informatyka+

Algorytmika i programowanie

Bazy danych

Multimedia, grafika i technologie internetowe

Sieci komputerowe

Tendencje w rozwoju informatyki i jej zastosowań

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

informatyka+

Wszechnica Poranna: Algorytmika i programowanie

Proste rachunki wykonywane za pomocą komputera

Maciej M Sysło

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Proste rachunki wykonywane za pomocą komputera

Rodzaj zajęć: Wszechnica Poranna
Tytuł: Proste rachunki wykonywane za pomocą komputera
Autor: prof. dr hab. Maciej M Sysło

Redaktor merytoryczny: prof. dr hab. Maciej M Sysło

Zeszyt dydaktyczny opracowany w ramach projektu edukacyjnego **Informatyka+** — ponadregionalny program rozwijania kompetencji uczniów szkół ponadgimnazjalnych w zakresie technologii informacyjno-komunikacyjnych (ICT).
www.informatykaplus.edu.pl
kontakt@informatykaplus.edu.pl

Wydawca: Warszawska Wyższa Szkoła Informatyki
ul. Lewartowskiego 17, 00-169 Warszawa
www.wwsi.edu.pl
rektorat@wwsi.edu.pl

Projekt graficzny: FRYCZ I WICHA

Warszawa 2009

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

WARSZAWSKA WYŻSZA SZKOŁA INFORMATYKI

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Proste rachunki wykonywane za pomocą komputera

Maciej M. Sysło
Uniwersytet Wrocławski, UMK w Toruniu
syslo@ii.uni.wroc.pl, syslo@mat.uni.torun.pl

Streszczenie

Komputery nie przestały być maszynami matematycznymi – jak kiedyś je nazywano – i obecnie służą również do wykonywania różnych obliczeń. Wykład jest poświęcony m.in. algorytmom obliczania: wartości dziesiętnej liczby binarnej, postaci binarnej liczby dziesiętnej, wartości wielomianu, największego wspólnego dzielnika dwóch liczb (algorytm Euklidesa) oraz wartości potęgi. Motywacją dla wprowadzenia tych algorytmów jest chęć objaśnienia metody szyfrowania informacji z kluczem publicznym RSA, powszechnie stosowanej w kryptografii komputerowej. Na warsztatach uczniowie zapoznają się z komputerowymi demonstracjami omówionych na wykładzie algorytmów oraz zaprogramują w języku Pascal lub C++ i przetestują wybrane algorytmy.

Te zajęcia są drugą częścią wprowadzenia do algorytmiki i programowania. Omawiane są więc również ogólne pojęcia z zakresu algorytmiki, jak specyfikacja problemu, schematy blokowe algorytmów, podstawowe struktury danych (ciąg i tablica) oraz pracochłonność (złożoność) algorytmów. Na warsztatach zostają wprowadzone podstawowe instrukcje języka programowania (iteracyjna i warunkowa oraz procedura i funkcja niestandardowa), wystarczające do zaprogramowania i uruchomienia komputerowych realizacji algorytmów omówionych na wykładzie. Przytoczono ciekawe przykłady zastosowań omawianych zagadnień.

Rozważania są prowadzone na elementarnym poziomie i do ich wysłuchania oraz wzięcia udziału w warsztatach wystarczy znajomość informatyki wyniesiona z gimnazjum oraz matematyki na poziomie szkoły średniej. Te zajęcia są adresowane do wszystkich uczniów w szkołach ponadgimnazjalnych, zgodnie bowiem z nową podstawą programową, kształceniem umiejętności algorytmicznego rozwiązywania problemów mają być objęci wszyscy uczniowie.

Spis treści

WPROWADZENIE

Komputery w dalszym ciągu są przede wszystkim maszynami matematycznymi, chociaż użytkownik widzi często teksty i obrazy, słyszy muzykę, ogląda filmy. Wykład jest poświęcony algorytmom wykonywania elementarnych obliczeń, takich jak: obliczanie wartości dziesiętnej liczby binarnej, postaci binarnej liczby dziesiętnej, wartości wielomianu, największego wspólnego dzielnika dwóch liczb, wartości potęgi.

Te zajęcia są drugą częścią wprowadzenia do algorytmiki i programowania. Omawiane są więc również ogólne pojęcia z zakresu algorytmiki, jak specyfikacja problemu, schematy blokowe algorytmów, podstawowe struktury danych (ciąg i tablica) oraz pracochłonność (złożoność) algorytmów. Na warsztatach zostają wprowadzone podstawowe instrukcje języka programowania (iteracyjna i warunkowa oraz procedura i funkcja niestandardowa), wystarczające do zaprogramowania i uruchomienia komputerowych realizacji algorytmów omówionych na wykładzie. Przytoczono ciekawe przykłady zastosowań omawianych zagadnień.

