Temat 6. Zapisywanie algorytmów na liczbach naturalnych w wybranych językach programowania

1. **Realizacja algorytmu Euklidesa w wersji z odejmowaniem**
2. **Badanie podzielności liczb naturalnych**
3. **Realizacja algorytmu Euklidesa w wersji z dzieleniem**
4. **Algorytm wyodrębniania cyfr danej liczby**

Warto powtórzyć

1. Czym jest algorytm?
2. Czym jest specyfikacja problemu (zadania)?
3. W jaki sposób można przedstawiać algorytmy?
4. Na czym polega przedstawienie algorytmu w postaci listy kroków?
5. Jakie są zasady budowania schematu blokowego?
6. W jaki sposób przedstawiamy w schemacie blokowym sytuację warunkową?
7. W jaki sposób przedstawiamy w schemacie blokowym iterację? Czym jest pętla?

1. Realizacja algorytmu Euklidesa w wersji z odejmowaniem

W wybranym środowisku programowania chcemy napisać program realizujący algorytm znajdowania największego wspólnego dzielnika dwóch niezerowych liczb naturalnych. W jaki sposób zastosować poznane instrukcje środowisk programowania Scratch i Baltie oraz języków C++ i Python do rozwiązania tego problemu?

Do znajdowania największego wspólnego dzielnika (**NWD**) dwóch liczb naturalnych służy **algorytm Euklidesa**.

N **NWD**

Największy wspólny dzielnik dwóch lub więcej liczb naturalnych (różnych od zera) to największa liczba naturalna, przez którą dzieli się bez reszty każda z tych liczb.

W realizacji algorytmu Euklidesa wykorzystamy instrukcje warunkowe i iteracyjne. W algorytmie tym występują bowiem sytuacje warunkowe i powtarzające się polecenia.

Na przykład dla liczb $a = 10$ i $b = 25$ NWD wynosi 5. Korzystając z algorytmu Euklidesa, obliczamy NWD dla tych liczb w następujący sposób:

- od większej liczby odejmujemy mniejszą:
  $b - a = 25 - 10 = 15$,
- zastępujemy większą liczbę otrzymaną różnicą, czyli teraz $a = 10$, $b = 15$,
- od większej liczby odejmujemy mniejszą:
  $b - a = 15 - 10 = 5$,
- zastępujemy większą liczbę otrzymaną różnicą, czyli teraz $a = 10$, $b = 5$,
- od większej liczby odejmujemy mniejszą (teraz większą liczbą jest liczba a):
  $a - b = 10 - 5 = 5$,
- zastępujemy większą liczbę otrzymaną różnicą, czyli teraz $a = 5$, $b = 5$.

Kończymy algorytm, ponieważ $a = b$. Przyjmujemy, że NWD jest równe pierwszej z liczb.

Największy wspólny dzielnik liczb (NWD) możemy wykorzystać do skracania ułamków zwykłych – licznik i mianownik dzielimy przez ich NWD. Jeśli licznik i mianownik ułamka $\frac{10}{25}$ podzielimy przez 5, otrzymamy $\frac{2}{5}$.

Algorytm Euklidesa – wersja z odejmowaniem

Zadanie: Przedstaw w postaci listy kroków algorytm znajdowania największego wspólnego dzielnika dwóch niezerowych liczb naturalnych.
Dane: Liczby naturalne: a, b ($a \neq 0$, $b \neq 0$).
Wynik: Wartość największego wspólnego dzielnika liczb a i b: *NWD*.
Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość liczby a.
3. Wprowadź wartość liczby b.
4. Sprawdź, czy $a = b$: jeżeli tak, idź do kroku 7.
5. Jeżeli $a > b$, to zmiennej a przypisz wartość wyrażenia $a - b$: $a = a - b$; w przeciwnym przypadku zmiennej b przypisz wartość wyrażenia $b - a$: $b = b - a$.
6. Idź do kroku 4.
7. Wyprowadź wynik: *NWD* jest równe a.
8. Zakończ algorytm.

Przykłady wykonania algorytmu

Dla $a = 25$ i $b = 15$

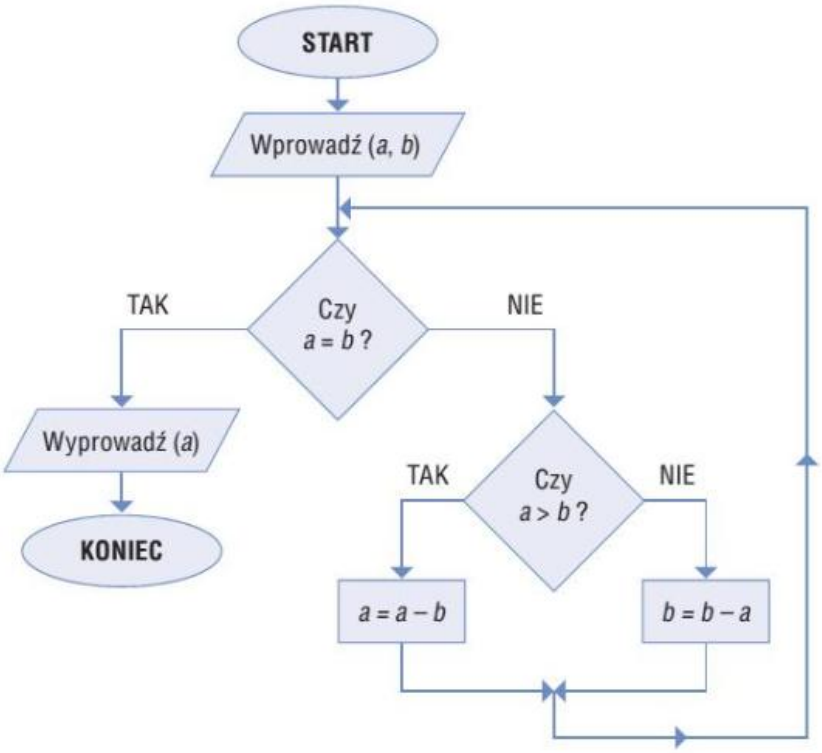
czy $a = b$?	$25 = 15$?	NIE	
czy $a > b$?	$25 > 15$?	TAK	$a = a - b = 25 - 15 = 10$
czy $a = b$?	$10 = 15$?	NIE	
czy $a > b$?	$10 > 15$?	NIE	$b = b - a = 15 - 10 = 5$
czy $a = b$?	$10 = 5$?	NIE	
czy $a > b$?	$10 > 5$?	TAK	$a = a - b = 10 - 5 = 5$
czy $a = b$?	$5 = 5$?	TAK	$\boldsymbol{NWD = a = 5}$

Dla $a = 4$ i $b = 8$

czy $a = b$?	$4 = 8$?	NIE	
czy $a > b$?	$4 > 8$?	NIE	$b = b - a = 8 - 4 = 4$
czy $a = b$?	$4 = 4$?	TAK	$NWD = a = 4$

Ćwiczenie 1. Sprawdzamy działanie algorytmu Euklidesa w wersji z odejmowaniem

Sprawdź działanie algorytmu Euklidesa w wersji z odejmowaniem i wyznacz *NWD* dla liczb: $a = 3$ i $b = 9$ oraz $a = 16$ i $b = 12$. Skorzystaj z podanej listy kroków.



