Studia Podyplomowe - Inzynieria Oprogramowania

Zbiór zadan z Podstaw Programowania C++

Dr inz G. Rogus

Spis treści

1 Instrukcje wejścia-wyjścia. Instrukcje warunkowe i wyboru

1. Napisz program, który wyświetli na ekranie Twoje imię i nazwisko oraz datę urodzenia.

2. Napisz program, sprawdzający czy dany rok jest przestępny.

3. Napisz program, który wyświetli wartość bezwzględną z podanej przez użytkownika liczby.

4. Napisz program, który wczyta dwie liczby rzeczywiste i wyświetli na ekranie ich sumę z dokładnością do trzech miejsc po przecinku.

5. Napisz program, który wczyta dwie liczby rzeczywiste i wyświetli na ekranie ich sumę, różnicę oraz iloczyn z dokładnością do dwóch miejsc po przecinku.

6. Napisz program, wczyta długości boków prostokąta i obliczy jego obwód oraz pole.

7. Napisz program, który wczyta od użytkownika długość promienia i obliczy obwód oraz pole koła.

8. Napisz program, wczyta długości boków prostokąta i i w zależności od wyboru użytkownika obliczy jego obwód lub pole.

9. Napisz program, który wczyta od użytkownika długość promienia i w zależności od wyboru użytkownika obliczy obwód lub pole koła.

10. Napisz program, który wczyta z klawiatury 4 liczby rzeczywiste, a następnie wyświetli informację ile z nich jest mniejszych od 0.

11. Napisz program, który wczyta z klawiatury 3 liczby rzeczywiste, a następnie wyświetli je uporządkowane rosnąco.

12. Napisz program, który wczyta z klawiatury 3 liczby rzeczywiste, a następnie wyświetli ich wartość środkową.

13. Napisz program, który dokona zamiany podanej przez użytkownika liczby naturalnej od 1 do 7 na odpowiadający jej dzień tygodnia. Obsłuż sytuację, gdy użytkownik poda wartość spoza zakresu.

14. Napisz program, który dokona zamiany podanej przez użytkownika liczby naturalnej od 1 do 12 na odpowiadający jej miesiąc. Obsłuż sytuację, gdy użytkownik poda wartość spoza zakresu.

15. Napisz program, który wczyta liczbę rzeczywistą i wydrukuje pierwiastek z tej liczby z dokładnością do 5 miejsc oraz kwadrat tej liczby. Pamiętaj, że pierwiastkować można tylko liczby nieujemne.

16. Napisz program, który wczytuje liczbę n (gdzie $0 <= n =< 10000$), i wypisuje jej słowny odpowiednik. Przykład: 22 – dwadzieścia dwa, 0 – zero, 5463 – pięć tysięcy czterysta sześćdziesiąt trzy.

17. Napisz program, który wczyta długości trzech odcinków i sprawdzi, czy da się z nich zbudować trójkąt.

18. Napisz program, który wczyta długości trzech odcinków i sprawdzi, jaki trójkąt można z nich zbudować (jakikolwiek, prostokątny, równoboczny, równoramienny).

19. Napisz program, który wczyta współrzędne dwóch punktów i sprawdzi, czy są one współliniowe.

20. Napisz program, który po podaniu odległości w centymetrach (liczba całkowita) wypisze tą odległość w metrach, decymetrach i centymetrach. Przykład: 123 cm to 1 metr, 2 decymetry i 3 centymetry.

21. Napisz program, który w zależności od wyboru użytkownika przeliczy wartość pomiędzy centymetrami a calami. Przeliczniki między jednostkami zdefiniuj za pomocą dyrektywy `#define`.

22. Napisz program, który w zależności od wyboru użytkownika będzie dokonywał przeliczania z centymetrów na cale i odwrotnie. Przeliczniki między jednostkami zdefiniuj za pomocą dyrektywy `#define`.

23. Napisz program, który oblicza rozwiązanie równania postaci: $ax^2+bx+c = 0$. Wartości a, b i c wczytaj od użytkownika. Rozpatrz wszystkie możliwe przypadki wartości współczynników a, b, c.

24. Napisz program, który sprawdzi w której ćwiartce układu współrzędnych leży punkt o współrzędnych podanych przez użytkownika.

25. Napisz program, który sprawdzi przez które ćwiartki układu współrzędnych przechodzi prosta $y = ax + b$. Współczynniki a i b wczytaj od użytkownika.

26. Napisz program, który wyświetli na ekranie wszytkie kody ASCII od 32 do 126 wraz ze znakami, kóre się pod nimi kryją.

2 Instrukcje iteracyjne

1. Napisz program rysujący na ekranie poziomą kreskę (ze znaków minus) o długości zadawanej z klawiatury.

2. Napisz program rysujący na ekranie linię o długości zadawanej z klawiatury. Znak, z którego zbudowana będzie linia podaje użytkownik.

3. Napisz program rysujący na ekranie za pomocą podanego znaku trójkąt równoramienny o zadanej przez użytkownika długości podstawy.

znak=*, długość podstawy 7	znak=*, długość podstawy 8
*	**
***	****
*****	******
*******	********

4. Napisz program wyświetlający na ekranie sekwencje kolejnych liczb całkowitych najpierw od 1 do 10, a następnie od 10 do 1.

5. Napisz program, który wylosuje 100 liczb. Podczas losowania na bieżąco obliczaj ich sumę, wartość minimalną i maksymalną. Po zakończeniu losowania wyświetl na ekranie następujący komunikat: *średnia wylosowanych liczb wynosi ***. Spośród nich wartość minimalna to ***, a maksymalna ****. Znaki ***** zastąp wyliczonymi wartościami.

6. Napisz program wyświetlający na ekranie kolejne 20 liczb parzystych począwszy od zadanej przez użytkownika liczby.

7. Napisz program wyświetlający na ekranie kolejne liczby całkowite typu `int` (od 0 do 100), które są podzielne bez reszty przez n (gdzie n jest zadawane z klawiatury).

8. Napisz program, który wczyta od użytkownika n liczb i zliczy ile z nich jest parzystych. Wartość n podawana jest przez użytkownika na początku działania programu.

9. Napisz program, który wczyta od użytkownika n liczb i znajdzie spośród nich element minimalny oraz maksymalny. Wartość n podawana jest przez użytkownika na początku działania programu.

10. Napisz program wczytujący kolejne liczby z klawiatury i kończący się gdy:

    (a) suma tych liczb przekroczy 100,

    (b) ilość podanych liczb ujemnych przekroczy 10,

    (c) dwie kolejne podane liczby będą miały identyczną wartość.

11. Napisz program, który będzie wczytywał od użytkownika liczby dopóki ich suma będzie mniejsza od 100. Suma nie powinna uwzględniać ostatniej z podawanych liczb (tej, która spowodowała przekroczenie wartości sumy równej 100). Po zakończeniu wczytywania wypisz na ekranie wyliczoną sumę.

12. Napisz program, który będzie wczytywał od użytkownika liczby dopóki ich suma będzie mniejsza od liczby X (X podaje użytkownik na początku działania programu), a następnie wyświetli wartość średniej arytmetycznej tych liczb. Suma nie powinna uwzględniać ostatniej z podawanych liczb (tej, która spowodowała przekroczenie wartości sumy równej X).

13. Napisz program, który wczytuje liczby całkowite podane przez użytkownika aż do momentu podania wartości 0, a następnie wyświetla ich iloczyn. Zero kończące podawanie danych nie jest uwzględniane w wyniku.

14. Napisz program, który wczytuje liczby całkowite podane przez użytkownika aż do momentu podania wartości 0, a następnie wyświetli iloczyn liczb parzystych. Zero kończące podawanie danych nie jest uwzględniane w wyniku.

15. Napisz program czytujący znaki z klawiatury aż do momentu naciśnięcia klawisza `'k'` lub `'K'`. W trakcie wczytywania należy zliczać ilość podanych znaków. Na zakończenie należy wyświetlić informacje:

    (a) ile było podanych znaków (przed naciśnięciem `'k'` lub `'K'`),

    (b) ile z tych znaków było dużymi literami.

16. Napisz program wczytujący liczby z klawiatury, z jednoczesnym zadawaniem pytania *"czy koniec wprowadzania t/n ?"*. Na zakończenie program powinien wyświetlić wartość: średnią, maksymalną, minimalną z podanych liczb.

17. Napisz program liczący iloczyn wszystkich liczb nieparzystych z domkniętego przedziału liczb całkowitych $< pierwsza, ..., ostatnia >$. Wartości zmiennych *pierwsza*, *ostatnia* należy wczytać z klawiatury.

18. Napisz program będący grą w zgadywanie. W pierwszym wariancie komputer losuje liczbę a użytkownik podaje możliwe wartości i dostaje odpowiedź: *"zgadłeś"*, *"za mało"*, *"za dużo"*. W drugim wariancie zgadującym jest komputer. Zgaduje losując liczby z przedziału wyznaczonego na podstawie poprzednich odpowiedzi (np. komputer losował z przedziału $< 0, 20 >$, odpowiedź: *"za mało"*, następne losowanie będzie z przedziału $< 10, 20 >$). Gra toczy się do momentu odgadnięcia.

