

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Zbiór zadań z informatyki. Nie tylko dla maturzystów

Autor: Monika Niedziela
ISBN: 83-7361-991-7
Format: B5, stron: 336



Zestaw zadań przydatny każdemu uczniowi i maturzyście

- Dokładnie omówienie zagadnień związanych z algorytmami, programowaniem i bazami danych
- Przykłady rozwiązań
- Zadania

Znacznie łatwiej i szybciej opanuje się poznawane zagadnienie, gdy jest to powiązane z rozwiązywaniem praktycznych przykładów. O wiele prościej jest zrozumieć znaczenie symboli i wzorów, gdy zastosuje się je w zadaniu – wtedy okaże się, że „nie taki diabeł straszny”. Opanowanie rzeczywistego zastosowania wiadomości teoretycznych jest szczególnie istotne w przypadku przedmiotów mocno powiązanych z wiedzą praktyczną. Takim przedmiotem niewątpliwie jest informatyka.

„Zbiór zadań z informatyki. Nie tylko dla maturzystów” to książka, dzięki której poznasz praktyczne zastosowanie wiadomości dotyczących systemów liczbowych, algorytmów, zasad programowania i możliwości wykorzystania baz danych. Każde zagadnienie jest przedstawione od strony teoretycznej i zilustrowane przykładami. Rozwiązując kolejne zadania, nauczysz się tego, co w informatyce najistotniejsze – myślenia algorytmicznego i obiektowego. Opanujesz metody przeliczania wartości pomiędzy systemem binarnym, ósemkowym, szesnastkowym i dziesiętnym, poznasz zastosowanie różnych typów danych, przedstawisz rzeczywiste problemy w sposób algorytmiczny, napiszesz proste programy i przekonasz się, jakie możliwości oferuje baza danych Access.

- Systemy liczbowe
- Typy i struktury danych
- Tworzenie algorytmów
- Podstawowe zasady programowania w języku Pascal
- Programowanie strukturalne i obiektowe
- Projektowanie relacyjnych baz danych
- Wprowadzanie i modyfikowanie danych
- Manipulowanie danymi za pomocą języka SQL



Spis treści

	Wprowadzenie	7
Część I	Metody reprezentowania informacji oraz struktury danych	9
Rozdział 1.	Reprezentacja liczb całkowitych	11
	Wprowadzenie	11
	System binarny	11
	System oktalny	12
	System heksadecymalny	14
	Reprezentacja binarna liczb ujemnych	16
	Zadania do samodzielnego rozwiązania	16
Rozdział 2.	Reprezentacja liczb rzeczywistych	19
	Wprowadzenie	19
	Ułamki dodatnie w notacji naturalnej	19
	Kod U2 dla ułamków	21
	Reprezentacja stałopozycyjna liczb	22
	Zmiennopozycyjna reprezentacja liczb	22
	Zadania do samodzielnego rozwiązania	24
Rozdział 3.	Predefiniowalne typy danych i ich podzbiory	25
	Wprowadzenie	25
	Typy liczbowe	26
	Typ logiczny	26
	Typy znakowe	26
	Typy okrojone i wycięte	27
	Zadania do samodzielnego rozwiązania	28
Rozdział 4.	Strukturalne typy danych	29
	Wprowadzenie	29
	Tablice	29
	Rekordy	31
	Pliki	31
	Zadania do samodzielnego rozwiązania	32

Rozdział 5. Typ wskaźnikowy oraz dynamiczne struktury danych	35
Wprowadzenie	35
Wskaźnik	35
Stos	37
Kolejka	37
Zadania do samodzielnego rozwiązania	38
Rozdział 6. Wprowadzenie do algorytmiki	39
Pojęcia podstawowe	39
Sposoby zapisu algorytmów	40
Elementy schematu blokowego	44
Zadania do samodzielnego rozwiązania	46
Rozdział 7. Algorytmy iteracyjne (pętle)	49
Wprowadzenie	49
Zadania do samodzielnego rozwiązania	57
Rozdział 8. Algorytmy klasyczne	59
Wprowadzenie	59
Zadania do samodzielnego rozwiązania	69
Rozdział 9. Algorytmy rekurencyjne	73
Wprowadzenie	73
Zadania do samodzielnego rozwiązania	78
Rozdział 10. Algorytmy „dziel i zwyciężaj”	81
Wprowadzenie	81
Zadania do samodzielnego rozwiązania	86
Rozdział 11. Algorytmy numeryczne	87
Wprowadzenie	87
Zadania do samodzielnego rozwiązania	92
Rozdział 12. Sortowanie	93
Wprowadzenie	93
Zadania do samodzielnego rozwiązania	103
Część III Programowanie	105
Rozdział 13. Wprowadzenie do programowania w Pascalu	107
Pierwszy program	107
Instrukcja warunkowa i wyboru	109
Instrukcja warunkowa if..then..else	109
Instrukcja wyboru case..of	110
Instrukcja złożona	111
Instrukcje iteracyjne	111
Instrukcja for..to..do	112
Instrukcja for..downto..do	113
Instrukcja while..do	113
Instrukcja repeat..until	115
Zadania do samodzielnego rozwiązania	117