Rys. 1. *Schemat blokowy algorytmu Euklidesa w wersji z odejmowaniem*

W schemacie blokowym algorytmu Euklidesa w wersji z odejmowaniem (rys. 1.) stosujemy dwa bloki warunkowe – jeden zagnieżdżony w drugim. Wykorzystujemy również pętlę.

Ćwiczenie 2. Analizujemy schemat blokowy algorytmu Euklidesa w wersji z odejmowaniem

Sprawdź działanie schematu blokowego pokazanego na rysunku 1. dla przykładowych danych: dla *a* równego *b* i dla *a* różnego od *b*. Wskaż bloki warunkowe. Zwróć uwagę na oznaczenie pętli w schemacie. Odpowiedz na pytania: *Które polecenia się powtarzają? Jaki jest warunek zakończenia pętli?*

Algorytm Euklidesa zaprogramujemy w środowiskach programowania Baltie (rys. 3a) i Scratch (rys. 3b), a następnie w języku C++ (rys. 4a) i języku Python (rys. 4b).

W algorytmie powtarzamy polecenia, dopóki liczby *a* i *b* są różne, dlatego w programie wykorzystamy odpowiednią instrukcję iteracyjną. Liczba powtórzeń (iteracji) nie jest z góry określona, dlatego w języku Scratch zastosujemy polecenie **powtarzaj aż**, a w środowisku programowania Baltie oraz w językach C++ (rys. 2a) i Python (rys. 2b) – instrukcję `while`. Wewnątrz pętli `while` musimy sprawdzać, która z liczb jest większa, dlatego zastosujemy instrukcję warunkową `if`. Użyjemy także niezbędnych zmiennych.

C++

```
while(warunek)
  lista_instrukcji;
kolejna_instrukcja;
```

Rys. 2a. *Ogólna postać instrukcji iteracyjnej* `while` *w języku C++*

Python

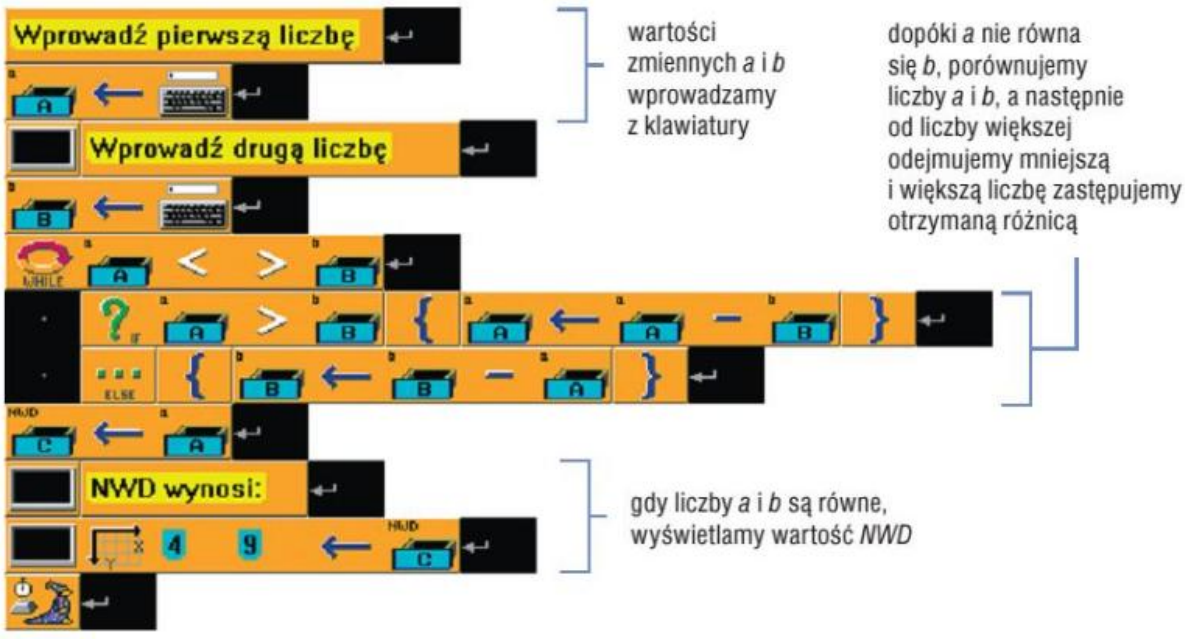
```
while warunek:
  lista_instrukcji
kolejna_instrukcja
```

