Uniwersytet Jana Długosza w Częstochowie

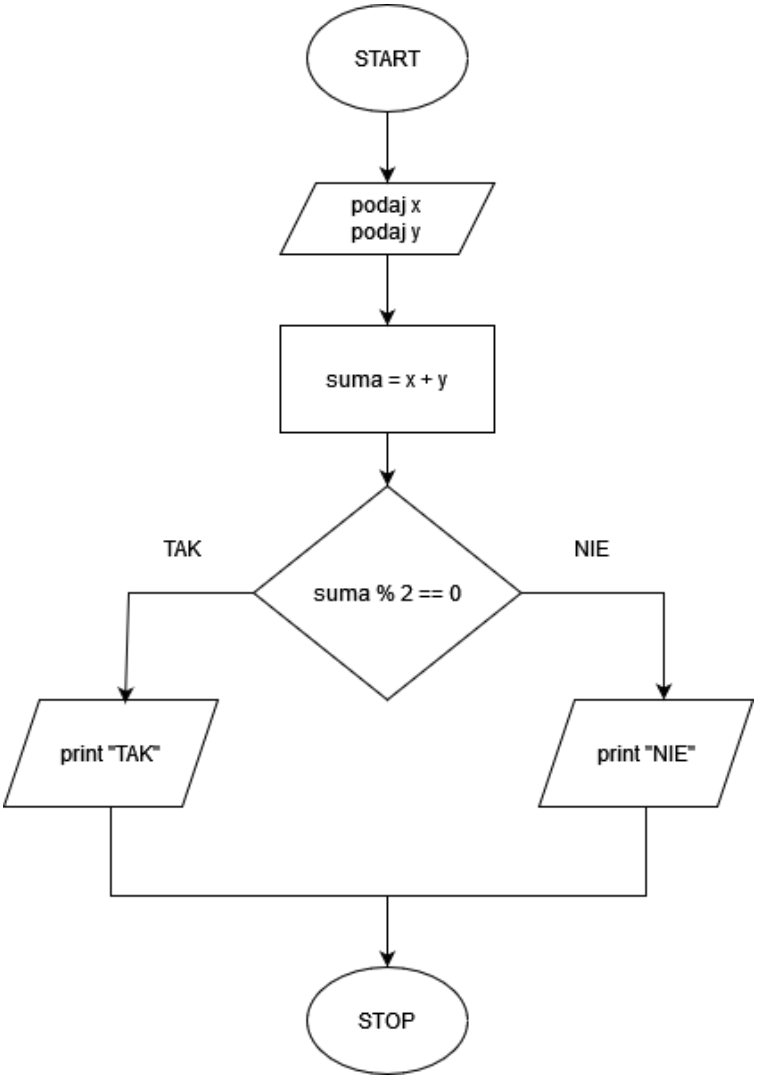
Mykhailo Hulii

Studia stacjonarne 1 stopnia,

2 rok informatyka, grupa 1

Zadanie 2

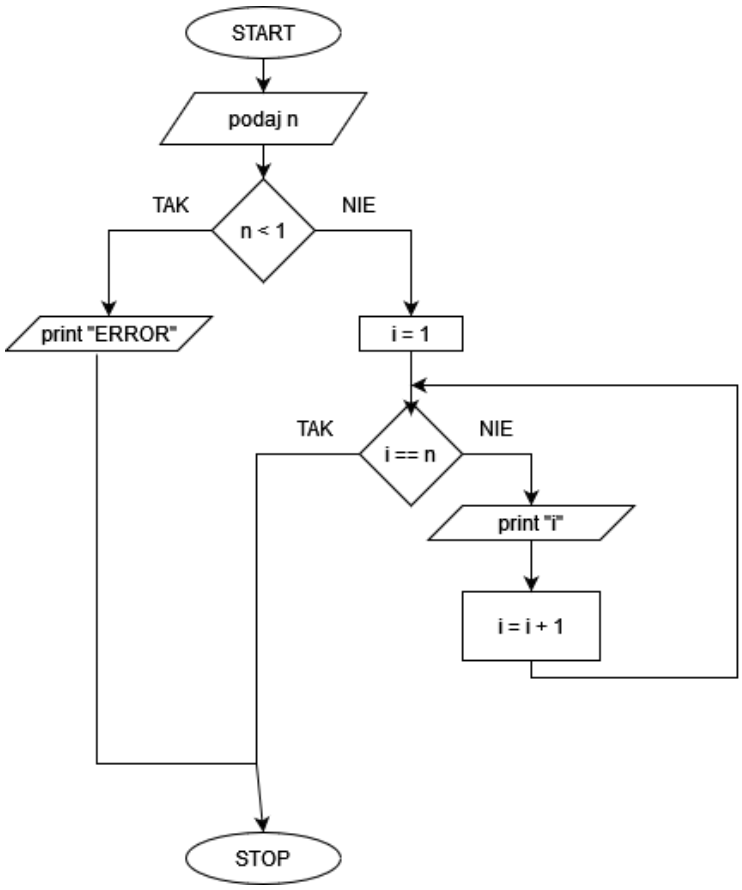
Utworzyć schemat blokowy algorytmu sprawdzającego czy suma podanych przez użytkownika dwóch liczb jest parzysta. W odpowiedzi algorytm ma wyświetlić TAK lub NIE. Utwórz także pseudokod.




```
START

Podaj x
Podaj y
suma = x + y

if suma % 2 == 0 {
  print("TAK")
else {
  print("Nie")
}

STOP
```