19. Korzystając z sita Eratostenesa napisz program, który sprawdzi, czy podana przez użytkownika liczba jest liczbą pierwszą.

20. Napisz program, który wyznaczy najmniejszą wspólną wielokrotność dwóch podanych przez użytkownika liczb.

21. Napisz program, który wyznaczy największy wspólny dzielnik dwóch podanych przez użytkownika liczb.

22. Napisz program, który wczyta liczbę naturalną n a następnie obliczy i wypisze sumę sześcianów $1^3 + 2^3 + ... + n^3$.

3 Funkcje

Do każdej z funkcji napisz program sprawdzający jej działanie.

1. Napisz funkcję `int maks(int x, int y, int z)` zwracającą wartość maksymalną spośród jej argumentów.

2. Napisz funkcję `int ktoraCwiartka(float x, float y)` która dla punktu o współrzędnych (x, y) zwróci wartość 1..4, identyfikującą jedną z ćwiartek układu współrzędnych wewnątrz której leży ten punkt. W przypadku, gdy punkt leży na którejkolwiek osi współrzędnych funkcja powinna zwrócić 0.

3. Napisz trzy warianty funkcji `void zamiana(int * a, int * b)` która zamieni wartości dwóch liczb wskazanych przez `a` i `b`:

   - zamienia bezwarunkowo,
   - zamienia tylko wtedy, gdy przynajmniej jeden z argumentów jest ujemny,
   - zamienia tylko wtedy, gdy jedna jest parzysta a druga nieparzysta.

4. Napisz cztery warianty funkcji `long int sum(int min, int max)`, która oblicza sumę liczb z $< min, max >$:

   - sumuje wszystkie liczby,
   - sumuje tylko liczby parzyste,
   - sumuje tylko liczby podzielne przez 3,
   - sumuje tylko takie, że suma aktualnej liczby i jej następnika jest podzielna przez 3.

5. Napisz funkcję `long int sumaAr(int n, int r)`, która wyznaczy sumę n elementów ciągu arytmetycznego o wyrazie początkowym równym 1 i zadanej różnicy `r`. Nie korzystaj z gotowego wzoru, tylko sumuj kolejne elementy ciągu.

6. Napisz funkcję `long int sumGeom(int n, int i)`, która wyznaczy sumę n elementów ciągu geometrycznego o wyrazie początkowym równym 1 i zadanym ilorazie `i`. Nie korzystaj z gotowego wzoru, tylko sumuj kolejne elementy ciągu.

7. Napisz funkcję `int wartoscMaks(int granica)` która znajdzie największą liczbę całkowitą n taką, że $1 + 2 + ... + n < granica$.

8. Napisz funkcję `int wartoscSilnia(int granica)` która znajdzie najmniejszą liczbę całkowitą taką, że $n! > granica$.

9. Napisz funkcję `long int suma(int n)`, która wyznaczy sumę szeregu: $1 - 2 + 3 - ... \pm n$.

10. Napisz funkcję `long int suma(int n)`, która wyznaczy sumę szeregu: $1! - 2! + 3! - ... \pm n!$.

11. Napisz funkcję `float suma(int n)`, która wyznaczy sumę szeregu: $1 - 3 + 5 - ... \pm (2n + 1)$.

12. Napisz funkcję `float suma(int n)`, która wyznaczy sumę szeregu: $1 - 1/4 + 1/7 - ... \pm 1/(3n + 1)$.

13. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $x - x2 + x3 - ... \pm xn$.

14. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $x + 2x2 + 3x3 + ... + nxn$.

15. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $1/x + 2/x^2 + 3/x^3 + ... + n/x^n$.

16. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $x/1! + x^2/2! + x^3/3! + ... + x^n/n!$.

17. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $(x + 1) + (x^2 + 2) + (x^3 + 3) + ... + (x^n + n)$.

18. Napisz funkcję `float suma(int n, float x)`, która wyznaczy sumę szeregu: $(x + 1) - (x^2 - 2) + (x^3 + 3) - ... \pm (x^n \pm n)$.

19. Napisz funkcję `float suma(float x, float epsilon)`, która dla `x` z przedziału $(0, 1)$ wyznaczy sumę szeregu: $1 + x/1 + x^2/2 + ... + x^i/i$. Sumowanie przerwij jeśli kolejny składnik będzie mniejszy od zadanej dokładności ε.

20. Napisz funkcję, która korzystając z sita Eratostenesa sprawdzi czy jej argument jest liczbą pierwszą. Sprawdź działanie tej funkcji pisząc program, wyświetli wszystkie liczby pierwsze z zadanego przez użytkownika przedziału.

21. Napisz funkcję sprawdzającą czy jej argument jest liczbą doskonałą. Liczba doskonała to taka, której suma dzielników jest równa tej liczbie (liczbami doskonałymi są $1 = 1, 6 = 1 + 2 + 3$). Sprawdź działanie tej

funkcji pisząc program wypisujący na ekran wszystkie liczby doskonałe z zadanego przez użytkownika przedziału.

22. Napisz funkcję sprawdzającą czy jej argument jest liczbą automorficzną. Liczba automorficzna to taka, która znajduje się na końcu swojego kwadratu (liczbami automorficznymi są $5, 6, 25$, bo $5^2 = 25, 6^2 = 36, 25^2 = 625$). Sprawdź działanie tej funkcji pisząc program wypisujący na ekran wszystkie liczby automorficzne z zadanego przez użytkownika przedziału.

4 Tablice jednowymiarowe o stałym rozmiarze

Do każdej z funkcji napisz program sprawdzający jej działanie.

1. Napisz program, wczytujący od użytkownika liczby do 10-elementowej tablicy liczb całkowitych a następnie wyświetlający zawartość tej tablicy na ekranie.

2. Napisz program, wczytujący losujący liczby do N-elementowej tablicy liczb całkowitych (zdefiniuj `N` za pomocą dyrektywy `#define`) a następnie wyświetlający zawartość tej tablicy na ekranie.

3. Napisz funkcję `void wypelnij1DU(int t1[N], int n)`, która będzie wypełniać przekazaną tablicę `t1` wartościami pobranymi od użytkownika. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

4. Napisz funkcję `void wypelnij1DL(int t1[N], int n)`, która będzie wypełniać przekazaną tablicę `t1` losowymi wartościami. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

5. Napisz funkcję `void wypelnij2DL2(int t1[N], int n, int min, int maks)` która będzie wypełniać przekazaną tablicę `t1` losowymi wartościami z przedziału $< min, maks >$. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

6. Napisz funkcję `void wypisz1D(int t1[N], int n)`, która wypisze przekazaną tablicę `t1` na ekranie. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

7. Napisz program, który po wczytaniu od użytkownika liczb do 10-elementowej tablicy liczb rzeczywistych wypisze ją na ekran a następnie obliczy i wyświetli na ekranie:

   (a) sumę wszystkich liczb,

   (b) sumę wszystkich liczb dodatnich,

   (c) sumę wszystkich liczb ujemnych,

   (d) sumę wszystkich liczb większych od `k` (`k` wczytaj od użytkownika),

   (e) sumę elementów o indeksach parzystych,

   (f) sumę elementów o indeksach nieparzystych.

   Każda wyliczona wartość powinna być wypisana w nowej linii i poprzedzona informacją co oznacza i jakie liczby brały udział w jej wyznaczaniu. Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.

8. Napisz program, który po wylosowaniu liczb do 10-elementowej tablicy liczb całkowitych obliczy i wyświetli na ekranie:

   (a) iloczyn wszystkich liczb,

   (b) iloczyn wszystkich liczb parzystych,

   (c) iloczyn wszystkich liczb nieparzystych,

   (d) iloczyn wszystkich liczb dodatnich,

   (e) iloczyn wszystkich liczb ujemnych,

   (f) iloczyn wszystkich liczb większych od `k` (`k` wczytaj od użytkownika).

   Każda wyliczona wartość powinna być wypisana w nowej linii i poprzedzona informacją co oznacza i jakie liczby brały udział w jej wyznaczaniu. Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.

9. Napisz program, który po wylosowaniu liczb z przedziału zadanego przez użytkownika do 10-elementowej tablicy liczb całkowitych obliczy i wyświetli na ekranie:

   (a) średnią arytmetyczną elementów parzystych,

   (b) średnią arytmetyczną elementów nieparzystych,

   (c) średnią arytmetyczną elementów nieujemnych,

   (d) średnią arytmetyczną elementów większych od `k` (`k` wczytaj od użytkownika).

Każda wyliczona wartość powinna być wypisana w nowej linii i poprzedzona informacją co oznacza i jakie liczby brały udział w jej wyznaczaniu. Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.

