7. Algorytm Euklidesa i działania na ułamkach

Algorytm Euklidesa, czyli metoda znajdowania największego wspólnego dzielnika dwóch liczb, jest uważany za najstarszy znany algorytm. Znalazł on zastosowanie w rozwiązywaniu różnych problemów, m.in. w kryptografii. W tym temacie poznasz różne wersje algorytmu Euklidesa oraz użyjesz go do wykonywania działań na liczbach wymiernych.

Cele lekcji

- Poznasz interpretację geometryczną algorytmu Euklidesa z wykorzystaniem odejmowania.
- Zapiszesz iteracyjnie algorytm Euklidesa w dwóch wersjach: z wykorzystaniem odejmowania i dzielenia.
- Dowiesz się, czym jest struktura w języku C++.
- Zastosujesz strukturę do reprezentacji liczb wymiernych.
- Wykorzystasz algorytm Euklidesa w działaniach na liczbach wymiernych.
- Nauczysz się używać funkcji typu `void`.
- Dowiesz się, czym są zmienne lokalne i globalne.
- Zrozumiesz, na czym polega przekazanie parametrów przez wartość.

Największy wspólny dzielnik (NWD) dwóch liczb całkowitych dodatnich to największa liczba całkowita, która dzieli obie te liczby bez reszty.

Największy wspólny dzielnik (NWD) dwóch liczb całkowitych dodatnich znajdowaliście już w szkole podstawowej na lekcjach matematyki, np. skracając ułamki. Aby wyznaczyć NWD, wystarczy rozłożyć obie liczby na czynniki pierwsze i obliczyć iloczyn czynników powtarzających się w obu rozkładach. Jest to jednak metoda mało wygodna do implementacji, a w przypadku dużych liczb – czasochłonna. Dlatego warto skorzystać z algorytmu opisanego przez Euklidesa ok. 300 r. p.n.e.