Rys. 2b. *Ogólna postać instrukcji iteracyjnej* `while` *w języku Python*

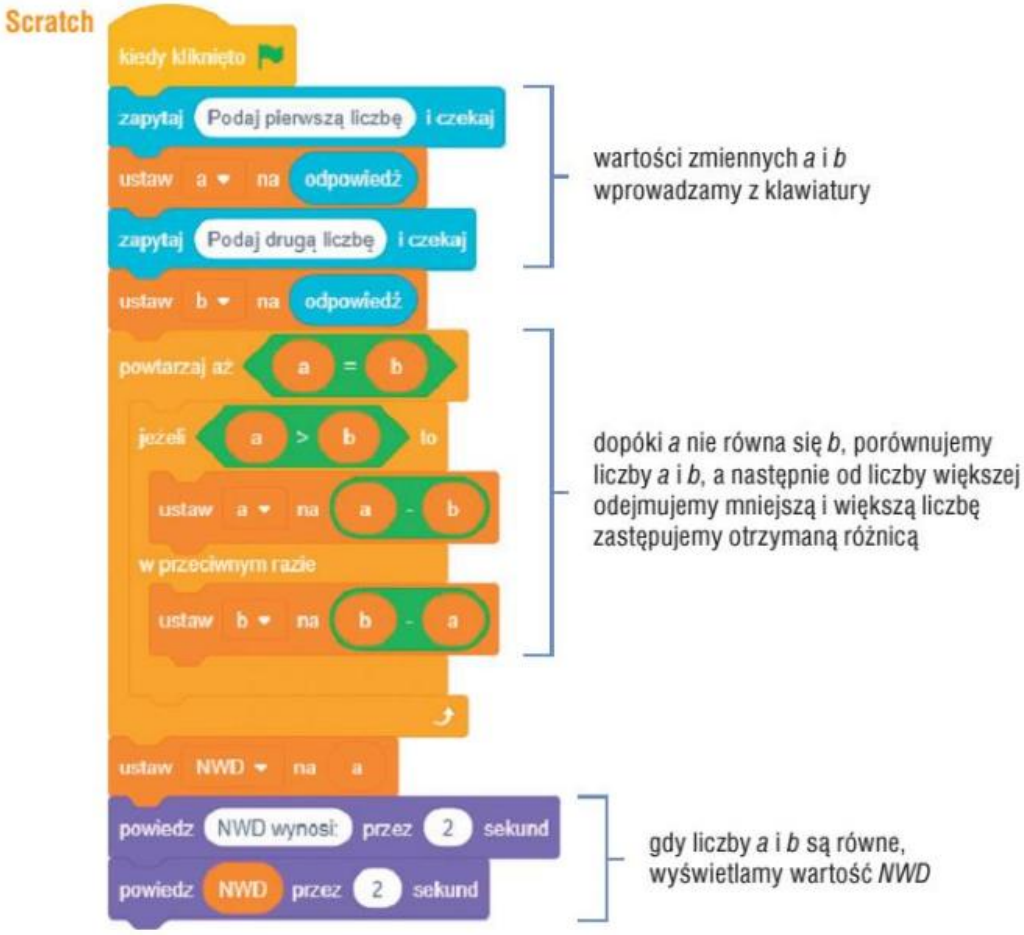
W środowisku programowania Baltie oraz w językach C++ i Python instrukcja `while` działa tak samo. Instrukcje w pętli są wykonywane (*lista_instrukcji*), dopóki *warunek* jest spełniony. Jeśli *warunek* jest fałszywy, jest wykonywana *kolejna_instrukcja*. W bloku *lista_instrukcji* powinna być instrukcja, która zmienia wartość *warunku* – w przeciwnym wypadku pętla nigdy się nie skończy.

Podobnie również działa instrukcja **powtarzaj aż** w języku Scratch.

Baltie



Rys. 3a. *Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Baltie) – ćwiczenie 3.*



Rys. 3b. *Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Scratch) – ćwiczenie 3.*

Ćwiczenie 3. Tworzymy program w środowisku programowania Baltie lub Scratch realizujący algorytm Euklidesa w wersji z odejmowaniem

1. Umieść w obszarze roboczym polecenia z rysunku 3a lub 3b.
2. Zapisz program w pliku pod nazwą *Euklides_odejmowanie*.
3. Uruchom program i sprawdź jego działanie dla różnych danych. Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń. Kiedy zostanie wykonane polecenie przypisania zmiennej *NWD* wartości zmiennej *a*?

Algorytmy, w których wykorzystujemy technikę iteracji (powtarzania tych samych operacji), to **algorytmy iteracyjne**.

C++

Euklides_odejmowanie.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      unsigned int a, b;
7
8      cout << "Podaj dwie liczby: " << endl;
9      cin >> a;
10     cin >> b;
11
12     while (a != b)
13     {
14         if (a > b)
15             a = a - b;
16         else
17             b = b - a;
18     }
19
20     cout << "NWD = " << a << endl;
21     return 0;
22 }
```

typ danych `unsigned int` określa nieujemne liczby całkowite (liczby naturalne)

wartości zmiennych *a* i *b* wprowadzamy z klawiatury

dopóki *a* nie równa się *b*, porównujemy liczby *a* i *b*, a następnie od liczby większej odejmujemy mniejszą i większą liczbę zastępujemy otrzymaną różnicą

gdy liczby *a* i *b* są równe, wyświetlamy wartość *NWD*

Rys. 4a. *Program realizujący algorytm Euklidesa w wersji z odejmowaniem (C++) – ćwiczenie 4.*

Python

Euklides_odejmowanie.py - C:/Python/Euklides_odejmowanie.py (3.9.0)

File Edit Format Run Options Window Help

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))

while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a

print("NWD =", a)

input("\n\nAby zakończyć naciśnij Enter")
```

wartości zmiennych *a* i *b* wprowadzamy z klawiatury

dopóki *a* nie równa się *b*, porównujemy liczby *a* i *b*, a następnie od liczby większej odejmujemy mniejszą i większą liczbę zastępujemy otrzymaną różnicą

gdy liczby *a* i *b* są równe, wyświetlamy wartość *NWD*

Rys. 4b. *Program realizujący algorytm Euklidesa w wersji z odejmowaniem (Python) – ćwiczenie 4.*

Ćwiczenie 4. Zapisujemy w języku C++ lub Python algorytm Euklidesa w wersji z odejmowaniem

1. Program pokazany na rysunku 4a lub 4b realizuje algorytm Euklidesa w wersji z odejmowaniem. Przepisz jeden z tych programów i zapisz go w pliku pod nazwą *Euklides_odejmowanie*. Uruchom kilkakrotnie program, testując go dla różnych wartości zmiennych *a* i *b*.
2. Objaśnij działanie programu, m.in. użycie poszczególnych instrukcji zgodnie z listą kroków lub schematem blokowym algorytmu.
3. Porównaj działanie instrukcji iteracyjnej w programach utworzonych w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

Ćwiczenie 5. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 4. tak, aby szukał NWD dla dziesięciu par liczb *a* i *b* wprowadzanych kolejno z klawiatury i wyświetlał wyniki na ekranie.
2. Zapisz program w pliku pod nazwą *Euklides_odejmowanie_10*.

2. Badanie podzielności liczb naturalnych

W wybranym środowisku programowania chcemy napisać program, realizujący algorytm badania podzielności liczb. W jaki sposób w środowiskach programowania Scratch i Baltie oraz językach C++ i Python obliczyć resztę z dzielenia i sprawdzić, czy jest równa zero?

Jeśli reszta z dzielenia dwóch liczb naturalnych jest równa zero, mówimy, że pierwsza z liczb (dzielna) jest podzielna przez drugą (dzielnik).
W językach programowania resztę z dzielenia możemy wyznaczyć za pomocą operatora modulo (**mod**, **%**)

W języku Scratch do obliczania reszty z dzielenia można wykorzystać **operator mod** (z grupy **Wyrażenia**), który w wersji Scratch 3 zastąpiono elementem **reszta z dzielenia** (rys. 5a).
Na przykład:

- dla $a = 21$ i $b = 4$ reszta z dzielenia a przez b wynosi 1:
  $reszta = a \textbf{ mod } b = 21 \textbf{ mod } 4 = 1$,
- dla $a = 42$ i $b = 7$ reszta z dzielenia a przez b wynosi 0:
  $reszta = a \textbf{ mod } b = 42 \textbf{ mod } 7 = 0$,
- dla $a = 245$ i $b = 8$ reszta z dzielenia a przez b wynosi 5:
  $reszta = a \textbf{ mod } b = 245 \textbf{ mod } 8 = 5$,
- dla $a = 8$ i $b = 16$ reszta z dzielenia a przez b wynosi 8:
  $reszta = a \textbf{ mod } b = 8 \textbf{ mod } 16 = 8$.