10. Napisz program, który po wylosowaniu liczb do N-elementowej tablicy liczb całkowitych (zdefiniuj `N` za pomocą dyrektywy `#define`) wyznaczy i wyświetli na ekranie:

    (a) największą wartość spośród elementów parzystych,

    (b) najmniejszą wartość spośród elementów nieparzystych,

    (c) średnią arytmetyczną elementów znajdujących się w przedziale $< a, b >$ (`a`, `b` wczytaj od użytkownika),

    (d) średnią arytmetyczną elementów podzielnych przez x znajdujących się w przedziale $< a, b >$ (`x`, `a`, `b` wczytaj od użytkownika).

    Każda wyliczona wartość powinna być wypisana w nowej linii i poprzedzona informacją co oznacza i jakie liczby brały udział w jej wyznaczaniu. Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.

11. Napisz funkcję `int min1D(int t[N], int n)`, która dla tablicy o wymiarze `N` danej jako parametr zwróci wartość elementu minimalnego. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

12. Napisz funkcję `int maks1D(int t[N], int n)`, która dla tablicy o wymiarze `N` danej jako parametr zwróci wartość elementu maksymalnego. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

13. Napisz program, który obliczy sumę dwóch tablic (wektorów) `N` elementowych. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy.

14. Napisz program, który sprawdzi, czy dwie wylosowane tablice `N` elementowe są takie same. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy.

15. Napisz program, który zliczy liczbę wystąpień elementów tablicy o rozmiarze `N`. Elementy tablicy losowane są z zadanego przez użytkownika przedziału. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy.

16. Napisz funkcję `int suma1D(int t[N], int n)`, która która dla tablicy o wymiarze `N` danej jako parametr zwróci sumę elementów. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

17. Napisz funkcję `float srednia1D(int t[N], int n)`, która która dla tablicy o wymiarze `N` danej jako parametr zwróci średnią arytmetyczną elementów. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

18. Napisz funkcję `float iloczynS(float t1[N], float t2[N], int n)`, obliczającą iloczyn skalarny dwóch wektorów `N` liczb, przekazanych jako jej parametry. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

19. Napisz funkcję `int ilePrzez3(int t1[N], int n)`, która sprawdzi ile razy w tablicy `N` liczb występują liczby podzielne przez trzy. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

20. Napisz funkcję `int ilePrzezX(int t1[N], int n, int X)`, która sprawdzi ile razy w tablicy `N` liczb występują liczby podzielne przez `X`. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

21. Napisz funkcję `int sumaIndP(int t1[N], int n)`, która wyznaczy sumę elementów o indeksach parzystych. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

22. Napisz funkcję `int sumaIndX(int t1[N], int n, int X)`, która wyznaczy sumę elementów o indeksach podzielnych przez `x`. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

23. Napisz funkcję `void przesun(int t[N], int n)`, która w tablicy `N` liczb danej jako parametr przesunie elementy cyklicznie o jedną pozycję "w dół". (tzn. pierwszy element na miejsce ostatniego, drugi na pierwszy, trzeci na drugi, ...). `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

24. Napisz funkcję `void zamien(int t[N], int n, int co, int naCo)`, która w tablicy `N` liczb przekazanej jako parametr zamieni wartości równe parametrowi `co` wartościami `naCo`. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

25. Napisz funkcję `void wypelnijB(int A[N], int B[N], int n)`, która na podstawie tablicy liczb całkowitych A wypełni tablicę B wg następującego wzoru: B[i] = 2*A[i]. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

26. Napisz funkcję `void wypelnijB2(int A[N], int B[N], int n)`, która na podstawie tablicy liczb całkowitych `A` wypełni tablicę `B` wg następującego wzoru: `B[i] = A[0]-A[1]+A[2]-...A[i]`. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

27. Napisz funkcję `void skaluj(int t[N], int n)`, która przeskaluje elementy tablicy `N` liczb danej jako parametr do przedziału $< 0, 1 >$. `N` jest zdefiniowane w programie za pomocą dyrektywy `#define` i oznacza maksymalny możliwy rozmiar tablicy. Parametr `n` to liczba elementów w tablicy ($n <= N$).

5 Tablice dwuwymiarowe o stałym rozmiarze

Do każdej z funkcji napisz program sprawdzający jej działanie.

1. Napisz program, wczytujący od użytkownika liczby do dwuwymiarowej tablicy liczb całkowitych o 3 wierszach i 4 kolumnach a następnie wyświetlający zawartość tej tablicy na ekranie.

2. Napisz program, wczytujący losujący liczby do dwuwymiarowej tablicy liczb całkowitych o `N` wierszach i `M` kolumnach (`N` i `M` zdefiniuj za pomocą dyrektywy `#define`) a następnie wyświetlający zawartość tej tablicy na ekranie.

3. Napisz funkcję `void wypelnij2DU(int t1[N][M], int n, int m)`, która będzie wypełniać przekazaną tablicę `t1` wartościami pobranymi od użytkownika. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

4. Napisz funkcję `void wypelnij2DL(int t1[N][M], int n, int m)`, która będzie wypełniać przekazaną tablicę `t1` losowymi wartościami. `N` i

`M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

5. Napisz funkcję `void wypelnij2DL2(int t1[N], int n, int m, int min, int maks)` która będzie wypełniać przekazaną tablicę `t1` losowymi wartościami z przedziału $< min, maks >$. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

6. Napisz funkcję `void wypisz2D(int t1[N][M], int n, int m)`, która wypisze przekazaną tablicę `t1` na ekranie. Każdy wiersz tablicy powinien być wypisany w oddzielnej linii, a elementy każdej kolumny powinny znaleźć się jeden pod drugim (z wyrównaniem do prawej strony). `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

7. Napisz funkcję `void wypelnij2D1(int t1[N][M], int n, int m)`, która wypełni tablicę `t1` daną kolejnymi liczbami od 1 do 5. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

8. Napisz funkcję `void wypelnij2D2(int t1[N][M], int n, int m, int a, int b)`, która wypełni tablicę `t1` kolejnymi liczbami z przedziału $< a, b >$. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

9. Napisz funkcję `void wypelnij2DLX(int t1[N][M], int n, int m)` (gdzie X jest kolejnym numerem wersji funkcji), która wypełni tablicę `t1` losowymi liczbami wg schematu:

   (a) w wierszach parzystych – liczby parzyste, w wierszach nieparzystych – nieparzyste,

   (b) w pierwszej kolumnie tablicy mogą znaleźć się liczby z przedziału $< 1;\ 10 >$, w drugim $< 11;\ 20 >$, etc.

   (c) w pierwszym wierszy tablicy mogą znaleźć się liczby z przedziału $< 1;\ 10 >$, w drugim wierszu z przedziału $<$ *min pierwszego wiersza*; *min pierwszego wiersza*$+10 >$, w trzecim z przedziału $<$ *min drugiego wiersza*; *min drugiego wiersza* $+ 10 >$, etc.

   `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

10. Napisz program wczytujący od użytkownika wymiary (`n` i `m`) dwuwymiarowej tablicy `t1[N][M]` i znajdujący w niej element minimalny spośród elementów:

    (a) całej tablicy,

    (b) głównej przekątnej,

    (c) każdego wiersza osobno,

    (d) każdej kolumny osobno.

    Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość -1.
    `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).
    Każda wyliczona wartość powinna być wypisana w nowej linii i poprzedzona informacją co oznacza i jakie liczby brały udział w jej wyznaczaniu. Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.
    Wskazówka: Wiersz lub kolumna tablicy dwywumiarowej może być traktowana jako tablica jednowymiarowa.

11. Napisz program, który wczyta od użytkownika wymiary (`n` i `m`) dwuwymiarowej tablicy `t[N][M]`, wypisze ją na ekran i sprawdzi czy jest ona symetryczna. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).
    Do wypełniania i wypisywania tablicy użyj funkcji zdefiniowanych w poprzednich zadaniach.

12. Napisz zestaw funkcji `double sredniaX(int t1[N][M], int n, int m)` (X jest kolejnym numerem wersji funkcji), które dla tablicy `t1` zawierającej nieujemne liczby całkowite obliczą średnią spośród elementów:

    (a) całej tablicy,

    (b) głównej przekątnej,

    (c) każdego wiersza osobno,

    (d) każdej kolumny osobno.

    Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość -1.
    `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

13. Napisz zestaw funkcji `int ileParzX(int t1[N][M], int n, int m)` (X jest numerem wersji funkcji), które dla tablicy `t1` zliczą ile jest elementów parzystych spośród elementów:

    (a) całej tablicy,

    (b) głównej przekątnej,

    (c) każdego wiersza osobno,

    (d) każdej kolumny osobno.

    Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość -1.
    `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

14. Napisz funkcję `int najwSrednia(int t1[N][M], int n, int m)`, która w tablicy `t1` znajdzie numer wiersza o największej średniej. Jeżeli ą dwa lub więcej wierszy o takiej średniej funkcja powinna zwrócić numer pierwszego z nich. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

15. Napisz funkcję `int zmienMniejsze(int t1[N][M], int n, int m)`, która w tablicy `t1` w każdej kolumnie zamieni elementy mniejsze od średniej tej kolumny. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

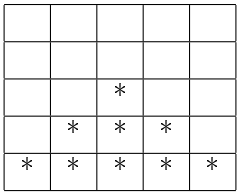
16. Napisz zestaw funkcji `int ileSumaX(int t1[N][M], int n, int m)` (X jest numerem wersji funkcji), które dla tablicy `t1` zawierającej nieujemne liczby całkowite obliczą sumę elementów leżących:

    (a) w górnej połowie tablicy,

    (b) w prawej górnej ćwiartce,

    (c) poniżej głównej przekątnej,

    (d) poniżej prawej i lewej przekątnej

    Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość -1.

`N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

17. Napisz funkcję `void wypelnij2DHank(int t1[N][M], int n, int m)`, która wypełni tablicę `t1` losowymi liczbami tak, aby była ona macierzą Hankela. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

18. Napisz funkcję `int sumaIlu(int t1[N][M], int n, int m, int k)`, która wyznaczy suma ilu pierwszych elementów tablicy `t1` danej jako parametr wyniesie `k`. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

19. Napisz funkcję `int transponuj(int t1[N][M], int n, int m)`, która dokona transpozycji macierzy `t1`. Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość 0. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

20. Napisz funkcję `int odbij(int t1[N][M], int n, int m)`, która odbije zawartość tablicy `t1` wokół jej głównej przekątnej. `N` i `M` są zdefiniowane w programie za pomocą dyrektywy `#define` i oznaczają maksymalny możliwy rozmiar tablicy. Parametry `n` i `m` to liczba elementów w tablicy ($n <= N$, $m <= M$).

6 Łańcuchy

Do każdej z funkcji napisz program sprawdzający jej działanie. Pamiętaj o zwolnieniu przydzielonej pamięci.

1. Napisz funkcję, która łańcuchu danym jako parametr zamieni wszystkie spacje na znaki podkreślenia. Łańcuch przekaż jako parametr funkcji.

2. Napisz funkcję, która obliczy liczbę wystąpień danego znaku w łańcuchu. Łańcuch oraz szukany znak przekaż jako parametry funkcji.

3. Napisz program, który sprawdza, czy dane słowo jest palindromem. Łańcuch wczytaj od użytkownika.

4. Napisz funkcję, która sprawdzi, czy dwa dowolne słowa są anagramami. Łańcuchy przekaż jako parametr funkcji.

5. Napisz funkcję, która sprawdzi czy k–elementowy ciąg znaków jest prefiksem drugiego ciągu n–elementowego. Obydwa ciągi znaków przekaż jako parametry funkcji.

6. Napisz funkcję, która sprawdzi czy k–elementowy ciąg znaków jest podciągiem drugiego ciągu n–elementowego. Obydwa ciągi znaków przekaż jako parametry funkcji.

7. Napisz funkcję, która usunie wszystkie spacje z danego łańcucha. Łańcuch znak przekaż jako parametr funkcji. Uwzględnij zmianę długości łańucha.

8. Napisz funkcję, która usunie co drugi znak w danym łańcuchu. Łańcuch przekaż jako parametr funkcji. Uwzględnij zmianę długości łańucha.
   Przykład:
   łańcuch wejściowy: `przykład`
   łańcuch po zmianie: `pzka`

9. Napisz funkcję, która wstawi spację co drugi znak w danym łańcuchu. Łańcuch znak przekaż jako parametr funkcji. Uwzględnij zmianę długości łańucha.
   Przykład:
   łańcuch wejściowy: `przykład`
   łańcuch po zmianie: `pr zy kł ad`

10. Napisz funkcję, która wypisze na ekranie znaki wprowadzonego łańucha w odwrotnej kolejności. Łańcuch przekaż jako parametr funkcji.

11. Napisz funkcję, która wyznaczy i zwróci liczbę słów w danym łańcuchu. Wyraz definiujemy jako ciąg liter oddzielonych białym znakiem lub znakiem przestankowym. Łańcuch przekaż jako parametr funkcji.
    Przykład:
    łańcuch wejściowy: `To jest przykład.`
    wynik: `3`

12. Napisz funkcję, która znajdzie i zwróci najdłuższy wyraz w danym łańcuchu. Wyraz definiujemy jako ciąg liter oddzielonych białym znakiem lub znakiem przestankowym. Łańcuch przekaż jako parametr funkcji.
    Przykład:
    łańcuch wejściowy: `To jest przykład.`
    wynik: `przykład.`

13. Napisz funkcję, która w danym łańcuchu podwoi wystąpienie każdego znaku. Łańcuch przekaż jako parametr funkcji. Uwzględnij zmianę długości łańucha.
    Przykład:
    łańcuch wejściowy: `kot`
    łańcuch po zmianach: `kkoott`

14. Napisz funkcję, która w danym łańcuchu usunie wszystkie wystąpienia danego podłańcucha. Łańcuch oraz szukany podłańcuch przekaż jako parametry funkcji. Uwzględnij zmianę długości łańucha.
    Przykład:
    łańcuch wejściowy: `caabcaadeea`
    podłańcuch: `aa`
    łańcuch wyjściowy: `cbcdeea`

15. Dane są dwie liczby całkowite `n` i `k`. Napisz funkcję, która usunie w danym łańcuchu co k-ty element n sąsiednich znaków. Łańcuch oraz wartości `n` i `k` przekaż jako parametry funkcji. Uwzględnij zmianę długości łańucha.
    Przykład:
    n=2
    k=3
    łańcuch wejściowy: `programowanie`
    łańcuch wyjściowy: `proamonie`

16. Napisz funkcję, która w danym łańcuchu zlikwiduje wszystkie wielokrotne wystąpienia tych samych znaków.
    Łańcuch przekaż jako parametr funkcji. Uwzględnij zmianę długości łańucha.
    Przykład:
    łańcuch wejściowy: `mmamma miiaa`
    łańcuch wyjściowy: `mama mia`.

17. Napisz fukcję, która zaszyfruje dany łańcuch. Szyfrowanie odbywa się na podstawie dwóch łańcuchów `s1` oraz `s2` i przebiega wg algorytmu: każdą literę w szyfrowanym łańcuchu należy w znaleźć w `s1`, odczytać jej pozycję a następnie zamienić ją na literę znajdującą się w łańcuchu `s2` na tej samej pozycji. Nieznalezione znaki zostawiamy bez zmian. Łańcuchy `s1` i `s2` oraz łańcuch do zaszyfrowania przekaż jako parametry funkcji.
    Przykład:
    s1: `bcdfghjklmnprstwz`
    s2: `zwtsrpnmlkjhgfdcb`
    szyfrowany łańuch: `ala ma kota`
    łańcuch po zaszyfrowaniu: `aoa ka moda`

18. Napisz funkcję, która w danym łańcuchu zamieni każde wystąpienie danego podłańcucha na `n` znaków. Łańuch, szukany podłańcuch, wartość `n` oraz znak przekaż jako parametry funkcji. Uwzględnij zmianę długości łańucha.
    Przykład:

łańcuch wejściowy: `caabcaadeea`
podłańcuch: `aa`
n: 3
znak: `*`
łańcuch wyjściowy: `c***bc***deea`

19. Napisz funkcję `void srev(char *dest, const char* src)`, która do łańcucha `dest` zapisze łańcuch `src` od tyłu.

20. Napisz własną wersję bibliotecznej funkcji
`int slen(const char* s);`

21. Napisz własną wersję bibliotecznej funkcji
`char* strcpy(char* s, const char* ct);`

22. Napisz własną wersję bibliotecznej funkcji
`char* strcat(char* s, const char* ct);`

23. Napisz własną wersję bibliotecznej funkcji
`int strcmp(const char* cs, const char* ct);`

24. Napisz własną wersję bibliotecznej funkcji
`char* strchr(const char* cs, int c);`

25. Napisz własną wersję bibliotecznej funkcji
`char* strrchr(const char* cs, int c);`

26. Napisz własną wersję bibliotecznej funkcji
`char* strdup(const char *s);`

27. Rozwiń biblioteczną wersję funkcji
`int strcmp(const char* cs, const char* ct);`
o możliwość porównywania z ignorowaniem wielkości liter.

7 Tablice tworzone dynamicznie

Do każdej z funkcji napisz program sprawdzający jej działanie. Pamiętaj o zwolnieniu przydzielonej pamięci.

1. Napisz funkcjke `void zmienIndeksy(int *t, int n`, która w jednowymiarowej tablicy `t` zamieni miejscami elementy o indeksach parzystych z elementami o indeksach nieparzystych. Parametr `n` to rozmiar tablicy.
Przykład:
przed zmianą: `1 5 2 7 9 3`
po zmianie: `5 1 7 2 3 9`

2. Napisz program, który wczyta znaki z klawiatury i zapisze je w dynamicznej tablicy znaków `s`. Koniec wczytywania jest sygnalizowany kropką.

3. Napisz program, który stworzy i wypisze dynamiczną tablicę dwuwymiarową reprezentującą tabliczkę mnożenia z zakresu od 0 do k, gdzie k jest wartością pobraną od użytkownika.

