DO NOWEJ PODSTAWY PROGRAMOWEJ

Część 2

PODRĘCZNIK dla szkół ponadpodstawowych

Informatyka Europejczyka

Zakres podstawowy

Danuta Korman,
Grażyna Szabłowicz-Zawadzka

Helion EDUKACJA

Spis treści

Rozdział 1.

ROZUMIENIE I ANALIZOWANIE PROBLEMÓW. WPROWADZENIE DO PROGRAMOWANIA W JĘZYKU PYTHON

Co już potrafisz?

Podajesz i wykorzystujesz **etapy rozwiązywania problemów za pomocą komputera**.

Potrafisz przedstawiać algorytmy w postaci **schematów blokowych i list kroków**.

Konstruujesz proste **algorytmy iteracyjne i rekurencyjne**.

Znasz podstawy **tekstowego języka programowania Python** oraz korzystasz ze **środowiska programowania** dla tego języka.

Sprawdzasz poprawność algorytmów poprzez testowanie dla przykładowych danych.

Czego się nauczysz?

Poznasz i będziesz stosować proste **algorytmy na liczbach, ciągach liczbowych i tekstach**.

Nauczysz się wykorzystywać **sekwencyjne typy danych w języku Python**: listy i napisy.

Dowiesz się, czym jest **kryptografia i kryptoanaliza**. Poznasz proste **metody przestawieniowe i podstawieniowe** stosowane w kryptografii.

Co pojawi się na poziomie rozszerzonym?

Poszerzysz znajomość **tekstowego języka programowania dopuszczonego na egzaminie maturalnym** z informatyki rozszerzonej. Może to być również język Python, którego podstawy poznajesz na lekcjach informatyki na poziomie podstawowym.

Będziesz omawiać oraz stosować **zaawansowane algorytmy iteracyjne** i **rekurencyjne** na liczbach, ciągach liczbowych i tekstach.

Zajmiesz się **analizą efektywności omawianych algorytmów**.

Zapoznasz się z **zadaniami z egzaminu maturalnego z informatyki** i będziesz je rozwiązywać.

Temat 1. Badanie, czy dana liczba jest liczbą pierwszą

Definicja

Liczbę naturalną n większą od 1 nazywamy **liczbą pierwszą**, jeśli ma tylko dwa dzielniki: 1 i n. Liczbę naturalną większą od 1, która nie jest liczbą pierwszą, nazywamy **liczbą złożoną**. Natomiast liczby 0 i 1 nie są ani liczbami złożonymi, ani pierwszymi.

Najprostszym algorytmem określającym, czy liczba n to liczba pierwsza, jest sprawdzenie, czy ma ona więcej niż dwa dzielniki. Należy więc zbadać, czy w przedziale $[2, n-1]$ znajduje się co najmniej jedna wartość całkowita, przez którą dzieli się liczba n.

Ćwiczenie 1.1.

Przeanalizuj podany kod programu i odpowiedz na pytania:

a) Jaki komunikat pojawi się na ekranie po uruchomieniu przedstawionego programu dla wpisanej z klawiatury liczby n o wartości 12, a jaki dla liczby n o wartości 16?

b) Co jest efektem działania podanego algorytmu dla dowolnej liczby naturalnej n?

```
n = int(input("podaj liczbę: "))
i = 1
while i * i < n:
    if n % i == 0:
        print(i, n // i)
    i += 1
if i * i == n:
    print(i)
```

Skonstruujmy **algorytm sprawdzający, czy liczba naturalna *n* jest liczbą pierwszą**. W tym celu sprawdzimy, czy w przedziale $[2, n-1]$ znajduje się co najmniej jedna wartość całkowita, przez którą dzieli się liczba n.

Specyfikacja:

Dane: liczba naturalna: $n > 1$.

Wynik: komunikat informujący, czy liczba n jest liczbą pierwszą.

Program w języku Python:

```
def liczba_pierwsza(n):
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

print(liczba_pierwsza(61))
```

Wynik:

True

Przedstawiona funkcja jest typu logicznego. Jeśli po jej wykonaniu otrzymamy wartość `True`, liczba n jest liczbą pierwszą. W przeciwnym razie n jest liczbą złożoną.

Spróbujmy zmodyfikować powyższy algorytm w taki sposób, aby poprawić jego złożoność obliczeniową. W obecnej postaci złożoność tej metody jest liniowa. Wynika to stąd, że liczba operacji dominujących, czyli porównań, wynosi $n-2$.