O Operator

Symbol określający rodzaj operacji wykonywanej przez komputer.

Scratch

Scratch 2

Scratch 3

Rys. 5a. *Obliczanie reszty z dzielenia (Scratch) – ćwiczenie 6.*

W środowisku Baltie oraz w językach C++ i Python do obliczenia reszty z dzielenia stosujemy operator **%**.
Na przykład:

- dla $a = 28$ i $b = 5$ reszta z dzielenia a przez b wynosi 3:
  reszta = a **%** $b = 28$ **%** $5 = 3$,
- dla $a = 100$ i $b = 5$ reszta z dzielenia a przez b wynosi 0:
  reszta = a **%** $b = 100$ **%** $5 = 0$.
- dla $a = 7$ i $b = 14$ reszta z dzielenia a przez b wynosi 7:
  reszta = a **%** $b = 7$ **%** $14 = 7$.

Baltie 

Rys. 5b. *Obliczanie reszty z dzielenia (Baltie) – ćwiczenie 6.*

Ćwiczenie 6. Tworzymy program w środowisku programowania Baltie lub Scratch obliczający resztę z dzielenia

1. W wybranym środowisku programowania utwórz program obliczający resztę z dzielenia liczby naturalnej *a* przez liczbę naturalną *b* i wyświetlający wynik (wartość zmiennej *reszta*) na ekranie. Liczby *a* i *b* należy wprowadzać z klawiatury.
2. Zapisz program w pliku pod nazwą *Reszta*.

C++
```
reszta = a % b;
```

Rys. 5c. *Obliczanie reszty z dzielenia (C++) – ćwiczenie 7.*

Python
```
reszta = a % b
```

Rys. 5d. *Obliczanie reszty z dzielenia (Python) – ćwiczenie 7.*

Ćwiczenie 7. Piszemy program w języku C++ lub Python obliczający resztę z dzielenia

1. Napisz program w języku C++ lub Python obliczający resztę z dzielenia liczby naturalnej *a* przez liczbę naturalną *b* i wyświetlający wynik (wartość zmiennej *reszta*) na ekranie. Liczby *a* i *b* należy wprowadzać z klawiatury.
2. Zapisz program w pliku pod nazwą *Reszta*.
3. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

Algorytm badania podzielności liczb

Zadanie: Przedstaw w postaci listy kroków algorytm badania podzielności liczby naturalnej *a* przez różną od zera liczbę naturalną *b*.
Dane: Liczby naturalne: a, b ($b \neq 0$).
Wynik: Reszta z dzielenia liczby *a* przez liczbę *b* oraz odpowiedni komunikat: „*b* jest dzielnikiem *a*" lub „*b* nie jest dzielnikiem *a*".
Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość dzielnej *a*.

3. Wprowadź wartość dzielnika *b*.
4. Zmiennej *reszta* przypisz wartość wyrażenia *a* **mod** *b*: *reszta* = *a* **mod** *b*.
5. Jeżeli wartość zmiennej *reszta* jest równa 0, wyprowadź na ekran informację: „*b* jest dzielnikiem *a*", w przeciwnym przypadku – „*b* nie jest dzielnikiem *a*".
6. Zakończ algorytm.

Przykłady wykonania algorytmu

Dla *a* = 15 i *b* = 6

reszta = 15 **mod** 6 = 3

czy *reszta* = 0? NIE 6 *nie jest dzielnikiem* 15

Dla *a* = 18 i *b* = 6

reszta = 18 **mod** 6 = 0

czy *reszta* = 0? TAK 6 *jest dzielnikiem* 18

Ćwiczenie 8. Analizujemy listę kroków algorytmu badania podzielności liczb

1. Zapoznaj się z listą kroków algorytmu badania podzielności liczb.
2. Wykonaj algorytm badania podzielności dla kilku par liczb.

Ćwiczenie 9. Tworzymy program w środowisku programowania Baltie lub Scratch realizujący algorytm badania podzielności liczb naturalnych

1. Otwórz program *Reszta* zapisany w ćwiczeniu 6. Zmodyfikuj go tak, aby realizował algorytm badania podzielności liczb zgodnie z podaną listą kroków. Jaką instrukcję zastosujesz, aby zrealizować punkt 5. listy kroków? Uzasadnij odpowiedź.
2. Zapisz program w pliku pod nazwą *Podzielność*.
3. Uruchom program i sprawdź jego działanie dla kilku różnych par liczb. Zakładamy, że wprowadzamy prawidłowe dane, czyli dzielna i dzielnik to liczby naturalne i dzielnik jest różny od zera.

Ćwiczenie 10. Zapisujemy w języku C++ lub Python algorytm badania podzielności liczb naturalnych

1. Otwórz program *Reszta* zapisany w ćwiczeniu 7. Zmodyfikuj go tak, aby realizował algorytm badania podzielności liczb zgodnie z podaną listą kroków. Jaką instrukcję zastosujesz, aby zrealizować punkt 5. listy kroków? Uzasadnij odpowiedź.
2. Zapisz program w pliku pod nazwą *Podzielność*.
3. Uruchom program i sprawdź jego działanie dla kilku różnych par liczb. Zakładamy, że wprowadzamy prawidłowe dane, czyli dzielna i dzielnik to liczby naturalne i dzielnik jest różny od zera.
4. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

Ćwiczenie 11. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 10. tak, aby sprawdzał podzielność liczb tylko dla dzielnika *b* różnego od zera, a w przypadku gdy dzielnik *b* jest równy zero – wyświetlał komunikat „Nie wolno dzielić przez zero".
2. Zapisz program w pliku pod nazwą *Podzielność_zmodyf*.

3. Realizacja algorytmu Euklidesa w wersji z dzieleniem

W wersji z odejmowaniem algorytmu Euklidesa odejmowanie posłużyło znalezieniu reszty z dzielenia dwóch liczb. W jaki sposób możemy zamiast odejmowania zastosować **dzielenie z resztą**?

Wersja z dzieleniem algorytmu Euklidesa polega na wykonywaniu kolejnych dzieleń liczby a (dzielnej) przez liczbę b (dzielnik), aż reszta z dzielenia osiągnie wartość 0. Na przykład dla liczb $a = 25$ i $b = 10$ NWD obliczamy w następujący sposób:

- dzielimy liczbę a przez b:
  $a / b = 25 / 10 = 2$ reszta 5,
- liczbę a zastępujemy liczbą b, a liczbę b resztą z poprzedniego dzielenia,
  czyli $a = 10$, $b = 5$,
- dzielimy *liczbę* a przez b:
  $a / b = 10 / 5 = 2$ reszta 0,
- liczbę a zastępujemy liczbą b, a liczbę b resztą z poprzedniego dzielenia,
  czyli $a = 5$, $b = 0$.