4. Napisz funkcję mnożącą dwie macierze dane jako parametry. Funkcja powinna zwrócić wskaźnik do macierzy wynikowej, a w przypadku gdy operacji nie da się wykonać - wartość NULL. Pamiętaj, że należy również wynieść poza funkcję wymiary nowej tablicy.

5. Napisz funkcję, która dokona transpozycji macierzy danej jako parametr. Pamiętaj, że operacja transpozycji wiąże się ze zmianą liczby wierszy i kolumn. Pamiętaj, że należy również wynieść poza funkcję wymiary nowej tablicy.

6. Napisz funkcję `int zamienWiersze(int **t, int n, int m, int w1, int w2)`, która w prostokątnej tablicy `t` zamieni miejscami wiersze `w1` i `w2`. Jeżeli wykonanie operacji jest niemożliwe funkcja powinna zwrócić wartość 0, w przeciwnym wypadku 1. Parametry `n` i `m` to liczba elementów w tablicy.
   Wykorzystaj fakt, że tablica dwuwymiarowa jest w rzeczywistości jednowymiarową tablicą wskaźników.
   Przykład:

   `w1`: `2`

   `w2`: `4`

   tablica przed zmianą:

1	3	5	2
3	9	5	9
8	2	1	6
7	7	4	3

   tablica po zmianie:

1	3	5	2
7	7	4	3
8	2	1	6
3	9	5	9

7. Dwuwymiarowa tablica nieregularna, to taka, w której długości poszczególnych wierszy różnią się między sobą. Napisz funkcję, która będzie odpowiednikiem funkcji z zadania 6 dla takiego rodzaju tablicy.
   Przykład:

`w1`: `2`

`w2`: `4`

tablica przed zmianą:

1	3	5	2
3	9	5	
8	2	1	
7	7		

tablica po zmianie:

1	3	5	2
7	7		
8	2	1	
3	9	5	

8. *Słowa Fibonacciego* to ciąg słów stosowany w informatyce teoretycznej mikedzy innymi do analizy złożoności algorytmów tekstowych. Ciąg ten opisany jest wzorem rekurencyjnym:

$$F_n = \begin{cases} a & n = 1 \\ b & n = 2 \\ F_{n-1} \cdot F_{n-2} & n > 2 \end{cases}$$

gdzie $\cdot$ oznacza konkatenację łańcuchów.
Napisz funkcję, która utworzy, wypełni i zwróci dynamiczną dwuwymiarową tablicę znaków, będącą ciągiem n słów Fibonacciego. Każde słowo stanowi pojedynczy wiersz tablicy. Liczba słów w ciągu jest parametrem funkcji.
Przykład:
Ciąg pięciu pierwszych słów wygląda następująco:

$$F_1 = b$$
$$F_2 = a$$
$$F_3 = ab$$
$$F_4 = aba$$
$$F_5 = abaab$$

9. Napisz funkcję, która utworzy, wypełni i zwróci dynamiczną dwuwymiarową tablicę, przechowującą kolejne wiersze *trójkąta Pascala*. Kolejnym wierszom trójkąta odpowiadają kolejne wartości n, k-ty wyraz w n-tym wierszu jest równy $\binom{n}{k} = \frac{n!}{(n-k)!k!}$. Wysokość trójkąta przekaż jako parametr funkcji.
Przykład:
Trójkąt Pascala dla $n = 5$

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

10. Napisz funkcję, która przekształci daną jako parametr dwuwymiarową tablicę przechowująca wartości *trójkąta Pascala* (zad. 9) w tablicę jednowymiarową. Pamiętaj, że należy również wynieść poza funkcję wymiar nowej tablicy.
    Przykład:
    tablica dwuwymiarowa:
    1
    1 1
    1 2 1
    1 3 3 1
    1 4 6 4 1
    tablica wynikowa: 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1

8 Typ strukturalny

Do każdej funkcji napisz program sprawdzający jej działanie. Pamiętaj o zwolnieniu przydzielonej pamięci.

1. Dana jest następująca struktura:

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

   Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` punktów. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

2. Korzystając z tablicy utworzonej w zadaniu 1 napisz funkcję, która wyświetli na ekran te pary punktów znajdujących się w tablicy, które są symetryczne względem osi OX lub OY. Wskaźnik do tablicy przekaż jako argument funkcji.

3. Dana jest następująca struktura:

   ```
   struct auto {
   double cena;
   int rocznik;
   double przebieg;
   char *marka;
   };
   ```

   Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` samochodów. Pamiętaj o sprawdzaniu poprawności danych: pola `cena`, `rocznik` oraz `przebieg` nie mogą być ujemne, pole

`marka` składa się wyłącznie z liter. Poszczególne wartości wczytuj do momentu, aż użytkownik poda poprawne dane. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

4. Napisz funkcję, która z tablicy utworzonej w zadaniu 3 wypisze na ekran dane o tych samochodach, które mają najniższy przebieg. Wskaźnik do tablicy przekaż jako argument funkcji.

5. Dana jest następująca struktura:

```
struct ksiazka{
  char *autor, *tytul;
  int rokWydania, liczbaStron;
  double cena;
  char wypozyczona;
};
```

Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` książek. Pamiętaj o sprawdzaniu poprawności danych: pole `autor` składa się wyłącznie z liter; pole `tytul` - z liter i cyfr; pola `cena`, `rokWydania` i `liczbaStron` to wartości większe od zera (przy czym `rokWydania` nie może być wyższy niż rok bieżący); pole `wypozyczona` przybiera tylko dwie wartości `'T'` albo `'N'`. Poszczególne wartości wczytuj do momentu, aż użytkownik poda poprawne dane. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

6. Napisz funkcję, która z tablicy utworzonej w zadaniu 5 wyświetli na ekranie wszystkie dane książek, które są tańsze niż wartość `x`. Wskaźnik do tablicy i wartość `x` przekaż jako parametry funkcji.

7. Napisz funkcję, która posortuje tablicę utworzoną w zadaniu 5 rosnąco ze względu na autora i tytuł. Wskaźnik do tablicy przekaż jako parametr funkcji.

8. Dana jest następująca struktura:

```
struct pracownik {
     char *nazwisko, *imie, *stanowisko;
  double pensja;
  char plec;
};
```

Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` pracowników. Pamiętaj o sprawdzaniu poprawności danych: pola `nazwisko`, `imie` i `stanowisko` składają się wyłącznie z liter; pole `pensja` to wartość większe od zera; pole `plec` przybiera tylko

dwie wartości `'K'` albo `'M'`. Poszczególne wartości wczytuj do momentu, aż użytkownik poda poprawne dane. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

9. Napisz funkcję, która w tablicy utworzonej w zadaniu 8 sprawdzi, czy w danej firmie na danym stanowisku większa jest średnia pensja kobiet czy mężczyzn. Wskaźnik do tablicy i nazwę stanowiska przekaż jako parametry funkcji.

10. Dana jest następująca struktura:

```
struct student {
  char *nazwisko, *imie;
  char indeks[10];
  int wiek;
  double stypendium;
  char zaliczenie;
};
```

Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` studentów. Pamiętaj o sprawdzaniu poprawności danych: pola `nazwisko` i `imię` składają się wyłącznie z liter; pole `indeks` składa się z dokładnie z 10 cyfr; pola `wiek` i `stypendium` to wartości większe od zera; pole `zaliczenie` przybiera tylko dwie wartości `'T'` albo `'N'` i mówi, czy student zaliczył semestr. Poszczególne wartości wczytuj do momentu, aż użytkownik poda poprawne dane. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

11. Napisz funkcję, która w tablicy utworzonej w zadaniu 10 wyszuka i zwróci dane studenta, który zaliczył semestr i ma najniższe stypendium. Wskaźnik do tablicy przekaż jako parametr funkcji.

12. Napisz funkcję, która z tablicy utworzonej w zadaniu 10 wyświetli na ekranie dane personalne wszystkich studentów, których nazwiska i imiona zaczynają się na tą samą literę. Wskaźnik do tablicy przekaż jako parametr funkcji.

13. Dana jest następująca struktura:

```
struct rozklad {
  char *skad, *dokad;
  time_t odjazd;
  double odleglosc;
  char pociag;
};
```

Napisz funkcję, która utworzy i wypełni danymi pobranymi od użytkownika tablicę `n` pozycji rozkładu jazdy pociągów. Pamiętaj o sprawdzaniu poprawności danych: pola `skad` i `dokad` składają się wyłącznie z liter, są to nazwy miast; pole `odleglosc` to wartość większa od zera; pole `pociag` oznacza typ pociągu, przybiera wartości `'O'`-osobowy, `'P'`-pośpieszny albo `'E'`-ekspes. Poszczególne wartości wczytuj do momentu, aż użytkownik poda poprawne dane. Wartość `n` oraz wskaźnik do tablicy przekaż jako parametry funkcji.

14. Napisz funkcję, która dla tablicy utworzonej w zadaniu 13 wyznaczy średnią długość trasy pociągów danego typu na określonej trasie. Wskaźnik do tablicy, typ pociągu oraz miasto początkowe i końcowe trasy przekaż jako parametry funkcji.

