetlić wartości z `a`, każda komórka w osobnym wierszu.
6) Usuń dynamicznie przydzieloną pamięć przy pomocy słowa kluczowego `delete`.

Zad. 72:
Zaprojektuj funkcję szukającą w ciągu jednego podanego zbioru znaków. Powinna ona pasować do prototypu:

```
char *szukajZnaki (char const *zrodlo, char const *znaki);
```

Funkcja powinna szukać w `zrodlo` pierwszego znaku, który pasuje do dowolnego ze znaków w ciągu `znaki`. Funkcja powinna zwrócić wskaźnik do pierwszego znalezionego znaku. Jeżeli nie zostanie znaleziony żaden znak, zwracany jest wskaźnik `NULL`. Jeżeli którykolwiek z argumentów jest wartością `NULL` lub pustym ciągiem, zwracany jest wskaźnik `NULL`.

Aby to zilustrować, załóżmy, że `zrodlo` wskazuje na ABCDEFG. Jeżeli zmienna `znaki` wskazuje na XYZ, JURY lub QQQQ, funkcja zwraca `NULL`. Jeżeli zmienna `znaki` wskazuje na XRCQEF, funkcja powinna zwrócić wskaźnik do znaku C w `zrodlo`. Ciągi będące argumentami funkcji nie są nigdy modyfikowane.

W funkcji `main()` sprawdź działanie funkcji wywołując ją z przykładami podanymi powyżej oraz z którymś argumentem `NULL`.

Nie możesz korzystać z funkcji manipulujących ciągami. W funkcji nie można korzystać z indeksów. Celem jest przećwiczenie operowania na wskaźnikach.

Zad. 73:
Napisz program, który prosi użytkownika o liczbę całkowitą `n`. Następnie program w pętli prosi użytkownika o wpisywanie danych dot. `n` osób: imię, nazwisko, wiek (tablica struktur). Po wprowadzaniu danych ma pojawić się menu:

```
1. Wypisz wg nazwisk
2. Wypisz wg imion
3. Wypisz wg wieku
4. koniec
?
```

Program ma wyświetlać dane osób zgodnie z wyborem. Program kończy się po wciśnięciu 4.

Zad. 74:

Liczba pierwsza to taka liczba naturalna większa od 1, która dzieli się przez jeden i przez samą siebie. Sito Erastotenesa to efektywny algorytm obliczania liczb pierwszych. Pierwszym krokiem algorytmu jest zapisanie wszystkich liczb od dwa do jakiejś wartości granicznej. W drugiej części algorytmu wykreśla się liczby, które nie są pierwsze.

Rozpoczynając od początku listy, należy znaleźć pierwszą niewykreśloną liczbę, czyli w tym przypadku 2. Teraz trzeba wykreślić co drugą liczbę z listy, ponieważ wszystkie one dzielą się przez dwa. Następny krok rozpoczynamy od początku listy. Pierwszą niewykreśloną liczbą jest trzy, więc trzeba wykreślić co trzecią liczbę. Następna wartość z listy, cztery, jest już wykreślona, więc jest pomijana. Po zakończeniu całego procesu w tablicy pozostaną niewykreślone tylko liczby pierwsze.

Napisz program implementujący ten algorytm. Użytkownik podaje wartość graniczną. Na tej podstawie tworzona jest tablica dynamiczna typu `bool`. Wartość każdego z elementu tablicy wskazuje, czy został wyświetlony czy nie. Ustaw na początek wszystkie elementy tablicy na `TRUE`, a następnie liczby do wykreślenia ustawiaj na `FALSE`. Możesz użyć indeksów do pobierania wskaźników na początek i na koniec tablicy, ale do odwoływania się do elementów tablicy użyj wskaźników.

Zwróć uwagę, że wszystkie liczby parzyste oprócz liczby dwa nie są pierwsze. Jeżeli trochę pomyślisz, możesz zoptymalizować zajętość pamięci, przechowując w tablicy tylko liczby nieparzyste. Dzięki temu powinieneś być w stanie znaleźć (około) dwa razy więcej liczb pierwszych, korzystając z tablicy tej samej wielkości.

Zad. 75:

Macierz jednostkowa to taka macierz kwadratowa, która jest wypełniona zerami poza główną przekątną, na której znajdują się jedynki. Na przykład:

```
1 0 0                    // ← Jest to macierz jednostkowa 3x3.
0 1 0
0 0 1
```

Zaprojektuj funkcję o nazwie `czyMacierzJednostkowa`, która jako argumentu (jedynego) wymaga macierzy licz całkowitych o rozmiarach 10x10 i zwraca wartość logiczną, wskazującą, czy macierz jest macierzą jednostkową. (Faza 1)

Zmień funkcję tak, by mogła operować na macierzach dowolnych rozmiarów. Pierwszy argument powinien być wskaźnikiem na `int`, natomiast drugi – rozmiarem macierzy. (Faza 2)

Popraw funkcję tak, by sprawdzała poprawność danych wejściowych (czy macierz jest kwadratowa) i zwracała wynik niezależnie od rozmiaru macierzy. Funkcja pobiera tylko jeden argument – macierz. (Faza 3)

Oddaj kody funkcji z wszystkich trzech faz.

Zad. 76:

Napisz funkcję generującą tablicę trójkątną (`double`) wypełnioną wartościami losowymi typu `double` z zakresu (10…100) o wysokości podawanej jako parametr przez użytkownika (tablica trójkątna ma w pierwszym wierszu 1 element, w drugim 2, itd.). Napisz drugą funkcję zwracającą sumę najmniejszych liczb w każdym wierszu. Osadź je w programie i pokaż jak je wywoływać.

Zad. 77:

Napisz funkcję, która jako argumenty przyjmuje trzy tablice typu `char`. Dwie pierwsze są znane i wypełnione. Do trzeciej tablicy należy przydzielić pamięć dynamicznie na podstawie dwóch pierwszych tablic tak, aby możliwe było dokonanie następującej operacji: najpierw przepisujemy zawartość pierwszej tablicy, następnie wstawiamy spację, potem przepisujemy zawartość drugiej tablicy, na końcu dodajemy kropkę: `tabChar3 ← tabChar1 + ' ' + tabChar2 + '.'`

Na koniec, w funkcji `main()` wyświetl zawartość trzeciej tablicy.

Np.:

```
tabChar1[] = {'A', 'l', 'a', ' ', 'm', 'a'};
tabChar2[] = {'k', 'o', 't', 'a'};
// po wywołaniu funkcji wyświetli zawartość tabChar3:
Ala ma kota.
```

Przetestuj działanie programu na powyższym przykładzie.