Rozważania ogólne na temat algorytmiki, algorytmicznego myślenia i rozwiązywania problemów z pomocą komputerów są zamieszczone w Dodatku (rozdz. 7). Materiał tam zawarty może być dobrym podsumowaniem zajęć.

Jako literaturę rozszerzającą prowadzone tutaj rozważania polecamy podręczniki [2], a zwłaszcza książki [5] i [6].

1 SYSTEM DZIESIĘTNY I SYSTEM BINARNY

Obecnie w powszechnym użyciu jest **system dziesiętny**, zwany też **systemem dziesiątkowym**. Komputery natomiast wszystkie operacje wykonują w **systemie binarnym**, zwanym również **systemem dwójkowym**. Oba systemy, to dwa przykłady tzw. **systemu pozycyjnego** zapisywania wartości liczbowych. Powinny one być Wam znane z lekcji matematyki. Przypomnijmy jednak jeden i drugi system.

Liczba zapisana w systemie dziesiętnym, np. 357, oznacza wartość, na którą składają się trzy setki, pięć dziesiątek i siedem jedności, a zatem wartość tej liczby można zapisać jako $357 = 3{\cdot}100 + 5{\cdot}10 + 7{\cdot}1$, a także jako $357 = 3{\cdot}10^2 + 5{\cdot}10^1 + 7{\cdot}10^0$.

A zatem, dziesiętna liczba naturalna złożona z n cyfr:

$$d_{n-1}\, d_{n-2} \ldots d_1\, d_0$$

oznacza wartość:

$$d_{n-1}10^{n-1} + d_{n-2}10^{n-2} + \ldots + d_1 10^1 + d_0 10^0. \qquad (1)$$

Liczba 10 w tej reprezentacji nazywa się **podstawą systemu liczenia**. Do zapisywania liczb w systemie dziesiętnym są używane cyfry: 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Ten sposób reprezentowania liczb nazywa się **system pozycyjnym**, gdyż w tym systemie znaczenie cyfry zależy od jej pozycji w ciągu cyfr, określającym liczbę. Na przykład, liczby 123 i 321 mają różną wartość, chociaż są złożone z tych samych cyfr, a liczba 111 jest złożona z trzech jedynek, z których każda ma inne znaczenie dla wartości tej liczby.

Dwa tysiące lat przed naszą erą Babilończycy stosowali system **kopowy** (tj. przy podstawie 60) – przypuszcza się, że stąd wziął się podział kąta pełnego na 360 stopni, godziny – na 60 minut, a minuty – na 60 sekund.

Ćwiczenie 1. Wszystkim są znane tzw. **cyfry rzymskie**: I, V, X, L, C, D. M. Oznaczają one odpowiednio liczby: 1, 5, 10, 50, 100, 500, 1000. W tym systemie, liczba III ma dziesiętną wartość 3, a zatem każda z cyfr w tej liczbie ma takie samo znaczenie, a wartość jest obliczana przez dodawanie wartości cyfr. Zapisz w tym systemie liczby 2009 i 2012. Na liczbach rzymskich trudno wykonuje się podstawowe działania: dodawanie, odejmowanie i mnożenie. Liczby te były stosowane przez Rzymian głównie do zapisywania wartości liczbowych, a nie do wykonywania na nich działań.

Za podstawę systemu liczenia można przyjąć dowolną liczbę naturalną p większą od 1, np. 2, 5, 8, 16 czy 60. Jeśli p jest podstawą systemu liczenia, to liczby w tym systemie są zapisywane za pomocą cyfr ze zbioru $\{0, 1, \ldots, p-1\}$.

W rozważaniach związanych z komputerami pojawiają się liczby w systemach o podstawach 2, 8 i 16. Naszą uwagę skupimy głównie na systemie o podstawie 2, a o innych systemach jest mowa w ćwiczeniach.