Zauważ, że nie ma konieczności sprawdzania wszystkich liczb z przedziału $[2, n-1]$. Załóżmy, że istnieje liczba x większa niż $\sqrt{n}$, która jest dzielnikiem liczby n. Wynika stąd, że musi istnieć liczba y, będąca również dzielnikiem liczby n, taka, że $n = x \times y$. Liczba y musiałaby jednak być mniejsza od $\sqrt{n}$, a to oznacza, że zostałaby znaleziona już w zakresie $[2, \sqrt{n}]$. W rzeczywistości wystarczy więc sprawdzenie, czy w zakresie $[2, \sqrt{n}]$ znajduje się wartość, która jest dzielnikiem liczby n.

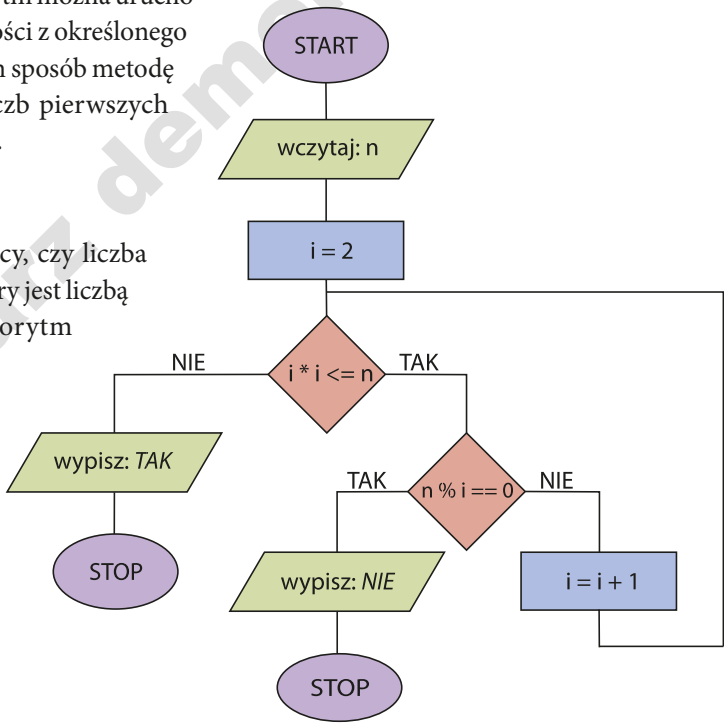
Na rysunku 1.1 został przedstawiony **schemat blokowy zmodyfikowanego algorytmu sprawdzającego, czy dana liczba jest liczbą pierwszą**.

Liczba operacji dominujących, czyli porównań, w tym algorytmie jest tym większa, im później zostaje znaleziona liczba będąca dzielnikiem n. Najwięcej porównań zostanie wykonanych w sytuacji, gdy n będzie liczbą pierwszą.

Funkcję realizującą ten algorytm można uruchomić w pętli generującej wartości z określonego przedziału. Uzyskujemy w ten sposób metodę wyznaczania wszystkich liczb pierwszych z podanego przedziału liczb.

Ćwiczenie 1.2.

Napisz program sprawdzający, czy liczba naturalna wpisana z klawiatury jest liczbą pierwszą. Skonstruuj algorytm zgodny ze schematem blokowym przedstawionym na rysunku 1.1.



Rysunek 1.1. Schemat blokowy algorytmu sprawdzającego, czy dana liczba jest liczbą pierwszą

Zadanie 1.1.

Napisz program, który dla liczby naturalnej *n* wpisanej z klawiatury wykonuje następujące operacje:

a) wypisuje wszystkie dzielniki naturalne liczby *n*,

b) wypisuje wszystkie dzielniki pierwsze liczby *n*.

Zadanie 1.2.

Napisz program sprawdzający, czy wpisana z klawiatury liczba naturalna jest podzielna przez 3. Wykonaj to zadanie bez wykorzystania operatora reszty z dzielenia. Zauważ, że gdy zsumujesz cyfry wczytanej liczby, potem cyfry powstałej sumy i będziesz powtarzać to sumowanie, aż uzyskasz jedną cyfrę, otrzymasz odpowiedź. Jeśli cyfrą, którą uzyskasz, będzie 3, 6 lub 9, to wczytana liczba jest podzielna przez 3.