Zad. 78:

Zaimplementuj funkcję `dzienRoku(dzien, miesiac, rok)`, która pobiera trzy argumenty będące datą, a zwraca, którym dniem roku jest podana data. Zadbaj o idiotoodporność funkcji – jeśli data, z którą została wywołana funkcja, nie istnieje, funkcja zwróci 0.

Pamiętaj o latach przestępnych – rok jest przestępny, jeśli jest podzielny przez 400 lub jest podzielny przez 4 i nie jest podzielny przez 100.

Wykorzystaj tablicę dwuwymiarową – pierwszy wiersz przechowuje liczbę dni miesięcy w roku normalnym, drugi wiersz w roku przestępnym.

W `main()` sprawdź działanie funkcji dla następujących danych:

```
dzienRoku(2014,05,27);   // 147
dzienRoku(2015,13,20);   // 0
dzienRoku(1983,02,29);   // 0
dzienRoku(1984,02,29);   // 60
dzienRoku(2000,02,29);   // 60
dzienRoku(2004,02,29);   // 60
dzienRoku(1900,02,29);   // 0
dzienRoku(2100,02,29);   // 0
```

Zad. 79:

Korzystając z mechanizmu wskaźników (nie korzystaj tym razem z STLa), zaimplementuj tzw. kolejkę priorytetową. Jest to prawie zwykła lista jednokierunkowa, ale tym razem przy wstawianiu podajemy dodatkowy parametr, tzw. „priorytet”. Im wyższy priorytet, tym wyżej w kolejce znajdzie się dany element. Elementy o tym samym priorytecie ustawiane są w kolejności napływania. Zaimplementuj odczyt z pliku: każdy element w nowej linijce, elementy to pary liczb typu `int` w formacie *„<priorytet> <spacja> <element>”*.

Zad. 80:

Napisz deklarację tablicy z inicjalizacją określonych komórek wartościami. Tablica powinna mieć nazwę `wartosciZnaki` i wymiary `3x6x4x5` oraz typ elementów `unsigned char`. Poniższe lokalizacje powinny być statycznie zainicjowane następującymi wartościami:

1,2,2,3 – `'A'`

2,4,3,2 – `'3'`

2,4,3,3 – `3`

2,1,1,2 – `0320`

1,1,1,1 – `' '` // (spacja)

1,4,2,3 – `'\n'`

2,5,3,5 – `125`

1,3,2,2 – `0xf3`

2,2,3,1 – `'\121'`

1,2,3,4 – `'X'`

2,2,1,1 – `'0'`

Wszystkie lokalizacje poza wymienionymi powinny być zainicjowane binarnym (nie znakowym) zerem.

Napisz program sprawdzający inicjalizację poprzez wydrukowanie na ekran całej tablicy. Ponieważ niektóre z wartości nie są drukowalne, wydrukuj wartości jako liczby całkowite (format ósemkowy albo szesnastkowy może być wygodniejszy).

ZADANIA C++: programowanie obiektowe

Zad. 1:
Skoro znamy konstruktory, to doczytaj o destruktorach.

Zad. 2:
Stwórz klasę *„Liczba zespolona”* i oprogramuj jej działanie zgodnie z regułami matematyki. Powinna posiadać pary metod `get-set`, trzy konstruktory i być rozdzielona do odpowiednich plików .h i .cpp.

Zad. 3:
Napisz klasę realizującą rozwiązywanie równania kwadratowego.
Klasa powinna posiadać pola prywatne `A`, `B`, `C`, reprezentujące współczynniki równania, wraz z odpowiednimi akcesorami i modyfikatorami.
Dodatkowo należy stworzyć funkcję składową `delta()`, obliczającą wyróżnik trójmianu oraz trzy funkcje obliczające miejsca zerowe.
Klasa powinna być wyposażona w konstruktor domyślny oraz trójparametrowy konstruktor ogólny.
Należy zastanowić się nad przypadkiem, gdy równanie staje się liniowym tzn. `A = 0` i zaproponować rozwiązanie tego problemu.

Obiekt stworzonej klasy może być wykorzystany następująco:

```
RownanieKwadratowe r;
double num;
cout << 'Podaj A:'; cin >> num; r.ustawA( num );
cout << 'Podaj B:'; cin >> num; r.ustawB( num );
cout << 'Podaj C:'; cin >> num; r.ustawC( num );
if( r.delta() > 0 )
     cout << "Pierwiastki rownania x1=" << r.obliczX1() <<
     " x2=" << r.obliczX2() << endl;
else
if(r.delta() = 0 )
     cout << "Pierwiastek podwójny x12=" << r.obliczX12()
     << endl;
else
     cout << "Brak pierwiastkow rzeczywistych" << endl;
```

Obiekty mogą być inicjowane następująco:

```
RownanieKwadratowe r1; // Konstruktor inicjuje A=B=C=0;
RownanieKwadratowe r2(3,2,1); // Konstruktor inicjuje A=3,B=2,C=1;
```

Zad. 4:
Napisz klasę `Macierz` i zaimplementuj na jej rzecz przeciążenie operatorów tak, aby realizowały następujące funkcje:
+ dodawanie macierzy,
- odejmowanie macierzy,
* mnożenie macierzy przez skalar.

Zad. 5:

Załóżmy, że gromadzimy wodę mineralną. Są trzy rodzaje butelek – o pojemności 2l (duże), średnie o pojemności 1l oraz małe o pojemności 0.5 litra. Stworzyć klasę MyWater z metodami:
- `void addLarge(int)` – dodaje do zapasu wody podaną jako argument liczbę dużych butelek.
- `void addMedium(int)` – dodaje do zapasu wody podaną jako argument liczbę średnich butelek.
- `void addSmall(int)` – dodaje do zapasu wody podaną jako argument liczbę małych butelek.

Dodatkowo należy zaimplementować metody umożliwiające uzyskanie informacji o tym, ile jest każdego rodzaju butelek, oraz jaka jest łączna pojemność zgromadzonej wody.
Pojemności butelek (dużych, małych, średnich) przedstawić jako pola statyczne. Dostarczyć metod pozwalających uzyskiwać informacje o tych pojemnościach oraz je zmieniać.
Klasę przetestować wyprowadzając dane na ekran:
- Mam teraz 6.5 litrów wody.
- Dużych butelek: 2
- Małych butelek: 3
- Średnich butelek: 1

Zad. 6:

Napisz program, w którym zdefiniujesz jedną z wymienionych hierarchii:
1. Pojazdów
2. Książek
3. Co Ci do głowy wpadnie :-)

Zadbaj o poprawność schematu dziedziczenia, odpowiednie zadeklarowanie pól i metod w zależności od poziomu szczegółowości, itp. Niezależnie od wszystkiego – prowadź ewidencję, ile obiektów danej klasy aktualnie istnieje.