9 Pliki binarne

Do każdej funkcji napisz program sprawdzający jej działanie. Do części zadań będziesz musiał w pierwszej kolejności utworzyć potrzebne pliki elementowe.

1. Napisz funkcję, która zapisze do pliku elementowego `n` losowych liczb typu `double`. Ścieżkę dostępu do pliku, wartość `n` oraz krańce przedziału, z którego losowane będą liczby przekaż jako parametry funkcji.

2. Korzystając z pliku utworzonego w zadaniu 1 napisz funkcję, która zapisze te liczby do dynamicznej tablicy jednowymiarowej (najpierw określ ile liczb znajduje się w pliku) a następnie na ekranie wyświetli średnią arytmetyczną tych liczb. Odczyt liczb do tablicy wykonaj jedną instrukcją, nie wykorzystuj do tego pętli. Ścieżkę dostępu do pliku przekaż jako parametr funkcji. Pamiętaj o zwolnieniu przydzielonej pamięci.

3. Korzystając z pliku utworzonego w zadaniu 1 napisz funkcję, która zmieni znak każdej liczby w tym pliku na przeciwny. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

4. Korzystając z pliku utworzonego w zadaniu 1 napisz funkcję, która zmieni wartość n-tej liczby w tym pliku poprzez zwiększenie jej wartości o `x`. Ścieżkę dostępu do pliku oraz wartości `n` i `x` przekaż jako parametry funkcji.

5. Korzystając z pliku utworzonego w zadaniu 1 napisz funkcję, która obliczy sumy kolejnych par liczb i zapisze je do drugiego pliku. Ścieżki dostępu do plików przekaż jako parametr funkcji.

6. Dana jest następująca struktura:

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

Napisz program, który utworzy `n`-elementową tablicę punktów, a następnie zapisze ją do pliku elementowego. Zapis tablicy wykonaj jedną instrukcją, nie wykorzystuj do tego pętli. Wartość `n` oraz ścieżkę dostępu do pliku pobierz od użytkownika.

7. Korzystając z pliku utworzonego w zadaniu 6, napisz funkcję, która utworzy 4 pliki elementowe ("pierwsza.bin", "druga.bin", "trzecia.bin", "czwarta.bin") odpowiadające 4 ćwiartkom układu współrzędnych, a następnie do każdego z nich zapisze punkty, których współrzędne znajdują się w danej ćwiartce. Ścieżkę dostępu do pliku z punktami przekaż jako parametr funkcji.

8. Dana jest następująca struktura:

```
struct auto {
  double cena;
  int rocznik;
  double przebieg;
  char marka[20];
};
```

Napisz funkcję, która wczyta do tablicy struktur pobrane od użytkownika dane o `n` samochodach, a następnie zapisze je do pliku elementowego. Zapis tablicy wykonaj jedną instrukcją, nie wykorzystuj do tego pętli. Ścieżkę dostępu do pliku oraz wartość `n` przekaż jako parametry funkcji.

9. Korzystając z pliku utworzonego w zadaniu 8 napisz funkcję, która wypisze na ekran średnią cenę i najstarszy rocznik samochodów każdej marki. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

10. Korzystając z pliku utworzonego w zadaniu 8 napisz funkcję, która do drugiego pliku elementowego przepisze te samochody, których rocznik jest wyższy od najczęściej powtarzającego się rocznika spośród wszystkich samochodów. Ścieżki dostępu do obu plików przekaż jako parametry funkcji.

11. Korzystając z pliku utworzonego w zadaniu 8 napisz funkcję, która wczyta zawartość pliku do dynamicznej tablicy struktur, a następnie dowolnym algorytmem posortuje tą tablicę ze względu na markę, rocznik oraz przebieg samochodu. W przypadku, gdy występuje więcej niż jeden samochód o takiej samej marce, roczniku i przebiegu, to jako

pierwszy powinien znaleźć się ten tańszy. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

12. Dana jest następująca struktura:

```
struct ksiazka{
  char *autor, *tytu\l{};
  int rokWydania, liczbaStron;
  double cena;
  char wypozyczona;
};
```

Zapisz do pliku elementowego tablicę utworzoną w zadaniu 5 rozdziału *Typ strukturalny*. Zapis tablicy wykonaj jedną instrukcją, nie wykorzystuj do tego pętli. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

13. Korzystając z pliku utworzonego w zadaniu 12 napisz funkcję, która wczyta niewypożyczone książki do tablicy struktur. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

14. Korzystając z pliku utworzonego w zadaniu 12 napisz funkcję, która do drugiego pliku zapisze wypożyczone książki zadanego autora. Ścieżki dostępu do pliku przekaż jako parametry funkcji.

15. Dany jest plik utworzony przy pomocy następującej struktury:

```
struct prostokat {
  double a, b;
};
```

Napisz funkcję, która na podstawie tego pliku utworzy plik zawierający tylko kwadraty. Ścieżki dostępu do obu plików przekaż jako parametry funkcji.

10 Pliki tekstowe

Do każdej funkcji napisz program sprawdzający jej działanie.

1. Utwórz na dysku plik tekstowy zawierający umieszczone po spacji nazwy 10 miast. Przepisz elementy tego pliku do nowego pliku tak, aby każde miasto znalazło się w nowej linii. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

2. Dany jest plik z liczbami naturalnymi oddzielonymi spacją. Napisz funkcję, która wyznaczy, ile liczb parzystych znajduje się w tym pliku. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

3. Napisz funkcję, który z dwóch plików utworzy nowy plik, zawierający najpierw znaki pierwszego pliku, a potem drugiego. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

4. Napisz funkcję, która podaje statystykę wystąpień znaków w danym pliku. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

5. Napisz funkcję, która w danym pliku tekstowym zamieni wszystkie wystąpienia słowa *l1* na *l2*. Ścieżkę dostępu do pliku oraz łańcuchy `l1` i `l2` przekaż jako parametry funkcji.

6. Napisz funkcję, która z dwóch plików zawierających alfabetyczne listy nazwisk utworzy trzeci plik zawierający posortowaną listę wszytkich tych nazwisk. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

7. Napisz funkcję, która odwróci plik tekstowy `in.txt`. Wynikiem działania programu ma być drugi plik tekstowy `out.txt`. W pierwszym wierszu pliku wyjściowego znajdzie się ostatni wiersz pliku wejściowego, w drugim – przedostatni, itd. Dla uproszczenia możesz założyć, że maksymalna długość linii w pliku to 128 znaków.

8. Napisz funkcję, która zapisze do pliku wynikowego z pliku tekstowego zawierającego słowa oddzielone spacją lub przecinkiem wszystkie wyrazy zaczynając się wielką literą. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

9. Napisz funkcję, która w pliku tekstowym zawierającym słowa oddzielone spacją, kropką lub przecinkiem znajdzie wszystkie wyrazy zaczynające się i kończące tą samą literą i zapisze je do pliku wynikowego. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

10. Napisz funkcję, która z dwóch plików tekstowych utworzy trzeci plik, który zawiera wspólne łańcuchy tych plików. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

11. Napisz funkcję, która zapisze do pliku tekstowego kolejne liczby ciągu Fibonacciego, aż do wyczerpania zakresu liczb typu `unsigned long`. Każdą liczbę zapisz w oddzielnej linii. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

12. Napisz funkcję, która na podstawie pliku tekstowego liczb całkowitych stworzy dwa pliki: jeden z elementami parzystymi i drugi z elementami nieparzystymi. Ścieżki dostępu do plików przekaż jako parametr funkcji.

13. Napisz fukcję zliczającą i wypisującą na ekran liczbę wystąpień każdego ze znaków w dokumencie tekstowym o kodach 32-126. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

14. Napisz funkcję, która zwróci informację o tym, ile różnych liter wystąpiło w pliku o nazwie przekazanej jako parametr funkcji.

15. Napisz funkcję, która zwróci informację o tym, który znak wystąpił najczęściej w pliku o nazwie przekazanej jako parametr funkcji.

16. Napisz funkcję, która znajdzie liczbę znaków alfanumerycznych w wybranym pliku tekstowym. Nazwę pliku przekaż jako parametr funkcji.

17. Napisz funkcję, która wyznaczy liczbę linii w wybranym pliku tekstowym. Nazwę pliku przekaż jako parametr funkcji.

18. Napisz program, który obliczy sumę liczb zapisanych w pliku linia pod linią. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

19. Napisz funkcję, która sprawdzi, ile razy występuje w danym pliku łańcuch *s*. Ścieżę dostępu do pliku oraz poszukiwany łańcuch przekaż jako parametry funkcji.

20. Napisz funkcję, która zwróci numer wiersza i kolumny, od którego w pliku tekstowym zawiera się dany łańcuch. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

21. Napisz funkcję, która obliczy liczbę zdań w pliku tekstowym. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

22. Plik tekstowy zawiera pewną ilość linii. Napisz funkcję, która do drugiego pliku tekstowego zapisze długości poszczególnych linii, w takiej kolejności w jakiej wystepuja one w pliku wejściowym. Ścieżki dostępu do plików przekaż jako parametr funkcji.