Na przykład dla liczby 32 415 należy obliczyć sumę 3 + 2 + 4 + 1 + 5 = 15, a następnie 1 + 5 = 6. Wynikiem jest cyfra 6, więc liczba 32 415 jest podzielna przez 3.

Zadanie 1.3.

Parą **liczb bliźniaczych** nazywamy dwie liczby pierwsze różniące się o 2. Przykładem liczb bliźniaczych są liczby 11 i 13, ponieważ obie są liczbami pierwszymi i różnica pomiędzy nimi wynosi 2. Przykładem liczb, które nie są bliźniacze, jest para liczb 15 i 17, ponieważ 15 jest liczbą złożoną.

a) Podaj specyfikację zadania i napisz program, który sprawdza, czy dwie liczby naturalne wpisane z klawiatury są liczbami bliźniaczymi.

b) Napisz program generujący wszystkie pary liczb bliźniaczych, które są nie większe od wpisanej z klawiatury liczby *k*.

Temat 2. Sekwencyjne typy danych — listy

Definicja

Lista to typ sekwencyjny zmienny, a więc możliwe jest **przypisywanie wartości pojedynczym elementom** tego typu. Do zapisu listy wykorzystujemy **nawiasy kwadratowe**, a poszczególne elementy rozdzielamy przecinkami. Listy **mogą zawierać wartości różnego typu**. Każdy element listy ma przyporządkowany **indeks**. Elementy listy **są numerowane od zera**.

Deklaracja listy ma następującą składnię:

```
lista = [wartości listy]
```

Przykład 2.1.

Przyjrzyjmy się przykładowej deklaracji listy:

```
T = [1.0, 2, "trzy", 4.0, 5]
```

Zadeklarowano 5-elementową listę zawierającą wartości różnego typu, które są ponumerowane od 0 do 4. Do elementów listy odwołujemy się poprzez ich indeksy. Uzyskaliśmy więc dostęp do następujących zmiennych: `T[0]`, `T[1]`, `T[2]`, `T[3]`, `T[4]`. Na przykład `T[0]` to odwołanie do pierwszego elementu listy o wartości 1,0, a `T[2]` to odwołanie do elementu listy o wartości „trzy".

W języku Python możemy stosować również indeksy ujemne, które umożliwiają odwoływanie się do poszczególnych elementów począwszy od ostatniego. Na przykład `T[-1]` to odwołanie do elementu o wartości 5, a `T[-4]` to odwołanie do elementu o wartości 2.

Przykład 2.2.

Przeanalizuj odwołania do elementów przykładowej listy podanej poniżej.

```
T = [1, 4, 0, 3, 2]
```

Uzyskujemy 5-elementową listę zawierającą liczby całkowite o wartościach:

`T[0]` = 1; `T[1]` = 4; `T[2]` = 0; `T[3]` = 3; `T[4]` = 2.

Dostęp do elementów listy można uzyskać również w sposób następujący:

`T[-1]` = 2; `T[-2]` = 3; `T[-3]` = 0; `T[-4]` = 4; `T[-5]` = 1.

Możesz wypisać na ekranie zarówno całą listę, `print(T)`, jak i pojedyncze elementy tej listy, na przykład `print(T[2])`.

Ćwiczenie 2.1.

Napisz program wypisujący na ekranie trzy elementy listy, których numery wprowadzane są z klawiatury.

Przykład 2.3.

W języku Python mamy dostęp do funkcji `len()`, której wartością jest liczba elementów listy. Na przykład dla listy `T = [1, 4, 0, 3, 2]` funkcja `len(T)` zwraca wartość 5, co jest równe liczbie elementów tej listy. Przeanalizuj przykład zastosowania tej metody do wypisywania wszystkich elementów listy.

```
T = [1, 4, 0, 3, 2]
print(T)
for i in range(len(T)):
    print(T[i], end='  ')
```

Po uruchomieniu tego programu pojawiają się następujące wyniki:

[1, 4, 0, 3, 2]

1 4 0 3 2

Przykład 2.4.