Zad. 7:

1. Wykonaj prostą grę polegającą na symulacji zachowań drogowych.
2. Zdefiniuj klasy użytkowników drogi (pieszych, samochodów, rowerów).
3. Zadbaj o ich dziedziczenie z abstrakcyjnej klasy bazowej.
4. Każdy z obiektów danej klasy potrafi się poruszać po planszy (konsoli) z różną prędkością. Każdy z obiektów ma również swój odpowiedni znaczek.
5. Obiekty poruszają się po planszy w sposób losowy, gra kończy się w momencie „kraksy”.
6. Zadbaj o krokową pracę (aby można było podglądać każde stadium).
7. Przeprowadź symulacje dla różnej wielkości planszy oraz dla różnej liczby użytkowników drogi.

Zad. 8:

Napisz klasę o nazwie `Motorowka`, której składnikami będą cena, moc i maksymalna prędkość. Dopisz również dwie funkcje: pobierającą i wypisującą parametry motorówki. Dodaj prywatne atrybuty dla wektora prędkości. Napisz dwie metody zmieniające wektor prędkości motorówki: jedną, nadająca jej zupełnie nowy wektor prędkości, oraz drugą, zachowującą kierunek, ale zmieniającą prędkość (długość wektora). Przyjmij, że motorówka nie pływa do tyłu. W funkcji głównej wywołaj poszczególne metody.

Zad. 9:

Zdefiniuj klasę `Kolo`, która będzie zawierać:

- pola prywatne:
  - promień koła (typ `float`),
  - kolor wypełnienia (typ `int`)
- konstruktor bezparametrowy inicjujący wartość początkową pola koła (dowolnie ustaloną),
- konstruktor z parametrami promień, kolor wypełnienia,
- metody publiczne:
  - `Oblicz_Pole(..)`, obliczająca pole powierzchni koła,
  - `Ustaw_Promien(..)`, zmieniająca promień koła,
  - `Ustaw_Kolor(..)`, zmieniającą kolor koła.

Zdefiniuj przeciążenie operatora == porównujące, czy dwa koła mają taką samą powierzchnię i kolor.

Uruchom w programie głównym sekwencję instrukcji dla dwóch obiektów `K1`, `K2`: konstruktor bezparametrowy, porównanie pól, ustawienie tego samego koloru, porównanie pól, ustawienie różnych kolorów, porównanie pól. Użyj na końcu konstruktora z parametrami dla obiektów `K3`, `K4` i pokaż różne warianty ich porównania.
Dodatkowo:
Spróbuj wykonać to samo z udziałem wskaźników na obiekty oraz uwalnianiem pamięci dynamicznej.

Zad. 10:

Stwórz klasę `Osoba`, która będzie mieć `Imie` i `Plec`. Stwórz klasę `Student`, która będzie dziedziczyć po klasie `Osoba` i dodatkowo będzie mieć daną składową `Punktacja`. Zaimplementuj metodę `wyswietlInformacje()`, która dla klasy `Osoba` wypisze imię i płeć. Niech klasa `Student` skorzysta z metody klasy nadrzędnej, a dodatkowo wypisze informacje nt. punktacji. Stwórz w klasie wspólny licznik wszystkich utworzonych obiektów.

W funkcji `main()` zademonstruj działanie wszystkich klas.
Dodatkowo:
Stwórz klasę `Punktacja`, która będzie posiadała trzy dane składowe: `maksymalnaPunktacja`, `otrzymanaPunktacja`, `procent`. Zastąp tą klasą odpowiednie dane składowe klasy `Student`.

Zad. 11:

Zaprojektuj klasę reprezentującą `Samolot`. Zaproponuj atrybuty dla tej klasy pozwalające na określenie jego rozmiaru i maksymalnej liczby pasażerów oraz wektora prędkości (składa się on z kierunku oraz prędkości). Atrybuty dla wektora prędkości mają być prywatne. Napisz dwie metody zmieniające wektor prędkości statku: jedną, nadająca mu zupełnie nowy wektor prędkości, oraz drugą, zachowującą kierunek, ale zmieniającą prędkość (długość wektora). Pamiętaj, że samolot nie lata do tyłu.
W funkcji `main()` zademonstruj działanie klasy.

Zad. 12:
Zdefiniuj klasę `Prostokat`, która będzie zawierać:

- pola prywatne typu całkowitego:
  - długości boków prostokąta,
  - kolor wypełnienia
- konstruktor bezparametrowy inicjujący wartość początkową pola prostokąta (dowolnie ustaloną),
- konstruktor z parametrami długość, wysokość i kolor wypełnienia prostokąta,
- metody publiczne:
  - `Daj_Dlugosc(..)`, zwracającą długość prostokąta,
  - `Daj_Wysokosc(..)`, zwracającą wysokość prostokąta.

Zdefiniuj przeciążenie operatora == porównujące, czy dwa prostokąty mają boki tej samej długości (mogą być podane w odwrotnej kolejności!) i kolor.

Uruchom w programie głównym sekwencję instrukcji dla dwóch obiektów `P1`, `P2`: konstruktor bezparametrowy, porównanie czy są takie same. Użyj też konstruktora z parametrami dla obiektów `P3`, `P4` oraz `P5`, `P6` i pokaż różne warianty porównania ich ze sobą.

Zad. 13:
1. Zdefiniować klasy reprezentujące płaskie figury geometryczne: `Kwadrat`, `Prostokat`, `Kolo`, `Trapez`. Klasy powinny posiadać pola przechowujące informacje niezbędne do obliczenia pól tych figur.
2. Pola powinny być prywatne, dostęp do nich powinien być zapewniany przez odpowiednie metody pobierania i ustawiania wartości. Klasy powinny posiadać konstruktor bezparametrowy oraz ogólny, zapewniające inicjalizację wszystkich pól.
3. Obliczenie pola danej figury powinna realizować odpowiednia funkcja składowa o nazwie `obliczPole` — wyznaczona wartość pola powinna być jej rezultatem.

Zad. 14:
Bazując na klasach opisu figur płaskich stworzonych wcześniej, zdefiniować ich klasy pochodne, reprezentujące bryły: sześcian, prostopadłościan, kula, graniastosłup o podstawie trapezu. W klasach pochodnych należy dodać wszelkie informacje konieczne dla obliczenia pól tych brył oraz należy przedefiniować funkcje składowe obliczania pola (funkcje `obliczPole` każdej z klas reprezentujących figurę płaską), tak by wyznaczały właściwe pola brył.

Zad. 15:
Zdefiniuj klasę napis. Zaimplementuj przeciążenie operatorów + oraz +=.

Zad. 16:
Zdefiniuj klasę napis. Zaimplementuj przeciążenie operatorów - oraz -=.