7.1. Algorytm Euklidesa z wykorzystaniem odejmowania

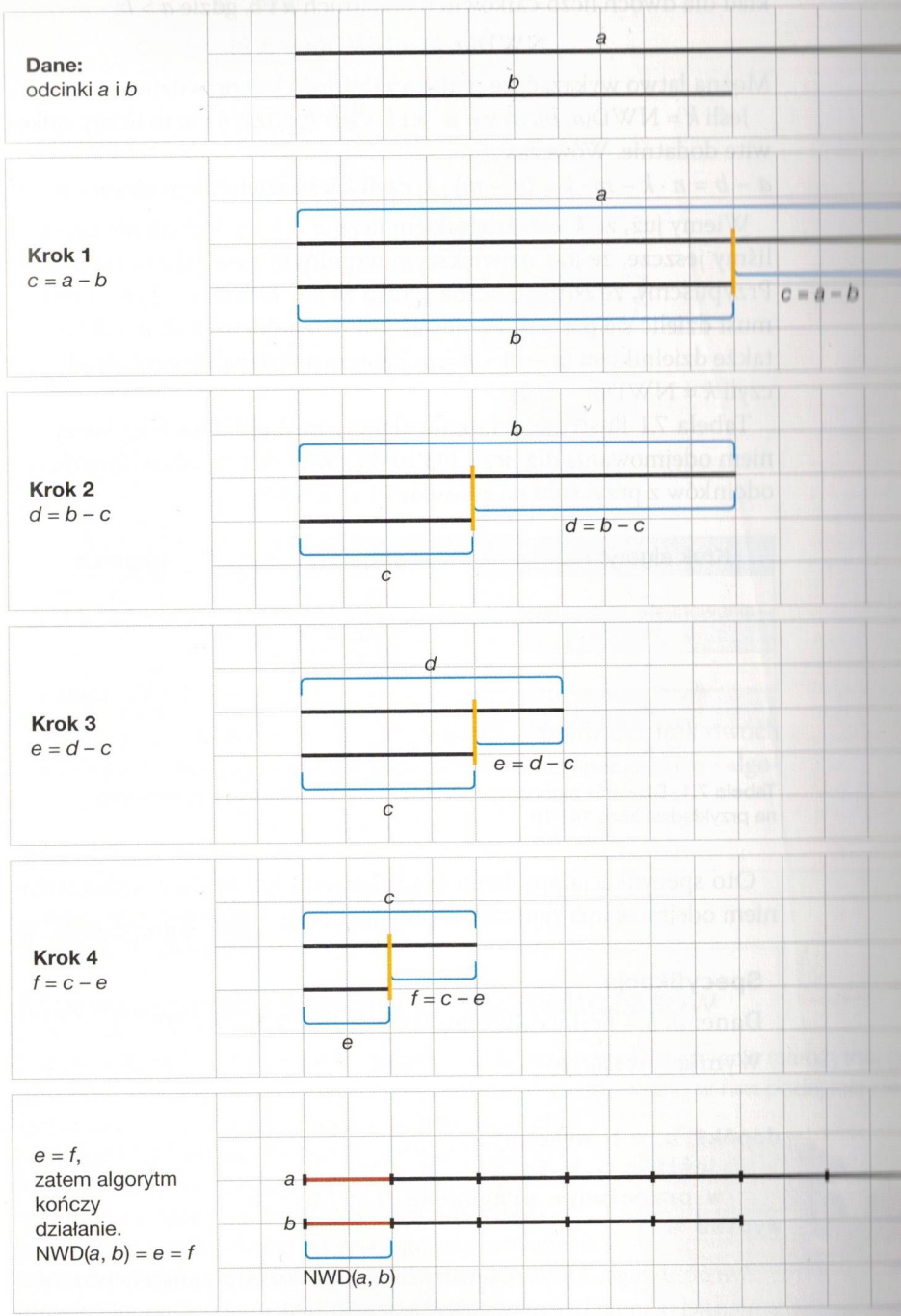
Algorytm Euklidesa

Euklides swój algorytm, zwany obecnie **algorytmem Euklidesa**, opisał pierwotnie dla odcinków, a nie dla liczb. Znajdował najdłuższy odcinek jednostkowy, który wyznaczał miarę dwóch odcinków, czyli odkładał się w każdym z odcinków całkowitą liczbę razy.

Interpretacja geometryczna algorytmu Euklidesa

Załóżmy, że mamy dwa odcinki *a* i *b* różnej długości. Najpierw od dłuższego odcinka odejmujemy krótszy. Następnie wykonujemy podobne odejmowanie, przy czym bierzemy pod uwagę krótszy z odcinków z poprzedniego kroku oraz różnicę odcinków z poprzedniego kroku. Odejmowanie odpowiednich odcinków powtarzamy tak długo, aż

Rysunek 7.1 przedstawia przykładową ilustrację działania algorytmu. Zauważ, że jeśli wyjściowe odcinki byłyby tej samej długości, nie musielibyśmy nic robić – poszukiwany odcinek miałby taką samą długość jak każdy z danych odcinków.



Algorytm Euklidesa, który znajduje największy wspólny dzielnik z wykorzystaniem odejmowania, opiera się na następującej własności: NWD dwóch liczb całkowitych dodatnich jest taki sam jak NWD mniejszej z tych liczb oraz różnicy większej i mniejszej liczby. Na przykład dla dwóch liczb całkowitch dodatnich a i b, gdzie $a > b$:

$$\mathrm{NWD}(a, b) = \mathrm{NWD}(a - b, b)$$

Można łatwo wykazać, że podana zależność jest prawdziwa.

Jeśli $k = \mathrm{NWD}(a, b)$, to $a = n \cdot k$ i $b = m \cdot k$, gdzie n i m to liczby całkowite dodatnie. Wówczas:

$a - b = n \cdot k - m \cdot k = (n - m) \cdot k$, czyli k jest dzielnikiem różnicy $a - b$.

Wiemy już, że k jest dzielnikiem liczb $a - b$ i b. Jednak nie wykazaliśmy jeszcze, że jest największym wspólnym dzielnikiem tych liczb. Przypuśćmy, że istnieje liczba s, taka że $s = \mathrm{NWD}(a - b, b)$. Liczba s musi dzielić się przez k. Ponieważ jest dzielnikiem liczb $a - b$ i b, jest także dzielnikiem $(a - b) + b = a$. Zatem s jest dzielnikiem k. Stąd $s = k$, czyli $k = \mathrm{NWD}(a - b, b)$.

Tabela 7.1 ilustruje działanie algorytmu Euklidesa z wykorzystaniem odejmowania dla liczb 14 i 10 (liczby te odpowiadają długościom odcinków z przykładu na rysunku 7.1 ze s. 103).

Krok algorytmu	Liczba *a*	Liczba *b*
–	14	10
1	14 – 10 = 4	10
2	4	10 – 4 = 6
3	4	6 – 4 = 2
4	4 – 2 = 2	2

Tabela 7.1. Działanie algorytmu Euklidesa z wykorzystaniem odejmowania na przykładzie liczb 14 i 10

Oto specyfikacja problemu oraz algorytm Euklidesa z wykorzystaniem odejmowania zapisany w pseudokodzie:

Specyfikacja

Dane: a, b – liczby całkowite, $0 < a \leqslant 2^{31} - 1$, $0 < b \leqslant 2^{31} - 1$.