23. Napisz fukcję zamieniającą wszystkie małe litery na wielkie w pliku tekstowym Funkcja ma zwrócić liczbę dokonanych zamian. Ścieżkę dostępu do pliku przekaż jako parametr funkcji.

24. W pliku tekstowym znajdują się informacje o rachunku z restauracji. Treść składa się zawsze z 3 wierszy. Każdy wiersz zawiera nazwę towaru, dwukropek, cenę w złotych. Nazwa towaru może się składać z dowolnej liczby słów oddzielonych odstępami, a słowa wyłącznie z liter. Napisz funkcję obliczającą całkowitą należność do zapłaty. Ścieżkę dostępu do pliku przekaż jako parametry funkcji.
    Przykładowy plik może wyglądać następująco:
    ```
    kanapka: 2.50
    szarlotka: 1.50
    woda mineralna: 1.50
    ```

25. Napisz funkcję, która porówna ze sobą dwa pliki tekstowe z dokładnością do białych znaków w obrębie linii. Funkcja zwraca numer linii, w

której nastąpiła pierwsza różnica, w przeciwnym wypadku wartość 0. Ścieżki dostępu do plików przekaż jako parametry funkcji.

26. Dane są dwa pliki tekstowe zawierające macierze liczb rzeczywistych. W pierwszym wierszu każdego pliku znajduje się informacja o liczbie wierszy i kolumn macierzy, a następnie w wierszach umieszczone są kolejne wartości. Napisz funkcję mnożącą te dwie macierze i zapisującą wynik do pliku wyjściowego w takiej samej postaci jak czynniki iloczynu. W przypadku, gdy operacji nie da się wykonać w pliku wyjściowym powinna znaleźć się tylko wartość 0. Ścieżki dostępu do plik'ow przekaż jako parametr funkcji.

11 Listy jednokierunkowe

Do każdej funkcji napisz program sprawdzający jej działanie. Do większości zadań będziesz musiał w pierwszej kolejności utworzyć potrzebne listy. Tam, gdzie tylko można korzystaj z wcześniej napisanych funkcji lub podziel treść zadania na funkcje pomocnicze i funkcję realizującą jego główną część. Pamiętaj również o usunięciu na końcu programu każdej stworzonej w nim listy.

1. Napisz funkcję wstawiającą element na początek jednokierunkowej listy liczb całkowitych. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

2. Napisz funkcję wstawiającą element na koniec jednokierunkowej listy liczb całkowitych. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

3. Napisz funkcję dodającą element do jednokierunkowej listy liczb całkowitych, tak aby była uporządkowana rosnąco. Zakładamy, że przed dodaniem każdego nowego elementu elementu lista jest uporządkowana. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

4. Napisz funkcję usuwającą wszystkie wystąpienia danego elementu z listy jednokierunkowej. Wskaźnik na początek listy przekaż jako parametr funkcji.

5. Napisz funkcję usuwającą wszystkie elementy listy począwszy od jej pierwszego elementu. Wskaźnik na początek listy przekaż jako parametr funkcji.

6. Napisz funkcję usuwającą wszystkie elementy listy począwszy od jej ostatniego elementu. Wskaźnik na początek listy przekaż jako parametr funkcji.

7. Napisz funkcję usuwającą pierwsze wystąpienie danego elementu na liście jednokierunkowej. Wskaźnik na początek listy przekaż jako parametr funkcji.

8. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, zwracającą wartość drugiego co do wielkości elementu tej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

9. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która zwróci średnią arytmetyczną z jej elementów. Wskaźnik na początek listy przekaż jako parametr funkcji.

10. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która zwróci wartość elementu minimalnego oraz elementu maksymalnego na tej liście. Wskaźnik na początek listy przekaż jako parametr funkcji.

11. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która wypisze na ekran elementy tej listy, które są niemniejsze niż średnia arytmetyczna wszystkich liczb. Wskaźnik na początek listy przekaż jako parametr funkcji.

12. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która sprawdzi czy jest ona posortowana. Wskaźnik na początek listy przekaż jako parametr funkcji.

13. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która sprawdzi ile spośród jej elementów mieści się w zadanym przedziale. Wskaźnik na początek listy przekaż jako parametr funkcji.

14. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która usunie z tej listy elementy, których wartość są mniejsze od średniej arytmetycznej wszystkich liczb. Wskaźnik na początek listy przekaż jako parametr funkcji.

15. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która usunie z tej listy każdy taki element, którego wartość jest równa elementowi poprzedniemu.

16. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która utworzy nową listę zawierającą co drugi element z listy źródłowej. Elementy na nowej liście muszą występować w takiej samej kolejności jak na liście źródłowej. Wskaźnik na początek listy przekaż jako parametr funkcji.

17. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która utworzy nową listę złożoną z tych elementów listy źródłowej, których wartość mieści się w zadanym przedziale. Wskaźnik na początek listy przekaż jako parametr funkcji.

18. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która utworzy i zwróci nową listę będącą lustrzanym odbiciem listy źródłowej. Wskaźnik na początek listy źródłowej przekaż jako parametr funkcji.

19. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która odwróci kolejność elementów danej listy jednokierunkowej i zwróci odwróconą listę. Nie korzystaj z pomocniczych typów (list, tablic). Wskaźnik na początek listy przekaż jako parametr funkcji.

20. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która zmieni kolejność elementów danej listy jednokierunkowej w taki sposób, że najpierw znajdą się liczby nieparzyste a potem parzyste. Wskaźnik na początek listy przekaż jako parametr funkcji.

21. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która posortuje daną listę algorytmem sortowania bąbelkowego. Podczas sortowania zamieniaj wartości elementów listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

22. Dana jest lista złożona z liczb całkowitych. Napisz funkcję, która posortuje daną listę algorytmem sortowania bąbelkowego. Podczas sortowanie zamieniaj wskaźniki na elementy listy a nie wartości. Wskaźnik na początek listy przekaż jako parametr funkcji.

23. Dana jest struktura:

    ```
    struct element {
      float a, b;
      struct element *nast;
    };
    ```

    Napisz funkcję zwracającą informację, którym z kolei na liście jest prostokąt o najkrótszym obwodzie. Jeżeli jest kilka prostokątów funkcja powinna zwrócić położenie pierwszego z nich. Wskaźnik na początek listy przekaż jako parametr funkcji.

24. Napisz funkcję, która wczyta na listę jednokierunkową zawartość pliku tekstowego, w taki sposób, by każde zdanie tego pliku stanowiło jeden element listy. Ścieżkę do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.

25. Zmodyfikuj funkcję z poprzedniego zadania (24) w taki sposób, aby każde przechowywane było na oddzielnej liście słów, a każdy element listy zawierającej zawartość pliku przechowywał wskaźnik na początek danego zdania. ścieżkę do pliku oraz wskaźnik na początek listy zdań przekaż jako parametry funkcji.
    Możesz wykorzystać następujące struktury:

```
struct listaSlow {
  char *slowo;
  struct listaSlow *nast;
};
struct listaZdanie {
  struct listaSlow *zdanie;
  struct listaZdan *nast;
};
```

26. Przechowywanie treści pliku na liście (zadania 24, 25) może wiązać się z wielokrotnym przechowywaniem jednego słowa w pamięci. Na podstawie pliku utwórz listę niepowtarzających się słów `listaSlow`, a następnie stwórz listę zdań, której elementami będą wskaźniki do elementów z listy słów.

27. Dana jest lista jednokierunkowa utworzona przy pomocy następującej struktury:

```
struct listaSlow {
  char *slowo;
  struct listaSlow *nast;
};
```

Napisz funkcję, która na jej podstawie utworzy nową listę, której elementami będzie słowo z listy źródłowej oraz liczba rzeczywista z przedziału $[0;1]$ określająca stopień podobieństwa tego słowa do wyrazu danego jako parametr funkcji. Podobie nstwo dwóch słów wyznaczamy jako iloraz liczby liter występujących w obydwu tych słowach do średniej arytmetycznej ich długości. Wskaźniki na poczatek list przekaż jako parametry funkcji.
Przyklad:

słowo1:	kot	kot	kot
słowo2:	tok	krokiew	pies
podobieństwo:	$\frac{3}{(3+3)/2} = 1$	$\frac{2}{(3+7)/2} = 1$	$\frac{0}{(3+4)/2} = 0$

28. Dany jest plik elementowy utworzony przy pomocy następującej struktury:

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

Napisz funkcję, która wczyta na listę jednokierunkową wszystkie punkty w takiej kolejności, w jakiej są umieszczone w pliku. Ścieżkę dostępu do pliku oraz wskaźnik na poczatek listy przekaż jako parametry funkcji.