Rozdział 14. Programowanie strukturalne	121
Wprowadzenie	121
Procedury	122
Funkcje	123
Zasięg zmiennych. Sposoby przekazywania parametrów	124
Moduły	125
Zadania do samodzielnego rozwiązania	127
Rozdział 15. Przetwarzanie plików	129
Wprowadzenie	129
Zadania do samodzielnego rozwiązania	132
Rozdział 16. Implementacja dynamicznych struktur danych	133
Wprowadzenie	133
Zastosowanie wskaźników	134
Stos	135
Kolejka	137
Zadania do samodzielnego rozwiązania	138
Rozdział 17. Programowanie obiektowe	139
Wprowadzenie	139
Pojęcie obiektu	139
Obiekty w modułach	141
Dziedziczenie	142
Zadanie do samodzielnego rozwiązania	143
Rozdział 18. Programowanie w Delphi	145
Wprowadzenie	145
Zadania do samodzielnego rozwiązania	152
Część IV Bazy danych	155
Rozdział 19. Projektowanie relacyjnej bazy danych	157
Wprowadzenie	157
Projekt tabeli	157
Zadania do samodzielnego rozwiązania	160
Właściwości pól	160
Zadania do samodzielnego rozwiązania	163
Normalizacja bazy danych	164
Zadania do samodzielnego rozwiązania	167
Relacje między tabelami	168
Podsumowanie	169
Zadania do samodzielnego rozwiązania	170
Rozdział 20. Formularze	173
Wprowadzenie	173
Proste modyfikacje formularzy	173
Zadania do samodzielnego rozwiązania	176
Zastosowanie formantów	178
Wprowadzenie	178
Zadania do samodzielnego rozwiązania	181
Rozdział 21. Zapytania (kwerendy)	183
Wprowadzenie	183
Proste kwerendy wyszukujące, sortujące i parametryczne	183
Zadania do samodzielnego rozwiązania	186

Kwerendy z polami wyliczanymi	187
Zadania do samodzielnego rozwiązania	188
Kwerendy grupujące i podsumowujące	189
Zadania do samodzielnego rozwiązania	190
Kwerendy krzyżowe	192
Zadania do samodzielnego rozwiązania	193
Kwerendy funkcjonalne (aktualizujące, dołączające, usuwające, tworzące tabelę)	194
Zadania do samodzielnego rozwiązania	195
Rozdział 22. Raporty	197
Wprowadzenie	197
Zadania do samodzielnego rozwiązania	200
Rozdział 23. Eksport i import danych	203
Wprowadzenie	203
Zadania do samodzielnego rozwiązania	205
Rozdział 24. Makra	207
Wprowadzenie	207
Zadania do samodzielnego rozwiązania	209
Rozdział 25. Administrowanie bazą danych	211
Wprowadzenie	211
Zadania administratora baz danych	211
Wielodostępne bazy danych, uprawnienia użytkowników i grup	213
Zadania do samodzielnego rozwiązania	216
Rozdział 26. SQL	219
Wprowadzenie	219
Tworzenie baz, tabel i indeksów	219
CREATE DATABASE	220
USE	220
DROP DATABASE	220
CREATE TABLE	220
DESCRIBE	221
CREATE INDEX	221
Wprowadzanie, usuwanie i modyfikowanie danych	222
INSERT	222
DELETE	223
Modyfikowanie danych	223
UPDATE	223
Wyszukiwanie danych	224
SELECT	224
Zadania do samodzielnego rozwiązania	227
Dodatki	231
Dodatek A Zadania typu maturalnego	233
Zadania do samodzielnego rozwiązania	235
Dodatek B Rozwiązania zadań	239
Skorowidz	327

Rozdział 6.