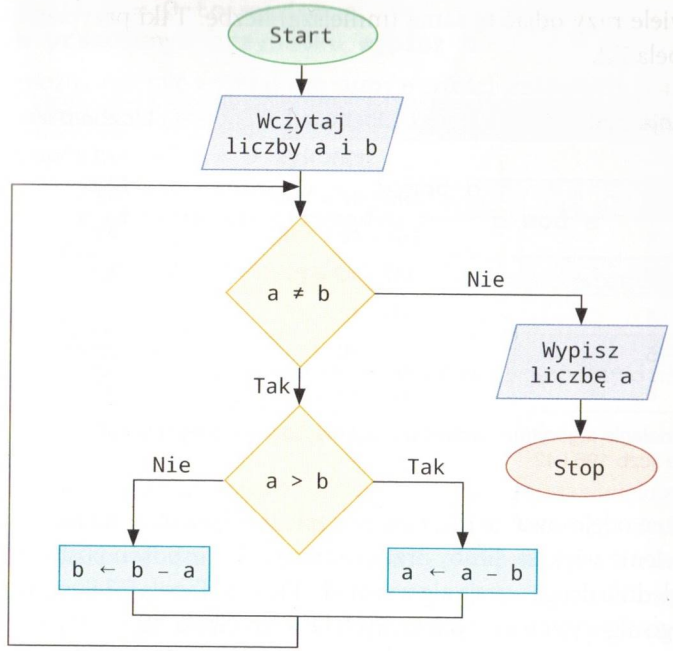
Wynik: $\mathrm{NWD}(a, b)$

```
dopóki a ≠ b wykonuj
    jeśli a > b to a ← a - b
    w przeciwnym przypadku b ← b - a
wypisz a
```

Zwróć uwagę, że w ostatniej linii pseudokodu zamiast wartości zmiennej a moglibyśmy wypisać wartość zmiennej b – po wykonaniu instrukcji w ostatnim kroku pętli

Jeśli chcielibyśmy przedstawić algorytm poszukiwania NWD dwóch liczb całkowitych dodatnich a i b w postaci **schematu blokowego**, wyglądałby on np. tak jak poniżej.

Schemat blokowy, s. 15



Rys. 7.2. Schemat blokowy algorytmu Euklidesa z wykorzystaniem odejmowania

Ćwiczenie 1

Napisz program wyznaczający największy wspólny dzielnik dwóch wczytanych liczb zgodnie z podanymi wcześniej specyfikacją i algorytmem.

A to ciekawe

Matematyczny bestseller wszechczasów

Elementy Euklidesa to traktat matematyczny, który opisuje osiągnięcia starożytnej geometrii. Jest on uznawany za dzieło wyjątkowe, ponieważ zastosowano w nim podejście dedukcyjne (od ogółu do szczegółu), precyzję definicji i dowodów twierdzeń. W traktacie znalazł się również algorytm, który jest wykorzystywany we współczesnej informatyce. Mimo że dzieło Euklidesa powstało ok. 300 r. p.n.e., używano go w kolejnych przekładach jako podręcznika do geometrii przez niemal 2000 lat. Uznaje się, że *Elementy* są drugą po Biblii książką

7.2. Algorytm Euklidesa z wykorzystaniem dzielenia

Zwróć uwagę, że jeśli w algorytmie Euklidesa wykorzystującym odejmowanie jedna liczba jest dużo większa od drugiej, to od większej liczby trzeba wiele razy odjąć tę samą (mniejszą) liczbę. Taki przypadek ilustruje tabela 7.2.

Iteracja pętli	Liczba *a*	Liczba *b*
–	196	42
1	196 – 42 = 154	42
2	154 – 42 = 112	42
3	112 – 42 = 70	42
4	70 – 42 = 28	42
5	28	42 – 28 = 14
6	28 – 14 = 14	14

Tabela 7.2. Działanie algorytmu Euklidesa z wykorzystaniem odejmowania na przykładzie liczb 196 i 42

Wielokrotne odejmowanie możemy zastąpić dzieleniem, a dokładniej resztą z dzielenia większej liczby przez mniejszą. Taki sposób postępowania uwzględnia druga wersja algorytmu Euklidesa. Tabela 7.3 ilustruje działanie tego algorytmu dla tych samych liczb co tabela 7.2.