Jeśli $p = 2$, to system nazywa się **binarnym** lub **dwójkowym** – cyfry liczby, zapisanej w tym systemie, mogą mieć wartość 0 lub 1 i nazywają się **bitami**. Liczba naturalna a dana w systemie dziesiętnym ma następującą postać w systemie binarnym dla pewnego n:

$$a = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0, \qquad (2)$$

Za prekursora systemu binarnego uważa się G.W. Leibniza, który w pracy opublikowanej w 1703 roku zilustrował wykonywanie czterech podstawowych działań arytmetycznych na liczbach binarnych.

gdzie współczynniki $b_{n-1}, b_{n-2}, \ldots, b_1, b_0$ są liczbami 0 lub 1. W skrócie piszemy zwykle $a = (b_{n-1}b_{n-2} \ldots b_1 b_0)_2$ i ciąg bitów w nawiasie nazywamy **binarnym rozwinięciem liczby** a. Na przykład mamy, $8 = (1000)_2$, $12 = (1100)_2$, $7 = (111)_2$. Cyfra b_{n-1} jest **najbardziej znaczącą**, a b_0 – **najmniej znaczącą** w rozwinięciu liczby a.

Sposoby wykonywania działań w systemach innych niż dziesiętny są prostym przeniesieniem zasad z systemu dziesiętnego.

Jeśli się dobrze przyjrzymy wzorom oznaczonym przez (1) i (2), to zauważymy, że przypominają one specjalne wielomiany – współczynnikami tych wielomianów są cyfry rozwinięcia, a argumentem wielomianu – jest podstawa systemu liczenia. Aby więc obliczyć wartość dziesiętną liczby, danej w innym systemie, poznamy algorytm, który służy do bardzo efektywnego obliczania wartości wielomianu.

2 OBLICZANIE WARTOŚCI WIELOMIANU – SCHEMAT HORNERA

Jednym z najważniejszych kroków w algorytmach jest powtarzanie tej samej czynności (operacji), czyli **iteracja**. W tym punkcie zilustrujemy iterację na przykładzie szybkiego sposobu obliczania wartości wielomianu.

Obliczanie wartości wielomianu o zadanych współczynnikach jest jedną z najczęściej wykonywanych operacji w komputerze. Wynika to z ważnego matematycznego faktu, zgodnie z którym każdą funkcję (np. funkcje trygonometryczne) można zastąpić wielomianem, którego postać zależy od funkcji i od tego, jaką chcemy uzyskać dokładność obliczeń.

Na przykład, obliczanie wartości funkcji $\cos x$ w przedziale $0 \le x \le \pi/2$ z błędem nie większym niż 0.0009, można zastąpić obliczaniem wartości następującego wielomianu:

$$\cos x = 1 - 0.49670x^2 + 0.03705x^4.$$

Ada Augusta, córka Byrona, uznawana powszechnie za pierwszą programistkę komputerów, przełomowe znaczenie maszyny analitycznej Ch. Babbage'a, pierwowzoru późniejszych komputerów, upatrywała właśnie „w możliwości wielokrotnego wykonywania przez nią danego ciągu instrukcji, z liczbą powtórzeń z góry zadaną lub zależną od wyników obliczeń".

Zacznijmy od prostych ćwiczeń.
Jeśli dane są wartości współczynników a, b i c wielomianu stopnia 2:

$$w(x) = ax^2 + bx + c,$$

to jego wartość można obliczyć wykonując zaznaczone mnożenia i dodawania: $a \cdot x \cdot x + b \cdot x + c$, czyli 3 mnożenia i dwa dodawania. Można także nieco inaczej – wyłączmy x z dwóch pierwszych składników – wtedy otrzymamy:

$$w(x) = (ax + b)x + c$$

i dla tej postaci obliczanie wartości tego wielomianu przyjmuje postać: $(a \cdot x + b) \cdot x + c$, czyli są wykonywane tylko 2 mnożenia i dwa dodawania.

W podobny sposób można przedstawić wielomian stopnia 3:

$$v(x) = ax^3 + bx^2 + cx + d$$

najpierw wyłączam x z pierwszych trzech wyrazów, a następnie wyłączamy x z dwóch pierwszych wyrazów w nawiasie:

$$v(x) = ax^3 + bx^2 + cx + d = v(x) = v(x) = (ax^2 + bx + c)x + d, = ((a\cdot x + b)\cdot x + c)\cdot x + d.$$

Widać z tego ostatniego wzoru, że wartość wielomiany stopnia 3 można obliczyć za pomocą 3 mnożeń i 3 dodawań.