Kończymy algorytm, ponieważ reszta z dzielenia wynosi 0. Przyjmujemy, że NWD jest równe ostatniej niezerowej reszcie, czyli NWD $= a = 5$.

Algorytm Euklidesa – wersja z dzieleniem

Zadanie: Przedstaw w postaci listy kroków algorytm znajdowania największego wspólnego dzielnika dwóch niezerowych liczb naturalnych w wersji z dzieleniem.
Dane: Liczby naturalne: a, b $(a \neq 0)$, $(b \neq 0)$.
Wynik: Wartość największego wspólnego dzielnika liczb a i b: *NWD*.
Lista kroków:

1. Zacznij algorytm.
2. Wprowadź wartość liczby a.
3. Wprowadź wartość liczby b.
4. Sprawdź, czy $b = 0$: jeżeli tak, idź do kroku 9.
5. Zmiennej *dzielnik* przypisz wartość zmiennej b: *dzielnik* $= b$.
6. Zmiennej b przypisz wartość reszty z dzielenia a przez b: $b = a$ **mod** b.
7. Zmiennej a przypisz wartość zmiennej *dzielnik*: $a =$ *dzielnik*.
8. Idź do kroku 4.
9. Wyprowadź wynik: *NWD* jest równe a.
10. Zakończ algorytm.

Uwaga: Zmienna *dzielnik* przechowuje dzielnik w trakcie wykonywania obliczeń.

Przykłady wykonania algorytmu

Dla $a = 9$ i $b = 3$

czy $b = 0$?	NIE	
dzielnik $= b = 3$		
$b = a$ **mod** $b = 9$ **mod** $3 = 0$		
$a =$ *dzielnik* $= 3$		
czy $b = 0$?	TAK	***NWD*** $= a = \mathbf{3}$

Dla $a = 28$ i $b = 35$

czy $b = 0$? NIE
$dzielnik = b = 35$
$b = a$ **mod** $b = 28$ **mod** $35 = 28$
$a = dzielnik = 35$
czy $b = 0$? NIE
$dzielnik = b = 28$
$b = a$ **mod** $b = 35$ **mod** $28 = 7$
$a = dzielnik = 28$
czy $b = 0$? NIE
$dzielnik = b = 7$
$b = a$ **mod** $b = 28$ **mod** $7 = 0$
$a = dzielnik = 7$
czy $b = 0$? TAK $NWD = a = 7$

Uwaga

Jeśli pod tę samą zmienną podstawimy inną wartość, poprzednia wartość znika, dlatego w niektórych przypadkach musimy wartość danej zmiennej przechować w tzw. **zmiennej pomocniczej**. W omawianym algorytmie jest to zmienna *dzielnik*, w której pamiętamy aktualny dzielnik.

Ćwiczenie 12. Sprawdzamy działanie algorytmu Euklidesa w wersji z dzieleniem

Sprawdź działanie algorytmu Euklidesa w wersji z dzieleniem i wyznacz *NWD* dla liczb: $a = 56$ i $b = 6$ oraz $a = 16$ i $b = 24$. Skorzystaj z podanej listy kroków.

W algorytmie powtarzamy polecenia, dopóki liczba *b* jest różna od zera, dlatego w programach wykorzystamy odpowiednią instrukcję iteracyjną. W języku Scratch zastosujemy polecenie **powtarzaj aż**, a w środowisku programowania Baltie oraz w językach C++ i Python – instrukcję `while`.

Ćwiczenie 13. Tworzymy program w środowisku programowania Baltie lub Scratch realizujący algorytm Euklidesa w wersji z dzieleniem

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 6a lub 6b. Uzupełnij je o polecenia wprowadzania z klawiatury danych *a* i *b* oraz wyprowadzania wyniku zgodnie z podaną listą kroków algorytmu Euklidesa w wersji z dzieleniem. Utwórz cztery zmienne: *a*, *b*, *dzielnik* i *NWD*.
2. Zapisz program w pliku pod nazwą *Euklides_dzielenie*.

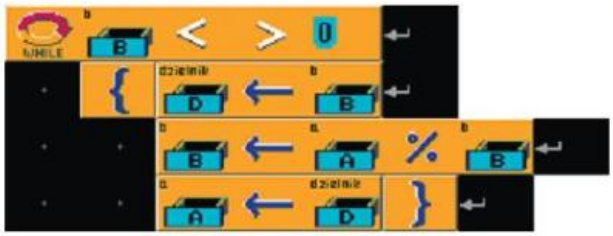
Wskazówka: Wprowadzanie wartości zmiennych, wyświetlanie wyniku i napisów możesz wzorować na programie realizującym algorytm Euklidesa w wersji z odejmowaniem (rys. 3a lub 3b).

Ćwiczenie 14. Zapisujemy w języku C++ lub Python algorytm Euklidesa w wersji z dzieleniem

1. Napisz w edytorze kodu źródłowego instrukcje pokazane na rysunku 7a lub 7b. Uzupełnij je o polecenia wprowadzania z klawiatury danych *a* i *b* oraz wyprowadzania wyniku zgodnie z podaną listą kroków algorytmu Euklidesa w wersji z dzieleniem.
2. Zapisz program w pliku pod nazwą *Euklides_dzielenie*.
3. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

Wskazówka: Wprowadzanie wartości zmiennych, wyświetlanie wyniku i napisów możesz wzorować na programie realizującym algorytm Euklidesa w wersji z odejmowaniem (rys. 4a lub 4b).

Baltie



dopóki *b* jest różne od 0, przypisujemy: zmiennej *dzielnik* wartość zmiennej *b*, zmiennej *b* resztę z dzielenia *a* przez *b*, zmiennej *a* wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik* przechowuje aktualny dzielnik

Rys. 6a. *Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Baltie) – ćwiczenie 13.*

Scratch

dopóki *b* jest różne od 0, przypisujemy: zmiennej *dzielnik* wartość zmiennej *b*, zmiennej *b* resztę z dzielenia *a* przez *b*, zmiennej *a* wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik* przechowuje aktualny dzielnik

Rys. 6b. *Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Scratch) – ćwiczenie 13.*

C++

```
while( b != 0 )
{
    dzielnik = b;
    b = a % b;
    a = dzielnik;
}
```

dopóki *b* jest różne od 0, przypisujemy: zmiennej *dzielnik* wartość zmiennej *b*, zmiennej *b* resztę z dzielenia *a* przez *b*, zmiennej *a* wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik* przechowuje aktualny dzielnik

Rys. 7a. *Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (C++) – ćwiczenie 14.*

Python

```
while b != 0:
    dzielnik = b
    b = a % b
    a = dzielnik
```

dopóki *b* jest różne od 0, przypisujemy: zmiennej *dzielnik* wartość zmiennej *b*, zmiennej *b* resztę z dzielenia *a* przez *b*, zmiennej *a* wartość zmiennej *dzielnik*

w trakcie wykonywania obliczeń zmienna *dzielnik* przechowuje aktualny dzielnik

Rys. 7b. *Fragment programu realizującego algorytm Euklidesa w wersji z dzieleniem (Python) – ćwiczenie 14.*

4. Algorytm wyodrębniania cyfr danej liczby

W wybranym środowisku programowania chcemy napisać program realizujący algorytm wyodrębniania cyfr danej liczby. Jak to zrobić w środowiskach programowania Scratch i Baltie oraz językach C++ i Python, wykorzystując obliczanie reszty z dzielenia?