Iteracja pętli	Liczba *a*	Liczba *b*
–	196	42
1	196 **mod** 42 = 28	42
2	28	42 **mod** 28 = 14
3	28 **mod** 14 = 0	14

Tabela 7.3. Działanie algorytmu Euklidesa z wykorzystaniem dzielenia na przykładzie liczb 196 i 42

wiedzieć
uklidesa
:aniem
)licza NWD
zb całkowitych
Algorytm
aniem
ia działa
tylko dla liczb
dodatnich.

Wynikiem dzielenia 196 przez 42 jest 4 i reszta 28. Część całkowita określa liczbę operacji odejmowania, a reszta z dzielenia – wynik ostatniego odejmowania. W tym przypadku cztery działania zastąpiliśmy jednym.

Zwróć uwagę na ostatni wiersz tabeli 7.3. Warunkiem zakończenia działania algorytmu nie jest tym razem otrzymanie dwóch takich samych wartości. Algorytm zakończy działanie, jeśli jedna ze zmiennych przyjmie wartość 0 (przed ostatnim dzieleniem większa liczba jest wielokrotnością mniejszej, a więc ta mniejsza jest największym wspólnym dzielnikiem). Oto zapis algorytmu w pseudokodzie:

```
dopóki a ≠ 0 oraz b ≠ 0 wykonuj
    jeśli a > b to a ← a mod b
    w przeciwnym przypadku b ← b mod a
jeśli a ≠ 0 to wypisz a
w przeciwnym przypadku wypisz b
```


Podany pseudokod można uprościć. Warunek powtarzania pętli zamienimy na warunek `a * b ≠ 0` (ponieważ jeśli `a ≠ 0 oraz b ≠ 0`, to `a * b ≠ 0`). Ponadto instrukcję warunkową:

```
jeśli a ≠ 0 to wypisz a
w przeciwnym przypadku wypisz b
```

można zastąpić wypisaniem sumy wartości zmiennych, ponieważ jedna ze zmiennych ma wartość zero. Oto zapis algorytmu po modyfikacji:

```
dopóki a * b ≠ 0 wykonuj
    jeśli a > b to a ← a mod b
    w przeciwnym przypadku b ← b mod a
wypisz a + b
```

Ćwiczenie 2

Napisz program wyznaczający największy wspólny dzielnik dwóch liczb z wykorzystaniem dzielenia.

Jeśli założymy, że wartość zmiennej a jest zawsze większa od wartości zmiennej b, algorytm można zapisać prościej, bez użycia instrukcji warunkowej. Zmienna pom w poniższym pseudokodzie jest zmienną pomocniczą, służącą do tymczasowego przechowania mniejszej wartości. Po wykonaniu dzielenia wartość tę przyjmuje zmienna a.

```
dopóki b ≠ 0 wykonuj
    pom ← b
    b ← a mod b
    a ← pom
wypisz a
```

Jeśli na początku wartość zmiennej a będzie mniejsza od wartości zmiennej b, to pierwsza iteracja pętli zamieni wartości zmiennych a i b. Działanie algorytmu w takim przypadku ilustruje tabela 7.4.

Iteracja pętli	Wartość zmiennej a	Wartość zmiennej b	Wartość zmiennej pom
1	42	196	–
2	196	42 **mod** 196 = 42	196
3	42	196 **mod** 42 = 28	42
4	28	42 **mod** 28 = 14	28
5	14	28 **mod** 14 = 0	14

Tabela 7.4. Działanie zmodyfikowanego algorytmu Euklidesa z wykorzystaniem dzielenia na przykładzie liczb 42 i 196

Ćwiczenie 3

Warto wiedzieć
Algorytm Euklidesa z wykorzystaniem dzielenia wykonuje nie więcej operacji dzielenia niż liczba cyfr mniejszej liczby pomnożona przez 5. Wykazał to francuski matematyk i inżynier Gabriel Lamé. Najwięcej operacji jest wykonywanych dla liczb, które są dwiema kolejnymi liczbami Fibonacciego. Liczby te omówimy w temacie 17.