Rozważmy teraz wielomian stopnia n:

$$w_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n \tag{3}$$

i zastosujmy grupowanie wyrazów podobnie, jak w wielomianie stopnia 3 – najpierw wyłączamy x ze wszystkich wyrazów z wyjątkiem ostatniego, a następnie stosujmy ten sam krok wielokrotnie do wyrazów, które będą pojawiały się w najgłębszych nawiasach, aż pozostanie jednomian:

$$\begin{aligned} w_n(x) &= a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = (a_0x^{n-1} + a_1x^{n-2} + \ldots + a_{n-1})x + a_n \\ &= ((a_0x^{n-2} + a_1x^{n-3} + \ldots + a_{n-2})x + a_{n-1})x + a_n = \\ &= (\ldots((a_0x + a_1)x + a_2)x + \ldots + a_{n-2})x + a_{n-1})x + a_n \end{aligned} \tag{4}$$

Ćwiczenie 2. Przedstaw w postaci (4) następujące wielomiany:

$w(x) = 3x^4 - x^3 + 5x^2 + 7x - 2$
$w(x) = x^5 - x^3 + 4x^2 + 3x - 1$
$w(x) = x^6 - x^3 + x$

Zapiszmy teraz **specyfikację**, czyli dokładny opis rozważanego tutaj problemu:

Problem: Obliczanie wartości wielomianu
Dane: n – nieujemna liczba całkowita – stopień wielomianu;
$a_0, a_1, \ldots a_n$ – $n+1$ współczynników wielomianu;
z – wartość argumentu.
Wynik: Wartość wielomianu (4) stopnia n dla wartości argumentu $x = z$.

Aby obliczyć ze wzoru (4) wartość wielomianu dla wartości argumentu z, należy postępować następująco (y oznacza pomocniczą zmienną):

$$\begin{aligned} &y := a_0 \\ &y := yz + a_1 \\ &y := yz + a_2 \\ &\ldots \\ &y := yz + a_{n-1} \\ &y := yz + a_n \end{aligned}$$

Wszystkie wiersze, z wyjątkiem pierwszego można zapisać w jednolity sposób – otrzymujemy wtedy:

$$\begin{aligned} &y := a_0 \\ &y := yz + a_i \quad \text{dla } i = 1, 2, \ldots, n. \end{aligned} \tag{5}$$

Ten sposób obliczania wartości wielomianu nazywa się **schematem Hornera**.

Uwaga. W opisie algorytmu pojawiło się polecenie (instrukcja) **przypisania**[1], np. $y := a_0$, w której występuje symbol :=, złożony z dwóch znaków: dwukropka i równości. Przypisanie oznacza nadanie wielkości (zmiennej) stojącej po lewej strony tego symbolu wartości równej wartości wyrażenia (w szczególnym przypadku to wyrażenie może być zmienną) znajdującego się po prawej stronie tego symbolu. Przypisanie jest stosowane na przykład wtedy, gdy należy zmienić wartość zmiennej, np. $i := i + 1$ – w tym przypadku ta sama zmienna występuje po lewej i po prawej stronie symbolu przypisania. Polecenie przypisania występuje w większości języków programowania, stosowane są tylko różne symbole i ich uproszczenia dla jego oznaczenia.

Schemat Hornera został podany przez jego autora w 1819 roku, chociaż znacznie wcześniej Isaac Newton stosował podobną metodę obliczania wartości wielomianów w swoich rachunkach fizycznych. W 1971 roku, A. Borodin udowodnił, że schemat Hornera jest optymalnym, pod względem liczby wykonywanych działań, algorytmem obliczania wartości wielomianu.

Ćwiczenie 3. Zastosuj schemat Hornera do obliczenia wartości wielomianów z ćwicz. 2 w punkcie $z = 1$.

Możemy więc teraz zapisać:

Algorytm: Schemat Hornera
Dane: n – nieujemna liczba całkowita (stopień wielomianu);
$a_0, a_1, \ldots, a_n$ – n+1 współczynników wielomianu;
z – wartość argumentu.
Wynik: $y = w_n(z)$ – wartość wielomianu (4) stopnia n dla wartości argumentu $x = z$.
Krok 1. $y := a_0$ – początkową wartością jest współczynnik przy najwyższej potędze.
Krok 2. Dla $i = 1, 2, \ldots, n$ oblicz wartość dwumianu $y := yz + a_i$

Wynika stąd, że aby obliczyć wartość wielomianu stopnia n wystarczy wykonać n mnożeń i n dodawań. Udowodniono, że jest to najszybszy sposób obliczania wartości wielomianu.