29. Dana jest lista jednokierunkowa, która przechowuje elementy następującej struktury:

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

Napisz funkcję, która z listy jednokierunkowej zapisze do pliku tekstowego te punkty, które znajdują się wewnątrz okręgu o zadanym środku i promieniu. Ścieżkę dostępu do pliku, współędne środka i promień okręgu oraz wskaźnik na poczatek listy przekaż jako parametry funkcji.

30. Dane są dwie listy jednokierunkowe zawierające liczby całkowite. Napisz funkcję, która scali te listy w jedną, w taki sposób, by liczby posortowane były rosnąco. Wskaźniki na początek wszystkich list przekaż jako parametry funkcji.

31. W pliku tekstowym znajdują się informacje o rachunkach z restauracji. Każdy rachunek zawiera maksymalnie 3 pozycje (ponumerowane od 1 do 3). Każdy wiersz rachunku zawiera nazwę towaru, dwukropek oraz cenę. Nazwa towaru może składać się z dowolnej liczby słów oddzielonych spacją, a słowa wyłącznie z liter. Napisz funkcję, która wczyta zawartość pliku na listę w taki sposób, aby każdy rachunek stanowił oddzielny element listy. Ścieżkę dostępu do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.
Przykładowy plik może wyglądać następująco:

```
1. kanapka: 4.50
2. szarlotka: 6.50
3. woda mineralna: 3.50 1. zupa: 2.20
2. sok: 3.00
1. kawa: 5.00
1. woda mineralna: 3.50
```

32. Napisz funkcję, która bazując na liście z poprzedniego zadania (31), utworzy nową listę poszerzoną o łączną wartość każdego rachunku. Wskaźniki na początek obu list przekaż jako parametry funkcji.

33. Dana jest dwuwymiarowa tablica liczb rzeczywistych Napisz funkcję, która wczyta na listę wartości z tej tablicy w następującej kolejności:

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Tablicę oraz wskaźnik na początek listy przekaż jako parametry funkcji.

34. Dany jest plik tekstowy zawierający ciągi liter. Napisz funkcję, która utworzy listę zawierającą alfabetyczny skorowidz wystąpień wszystkich słów w danym tekście, wraz z liczbą wystąpień danego słowa (przez słowo rozumiemy każdy ciąg znaków niezawierający białych znaków. Ścieżkę dostępu do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.
    Przykład:
    plik: `pies awokado krowa kot krowa pies pies awokado`
    lista:
    ```
    awokado 2
    kot 1
    krowa 2
    pies 3
    ```

35. Dana jest jednokierunkowa lista złożona z liczb całkowitych. Napisz funkcję, która przesunie na tej liście wszystkie elementy o jedną pozycję do przodu (a ostatnią umieści na początki listy). Nie używaj do tego żadnych dodatkowych list ani tablic. Wskaźnik na początek listy przekaż jako parametr funkcji.

12 Listy dwukierunkowe

Do każdej funkcji napisz program sprawdzający jej działanie. Do większości zadań będziesz musiał w pierwszej kolejności utworzyć potrzebne listy. Tam, gdzie tylko można korzystaj z wcześniej napisanych funkcji lub podziel treść zadania na funkcje pomocnicze i funkcję realizującą jego główną część. Pamiętaj również o usunięciu na końcu programu każdej stworzonej w nim listy.

1. Napisz funkcję wstawiającą element na początek dwukierunkowej listy liczb całkowitych. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

2. Napisz funkcję wstawiającą element na koniec dwukierunkowej listy liczb całkowitych. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

3. Napisać funkcję dodającą element do dwukierunkowej listy liczb całkowitych, tak aby była uporządkowana rosnąco. Zakładamy, że przed dodaniem każdego nowego elementu elementu lista jest uporządkowana. Uwzględnij przypadek wstawiania do pustej listy. Wskaźnik na początek listy przekaż jako parametr funkcji.

4. Napisz funkcję usuwającą wszystkie elementy dwukierunkowej listy liczb całkowitych począwszy od jej pierwszego elementu. Wskaźnik na początek listy przekaż jako parametr funkcji.

5. Napisz funkcję usuwającą wszystkie elementy dwukierunkowej listy liczb całkowitych począwszy od jej ostatniego elementu. Wskaźnik na początek listy przekaż jako parametr funkcji.

6. Napisz funkcję usuwającą pierwsze wystąpienie danego elementu na dwukierunkowej liście liczb całkowitych. Wskaźnik na początek listy przekaż jako parametr funkcji.

7. Napisz funkcję usuwającą wszystkie wystąpienia danego elementu z dwukierunkowej listy liczb całkowitych. Wskaźnik na początek listy przekaż jako parametr funkcji.

8. Dana jest dwukierunkowa lista znaków (każdy element listy zawiera jedną literę). Napisz funkcję, która sprawdzi czy ta lista jest symetryczna (palindrom). Wskaźnik na początek listy przekaż jako parametr funkcji.

9. Dane są dwie dwukierunkowe listy znaków (każdy element listy zawiera jedną literę): `s1` i `s2`. Napisz funkcję, która sprawdzi, czy zawartość listy `s2` można traktować jako anagram elementów listy `s1`. Wskaźniki na początek list przekaż jako parametry funkcji.

10. Dana jest dwukierunkowa lista złożona z liczb całkowitych. Napisz funkcję, która zapisze do pliku tekstowego informacje w następującej postaci: *wartość elementu - liczba elemenów mniejszych od niego*.
    Każda wartość może pojawić się w pliku tylko raz. Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Ścieżkę dostępu do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.
    Przykład:

    ```
    lista:  1 3 5 4 9 3 9 8 4
    plik:   1 - 0
            3 - 1
            5 - 5
            4 - 3
            9 - 7
            8 - 6
    ```

11. Dana jest dwukierunkowa lista złożona z liczb całkowitych. Napisz funkcję, która posortuje daną listę dowolnym znanym Ci algorytmem sortowania (poza sortowaniem bąbelkowym). Podczas sortowania zamieniaj wskaźniki na elementy listy a nie wartości. Wskaźnik na początek listy przekaż jako parametr funkcji.

12. Dany jest plik elementowy utworzony przy pomocy następującej struktury:

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

Napisz funkcję, która wczyta na listę dwukierunkową wszystkie punkty w takiej kolejności, by były one posortowane względem odległości od początku układu współrzędnych. W przypadku, gdy dwa (lub więcej) punkty znajdują się w tej samej odległości, powinny być posortowane rosnąco najpierw względem współrzędnej `x`, a potem rosnąco względem współrzędnej `y`. Ścieżkę dostępu do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.

13. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

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

Napisz funkcję, która znajdzie i wypisze na ekranie współrzędne takich dwóch punktów, między którymi odległość jest maksymalna. Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Wskaźnik na początek listy przekaż jako parametr funkcji.

14. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

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

Napisz funkcję, która znajdzie i wypisze na ekranie współrzędne tych punktów, które mają na liście swój odpowiednik symetryczny względem początku układu współrzędnych. Nie wypisuj dwa razy tej samej pary. Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Wskaźnik na początek listy przekaż jako parametr funkcji.

15. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

```
struct prostokat {
  double a, b;
};
```

Napisz funkcję, która zwróci informacje o tym, ile pól prostokątów na liście jest unikalnych. Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Wskaźnik na początek listy przekaż jako parametr funkcji.

16. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

```
struct prostokat {
  double a, b;
};
```

Napisz funkcję, która zwróci najczęściej powtarzające się pole prostokąta na liście. Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Wskaźnik na początek listy przekaż jako parametr funkcji.

17. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

```
struct prostokat {
  double a, b;
};
```

Napisz funkcję, która zapisze do pliku tekstowego informacje w następującej postaci: *pole protokąta - liczba wystąpień na liście*.
Wykorzystaj właściwość listy dwukierunkowej, która umożliwia przeszukanie listy w obydwu kierunkach; nie wprowadzaj dodatkowych struktur. Ścieżkę dostępu do pliku oraz wskaźnik na początek listy przekaż jako parametry funkcji.

18. Dana jest lista dwukierunkowa, która przechowuje elementy następującej struktury:

```
struct student{
  char *nazwisko, *imie;
  int rokUr, miesUr, dzienUr;
  int ilePrzedmiotow;
  double *oceny;
};
```

Pole `ilePrzedmiotow` informuje o liczbie przedmiotów, z których student uzyskał ocenę. Tablica `oceny` zawiera wszystkie oceny otrzymane

przez studenta; jej rozmiar jest równy `ilePrzedmiotow`. Napisz funkcję, która posortuje tą listę w kolejności alfabetycznej względem nazwisk. W przypadku, gdy nazwiska studentów są takie same, należy posortować ich alfabetycznie względem imienia. Jeżeli nazwiska i imiona studentów są takie same, to pierwszy na liście powinien znaleźć się starszy student. Wskaźnik na początek listy przekaż jako parametr funkcji.

19. Korzystając z listy z zadania 18 napisz funkcję, która zwróci dane studenta z najwyższą średnią ocen. W przypadku, gdy kilku studentów uzyskało taką samą średnią, należy wybrać tego, który otrzymywał lepsze oceny. Wskaźnik na początek listy przekaż jako parametr funkcji.