Zadanie 3a

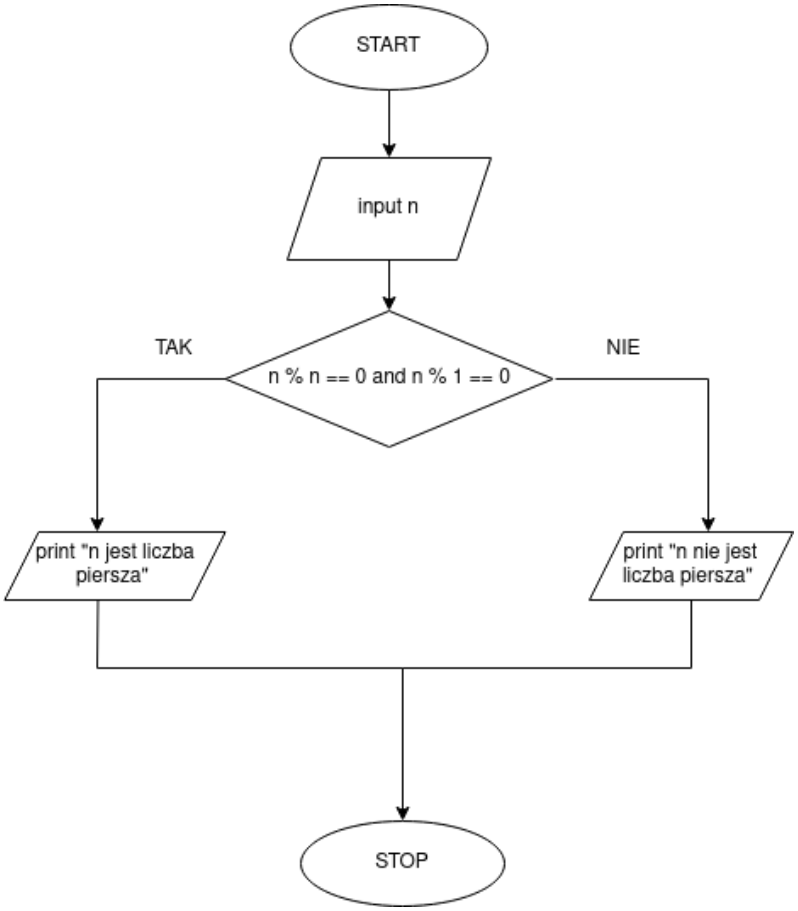
Utworzyć schemat blokowy i pseudokod algorytmu pozwalającego na ekranie wyświetlić liczby od 1 do n włącznie, gdzie n to wartość podana przez użytkownika. W pseudokodzie zastosować rozwiązanie typu polecenie goto i odpowiednią etykietę. Zapewnić kontrolę zmiennej n. Jeśli użytkownik wprowadzi wartość mniejszą niż 1 powinien otrzymać stosowną informację.




```
START
podaj n

if (n < 1) {
  print "ERROR"
}
else {
  i = 1
  :petla_1
  if (i == n) {
    end
  }
  else {
    print i
    i = i + 1
    goto  petla_1
   }
}
STOP
```

Zadanie 3

Opracować schemat blokowy oraz pseudokod algorytmu (wersja tzw. naiwna) sprawdzającego czy podana liczba jest liczba pierwszą.




```
START

input n

if (n % n == 0 and n % 1 ==0) {
  print "Liczba jest liczba piersza"
}
else {
  print "Liczba nie jest liczba piersza"
}

STOP
```

Zadanie 5

Zapisać poniższy pseudokod za pomocą schematu blokowego. Poprawić czytelność kodu przez odpowiednie sformatowanie go.