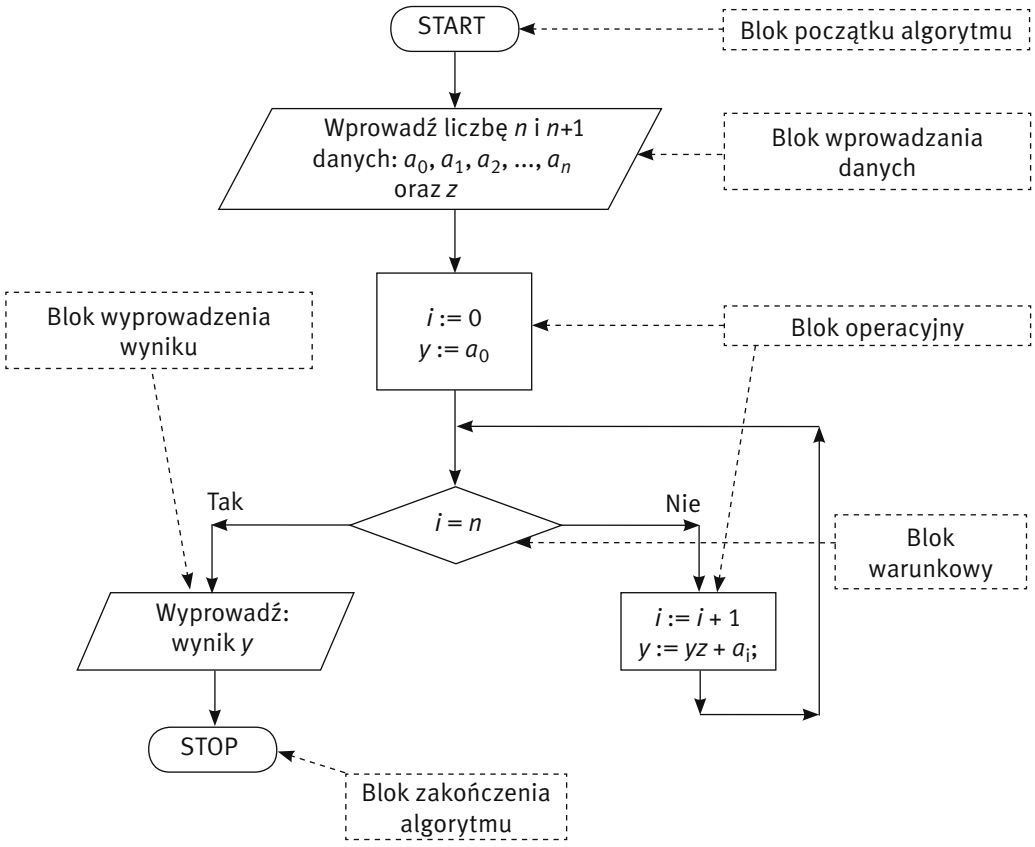
Schemat blokowy schematu Hornera
Schemat blokowy algorytmu (zwany również **siecią działań** lub **siecią obliczeń**) jest graficznym opisem: działań składających się na algorytm, ich wzajemnych powiązań i kolejności ich wykonywania. W informatyce miejsce schematów blokowych jest pomiędzy opisem algorytmu w postaci listy kroków, a programem, napisanym w wybranym języku programowania. Należą one do kanonu wiedzy informatycznej, nie są jednak niezbędnym jej elementem, chociaż mogą okazać się bardzo przydatne na początkowym etapie projektowania algorytmów i programów komputerowych. Z drugiej strony, w wielu publikacjach algorytmy są przedstawiane w postaci schematów blokowych, pożądana jest więc umiejętność ich odczytywania i rozumienia. Warto nadmienić, że ten sposób reprezentowania algorytmów pojawia się w zadaniach maturalnych z informatyki.

Na rys. 1 jest przedstawiony schemat blokowy algorytmu **Schemat Hornera**. Jest on zbudowany z **bloków**, których kształty zależą od rodzaju wykonywanych w nich poleceń. I tak mamy:
- blok początku i blok końca algorytmu;
- blok wprowadzania (wczytywania) danych i wyprowadzania (drukowania) wyników – bloki te mają taki sam kształt;
- blok operacyjny, w którym są wykonywane operacje przypisania;
- blok warunkowy, w którym jest formułowany warunek;
- blok informacyjny, który może służyć do komentowania fragmentów schematu lub łączenia ze sobą części większych schematów blokowych.