Wprowadzenie do algorytmiki

Pojęcia podstawowe

Powszechnie przez algorytm rozumie się schemat mechanicznego rozwiązania określonego problemu. Posługując się bardziej naukowymi pojęciami, można powiedzieć, że **algorytm to skończony ciąg operacji na obiektach, ze ściśle ustalonym porządkiem wykonania, dający możliwość realizacji zadania z określonej klasy.**

Algorytm w sensie informatycznym opiera się na określonych **danych** i daje oczekiwane **wyniki**. Ponadto powinien mieć następujące cechy:

- ❖ **Poprawność** — dla każdego zestawu poprawnych danych wejściowych algorytm powinien dawać poprawne wyniki.
- ❖ **Skończoność** — dla każdego zestawu poprawnych danych wejściowych algorytm powinien dawać wyniki w skończonej liczbie kroków.
- ❖ **Określoność** — algorytm nie może pozostawiać wątpliwości co do wyboru kolejnej operacji (najlepszym sprawdzianem tej cechy jest łatwość implementacji algorytmu).
- ❖ **Efektywność (sprawność)** — algorytm powinien prowadzić do rozwiązania jak najniższym kosztem, czyli w jak najmniejszej liczbie kroków. W informatyce cecha ta sprowadza się do optymalizacji zajętości pamięci przez struktury danych wykorzystywane w algorytmie oraz do optymalizacji złożoności obliczeniowej (liczby wykonanych operacji).
- ❖ **Ogólność (uniwersalność)** — algorytm powinien rozwiązywać nie tylko jedno szczególne zadanie, ale całą klasę zadań.

Sposoby zapisu algorytmów

Algorytm można zapisać w postaci:

- ❖ listy kroków,
- ❖ schematu blokowego,
- ❖ pseudokodu.

Każdy algorytm powinien posiadać **specyfikację**, w której określamy **dane**, z jakich algorytm korzysta, **wyniki**, które powinien dawać, oraz **zmienne pomocnicze** niezbędne do realizacji algorytmu. Uniwersalny algorytm operuje nie na stałych (liczbowych, tekstowych), lecz na **zmiennych**. Zmienna to po prostu pojemnik na dane. Może być oznaczona dowolną literą lub łańcuchem znaków (np. `a`, `bok_a`).



W specyfikacji poza wypisaniem nazw zmiennych należy określić ich funkcję w algorytmie oraz typ (liczba całkowita, liczba rzeczywista, łańcuch znaków, wartość logiczna). Czasem należy także określić warunki, które muszą spełniać dane wejściowe.

W algorytmach spotykamy się także z **wyrażeniami**, w skład których poza **zmiennymi** i **stałymi** wchodzi operatory (+, -, *, /) i funkcje matematyczne.

Algorytm w postaci listy kroków jest intuicyjny i czytelnik najlepiej go opanuje, analizując kolejne przykłady. Algorytm w postaci **pseudokodu** ma strukturę bardziej złożoną, jego budowę czytelnik pozna czytając komentarze. Każdy z algorytmów zostanie zapisany także w postaci **schematu blokowego**, którego elementy zostaną omówione pod przykładami (rysunek 6.4).

Należy zwrócić uwagę, że budując wyrażenie warunkowe, korzystamy z **operatorów relacyjnych** (=, <, >, <=, >=, <>). W algorytmach stosuje się także **instrukcję przypisania** o postaci :=. Polega ona na podstawieniu wartości znajdującej się po prawej stronie pod zmienną z lewej strony, np. `a:=b+5` oznacza podstawienie pod zmienną `a` wartości zmiennej `b` zwiększonej o 5, tzn. jeśli `b = 7`, to `a = 12`.

Zadania znajdujące się pod przykładami najlepiej rozwiązać na zasadzie analogii.

Przykład 6.1. Algorytm liniowy

Sformułuj algorytm wyliczający pole prostokąta o bokach podanych przez użytkownika.

Sposób rozwiązania

Długości boków podane przez użytkownika zapamiętamy w zmiennych `a` i `b`. Następnie wyliczymy `p` jako iloczyn `a` i `b`. Wyliczone `p` wyświetlimy na ekranie. Będzie to prosty algorytm liniowy.