Start

Suma:=0

Podaj n

i:=0

Dopóki (i<n) wykonuj:

{

Wczytaj(a)

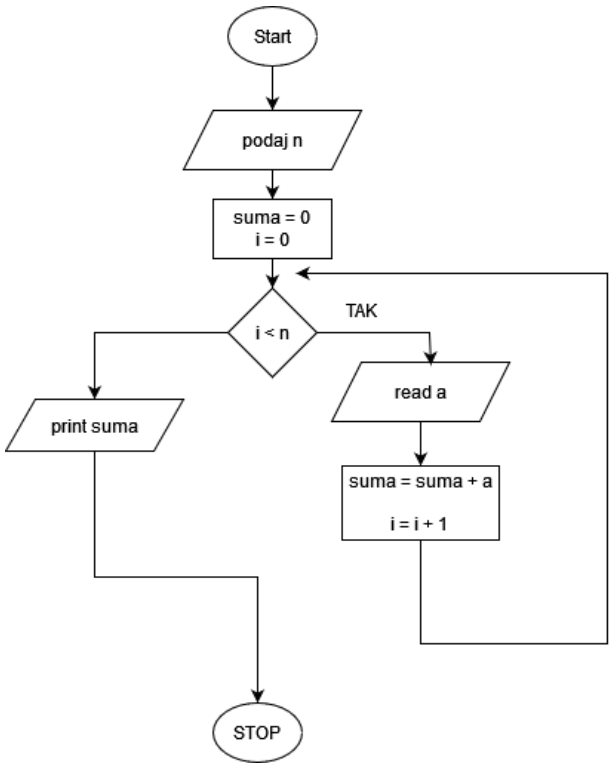
Suma := Suma + a

i := i + 1

}

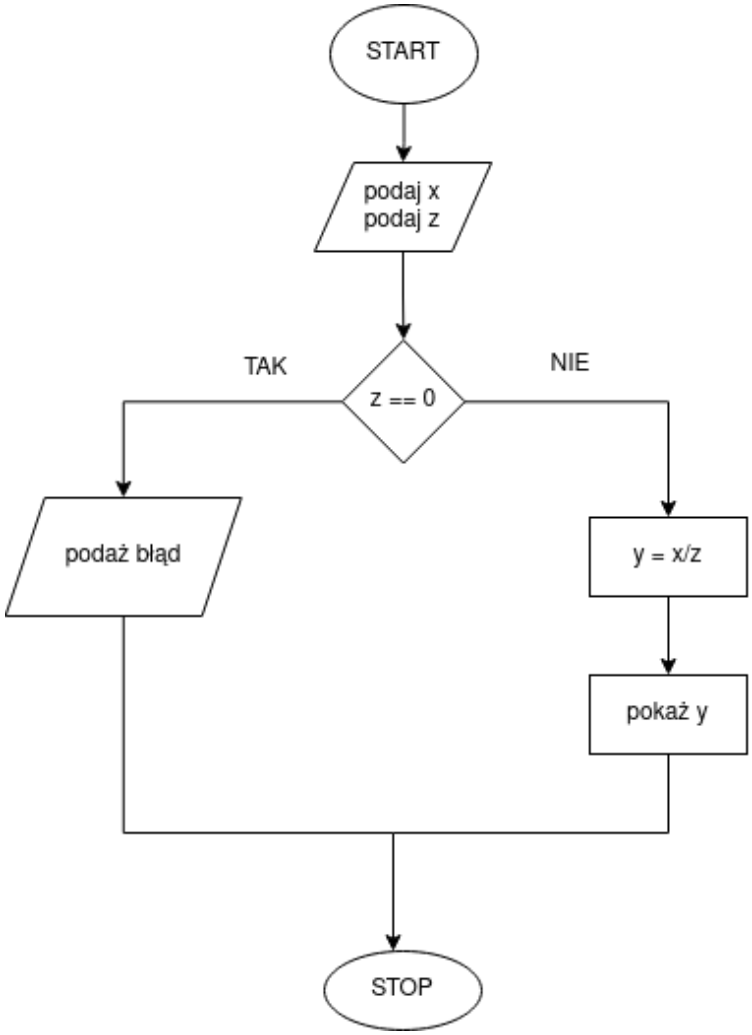
Wypisz(Suma)