W języku Python elementy listy możemy wprowadzać z klawiatury. Wykorzystujemy do tego metodę `append()`, co przedstawiono poniżej:

```
T = []
n = int(input("podaj liczbę elementów listy: "))
for i in range(n):
    T.append(int(input()))
print("lista:", T)
```

Po uruchomieniu tego programu pojawiają się następujące wyniki:

podaj liczbę elementów listy: 6
9
7
8
6
7
5
lista: [9, 7, 8, 6, 7, 5]

Ćwiczenie 2.2.

Napisz program, który wypisze na ekranie wszystkie elementy listy wprowadzonej z klawiatury w odwrotnej kolejności.

Przykład 2.5.

Przeanalizuj kody programów wykonujących następujące operacje:

a) obliczenie sumy wszystkich elementów listy,

b) obliczenie iloczynu tych elementów listy, które są mniejsze od 6,

c) obliczenie liczby tych elementów listy, których numer nie jest podzielny przez 5,

d) wyzerowanie tych elementów listy, które mają nieparzysty numer zawarty w przedziale [3, 7], i wyświetlenie zmodyfikowanej listy.

Rozwiązanie 1.:

```
T = [3, 4, 5, 5, 7, 9, 4, 2, 1]

# zad_a
s = 0
for i in range(len(T)):
    s += T[i]
print("suma =", s)
```

```
# zad_b
s = 1
for i in range(len(T)):
    if T[i] < 6:
        s *= T[i]
print("iloczyn =", s)

# zad_c
s = 0
for i in range(len(T)):
    if i % 5 != 0:
        s += 1
print("liczba elementów =", s)

# zad_d
for i in range(3, 8, 2):
    T[i] = 0
print("wyzerowana lista =", T)
```

Wyniki:

suma = 40

iloczyn = 2400

liczba elementów = 7

wyzerowana lista = [3, 4, 5, 0, 7, 0, 4, 0, 1]

Rozwiązanie 2.:

```
T = [3, 4, 5, 5, 7, 9, 4, 2, 1]

# zad_a
s = 0
for k in T:
    s += k
print("suma =", s)

# zad_b
s = 1
for k in T:
    if k < 6:
        s *= k
print("iloczyn =", s)
```

Wyniki:

suma = 40

iloczyn = 2400

Zwróć uwagę na dwie propozycje rozwiązania dla podpunktów a) i b) tego zadania. Dlaczego nie można w ten sposób rozwiązać pozostałej części zadania?

Zadanie 2.1.

Napisz program wykonujący następujące operacje na liście zawierającej liczby całkowite:

a) wypisanie wszystkich elementów listy,

b) obliczenie liczby tych elementów listy, których numer jest parzysty,

c) zwiększenie o 2 wartości tych elementów listy, których wartość jest mniejsza od 5,

d) obliczenie iloczynu tych elementów listy, których wartość jest równa 3,

e) obliczenie sumy tych elementów listy, których numer jest podzielny przez 3.

Zadanie 2.2.

Napisz program wykonujący następujące operacje na liście zawierającej liczby całkowite:

a) obliczenie liczby elementów listy równych wartości wczytanej z klawiatury,

b) obliczenie średniej arytmetycznej wszystkich elementów listy,

c) obliczenie średniej arytmetycznej tych elementów tablicy, których wartość jest nieparzysta.

Przykład 2.6.

W tabeli 2.1 podano dodatkowe informacje na temat list i ich indeksowania. Na podstawie przykładów pokazanych w tej tabeli przeanalizuj sposób indeksowania list i porównaj go z zasadami generowania ciągu liczbowego za pomocą metody `range()`. Czy zauważasz podobieństwa? Co oznaczają kolejne wartości oddzielone dwukropkami w indeksie listy?

Tabela 2.1. Indeksowanie list

Polecenie	Wyniki
`lista = [9, 2, 3, 4, 5, 6, 7, 8, 1, 10]`	
`print(lista)`	*[9, 2, 3, 4, 5, 6, 7, 8, 1, 10]*
`print(lista[2:])`	*[3, 4, 5, 6, 7, 8, 1, 10]*
`print(lista[3::2])`	*[4, 6, 8, 10]*
`print(lista[::-1])`	*[10, 1, 8, 7, 6, 5, 4, 3, 2, 9]*
`print(lista[::-2])`	*[10, 8, 6, 4, 2]*
`print(lista[::3])`	*[9, 4, 7, 10]*
`print(lista[:4:-2])`	*[10, 8, 6]*
`print(lista[:4:2])`	*[9, 3]*
`print(lista[0:5:2])`	*[9, 3, 5]*
`print(lista[3:9:2])`	*[4, 6, 8]*

Ćwiczenie 2.3.