Nie istnieje pełny układ zasad poprawnego konstruowania schematów blokowych. Można natomiast wymienić dość naturalne zasady, wynikające z charakteru bloków:

[1] Polecenie przypisania jest czasem nazywane niepoprawnie podstawieniem

- schemat zawiera dokładnie jeden blok początkowy, ale może zwierać wiele bloków końcowych – początek algorytmu jest jednoznacznie określony, ale algorytm może się kończyć na wiele różnych sposobów;
- z bloków: początkowego, wprowadzania danych, wyprowadzania wyników, operacyjnego wychodzi dokładnie jedno połączenie, może jednak wchodzić do nich wiele połączeń;
- z bloku warunkowego wychodzą dwa połączenia, oznaczone wartością warunku: TAK i NIE;
- połączenia wychodzące mogą dochodzić do bloków lub do innych połączeń.



Rysunek 1.
Realizacja schematu Hornera w postaci schematu blokowego

Ćwiczenie 4. Zakreśl na schemacie blokowym na rys. 1 fragmenty odpowiadające poszczególnym krokom w opisie algorytmu **Schemat Hornera**. Zauważ, że ten schemat blokowy zawiera również fragmenty odpowiadające wczytywaniu danym i wypisywaniu wyników.

Ćwiczenie 5. Blok wprowadzania danych w schemacie na rys. 1 polega na wczytaniu liczby n a później na wczytaniu $n + 1$ liczb a_i. Narysuj szczegółowy schemat blokowy tego bloku.

Schematy blokowe mają wady, trudne do wyeliminowania. Łatwo konstruuje się z ich pomocą algorytmy dla obliczeń nie zawierających iteracji i warunków, którym w schematach odpowiadają rozgałęzienia, nieco trudniej dla obliczeń z rozgałęzieniami, a trudniej dla obliczeń iteracyjnych. Za pomocą schematów blokowych nie można w naturalny sposób zapisać rekurencji oraz objaśnić znaczenia wielu pojęć związanych z algorytmiką, takich np. jak programowanie z użyciem procedur, czyli podprogramów z parametrami. .

W dalszej części zajęć rzadko będziemy korzystać ze schematów blokowych, wystarczy nam bowiem umiejętność programowania.

Schemat Hornera ma wiele zastosowań. Może być wykorzystany m.in. do:
- obliczania wartości dziesiętnej liczb danych w innym systemie pozycyjnym (rozdz. 3);
- szybkiego obliczania wartości potęgi (rozdz. 5).

Reprezentowanie danych w algorytmach

Zanim podamy komputerową realizację pierwszego algorytmu – schematu Hornera, musimy ustalić, w jaki sposób będą reprezentowane w algorytmie dane i jak będziemy je podawać do algorytmu.

Danymi dla schematu Hornera są stopień wielomianu n, zapisane w postaci ciągu $n + 1$ liczb współczynniki wielomianu $a_0, a_1, a_2, \ldots, a_n$ oraz liczba z, dla której chcemy obliczyć wartość wielomianu. Stopień wielomianu jest liczbą naturalną (czyli dodatnią liczbą całkowitą), a pozostałe liczby mogą być całkowite lub rzeczywiste, czyli np. dziesiętne (tj. z kropką). Rodzaj danych liczb nazywa się **typem danych**. Przyjmujemy, że poza stopniem wielomianu, pozostałe dane są rzeczywiste.

Dla wygody będziemy zakładać, że wiemy, ile będzie danych i ta liczba danych występuje na początku danych – jest nią liczba n, stopień wielomianu.

Zbiór danych, który jest przetwarzany za pomocą algorytmu, może być podawany (czytany) z klawiatury, czytany z pliku lub może być zapisany w algorytmie w odpowiedniej strukturze danych.

Komputerowa realizacja schematu Hornera – dane z klawiatury

Zapiszemy teraz schemat Hornera posługując się poleceniami języka Pascal (odpowiedni program w języku C++ zostanie podany na zajęciach warsztatowych). Przyjmujemy na początku, że dane są podawane z klawiatury – na początku należy wpisać liczbę wszystkich danych, a po niej kolejne elementy danych w podanej ilości. Po każdej danej liczbie naciskamy klawisz Enter.