Koniec



Zadanie 7

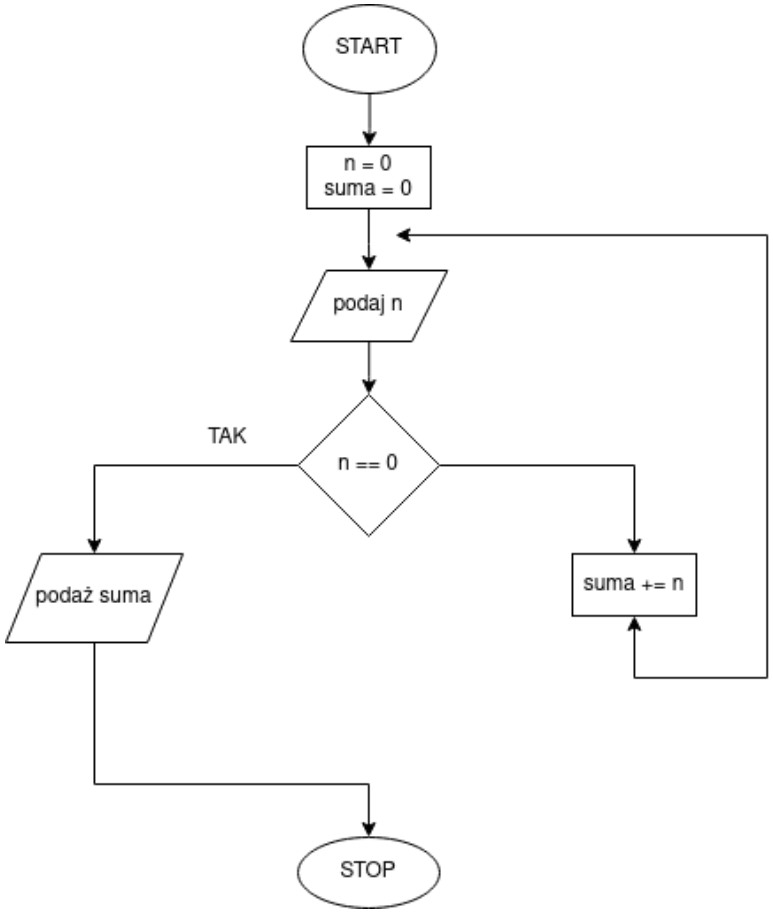
Mamy równanie y=x/z. Utworzyć pseudokod i schemat blokowy algorytmu realizującego zadanie: wczytać dane x i z a następnie znaleźć rozwiązanie o ile istnieje. Zadbaj o odpowiednie komunikaty dla użytkownika.




```
START

podaj x
podaj z

if (z == 0) {
  y = x/z
}
else {
  podaż błąd
}

STOP
```

Zadanie 11

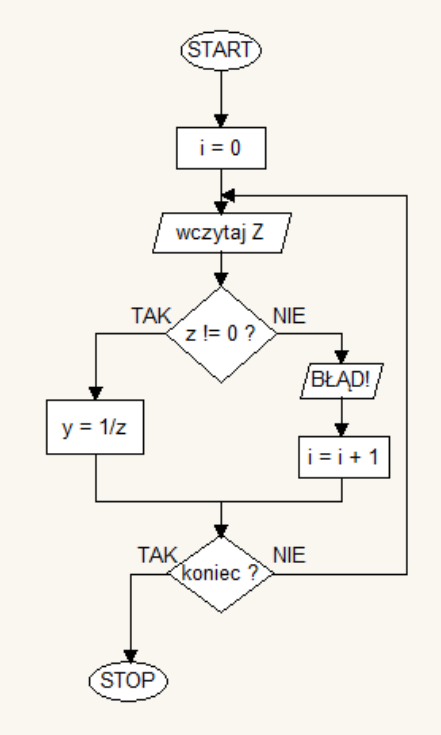
Narysować schemat blokowy programu wyznaczającego sumę liczb podawanych z klawiatury tak długo, aż użytkownik nie wprowadzi liczby zero. Liczba ta jest ogranicznikiem ciągu. Utworzyć także pseudokod.




```
START

n = 0
suma = 0

:loop1
podaj n
if (n == 0)  {
  pokaż suma
}
else {
  suma += n
  goto loop1
}

STOP
```