Rozwiązanie

Poniżej przedstawiono ten sam algorytm zapisany w postaci listy kroków, pseudokodu i schematu blokowego (rysunek 6.1).

Specyfikacja algorytmu:

Dane:

a — pierwszy bok prostokąta, liczba rzeczywista większa od 0;

b — drugi bok prostokąta, liczba rzeczywista większa od 0.

Wynik:

p — pole prostokąta, liczba rzeczywista.

Lista kroków:

Krok 1. Wczytaj wartość pierwszego boku. Zapisz ją pod zmienną a .

Krok 2. Wczytaj wartość drugiego boku. Zapisz ją pod zmienną b .

Krok 3. Oblicz p jako iloczyn a i b .

Krok 4. Wyświetl p .

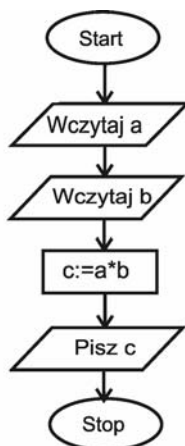
Pseudokod:

Program kwadrat	<i>{nagłówek programu}</i>
zmienne	
a, b, c : rzeczywiste	<i>{deklaracje zmiennych}</i>
początek	<i>{początek programu właściwego}</i>
czytaj(a)	<i>{wczytanie danych}</i>
czytaj(b)	
$c := a * b$	<i>{instrukcja przypisania}</i>
pisz(c)	<i>{wyświetlenie wyniku}</i>
koniec	<i>{koniec programu właściwego}</i>

Schemat blokowy:

Rysunek 6.1.

Schemat blokowy do przykładu 6.1



Przykład 6.2. Algorytm z warunkiem

Sformułuj algorytm zwracający mniejszą z dwóch liczb podanych przez użytkownika.

Sposób rozwiązania:

Wartości podane przez użytkownika zapamiętamy w zmiennych a i b . Następnie sprawdzimy, czy $a < b$ — jeśli tak, to w zmiennej c zapamiętamy wartość a ; jeśli nie, to w zmiennej c zapamiętamy wartość b . Następnie wyświetlimy c na ekranie. Będzie to **algorytm z warunkami** (rozgałęzieniami), gdyż w zależności od spełnienia bądź niespełnienia warunku ($a < b$) różne jest jego działanie.

Rozwiązanie

Podobnie jak w przykładzie 6.1, poniżej przedstawiono ten sam algorytm zapisany w postaci listy kroków, pseudokodu i schematu blokowego (rysunek 6.2).

Specyfikacja algorytmu:**Dane:**

a, b — liczby podane przez użytkownika, liczby całkowite.

Wynik:

c — mniejsza z liczb a i b , liczba całkowita.

Lista kroków:

Krok 1. Wczytaj wartość pierwszej liczby. Zapisz ją pod zmienną a .

Krok 2. Wczytaj wartość drugiej liczby. Zapisz ją pod zmienną b .

Krok 3. Jeśli $a < b$, to do c przypisz wartość a i przejdź do kroku 5.

Krok 4. Do c przypisz wartość b .

Krok 5. Wypisz wartość c .

Krok 6. Koniec.

Pseudokod:

```

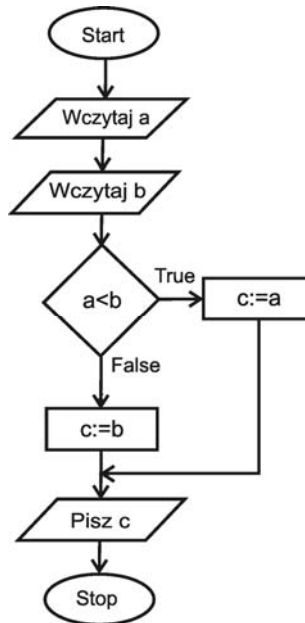
Program mniejsza
zmienne
  a, b, c: całkowite
początek
  czytaj(a,b)
  jeżeli a<b to c:=a  {sprawdzenie warunku oraz działanie wykonywane w przypadku
                    gdy warunek jest prawdziwy}
    w przeciwnym przypadku c:=b  {działanie wykonywane, gdy warunek nie jest prawdziwy}
  pisz(c)
koniec

```

Schemat blokowy:

Rysunek 6.2.

Schemat blokowy do przykładu 6.2



Przykład 6.3. Algorytm z zagnieżdżonymi warunkami

Sformułuj algorytm sprawdzający, czy z boków o długościach podanych przez użytkownika można utworzyć trójkąt.