Zad. 17:

Stwórz książkę adresową do przechowywania danych osób, opartą na obiektach. Program ma posiadać następujące menu:

1. Dodaj nową osobę.
2. Edytuj/usuń osoby.
3. Wyszukaj osoby.
4. Wyświetl osoby.
5. Importuj dane z pliku.
6. Eksportuj dane do pliku.
7. Zakończ program.

Dodaj nową osobę umożliwia dodanie nowej osoby do bazy danych na podstawie parametrów takich jak: Imię i nazwisko, data urodzenia, pełny adres, oraz opcjonalnie: numer gg, adres e-mail, adres www.
Edycja powoduje wyświetlenie wszystkich dostępnych osób i na podstawie tej listy ma być możliwość wyboru konkretnej osoby do edycji lub do usunięcia. Usunięcie osoby ma być potwierdzane komunikatem *„Czy jesteś pewien?”*.
Wyszukiwanie osób ma być możliwe po nazwisku, dacie urodzenia lub mieście zamieszkania.
Wyświetl osoby – wyświetla wszystkie osoby z książki z możliwością sortowania według imienia, nazwiska, roku urodzenia lub miasta.
Import i eksport kolejno wczytują lub zapisują dane do pliku tekstowego.
Ma być też możliwość wyświetlenia liczby zapisanych osób. Dodatkowo każda osoba powinna mieć swój unikalny identyfikator.
Każda osoba to obiekt danej klasy. Książka adresowa agreguje obiekty klasy osoba. Należy wykorzystać pola i 1 metodę statyczną, co najmniej 2 konstruktory oraz pamiętać o hermetyzacji.

Zad. 18:

Napisać klasę `Zamowienie`, która umożliwia przechowanie i zarządzanie informacjami związanymi z zamówieniem na towar. Klasa ta powinna zawierać następujące **pola prywatne**:
• `int * towary` – tablica z listą towarów w zamówieniu (towar reprezentowany przez identyfikator będący liczbą całkowitą),
• `double * ceny` – tablica z cenami poszczególnych towarów w zamówieniu (każda cena przypisana jest do towaru zapisanego pod tym samym indeksem w tabeli),
• `int maxPoz` – maksymalna liczna pozycji - zamówienie nie może mieć więcej niż `maxPoz` pozycji (towarów),
• `int ilePoz` – aktualna liczba pozycji - przechowuje aktualną liczbę pozycji zamówienia.
Konstruktory:
• Konstruktor **ogólny** z parametrem (`int maxPoz`) określającym maksymalną liczbę pozycji w zamówieniu. Konstruktor powinien nadać odpowiednie wartości wszystkim polom klasy i pozwolić na stworzenie tablic (towary, ceny).
• Konstruktor **kopiujący**, umożliwiający stworzenie zamówienia na podstawie już istniejącego.
Metody publiczne:
• `void dodajPozycje(int idTowaru, double cenaTowaru)` – funkcja dodaje towar do listy elementów zamówienia, jeżeli jest jeszcze na niej miejsce.
• `int najdrozszyTowar()` – funkcja zwraca identyfikator towaru (z tablicy towary), którego koszt jest największy. W przypadku, gdy zamówienie jest puste wynikiem jest -1.
• `double kosztCalkowity()` – funkcja zwraca sumaryczny koszt zamówienia.
• `void wyswietl()` – funkcja wyświetla całość zamówienia, a więc identyfikatory wszystkich towarów i odpowiadające im ceny oraz koszt całego zamówienia.

Zad. 19:

Poniżej podana została klasa `Lista`, która służy do przechowywania listy liczb całkowitych, uzupełnij metody tej klasy.

```
class Lista {
private:
      int liczby[100]; // tablicy na przechowywane liczby
      int pojemnosc; // liczba elementów, które można // pomieścić na liście
      int rozmiar; // liczba elementów aktualnie // przechowywana na liście
public:
      Lista();
      // dodaje nową liczbę do listy
      void dodaj(int liczba);
      // zwraca element listy znajdujący sie na podanym indeksie
      int dajElement(int indeks);
      // wyświetla aktualny rozmiar listy, tj. liczbę elementów
      int wezRozmiar();
};
```

Przykład użycia klasy lista:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "Lista.h"
using namespace std;
int main()
{
      Lista l;
      srand(time(NULL));
      for(int i = 0; i < 20; i++)
            l.dodaj( rand() % 100 );
      cout << "Zawartosc listy: ";
      for(int i = 0; i < l.wezRozmiar(); i++)
            cout << l.dajElement(i) << " ";
      cout << endl;
      return 0;
}
```

Zad. 20:

Napisać dwie klasy, których celem będzie liczenie **pola** i **objętości** następujących figur:

- Kwadrat
- Sześcian

Każda klasa powinna posiadać dwa konstruktory: **domyślny** i **ogólny**, które odpowiednio zerują wartości pól klasy lub ustawiają je na wartości podane podczas tworzenia obiektu. Klasa sześcian powinna dodatkowo zawierać konstruktor kopiujący i rzutujący. Proszę zadbać o to, by możliwa do ustalenia **długość boków** była większa od **zera** – nie można ustawić np. kwadratu o długości boku zero. Jeżeli jednak użytkownik będzie próbował ustawić taką długość boku, proszę tej wartości nie akceptować (nie przypisywać do pola), a przy próbie wyliczenia objętości lub pola powierzchni danej figury wyświetlać komunikat *„Operacja niemożliwa do zrealizowania. Sprawdź ustawioną długość boków”*. Niedozwolone jest deklarowanie pól klasy w sekcji `public`.

Zad. 21:

Zdefiniuj klasę `Przedmiot` reprezentującą przedmiot ćwiczeniowy studenta. Klasa powinna zawierać:

- pola prywatne:
- `double oceny[];` // oceny studenta
- `int nr_indeksu;` // numer indeksu studenta
- `int liczba_ocen;` // liczba wprowadzonych ocen
- `int limit_ocen;` // maksymalna liczba ocen

- publiczny konstruktor z dwoma parametrami:
- `int nr_indeksu, int limit_ocen`

- metody publiczne:
- `double obliczSrednia();` // oblicza średnią arytmetyczną ocen
- `bool zaliczenie();` // określa, czy student otrzyma zaliczenie
- `void dodajOcene(double ocena);` // dodaje nową ocenę
- `void zmienOcene(double ocena, int id);` // zmienia ocenę
- `void pisz();` // wypisuje na ekranie informację

Dodatkowe informacje:

- Funkcja `obliczSrednia` oblicza i zwraca średnią arytmetyczną wpisanych już ocen.
- Funkcja `zaliczenie` ustala, czy student otrzyma zaliczenie (średnia wszystkich uzupełnionych ocen większa lub równa 3.0). Jeśli otrzyma, to zwraca `true`, w przeciwnym razie `false`.
- Funkcja `dodajOcene` dodaje nową ocenę (jeśli jest to możliwe – nie przekroczy limitu).
- Funkcja `zmienOcene` zmienia ocenę na podanej pozycji (`id` liczonej od 0), ale tylko jeśli taka ocena istnieje.
- Funkcja `pisz` wyświetla na ekran studenta, wszystkie uzupełnione oceny i informację, czy aktualnie dostałby zaliczenie (*tak* lub *nie*).

Zad. 22:

Mamy daną klasę:

```
class wektor    {
public:
    double x, y, z;
    wektor (double xx = 0, double yy = 0, double zz = 0){
        x = xx;
        y = yy;
        z = zz;
    };
};
```

Przeciążyć operatory, aby realizowały następujące funkcje:
+ dodawanie wektorów,
- odejmowanie wektorów,
* iloczyn skalarny.
Sprawdź iloczyn skalarny na następującym przykładzie: (1, -3, -5) + (4, 3, -1) = 0.

Zad. 23:

Stwórz klasę reprezentującą liczby wymierne. Zaimplementuj podstawowe działania arytmetyczne przeciążenie operatorów wejścia/wyjścia (tj. << oraz >>).

Zad. 24:
Utwórz klasę reprezentującą przedziały otwarte liczb rzeczywistych. Zaimplementuj funkcje wyznaczania: części wspólnej, arytmetycznej sumy i iloczynu przedziałów.

Zad. 25:
Stwórz klasy reprezentujące: punkt, odcinek i prostą. Zapewnij metody pozwalające znajdować odległość punkt od innego punktu oraz prostej, a także metodę wyznaczająca długość odcinka.

Punkt: A = (xa, ya)

Punkt: B = (xb, yb)

Prosta: $c \cdot x + d \cdot y + e = 0$

Odległość punktów od siebie: $\sqrt{(xb-xa)^2+(yb-ya)^2}$

Odległość punktu A od prostej: $\frac{|c \cdot xa + d \cdot ya + e|}{\sqrt{c^2+d^2}}$

Zad. 26:
Stwórz wzorzec klasy zawierający wspólny licznik wszystkich obiektów utworzonych z wszystkich klas konkretyzowanych z tego wzorca.

Zad. 27:
Stwórz typ `para` zawierający publiczne pola `x` oraz `y` typu `double` (jako klasę albo jako strukturę). Typ ten reprezentuje parę liczb rzeczywistych, traktowaną zależnie od kontekstu jako punkt bądź wektor.

Stwórz klasę okręgu, tak aby obiekty tej klasy można było stworzyć na każdy z poniższych sposobów:

- podając środek oraz promień okręgu,
- podając środek oraz punkt leżący na okręgu,
- podając dwa przeciwległe punkty na okręgu,
- podając trzy punkty leżące na okręgu,
- podając trzy punkty stanowiące wierzchołki trójkąta opisanego na tym okręgu.

Zad. 28:
Zdefiniuj typ `wektor4` jako wektor czterech liczb rzeczywistych. Zdefiniuj dla tego typu operatory: +, -, *, /, =, +=, -=, *= i /= dla kombinacji wektorów i liczb rzeczywistych.
http://pl.wikipedia.org/wiki/Wektor

Zad. 29:
Zdefiniuj klasę `wektor`, w której rozmiar będzie argumentem konstruktora `wektor::wektor(int)`. Zdefiniuj dla tego typu operatory: +, -, *, /, =, +=, -=, *= i /= dla kombinacji wektorów i liczb rzeczywistych.
http://pl.wikipedia.org/wiki/Wektor

Zad. 30:
Zdefiniuj:
```
class podstawowa {
public:
     virtual void jaJestem() {
          cout << "podstawowa\n";
     }
};
```
Wyprowadź z klasy podstawowa dwie klasy pochodne i w każdej zdefiniuj funkcję `jaJestem()`, wypisującą nazwę klasy. Utwórz obiekty tych klas i wywołaj na nich `jaJestem()`. Przypisz adresy obiektów klas pochodnych wskaźnikom typu `podstawowa*` i – korzystając z tych wskaźników – wywołaj `jaJestem()`.

Zad. 31:
Zdefiniuj klasę `trojkat` i klasę `okrag`. Obie klasy muszą zawierać klasę `figura` jako publiczną klasę podstawową.

Zad. 32:
1. Stworzyć dwie klasy. Każda ma po 3 pola, które są liczbami naturalnymi. Dwa z nich są polami prywatnymi.
2. Zainicjować po obiekcie danej klasy. Wyświetlić w wątku głównym wszystkie pola (stworzyć potrzebne metody).
3. Stworzyć metodę, której atrybutem będzie obiekt drugiej klasy. Metoda ma dodać do każdego pola obiektu swoje pierwsze pole, podzieli przez drugie (jeśli zero, obsłużyć wyjątek), a następnie zapiszę wyniki tych działać w pola obiektu, który został jej przekazany.

Zad. 33:
Napisz definicje instrukcji inicjujących do poniższej klasy:
```
class Vector
{
     private:
          double x;
          double y;
     public:
          Vector();
          Vector(double, double);
};
```
Klasa ma reprezentować wektor w przestrzeni dwuwymiarowej, a instrukcje inicjujące mają realizować inicjalizację tego wektora. Pierwsze instrukcje inicjujące powinny ustawiać wektor na wartość domyślną (0,0).

Dopisz konstruktor kopiujący.
```
Vector(const Vector&);
```

Zad. 34:
Napisz obiektową wersję gry Kółko i krzyżyk. *Wskazówki*: dobrym kandydatem na obiekt jest oczywiście plansza. Zdefiniuj też klasę graczy, przechowującą ich imiona (niech program pyta się o nie na początku gry).

Zad. 35:
Poniższa klasa miała implementować dowolnej wielkości tablicę obiektów klasy Vector z poprzedniego ćwiczenia. Niestety okazało się, że powoduje wycieki pamięci – programista zapomniał o napisaniu destruktora:

```
class VectorsArray
{
     public:
          Vector* vectors;

          VectorsArray(size_t);
          Vector GetVector(size_t);
          size_t GetSize();
          size_t size;
};