Program, który jest zapisem schematu Hornera w języku Pascal, jest umieszczony w drugiej kolumnie w tab. 1. Język Pascal jest zrozumiały dla komputerów, które są wyposażone w specjalne programy, tzw. **kompilatory**, przekładające programy użytkowników na język wewnętrzny komputerów. Program w tab. 1 bez większego trudu zrozumie także człowiek dysponujący opisem schematu Hornera w postaci listy kroków. Kilka tylko słów komentarza. W wierszach nr 2 i 3 znajdują się **deklaracje** zmiennych – komputer musi wiedzieć, jakimi wielkościami posługuje się algorytm i jakie to są liczby, czyli jakiego są one **typu** – `integer` oznacza liczby całkowite (np. stopień wielomianu jest liczbą całkowitą), `real` oznacza liczby rzeczywiste, czyli np. liczby, które zawierają kropkę dziesietną. Polecenia w językach programowania nazywają się **instrukcjami**. Jeśli chcemy z kilku instrukcji zrobić jedną, to tworzymy z nich **blok**, umieszczając na jego początku słowo `begin`, a na końcu – `end`. Pojedyncze instrukcje kończymy **średnikiem**. Na końcu programu stawiamy kropkę. Pokrótce to wszystkie podstawowe zasady pisania programów w języku Pascal dla komputerów.

Jedna instrukcja wymaga wytłumaczenie, chociaż również jest dość oczywista. W wierszach 9 – 12 znajdują się instrukcje, które realizują Krok 2 algorytmu polegający na wykonaniu jednej iteracji schematu Hornera. Dodatkowo, wcześniej jest czytany kolejny współczynnik wielomianu. Instrukcja, służąca do wielokrotnego wykonania innych instrukcji nazywa się **instrukcją iteracyjną** lub **instrukcją pętli**. W programie w tab. 1 ta instrukcja zaczyna się w wierszu nr 9 a kończy w wierszu nr 12:

```
for i:=1 to n do begin
...
end
```

Ta instrukcja iteracyjna, jak napisaliśmy, służy do powtórzenia dwóch instrukcji:

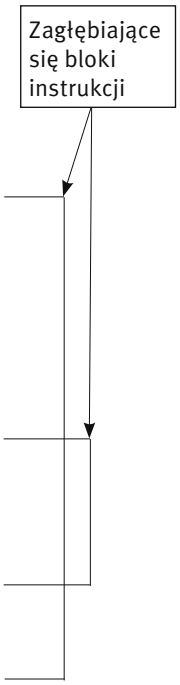
```
read(a);
y:=y*z+a
```

Inne typy instrukcji iteracyjnej będą wprowadzane sukcesywnie.

Uwaga. Ponieważ założyliśmy, że dane są podawane z klawiatury, zmieniliśmy ich kolejność w stosunku do specyfikacji – liczba z, dla której jest obliczana wartość wielomianu, jest podawana na początku, przed współczynnikami wielomianu.

Tabela 1.
Program w języku Pascal (druga kolumna)

Lp	Program w języku Pascal	Odpowiedniki instrukcji po polski
1.	`Program Horner;`	`nazwa programu;`
2.	`var i,n :integer;`	`zmienne i,n: naturalne;`
3.	`var a,y,z:real;`	`zmienne a,y,z: rzeczywiste;`
4.	`begin`	`początek`
5.	`read(n);`	`czytaj(n);`
6.	`read(z);`	`czytaj(z);`
7.	`read(a);`	`czytaj(a);`
8.	`y:=a;`	`początkowa wartość wielomianu`
9.	`for i:=1 to n do begin`	`dla i:=1 do n wykonaj początek`
10.	`read(a);`	`czytaj(a);`
11.	`y:=y*z+a`	`modyfikacja wartości wielomianu`
12.	`end;`	`koniec iteracji;`
13.	`write(y)`	`drukuj(y) - wartość wielomianu`
14.	`end.`	`koniec. – na końcu stawiamy kropkę`



W wyniku wykonania tego programu, wypisywana jest na ekranie wartość wielomianu *y* w punkcie *z*. Ta wartość jest wypisywana w **postaci normalnej**, np. liczba 30 jest wypisywana jako:

3.00000000000000E+001

czyli 3.00000000000000·10^1. Nie jest to najwygodniejsza postać liczb, zwłaszcza liczb o niewielkich wartościach. Aby wybrać inny format wyprowadzanych liczb, możemy napisać:

```
write(y:2:2)
```

co będzie oznaczać, że wartość *y* zostanie wyświetlona (wydrukowana, wyprowadzona) z dwoma cyframi przed kropką i z dwoma cyframi po kropce.

Ćwiczenie 6. Uruchom program `Horner` i wykonaj obliczenia dla wybranego wielomianu i kilku jego argumentów.

Komputerowa realizacja schematu Hornera – dane w tablicy

Zmodyfikujemy teraz nasz pierwszy program tak, aby:

- stopień i współczynniki wielomianu były czytane na początku i przechowywane w programie – do przechowania w programie współczynników użyjemy **tablicy**, która jest synonimem ciągu w języku programowania;
- można było liczyć wartość wielomianu o wczytanych współczynnikach dla wielu argumentów – zakładamy w tym celu, że ciąg argumentów jest zakończony liczbą 0 (jest to tak zwany **wartownik** ciągu, gdyż jego rolą jest pilnowanie końca ciągu).