Sposób rozwiązania:

Długości boków podane przez użytkownika zapamiętamy w zmiennych a , b , c . Aby z trzech odcinków można było zbudować trójkąt, żadna z sum dwóch dowolnych boków nie może być mniejsza od trzeciego boku. Posłużymy się zmienną logiczną *trojkat*, której na początku przypiszemy wartość fałsz (*false*). Jeśli będzie zachodziła każda z zależności: $a+b > c$, $b+c > a$, $c+a > b$, to zmiennej *trojkat* przypiszemy wartość prawda (*true*) i wyświetlimy komunikat, że z podanych boków da się zbudować trójkąt. Algorytm w postaci pseudokodu i schematu blokowego będzie **algorytmem z zagnieżdżonymi warunkami**.

Rozwiązanie

Podobnie jak w poprzednich przykładach, przedstawiono ten sam algorytm zapisany w postaci listy kroków, pseudokodu i schematu blokowego (rysunek 6.3).

Specyfikacja algorytmu:

Dane:

a, b, c — liczby rzeczywiste podane przez użytkownika, długości odcinków.

Wynik:

trojkat — informacja, czy z odcinków o podanych długościach można utworzyć trójkąt, wartość logiczna.

Lista kroków:

Krok 1. Wczytaj wartości zmiennych a, b, c .

Krok 2. Zmiennej trojkat przypisz wartość false.

Krok 3. Jeśli $a+b>c$, to jeśli $a+c>b$, to jeśli $c+b>a$, to zmiennej trojkat przypisz wartość true.

Krok 4. Jeżeli zmienna trojkat ma wartość true, to wypisz 'Z podanych boków można zbudować trójkąt'; w przeciwnym przypadku wypisz 'Z podanych boków nie można zbudować trójkąta'.

Krok 5. Koniec.

Komentarz:

Najpierw sprawdzamy warunek $a+b>c$ — jeśli zachodzi, to sprawdzamy, czy $a+c>b$ — jeśli tak, to sprawdzamy, czy $c+b>a$ — jeśli tak, to trojkat :=true. Jak widać, w przypadku gdy z podanych boków można zbudować trójkąt, sprawdzane są wszystkie 3 warunki i wszystkie są prawdziwe.

Schemat blokowy (rysunek 6.3).**Pseudokod:**

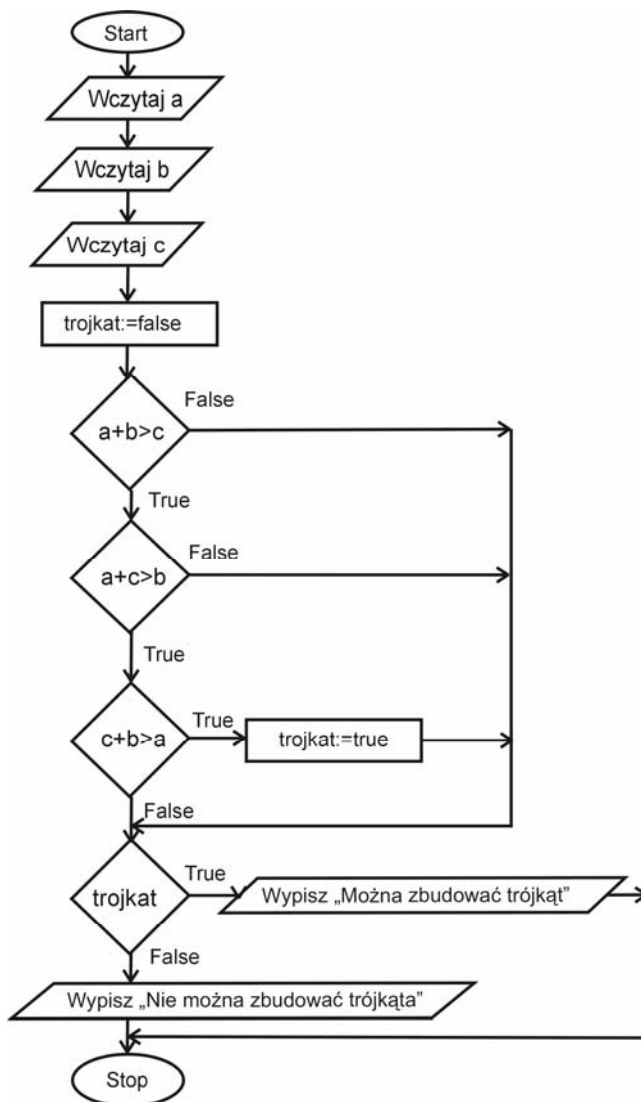
```

Program czytrojkat
zmienne
  a, b, c: rzeczywiste
  trojkat: logiczna
początek
  czytaj(a,b,c)
  trojkat:=false
  jeżeli a+b>c to
    początek
      jeżeli a+c>b to
        jeżeli b+c>a to trojkat:=true
    koniec
  jeżeli trojkat to pisz('Można zbudować trójkąt')
  w przeciwnym wypadku pisz('Nie można zbudować trójkąta')
koniec