W systemie dziesiętnym używamy potęg liczby 10. Liczbę naturalną w systemie dziesiętnym możemy rozpisać w następujący sposób:

$$647 = 6 \cdot 100 + 4 \cdot 10 + 7 \cdot 1 = 6 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0.$$

Cyfry danej liczby naturalnej możemy wyodrębniać od najmniej znaczącej (dla liczby 647 cyfrą tą jest cyfra jedności 7) lub od najbardziej znaczącej (dla 647 – to cyfra setek 6).

Do obliczania reszty z dzielenia zastosujemy operator **modulo (mod, %)**.

Aby wyodrębnić cyfry danej liczby naturalnej (od najmniej znaczącej), powtarzamy obliczanie reszty z dzielenia tej liczby przez 10 (pierwsza reszta jest cyfrą jedności), następnie od liczby odejmujemy resztę i wynik dzielimy przez 10, aż do otrzymania wyniku odejmowania równego zero.

Przykład wykonania algorytmu

Dla *liczba* = 647

cyfra = *liczba* **mod** 10 = 647 **mod** 10 = **7**
liczba = (*liczba* – *cyfra*) / 10 = (647 – 7) / 10 = 640 / 10 = 64
czy *liczba* = 0? NIE
cyfra = *liczba* **mod** 10 = 64 **mod** 10 = **4**
liczba = (*liczba* – *cyfra*) / 10 = (64 – 4) / 10 = 60 / 10 = 6
czy *liczba* = 0? NIE
cyfra = *liczba* **mod** 10 = 6 **mod** 10 = **6**
liczba = (*liczba* – *cyfra*) / 10 = (6 – 6) / 10 = 0 / 10 = 0
czy *liczba* = 0? TAK

Uwagi: Zmienna *cyfra* służy do pamiętania wartości kolejnych cyfr (tu: reszt z dzielenia kolejnych wartości zmiennej *liczba* przez 10). Wartość zmiennej *liczba*, której cyfry wyodrębniamy, zmienia się w czasie wykonywania algorytmu.

Ćwiczenie 15. Piszemy specyfikację zadania i listę kroków algorytmu wyodrębniania cyfr danej liczby

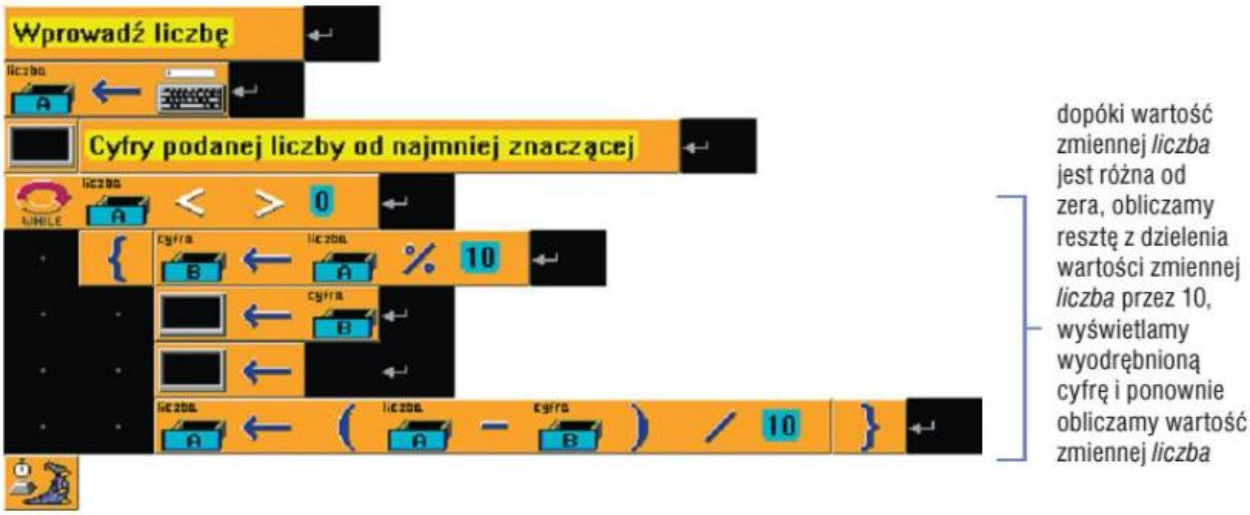
1. Zapoznaj się z opisem algorytmu wyodrębniania cyfr danej liczby i z podanym przykładem wykonania algorytmu. Wykonaj algorytm dla liczby 8945.
2. Zapisz w zeszycie przedmiotowym specyfikację zadania i listę kroków tego algorytmu.

Wskazówka: Wzoruj się na zapisie specyfikacji zadań i na listach kroków innych algorytmów opisanych w tym temacie.

W algorytmie wyodrębniania cyfr danej liczby powtarzamy polecenia, dopóki wartość zmiennej *liczba* jest różna od zera. W programach wykorzystamy instrukcje iteracyjne, których używa się, gdy liczba powtórzeń (iteracji) nie jest z góry określona.