Na podstawie przykładu 2.6 napisz program, w którym przetestujesz różne możliwości indeksowania listy wpisanej z klawiatury.

W tabeli 2.2 podano wybrane dodatkowe metody stosowane na listach, które możesz wykorzystywać w programach.

Tabela 2.2. Wybrane metody stosowane na listach

Polecenie	Opis polecenia i wyniki
`lista = [9, 2, 3, 4, 5, 6, 7, 8, 1, 10]`	
`lista.append(11)`	dołączanie elementu do listy, wynik: *[9, 2, 3, 4, 5, 6, 7, 8, 1, 10, 11]*
`lista.extend([12, 13])`	dołączanie listy *[12, 13]* do listy *[9, 2, 3, 4, 5, 6, 7, 8, 1, 10]*, wynik: *[9, 2, 3, 4, 5, 6, 7, 8, 1, 10, 12, 13]*
`lista.count(9)`	obliczanie, ile razy podana wartość 9 występuje na liście, wynik: *1*
`lista.index(8)`	wyznaczanie pozycji (licząc od 0) pierwszego wystąpienia podanej wartości 8, wynik: *7*
`lista.insert(3, 33)`	wstawianie liczby 33 do listy na podaną pozycję 3., wynik: *[9, 2, 3, 33, 4, 5, 6, 7, 8, 1, 10]*
`lista.pop(4)`	zwracanie wartości z podanej pozycji 4. i usunięcie tego elementu z listy, wynik: *[9, 2, 3, 4, 6, 7, 8, 1, 10]*
`lista.remove(2)`	usunięcie z listy pierwszej znalezionej wartości 2, wynik: *[9, 3, 4, 5, 6, 7, 8, 1, 10]*
`lista.reverse()`	odwracanie kolejności elementów listy, wynik: *[10, 1, 8, 7, 6, 5, 4, 3, 2, 9]*
`lista.sort()`	porządkowanie listy w porządku niemalejącym, wynik: *[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]*

Ćwiczenie 2.4.

Napisz program, który wykonuje następujące operacje na wczytanej z klawiatury liście zawierającej liczby całkowite. Zastosuj na listach metody podane w tabeli 2.2:

a) obliczenie, ile razy liczba 6 występuje w liście,

b) wyznaczenie pozycji pierwszego wystąpienia liczby 8 w liście,

c) wstawienie liczby 15 do listy na pozycję 0,

d) usunięcie z listy pierwszej znalezionej liczby 10,

e) odwrócenie kolejności elementów listy,

f) uporządkowanie listy w porządku niemalejącym.

Zadanie 2.3.

Napisz program wykonujący następujące operacje na 12-elementowej liście zawierającej liczby całkowite, której wartości wpisywane są z klawiatury:

a) wypisanie wszystkich elementów listy,

b) obliczenie liczby tych elementów listy, których numer nie jest podzielny przez 5,

c) zwiększenie o 2 wartości tych elementów listy, które mają nieparzysty numer zawarty w przedziale [3, 9],

d) obliczenie iloczynu tych elementów listy, których wartość jest równa 5,

e) obliczenie liczby tych elementów listy, których wartość nie zawiera się w przedziale [5, 8].

Zadanie 2.4.

Napisz program wykonujący następujące operacje na 10-elementowej liście zawierającej liczby całkowite, której wartości wpisywane są z klawiatury:

a) wypisanie wszystkich elementów listy,

b) obliczenie sumy tych elementów listy, których wartość jest większa od 6,

c) wypisanie tych elementów listy, których indeks jest zawarty w przedziale [3, 7],

d) wyzerowanie tych elementów listy, których indeks jest podzielny przez 3,

e) obliczenie średniej arytmetycznej wszystkich elementów listy i wypisanie tych elementów, których wartość jest mniejsza od wyznaczonej średniej.

Zadanie 2.5.

Napisz program wykonujący następujące operacje na 9-elementowej liście zawierającej liczby rzeczywiste, której wartości wprowadzane są w programie:

a) wypisanie wszystkich elementów listy,

b) obliczenie sumy tych elementów listy, których indeks jest podzielny przez 4,

c) zmniejszenie o 5 wartości tych elementów listy, które mają wartość większą od 0,

d) wypisanie tych elementów listy, których indeks jest nieparzysty i zawiera się w przedziale [1, 5],

e) obliczenie liczby tych elementów listy, których wartość jest zawarta w przedziale [5, 21].