VectorsArray::VectorsArray(size_t argSize)
: size(argSize)
, vectors(new Vector[argSize])
{
}
Vector VectorsArray::GetVector(size_t i)
{
     return vectors[i];
}
size_t VectorsArray::GetSize()
{
   return size;
}
```

Do powyższej klasy dopisz definicję destruktora. Nie zapomnij o dealokacji pamięci!

Zad. 36:
Zaimplementuj listę jednokierunkową jako klasę.

Zad. 37:
Zaimplementuj listę dwukierunkową jako klasę.

Zad. 38:
Zaimplementuj drzewo przeszukiwań binarnych BST jako klasę.

Zad. 39:
Utwórz klasę `Plik`. Klasa powinna zawierać:

- pole prywatne - wskaźnik na zmienną plikową typu `FILE`,
- konstruktor z parametrem otwierający plik dyskowy o podanej nazwie - wskaźnik na nazwę,
- destruktor zamykający plik dyskowy.

Uruchom program z obsługą obiektową pliku.

Zad. 40:

Napisz program do prowadzenia dokumentacji i wykonywania analiz statystycznych dla klasy 20 uczniów. Informacja o każdym uczniu zawiera: ID, imię, płeć, wyniki kolokwiów (3 kolokwia w semestrze) i wynik końcowy (średnią z kolokwiów). Użyj do tego celu klasy – zaimplementuj odpowiednie funkcjonalności, z których będzie można później skorzystać.
W funckji `main()` program poprosi użytkownika, aby wybrać z menu:

```
================================================
                        MENU
================================================
1. Dodaj wpis.
2. Usuń wpis.
3. Aktualizuj wpis.
4. Wyświetl wszystkie wpisy.
5. Oblicz średnią wyników wybranego studenta.
6. Wyświetl średnią wyników wybranego studenta.
7. Oblicz i wyświetl średnią wszystkich studentów.
8. Wyświetl najlepszy i najgorszy wynik (wraz z danymi studentów,
którzy je osiągnęli).
Wpisz swój wybór:
```

Zad. 41:

Utwórz klasę `Punkt`. Klasa powinna zawierać:

- 2 pola prywatne typu całkowitego (współrzędne punktu),
- konstruktor bezparametrowy inicjujący dowolne wartości początkowe punktu,
- konstruktor z parametrami (współrzędne punktu),
- metodę publiczną - zwracającą współrzędną `x`,
- metodę publiczną - zwracającą współrzędną `y`.

Zdefiniuj przeciążenie operatora == porównujące dwa punkty, czy się nakładają.

Uruchom program dla dwóch różnych obiektów, porównaj je, a następnie dokonaj zmian w jednym z nich, by uczynić obiekty identycznymi. Spróbuj wykonać te same działania w wersji z dynamicznym przydziałem pamięci.

Zad. 42:

Zaprojektuj klasę reprezentującą statek. Zaproponuj atrybuty dla tej klasy pozwalające na określenie jego rozmiaru i maksymalnej liczby pasażerów oraz wektora prędkości. Atrybuty dla wektora prędkości mają być prywatne. Napisz dwie metody zmieniające wektor prędkości statku: jedną, nadająca mu zupełnie nowy wektor prędkości, oraz drugą, zachowująca kierunek, ale zmieniająca prędkość (długość wektora). Przyjmij, że statek nie pływa do tyłu.
W funkcji `main()` zademonstruj działanie klasy.

Zad. 43:

Napisz klasę o nazwie `Samochod`, której składnikami będą cena, moc i maksymalna prędkość.
Dopisz również dwie funkcje: pobierającą i wypisującą parametry samochodu. W funkcji głównej wywołaj je.

Zad. 44:

Zaprojektuj klasę `Klient`: dane osobowe klienta – imię, nazwisko, e-mail, nr telefonu.

Zaprojektuj klasę `Rezerwacja`:

1. numer referencyjny;
2. `Klient`;
3. data rezerwacji;
4. data przyjazdu;
5. rodzaj pokoju;
6. liczba nocy;
7. uwagi;
8. rabat.

Dodaj metody dostępowe do zmiennych (pól) klasy.

Utwórz konstruktory pozwalające utworzyć obiekty.

Zwróć uwagę, że wartość pola numer referencyjny rezerwacji klasy `Rezerwacja` jest nadawany w momencie tworzenia obiektu (np. na podstawie znacznika czasu lub inny dowolny ciąg znaków) i później ta wartość nie może być zmieniona. Również jako data rezerwacji jest ustawiana data bieżąca w momencie tworzenia obiektu.

Zabezpiecz, aby nie można było utworzyć rezerwacji dla daty przyjazdu, która już minęła.

Dodaj metodę obliczającą kwotę noclegu na podstawie następujących stawek oraz rabatu:

Rodzaj pokoju:

A 290,00

B 330,00

C 410,00

D 450,00

E 510,00

F 520,00

G 1150,00

Zad. 45:

Zaimplementuj klasę `Ulamek` z prywatnymi polami całkowitymi `Licznik` i `Mianownik`. Należy odpowiednio przeciążyć operatory `*`, `-`, `+`, `*=`, `=`, `-=`, `+=`, `--`, `++`, `<`, `>`, `==`, `<<` (wypisywanie do strumienia) oraz zaimplementować konstruktory. Mianownik nie może być równy zero. Ułamki mają być wypisywane w postaci: `Licznik/Mianownik`, np.: „`3/4`”, „`-79/8`”.

W funkcji `main()` powołaj do życia kilka obiektów oraz zademonstruj działanie przeciążonych operatorów klasy `Ulamek`.

Zad. 46:

Zaimplementuj klasę `Data`, której składowymi prywatnymi będą: `dzien`, `miesiac`, `rok`. Utwórz funkcję `pobierzDate()`, która będzie pobierać datę od użytkownika.

Przeciąż operator << aby wypisywał datę zapisaną w obiekcie klasy. Jeśli dzień lub miesiąc będą jednocyfrowe, to dopisz przed nimi „0”.

Stwórz konstruktor bezparametrowy, który domyślnie ustawia datę na 1 stycznia 1970.

Stwórz konstruktory: jeden pobierający tylko rok (pozostałe składniki domyślnie ustawia na 1 stycznia), drugi pobierający tylko dzień i miesiąc (rok domyślnie ustawia na 1970).

Stwórz metody pozwalające na zmianę każdego pola osobno. Możesz z nich skorzystać w konstruktorach.

Zadbaj o poprawność wprowadzanych danych.

W funkcji `main()` zademonstruj działanie zaimplementowanych funkcjonalności.

Zad. 47:

Dany jest układ równań liniowych

$$A_1x + B_1y = C_1$$
$$A_2x + B_2y = C_2$$

Rozwiązanie układu równań może polegać na wyliczeniu odpowiednich wyznaczników W, Wx, Wy a następnie ich iloczynów — zgodnie z informacjami poznanym na zajęciach z matematyki.
Należy zaprojektować i zaimplementować obiektowy program pozwalający na rozwiązywanie dowolnego układu takich równań. Program powinien umożliwiać wczytanie współczynników A_1, B_1, C_1, A_2, B_2, C_2, następnie powinien wyznaczyć rozwiązania równań metodą wyznacznikową. Należy identyfikować i prawidłowo zareagować na sytuację, gdy układ jest nieokreślony.

Zad. 48:

Napisać klasę `Oceny` reprezentującą informacje o ocenach studentów. Klasa powinna zawierać:
• pole prywatne: `float[][] oceny` – tablica z listą ocen studentów (pierwszy wymiar – numer studenta, drugi wymiar – numer przedmiotu),
• konstruktor z dwoma parametrami (`int liczbaStudentów`, `int liczbaPrzedmiotow`) – powinien tworzyć i zainicjować odpowiednie tablice.
Metody publiczne:
• `float ObliczSrednia(int student)` – oblicza i zwraca średnią arytmetyczną podanego studenta dla wpisanych już ocen (brak oceny to -1 w odpowiednim elemencie tablicy),
• `boolean ZaliczonySemestr(int student)` – zwraca `false`, jeśli średnia ocen jest mniejsza od 3.0 lub brak oceny z któregoś z przedmiotów, w przeciwnym wypadku `true`,
• `void WpiszOcene(int student, int przedmiot, float ocena)` – dodaje nową ocenę,
• `void zmienOcene(int student, int przedmiot, float ocena)` – zmienia ocenę na podanej pozycji wpisując średnią oceny podanej i istniejącej, ale tylko jeśli taka jest już wpisana,
• `void Pisz()` – wyświetla na ekran oceny studentów (w jednym wierszu: numer albumu, lista ocen różnych od -1, średnia ocen).

Zad. 49:

Zaimplementuj obiektową wersję funkcji `dzienRoku(dzien, miesiac, rok)`, która pobiera trzy argumenty będące datą, a zwraca, którym dniem roku jest podana data. Zadbaj o idiotoodporność funkcji – jeśli data, z którą została wywołana funkcja, nie istnieje, funkcja zwróci 0.

Pamiętaj o latach przestępnych – rok jest przestępny, jeśli jest podzielny przez 400 lub jest podzielny przez 4 i nie jest podzielny przez 100.

Wykorzystaj tablicę dwuwymiarową – pierwszy wiersz przechowuje liczbę dni miesięcy w roku normalnym, drugi wiersz w roku przestępnym.

W `main()` sprawdź działanie funkcji dla następujących danych:

```
dzienRoku(2014,05,27);   // 147
dzienRoku(2015,13,20);   // 0
dzienRoku(1983,02,29);   // 0
dzienRoku(1984,02,29);   // 60
dzienRoku(2000,02,29);   // 60
dzienRoku(2004,02,29);   // 60
dzienRoku(1900,02,29);   // 0
dzienRoku(2100,02,29);   // 0
```

Zad. 50:

Napisz klasę pozwalającą na obliczenie podatku dochodowego dla osoby fizycznej. Uwzględnij doliczenia i odliczenia od podatku, a także skalę podatkową wg wzoru:
Dochód < 1800 zł -> podatek = 0;
Dochód >=1800 zł oraz dochód <100000 zł -> podatek = 15% dochodu
Dochód >=100000 zł -> podatek = 40% dochodu
Wszystkie pola w klasie mają być prywatne, a do ich ustawiania i odczytywania użyj selektorów i modyfikatorów. Napisz konstruktor, który zeruje pola oraz konstruktor, w którym podajesz dochód oraz doliczenia.
Prototyp:

```
Podatnik kowalski = new Podatnik();
kowalski.setDochod(1587);
kowalski.setDoliczenia(-5);
cout << kowalski.ObliczPodatek() << endl;
Podatnik iksinski = new Podatnik(48765, 120);
cout << iksinski.ObliczPodatek();
```

Zad. 51:

Napisz klasę służącą do analizy częstości występowania znaków w tekście. Program ma nic nie wyświetlać na ekran, a jedynie zapisywać wyniki analizy do pliku tekstowego. Prototyp użycia:

```
Analizator a = new Analizator("testowy.txt");
a.zapisz("testowy");
```

Zad. 52:

Napisz klasę `Serwis` wspomagającą obsługę serwisową notebooków. W konstruktorze domyślnym powinna być zerowana prywatna tablica `naprawy[][]` oraz ustawiane statyczne pole `liczbaZgloszen` na 0. Zakładamy, że pierwszy wymiar tablicy to identyfikatory komputerów (max 100), a drugi – daty napraw (max 5), Dodatkowo napisz metody:

a) `boolean dodajNaprawe(int nrKomputera)`: funkcja ta pobiera aktualne naprawy komputera o indeksie `nrKomputera` (wiersz z tabeli), a następnie w kolejnym wolnym miejscu dopisuje dzisiejszą datę. Jeśli brakuje miejsca na naprawę, funkcja powinna zwrócić `false`; w przeciwnym wypadku `true`.

a) `int liczDoZlomowania()`: funkcja liczy, ile komputerów już nie może być naprawionych (brak miejsca w tablicy na naprawy)

a) `String ostatniaNaprawa(int komputer)`: funkcja ma zwrócić `string`a z datą ostatniej naprawy danego komputera

a) `double średnioNapraw()`: funkcja zwraca informacje o średniej liczbie napraw komputerów w bazie. UWAGA: komputer nigdy nie naprawiany nie wchodzi do średniej!

Zad. 53:

Napisz funkcję `odczytaj()`. Jej funkcjonalnością będzie odczytywanie informacji o osobach zawartych w pliku tekstowym do oddzielnych instancji (obiektów) klasy `Osoba`.

```
Public class Osoba {
     int wiek;
     string imię
     string nazwisko;
     string[] dzieci;
     double dochod;
}
```

Plik wejściowy ma następującą postać:

```
wiek  imię  nazwisko  liczba_dzieci  imiona_dzieci_rozdzielone_tabem   dochód
```

Przykładowo:

```
35   Jan  Kowalski  3    Ania Zdzisio   Joanna          5000
```

Zad. 54:

W pliku **obiektów (nie rekordków, nie tekstowym)** `równania.obj` zapisane są obiekty klasy `Rownanie`:

```
class Rownanie {
     String wyrażenie;
     Boolean czyPoprawne;
}
```

Napisz program, w którym: 1) odczytasz strukturę obiektów do tablicy zadeklarowanej lokalnie (przyjmij, że będzie ich nie więcej niż 100), 2) sprawdzisz poprawność wyrażenia (wyrażenie jest poprawne, jeśli zawiera tyle samo liczb, co liter; inne znaki są pomijane), 3) uzupełnisz pole `czyPoprawne` odpowiednią informacją, 4) zapiszesz obiekty z powrotem do tego samego pliku, z którego odczytałeś nadpisując zawartość, 5) wypiszesz na ekran informację o liczbie poprawnych obiektów.

Zad. 55:

Napisz program „generator losowych numerów telefonów”. Program ma umożliwiać generowanie podanej przez użytkownika liczby numerów telefonów z konkretnej strefy numeracyjnej. Dodaj konstruktor domyślny zerujący wszystkie pola oraz pole statyczne, którego wartość będzie mówić o liczbie obiektów klasy `Telefony` stworzonych do tej pory. Przykładowe zastosowanie:

```
for(int i=11;i++;i<33) {

Telefony Tel = new Telefony();

Tel->setWojewództwo(i); //ustawienie strefy numeracji na i
     oraz wyczyszczenie tablicy wewnętrznej z numerami
     telefonów
Tel->generate(1000); //wygenerowanie 1000 numerów,
     zapisywanych w tablicy (odpowiednio zadeklarowanej
     wcześniej).