Poniżej znajduje się kod źródłowy programu wyznaczającego największy wspólny dzielnik dwóch liczb całkowitych dodatnich z wykorzystaniem dzielenia. Zwróć uwagę na warunek pętli w linii 8: zapis b!=0 oznacza „wartość zmiennej b jest różna od zera".

d źródłowy ajdującego wóch liczb dodatnich rzystaniem dzielenia

```
1.  #include <iostream>
2.
3.  using namespace std;
4.
5.  int NWD(int a, int b)
6.  {
7.      int pom;
8.      while (b!=0)
9.      {
10.         pom=b;
11.         b=a%b;
12.         a=pom;
13.     }
14.     return a;
15. }
16.
17. int main()
18. {
19.     int a, b;
20.     cout<<"a = "; cin>>a;
21.     cout<<"b = "; cin>>b;
22.     cout<<"NWD("<<a<<","<<b<<") = "<<NWD(a,b);
23.     return 0;
24. }
```

Do obliczeń funkcja NWD wykorzystuje zmienną pomocniczą zadeklarowaną w obrębie funkcji. Takie zmienne nazywamy **zmiennymi lokalnymi**. Zmienne deklarowane poza definicją funkcji i dostępne dla różnych funkcji nazywamy **zmiennymi globalnymi**.

a lokalna

globalna

Istnieją różne sposoby przekazania parametrów do funkcji. Ten, który stosowaliśmy w dotychczasowych przykładach, nazywa się **przekazaniem parametru przez wartość**. Parametry są w nim traktowane jak zmienne lokalne, czyli deklarowane w funkcji. Podczas wywołania funkcji tworzona jest kopia parametru i nadawana jest mu wartość początkowa **parametru aktualnego**. Oznacza to, że wartości parametrów są chronione, a parametrem aktualnym może być nie tylko wartość zmiennej, lecz także stała lub wartość wyrażenia. Nawet jeśli zmieniamy wartość parametru podczas działania funkcji (w funkcji NWD zmieniają się wartości parametrów a i b), to te zmiany są lokalne. Po zakończeniu wykonywania funkcji parametry a i b mają

arametru wartość

aktualny, s. 65

Zapamiętaj

Algorytm Euklidesa pozwala na znalezienie największego wspólnego dzielnika dwóch liczb całkowitych dodatnich. Odejmowanie od większej liczby mniejszej można zastąpić resztą z dzielenia większej liczby przez mniejszą.

7.3. Reprezentacja liczb wymiernych

W pamięci komputera **liczby wymierne** nie mają żadnej specjalnej reprezentacji – podobnie jest w językach programowania. Dlatego liczby wymierne należy przedstawiać jako pary liczb całkowitych. W języku C++ można to zrobić na kilka sposobów.

Liczba wymierna to liczba, którą można przedstawić w postaci ilorazu dwóch liczb całkowitych, czyli ułamka zwykłego.

Jednym z nich jest zapisanie licznika i mianownika jako powiązanych ze sobą zmiennych typu całkowitego. Dzięki temu będzie wiadomo, że reprezentują daną liczbę wymierną, a nie dwie niezależne liczby całkowite. Można w tym celu wykorzystać **strukturę**. Struktura jest złożonym typem danych, który pozwala zgrupować wiele zmiennych różnych typów pod jedną nazwą. Jednocześnie każda ze zmiennych ma także własną nazwę. Oto przykład definicji struktury o nazwie wymierna, grupującej dwie zmienne typu całkowitego o nazwach licz i mian (od słów „licznik" i „mianownik").

Struktura

```
struct wymierna
{
    int licz;
    int mian;
};
```

Po zdefiniowaniu w programie typu danych o nazwie wymierna będziemy mogli deklarować zmienne tego typu. Poniżej znajduje się przykład deklaracji zmiennych a i b typu wymierna, czyli ułamków zwykłych a i b.

```
wymierna a, b;
```

Aby odwołać się do wartości przechowywanych w strukturze, korzystamy z notacji z kropką, np. a.licz, co w naszym przypadku oznacza licznik ułamka a.

Problemem, którym się teraz zajmiemy, będzie skrócenie ułamka właściwego tak, aby w liczniku i mianowniku otrzymać **liczby względnie pierwsze**. Oto specyfikacja problemu:

Liczby względnie pierwsze to liczby całkowite, których największym wspólnym dzielnikiem jest liczba 1.

Specyfikacja

Dane: a – liczba wymierna postaci $\frac{l}{m}$, gdzie l i m to liczby całkowite dodatnie, $l < m$.

Wynik: skrócony ułamek a, $a = \frac{p}{q}$, gdzie p i q to liczby względnie

Dobra rada

Przy definiowaniu struktury pamiętaj o dodaniu średnika po nawiasie zamykającym. Po ciągu instrukcji ujętych w nawiasy klamrowe, np. w definicjach funkcji lub pętlach, nie trzeba tego robić.