Temat 3. Pozycyjne systemy liczbowe

3.1. Systemy liczbowe

Definicja

Systemem liczbowym nazywamy zbiór zasad określających sposób zapisywania i nazywania liczb.

Definicja

Pozycyjny system liczbowy to system, w którym wartość cyfry zależy od miejsca, w jakim znajduje się ona w danej liczbie. Miejsce to nazywamy pozycją.

W życiu codziennym korzystamy z **systemu dziesiętnego**, zwanego decymalnym, którego podstawą jest dziesięć. Do najważniejszych pozycyjnych systemów liczbowych wykorzystywanych w informatyce należą:

- system dwójkowy, czyli binarny,
- system ósemkowy, czyli oktalny,
- system szesnastkowy, czyli heksadecymalny.

Podstawą **systemu binarnego**, określającą liczbę cyfr, jest dwa. System ten korzysta więc z dwóch cyfr, którymi są 0 i 1.

System oktalny ma podstawę osiem, stąd cyframi są tutaj 0, 1, 2, 3, 4, 5, 6, 7.

Podstawą **systemu heksadecymalnego** jest szesnaście, a więc w systemie tym korzystamy z szesnastu cyfr. Cyframi tego systemu są: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F*. Wykorzystanie liter w zapisie cyfr podyktowane jest koniecznością jednoznacznej notacji liczby w tym systemie. Litery odpowiadają cyfrom, których wartości zapisane w układzie dziesiętnym są liczbami dwucyfrowymi:

$A_{16} = 10_{10}$;

$B_{16} = 11_{10}$;

$C_{16} = 12_{10}$;

$D_{16} = 13_{10}$;

$E_{16} = 14_{10}$;

$F_{16} = 15_{10}$.

Gdybyśmy nie korzystali z liter, zapis 112_{16} mógłby oznaczać liczbę 112_{16} lub $B2_{16}$, lub $1C_{16}$.

Wskazówka

W języku Python można skorzystać z **wbudowanych funkcji, które wykonują konwersję** liczb całkowitych z systemu dziesiętnego na liczby w innym systemie pozycyjnym.

```
x = 209
print(oct(x), bin(x), hex(x))
```

Wynikiem działania tego programu jest poniższy komunikat:

0o321 0b11010001 0xd1

Oznacza on, że liczba 209 zapisana w systemie decymalnym jest równa liczbie 321 w systemie oktalnym, 11010001 w systemie binarnym oraz *D*1 w systemie heksadecymalnym.

Wskazówka

Przy wykonywaniu konwersji i działań arytmetycznych w różnych systemach liczbowych można zastosować udostępnioną w systemie Windows **aplikację Kalkulator**. Program ten umożliwia przeprowadzanie obliczeń w następujących systemach: decymalnym (czyli dziesiętnym), binarnym, oktalnym i heksadecymalnym. Wykonywać można zarówno konwersję pomiędzy wymienionymi systemami, jak i operacje arytmetyczne.

Ćwiczenie 3.1.

Napisz program, który wykona konwersję wpisanej z klawiatury liczby całkowitej podanej w systemie dziesiętnym na liczbę we wskazanym systemie. Skorzystaj z funkcji wbudowanych języka Python:

a) systemu ósemkowego,

b) systemu dwójkowego,

c) systemu szesnastkowego.

Zadanie 3.1.

Napisz program, który dla wpisanej z klawiatury liczby naturalnej *n* wykonuje następujące operacje:

a) wypisuje cyfrę jedności liczby *n*,

b) wypisuje cyfrę setek liczby *n*,

c) wypisuje kolejne cyfry liczby *n*, rozpoczynając od cyfry jedności,

d) oblicza sumę cyfr liczby *n*.

Zadanie 3.2.

Liczba pseudobinarna to liczba, która w systemie dziesiętnym jest zapisana tylko za pomocą cyfr 1 lub 0. Przykładami takich liczb są 110011, 10110, natomiast liczbami pseudobinarnymi nie są liczby 345, 2091. Podaj specyfikację zadania i napisz program, który sprawdza, czy liczba wpisana z klawiatury jest liczbą pseudobinarną.