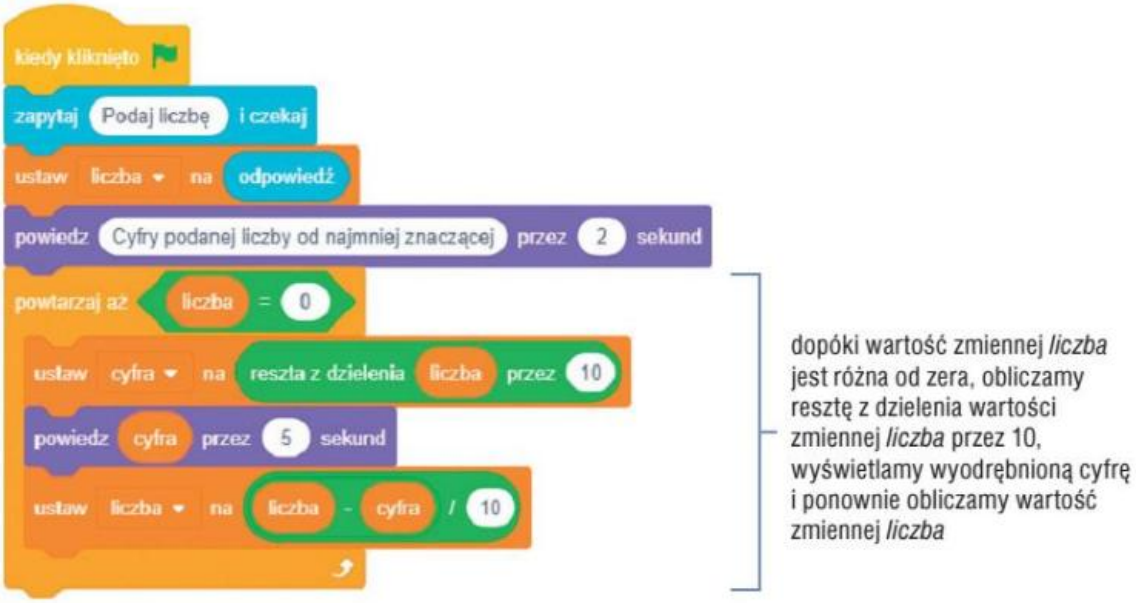
W środowisku programowania Baltie polecenie `while`, a w języku Scratch – **powtarzaj aż**.

Baltie



Rys. 8a. *Fragment programu realizującego algorytm wyodrębniania cyfr danej liczby (Baltie) – ćwiczenie 16.*

Scratch



Rys. 8b. *Fragment programu realizującego algorytm wyodrębniania cyfr danej liczby (Scratch) – ćwiczenie 16.*

Ćwiczenie 16. Tworzymy program w środowisku programowania Baltie lub Scratch realizujący algorytm wyodrębniania cyfr danej liczby

1. Umieść w obszarze roboczym polecenia pokazane na rysunku 8a lub 8b.
2. Zapisz program w pliku pod nazwą *Cyfry*.
3. Uruchom program i sprawdź jego działanie dla różnych danych.
4. Objaśnij działanie programu, m.in. uzasadnij użycie poszczególnych poleceń.
5. Odpowiedz na pytania: *Ile razy powtarzają się polecenia umieszczone w pętli dla liczby trzycyfrowej, a ile – dla jednocyfrowej? Dla jakiej wartości zmiennej* liczba *kończą się obliczenia?* Uzasadnij odpowiedzi.

Wskazówka: Aby w środowisku Baltie cyfry były wyświetlane z pewnym odstępem, można rozdzielić je czarnym przedmiotem  w instrukcji przypisania:

 Czarny przedmiot to pierwszy przedmiot umieszczony w pierwszym wierszu w banku 0. Należy odróżniać go od pustego pola  (element z białą kropeczką pośrodku).

W programach w językach C++ i Python zastosujemy odpowiednie instrukcje iteracyjne, których używamy, gdy liczba powtórzeń (iteracji) nie jest z góry określona. W języku C++ wykorzystamy instrukcję **`do ... while`**, w której warunek (czy *liczba* jest różna od 0?) sprawdzamy na końcu. W języku Python zastosujemy poznaną instrukcję iteracyjną **`while`**.

C++

```
do
  lista_instrukcji;
while(warunek);
kolejna_instrukcja;
```

Rys. 9. *Ogólna postać instrukcji* **`do ... while`** *w języku C++*

Instrukcje w pętli (*lista_instrukcji*) są wykonywane, dopóki warunek jest prawdziwy. Jeśli warunek jest fałszywy – wykonywana jest *kolejna_instrukcja*. Niezależnie od wartości początkowej warunku, *lista_instrukcji* zostanie wykonana przynajmniej raz. Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja **`do ... while`**) lub więcej instrukcji ujętych w blok **`{}`**.

Ćwiczenie 17. Zapisujemy w języku C++ lub Python algorytm wyodrębniania cyfr danej liczby

1. Programy pokazane na rysunkach 10a i 10b realizują algorytm wyodrębniania cyfr danej liczby, zaczynając od ostatniej cyfry (cyfry jedności). Przepisz wybrany program i zapisz go w pliku pod nazwą *Cyfry*.
2. Uruchom kilkakrotnie program, testując go dla różnych wartości zmiennej *liczba*.
3. Objaśnij działanie programu, m.in. użycie poszczególnych instrukcji. Porównaj programy utworzone w środowisku programowania Baltie i języku C++ lub w języku Scratch i języku Python. Co zauważasz? Uzasadnij odpowiedź.

C++

[*] Cyfry.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    unsigned int liczba, cyfra;

    cout << "Podaj liczbe: ";
    cin >> liczba;
    cout << "Cyfry liczby " << liczba << " od ostatniej cyfry:" << endl;
    do
    {
        cyfra = liczba % 10;
        cout << cyfra << endl;
        liczba = (liczba - cyfra) / 10;
    }
    while (liczba != 0 );

    return 0;
}
```

dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

***Rys. 10a**. Program realizujący algorytm wyodrębniania cyfr danej liczby (C++) – ćwiczenie 17.*

Python

Cyfry.py - C:\Python\Cyfry.py (3.9.0)

File Edit Format Run Options Window Help

```python
liczba = int(input("Podaj liczbę: "))

print("Cyfry liczby", liczba, "od ostatniej:")

if liczba == 0:
    print(liczba)
else:
    while liczba != 0:
        cyfra = liczba % 10
        print(cyfra)
        liczba = (liczba - cyfra) // 10

input("\n\nAby zakończyć naciśnij Enter")
```

dopóki wartość zmiennej *liczba* jest różna od zera, obliczamy resztę z dzielenia wartości zmiennej *liczba* przez 10, wyświetlamy wyodrębnioną cyfrę i ponownie obliczamy wartość zmiennej *liczba*

***Rys. 10b**. Program realizujący algorytm wyodrębniania cyfr danej liczby (Python) – ćwiczenie 17.*

C++

C:\C++\Cyfry.exe

```
Podaj liczbe: 746
Cyfry liczby 746 od ostatniej:
6
4
7
```

***Rys. 11a**. Wynik działania programu realizującego algorytm wyodrębniania cyfr danej liczby (C++) – ćwiczenie 17.*

Python

C:\WINDOWS\py.exe