```

Rysunek 6.3.

Schemat blokowy do przykładu 6.3

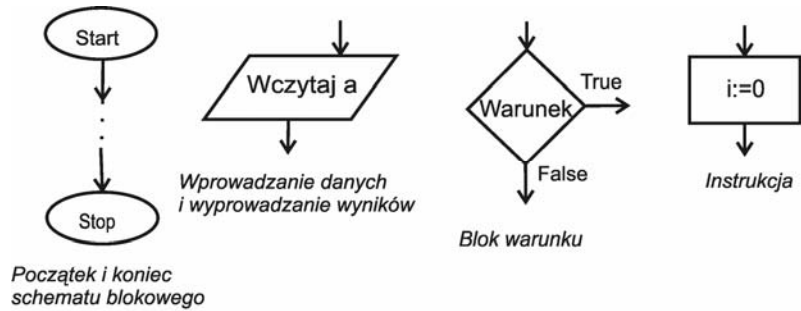


Elementy schematu blokowego

Reguły, według których należy budować schematy blokowe (rysunek 6.4):

- ❖ Każdy algorytm ma tylko jeden początek.
- ❖ Wszystkie drogi muszą się kończyć na bloku *Stop*.
- ❖ Wszystkie bloki muszą mieć zgodną z ich budową liczbę wejść i wyjść.
- ❖ Każda linia musi być opatrzona strzałką informującą o kierunku przepływu informacji.

Rysunek 6.4.
Elementy schematu
blokowego



Zadania do samodzielnego rozwiązania

Zadanie 6.1.

Napisz algorytm w postaci listy kroków i pseudokodu, wyliczający pole kwadratu o boku podanym przez użytkownika.

Zadanie 6.2.

Napisz algorytm w postaci listy kroków, według którego sporządzasz swoją ulubioną potrawę.

Zadanie 6.3.

Napisz algorytm w postaci schematu blokowego i pseudokodu, zwracający wartość bezwzględną liczby podanej przez użytkownika.

Zadanie 6.4.

Napisz algorytm w postaci listy kroków i schematu blokowego, informujący, czy liczba podana przez użytkownika jest większa, mniejsza czy równa zero.

Zadanie 6.5.

Zmodyfikuj algorytm z przykładu 6.1 tak, by w przypadku wprowadzenia danych mniejszych lub równych zero nie było wyliczane pole, lecz by był wyświetlany odpowiedni komunikat.

Zadanie 6.6.

Napisz algorytm w postaci listy kroków, pseudokodu i schematu blokowego, informujący, czy liczba podana przez użytkownika jest liczbą parzystą czy nieparzystą. Użyj operatora mod, który zwraca resztę z dzielenia całkowitego. Np. $5 \bmod 2 = 1$, $4 \bmod 2 = 0$.

Zadanie 6.7.

Utwórz w postaci listy kroków wraz ze specyfikacją algorytm wyliczający pierwiastek równania liniowego o postaci $ax+b=0$; gdzie a, b podaje użytkownik. Weź pod uwagę fakt, że użytkownik może podać $a=0$ lub $b=0$.

Zadanie 6.8.

Narysuj w postaci schematu blokowego wraz ze specyfikacją algorytm wyliczający pierwiastki równania kwadratowego o postaci $ax^2+bx+c=0$; gdzie a, b, c podaje użytkownik.

Zadanie 6.9.

Rozbuduj algorytm z przykładu 6.3. Nowy algorytm, w przypadku odcinków, z których można utworzyć trójkąt, ma liczyć jego pole według wzoru Herona:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ gdzie: } p = \frac{a+b+c}{2}.$$