Zadanie 12

Opisać, za pomocą pseudokodu, działanie algorytmu zawartego na poniższym diagramie. Dodać informację wyjściową o ilości błędów.




```
START

i = 0

:loop1

podaj z

if (z != 0) {
  y = 1/z
}
else {
  podaż błąd
  i = i+1
}

if (czy koniec?) {
  goto STOP
}
else {
  goto loop1
}

STOP
```

Zadanie 17

a) Podaj różnicę między kompilacją a interpretacją.

b) Podaj różnicę między błędem kompilacji a wykonania

c) Podaj różnicę między językiem niskiego a wysokiego poziomu

d) Jakie jest zastosowanie komentarzy

e) Czym jest pseudokod?

f) czym się różni kod źródłowy od wynikowego?

g) Dla czego języki interpretowane są wolniejsze od kompilowanych?

a) Podaj różnicę między kompilacją a interpretacją?

Kompilacja:

1) Kompilacja spoczątku kompiluje kod, a potem wykonuje.

2) Kod skompilowany jest specyficzny dla danego procesora lub systemu operacyjnego.

Interpretator:

1) Interpretator czyta linije i wykonuje.

2) Kod źródłowy jest zazwyczaj bardziej przenośny, ponieważ interpreter jest dostosowany do różnych platform

---

b) Podaj różnicę między błędem kompilacji a wykonania?

Kompilacja:

1) Jeśli będą błedy, to kod ne skompiluje się.

Interpretator:

1) Kod wykonuję się do pierszego błęda.

---

c) Podaj różnicę między językiem niskiego a wysokiego poziomu?

Język niskiego poziomu:

1) Języki niskiego poziomu oferują niewielki poziom abstrakcji i są bliskie językowi maszynowemu. Programista musi być bardziej zaawansowany i bardziej precyzyjny w kodowaniu operacji na poziomie sprzętu, takich jak zarządzanie pamięcią czy rejestrami procesora.

2) Kod niskopoziomowy może być bardziej wydajny, ponieważ programista ma większą kontrolę nad operacjami na poziomie sprzętu. Może zoptymalizować kod pod kątem wydajności.

Język wysokiego poziomu:

1) Języki wysokiego poziomu oferują wyższy stopień abstrakcji, co oznacza, że pozwalają programiście skupić się bardziej na problemie, który ma rozwiązać, a mniej na szczegółach sprzętowych. Programista używa bardziej ogólnych konstrukcji językowych, co ułatwia tworzenie kodu.

2) Kod wysokopoziomowy jest zazwyczaj mniej wydajny, ponieważ kompilatory lub interpretery języków wysokiego poziomu wprowadzają pewien narzut związany z abstrakcją. Niemniej jednak, dla wielu zastosowań, różnica w wydajności jest pomijalna.

---

d) Jakie jest zastosowanie komentarzy?

1) dokumentacja

2) Wyłączanie kodu

3) Tłumaczenie

---

e) Czym jest pseudokod?

To sposób zapisu naszego algorytmu w nieistniejącym języku programowania najczęściej podobnym do naszego języka i gramatyki.

---

f) Czym się różni kod źródłowy od wynikowego?

1) Kod żródłowy to ten, ktory napisał programist

2) Kod wynikowy to ten, który przetwarza się w kod żródłowy, czyli binary (maszynowy)

---

g) Dla czego języki interpretowane są wolniejsze od kompilowanych?

1) Tłumaczenie w czasie rzeczywistym

2) Brak optymalizacji kompilacji

3) Wielokrotne analizy kodu

4) Brak kompilacji do kodu maszynowego

5) Brak kontroli nad sprzętem

1, 8, 9, 10, 13, 14, 15, 16 zadanie nie robimy