```
Podaj liczbę: 746
Cyfry liczby 746 od ostatniej:
6
4
7
```

***Rys. 11b**. Wynik działania programu realizującego algorytm wyodrębniania cyfr danej liczby (Python) – ćwiczenie 17.*

Ćwiczenie 18. Modyfikujemy program

1. Zmodyfikuj program zapisany w ćwiczeniu 17, aby wyodrębniał cyfry dziesięciu liczb wprowadzanych z klawiatury i wyświetlał kolejne wyniki na ekranie. Zapisz program w pliku pod nazwą *Cyfry_10*.
2. Odpowiedz na pytanie: *Ile razy powtarzają się polecenia w pętli* `do ... while` *lub* `while` *dla liczby trzycyfrowej, a ile razy – dla jednocyfrowej?* Uzasadnij odpowiedź.
3. C++ Zamień instrukcję: `liczba = (liczba - cyfra) / 10;`
   na instrukcję: `liczba = liczba / 10;`
   Python Zamień instrukcję: `liczba = (liczba - cyfra) // 10`
   na instrukcję: `liczba = liczba // 10.`
   Czy zmieniło się działanie programu? Uzasadnij odpowiedź.

Warto zapamiętać

- Największy wspólny dzielnik (NWD) dwóch liczb naturalnych możemy znaleźć, stosując algorytm Euklidesa. Algorytm ten możemy zrealizować w dwóch wersjach: z odejmowaniem i z dzieleniem.
- Liczba *b* jest dzielnikiem liczby *a*, jeśli reszta z dzielenia *a* przez *b* jest równa zero. W programie badania podzielności liczb możemy wykorzystać operator modulo do obliczania reszty z dzielenia, np. **%**, **mod**.
- W algorytmie wyodrębniania cyfr danej liczby naturalnej wykorzystujemy dzielenie z resztą danej liczby przez dziesięć. Reszty z dzielenia są kolejnymi cyframi liczby.

Pytania i polecenia

1. Do czego służy algorytm Euklidesa?
2. Na czym polega algorytm Euklidesa w wersji z odejmowaniem?
3. Wyjaśnij na przykładzie liczb 36 i 8, dlaczego warunkiem zakończenia algorytmu Euklidesa w wersji z odejmowaniem jest $a = b$.
4. Na czym polega algorytm badania podzielności liczb naturalnych?
5. W jaki sposób można obliczyć w wybranym środowisku programowania resztę z dzielenia dwóch liczb naturalnych? Podaj przykład. Pamiętaj, że dzielnik nie może być równy zero.
6. Jak sprawdzić, czy liczba *y* jest dzielnikiem liczby *x*? Przedstaw na przykładzie.
7. Na czym polega algorytm Euklidesa w wersji z dzieleniem?
8. Czym różni się wersja algorytmu Euklidesa z odejmowaniem od wersji z dzieleniem? Wyjaśnij na przykładzie liczb 36 i 8.
9. Na czym opiera się idea algorytmu wyodrębniania cyfr danej liczby?

Zadania

1. Napisz listę kroków algorytmu sprawdzania, czy dana liczba naturalna jest parzysta. Jeśli liczba jest parzysta, wyświetlaj napis „Podana liczba jest parzysta", a gdy nieparzysta – „Podana liczba jest nieparzysta".
   Wskazówka: Liczba parzysta jest podzielna przez 2.

2. Narysuj schemat blokowy algorytmu sprawdzania, czy dana liczba naturalna jest parzysta.
3. Napisz program sprawdzający, czy wprowadzona z klawiatury liczba jest parzysta. Skorzystaj z listy kroków utworzonej w zadaniu 1. i schematu blokowego narysowanego w zadaniu 2. Zapisz program w pliku pod nazwą *Parzyste*.
4. Narysuj schemat blokowy algorytmu badania podzielności liczb. Skorzystaj z listy kroków podanej w punkcie 2. tematu.
5. Napisz program sprawdzający, czy wprowadzona z klawiatury liczba jest podzielna przez 3. Jeśli tak, wyprowadź komunikat: „Podana liczba jest podzielna przez 3". Jeśli nie – „Podana liczba nie jest podzielna przez 3". Dodatkowo w komunikacie możesz wyświetlać wartość tej liczby. Zapisz program w pliku pod nazwą *Podzielność_3*.
6. Dysponujesz pewną kwotą (*kwota*) i planujesz zakupy. Ceny kolejnych wydatków (*cena*) wprowadzasz z klawiatury. Zakupy planujesz dotąd, aż wartość zmiennej *kwota* będzie równa lub mniejsza od zera. Napisz program realizujący to zadanie. Wartość zmiennych *kwota* i *cena* mają być wprowadzane z klawiatury po uruchomieniu programu. Zapisz program w pliku pod nazwą *Zakupy*.
   Wskazówka: Zastanów się, którą instrukcję iteracyjną najlepiej zastosować.

Dla zainteresowanych

7. Na podstawie listy kroków podanej w punkcie 3. tematu narysuj schemat blokowy algorytmu Euklidesa w wersji z dzieleniem.
8. Napisz program zliczający oddzielnie liczby parzyste i nieparzyste spośród *n* liczb naturalnych wprowadzanych z klawiatury. Wartości liczników *licznik_p* (liczby parzyste) i *licznik_np* (liczby nieparzyste) wyświetl na ekranie z odpowiednim komunikatem. Zapisz program w pliku pod nazwą *Liczby*.
   Wskazówka: Aby zliczać wprowadzane liczby (np. parzyste), można zastosować instrukcję: `licznik_p = licznik_p + 1`.
9. Zmodyfikuj program *Zakupy* zapisany w zadaniu 6. w następujący sposób: jeśli spróbujemy przekroczyć dostępną kwotę, czyli wydać więcej, program na to nie pozwoli, tylko zapyta o kolejną wartość zmiennej *cena*, więc musimy zaplanować zakup czegoś tańszego. Program zakończy działanie, gdy zmienna *kwota* będzie równa zero. Zapisz plik pod tą samą nazwą.
10. Zastanów się, jak sprawdzić podzielność jednej liczby przez inną bez konieczności obliczania reszty z dzielenia. Zapisz algorytm w postaci listy kroków, schematu blokowego lub programu.
11. Narysuj schemat blokowy algorytmu wyodrębniania cyfr danej liczby.
12. Napisz program wyodrębniania cyfr danej liczby od najbardziej znaczącej do najmniej znaczącej. Zapisz program w pliku pod nazwą *Cyfry1*.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Euklides z Aleksandrii był greckim matematykiem, który ok. 300 lat p.n.e. stworzył swoje główne dzieło pt. „Elementy". W dziele tym opisał m.in. sposób obliczania największej wspólnej miary dwóch odcinków (czyli największego wspólnego dzielnika dwóch liczb naturalnych). W tamtym czasie nie znano pojęcia *algorytmu*, jednak opisana przez Euklidesa metoda dzisiaj mogłaby zostać uznana za algorytm. „Elementy" były przez wiele wieków jedynym rzetelnym podręcznikiem do nauki geometrii.