Tel->show(); //wypisuje na ekran numery zapisane w tablicy
}

cout << Telefony::ile << endl; //wyświetli... ile?
```

Zad. 56:

Napisz klasę `Pożyczkobiorca`. Będzie ona pomocna przy udzielaniu pożyczki dla konsumentów przez instytucje bankowe. Klasa ma zawierać pola określające imię, nazwisko, wiek i PESEL klienta (do ich ustawiania użyj **enkapsulacji**). **Pola można ustawiać „naraz” albo każde oddzielnie.** Oprócz tego, również za pomocą enkapsulacji, klasa ma umożliwiać ustawienie parametru `zarobki netto`. Po ustawieniu zarobków, system ma automatycznie wyliczać możliwą wartość pożyczki dla klienta ze wzoru:

*pożyczka = (pensja / 7) * (65 - wiek)*

Przykładowe działanie:

```
Pożyczkobiorca Osoba1 = New Pożyczkobiorca();
Osoba1.setDane(„Jan”, „Kowalski”, 40, 40041416732, 4500); //w
     tym momencie ma się już wyliczać pożyczka
Osoba1.getPożyczka(); //wypisze wartość pożyczki na ekran
Pożyczkobiorca Osoba2 = New Pożyczkobiorca();
Osoba2.setImię(„Anna”);
Osoba2.setNazwisko(„Iksińska”);
Osoba2.setWiek(40);
Osoba2.setZarobki(10543); // w tym momencie ma się wyliczać
     pożyczka
```

Zad. 57:

Napisz klasę „generator losowych numerów telefonów”. Klasa ma umożliwiać generowanie podanej przez użytkownika liczby numerów telefonów z konkretnej strefy numeracyjnej. Przykładowe zastosowanie klasy `LosoweNumeryTelefonu`:

```
LosoweNumeryTelefonu randomTel = new LosoweNumeryTelefonu();

for(int i=11;i++;i<33) {

randomTel.setWojewództwo(i); //ustawienie strefy numeracji 32
randomTel.generate(1000); //wygenerowanie 1000 numerów,
     zapisywanych lokalnie w tablicy obiektu, a nie klasy

randomTel.save(i+„plik.tel”) //zapis do pliku tekstowego –
     ścieżka podawana w parametrze
}
```

Wynikiem działania powyższego kodu powinno być otrzymanie kilkunastu (ilu? - odpowiedz) plików z losowo wybranymi numerami telefonów (1000 sztuk) z poszczególnych stref numeracyjnych (11, 12, …, 32).
Skup się na w/w funkcjach, nie dodawaj dodatkowych funkcjonalności!

Zad. 58:

Napisz funkcję `zapisz jako()`, która będzie częścią klasy `Osoba`. Jej funkcjonalnością będzie zapisywanie informacji o osobach do pliku tekstowego. Załóż, że za każdym razem plik będzie miał nazwę złożoną z imienia i nazwiska danej osoby. Prototyp klasy `Osoba` przedstawia się następująco:

```
public class Osoba  {
     int wiek;
     String imię;
     String nazwisko;
     String[] dzieci = new String[10];
     double dochód;
}
```

Zad. 59:

Zaprojektuj dwupoziomową hierarchię klas i zapisz ją w pliku nagłówkowym:

1. Zaimplementuj klasę główną.
2. W klasie głównej zaimplementuj trzy konstruktory: bezparametrowy, parametrowy, kopiujący.
3. W klasie głównej zaimplementuj destruktor.
4. Klasa ma posiadać pola i metody prywatne i publiczne (co najmniej po jednym z każdego).
5. W klasie głównej zdefiniuj metodę wirtualną oraz niewirtualną. Nadpisz obie metody w klasie potomnej.
6. Zaimplementuj trzy klasy dziedziczące po klasie głównej.
7. W klasie głównej lub potomnej zdefiniuj składnik statyczny (np. zliczający liczbę obiektów danej klasy).
8. W klasie potomnej zaimplementuj przeciążenie trzech wybranych operatorów.
9. Dodaj, cokolwiek jeszcze ciekawego Ci przyjdzie do głowy :-)

W `main()` powołaj do życia kilka obiektów stworzonych klas. Pokaż na ich przykładzie działanie zaimplementowanych funkcjonalności z punktów 1-8.

`Figura`, podklasy: `Trojkat, Prostokat`, `Romb`

`Zwierze`, podklasy: `Pies`, `Kot`, `Chomik`

`Pies`, podklasy: `Jamnik`, `Terier`, `Husky.`

`Parzystokopytne`, podklasy: `Jelen`, `Zyrafa`, `Wielblad`

`Ptak`, podklasy: `Golab`, `Wrobel`, `Kos`

`Budynek`, podklasy: `Biblioteka`, `Urzad`, `Szpital`

`Mebel`, podklasy: `Szafa`, `Lozko`, `Krzeslo`

`Sztucce`, podklasy: `Widelec`, `Lyzka`, `Noz`

`CialoNiebieskie`, podklasy: `Gwiazda`, `Planeta`, `Satelita`

`Bog`, podklasy: `Zeus`, `Herkules`, `Apollo`

`Roslina`, podklasy: `Kwiat`, `Drzewo`, `Krzew`

`DzieloSztuki`, podklasy: `Obraz`, `Plakat`, `Rzezba`

`Przedmiot`, podklasy: `Informatyka`, `Fizyka`, `Matematyka`

`Pojazd`, podklasy: `Samochod`, `Autobus`, `Motor`

`Taniec`, podklasy: `ChaCha`, `Rumba`, `Walc`

`Plyta`, podklasy: `CD`, `DVD`, `BluRay`

`UrzadzenieMobilne`, podklasy: `Laptop`, `Smartfon`, `Notebook`

ZADANIA C++: STL

Zad. 1:
Wczytaj plik tekstowy, rozbij go na poszczególne litery i przedstaw analizę częstości występowania tychże (map).

Zad. 2:
Napisz program do generowania losowych liczb korzystając z vectora. Użytkownik podaje, ile tych liczb ma być. Posortuj je malejąco.

Zad. 3:
Zadeklaruj dwa stosy w programie. Za pomocą drugiego stosu, odwróć elementy ze stosu pierwszego.