Przy tych założeniach, program będący **implementacją**[2] schematu Hornera, może mieć następującą postać:

```
Program Horner_tablica;
 var i,n:integer;
     y,z:real;
     a  :array[0..100] of real;
     {Przyjmujemy, ze wielomian ma stopien co najwyzej 100}
begin
 read(n);
 for i:=0 to n do read(a[i]);
 writeln('z      y');
 read(z);
 while z <> 0 do begin
  y:=a[0];
  for i:=1 to n do
   y:=y*z+a[i];
  write('    ',y:2:5);
  writeln;
  read(z)
 end
end.
```

Skomentowania wymaga użyta instrukcja warunkowa:

```
 while z <> 0 do begin
  y:=a[0];
  for i:=1 to n do
   y:=y*z+a[i];
  write('    ',y:2:5);
  writeln;
  read(z)
 end
```

W bloku tej instrukcji jest wykonywany schemat Hornera dla kolejnych wartości argumentu *z* tak długo, jak długo czytana liczba *z* jest różna od 0 (`z <> 0`).

Dodatkowo, w programie wprowadziliśmy drukowanie wyników w postaci tabelki.

Ćwiczenie 7. Uruchom program `Horner_tablica` i wykonaj obliczenia dla wybranego wielomianu i kilku jego argumentów. Popraw wygląd wyników.

3 ZAMIANA LICZBY BINARNEJ NA DZIESIĘTNĄ

Binarne rozwinięcie dziesiętnej liczby naturalnej (prawa strona we wzorze (2)) przypomina swoją postacią wielomian (patrz wzór (3)), gdzie a jest wartością wielomianu stopnia $n-1$ o współczynnikach $b_{n-1}, b_{n-2}, \ldots, b_1, b_0$ należących do zbioru {0, 1}, dla wartości argumentu $x = 2$. Z tej interpretacji rozwinięcia (2) wynika sposób obliczania dziesiętnej wartości liczby naturalnej a, gdy jest dane jej binarne rozwinięcie $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$. Wystarczy zastosować schemat Hornera, który dla wielomianu, danego wzorem (2), przyjmuje następującą postać:

$$\begin{aligned} a &= b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0 \\ &= (\ldots(b_{n-1}2 + b_{n-2})2 + \ldots + b_1)2 + b_0 \end{aligned} \qquad (6)$$

[2] Terminem **implementacja** określa się w informatyce realizację algorytmu w postaci programu komputerowego.

Ćwiczenie 8. Napisz program służący do obliczania dziesiętnej wartości liczby *a* danej w postaci binarnego rozwinięcia $(b_{n-1}b_{n-2} \ldots b_1b_0)_2$.
Wskazówka. Zmodyfikuj odpowiednio program `Hornera_tablica`. Zwróć uwagę, że kolejne cyfry rozwinięcia binarnego liczby w postaci (2) są ponumerowane odwrotnie niż współczynniki wielomianu we wzorze (3) i zauważ przy tym, że indeks współczynnika odpowiada potędze liczby 2, przed którą stoi ten współczynnik. Nie powinno to jednak utrudnić Ci wykonania tego ćwiczenia.

Postać rozwinięcia binarnego (6) sugeruje bardzo prosty sposób obliczania wartości liczby *a* za pomocą kalkulatora.

Ćwiczenie 9. Oblicz za pomocą kalkulatora dziesiętną wartość liczby *a*, przedstawionej w postaci rozwinięcia binarnego. Wykorzystaj algorytm wynikający z postaci (6). Zauważ, że nie musisz korzystać z dodatkowej pamięci kalkulatora.
Wskazówka. Możesz skorzystać z kalkulatora dostępnego wśród akcesoriów środowiska Windows.

Wszystkie fakty podane powyżej przenoszą się na rozwinięcia liczby w systemie przy dowolnej podstawie *p*.

Ćwiczenie 10. Zmodyfikuj program napisany w ćwicz. 8 tak, aby mógł być stosowany do obliczania dziesiętnej wartości liczby *a* danej w postaci rozwinięcia przy podstawie *p*.

4 OTRZYMYWANIE BINARNEGO ROZWINIĘCIA LICZBY DZIESIĘTNEJ

Interesuje nas teraz czynność odwrotna – otrzymanie binarnego rozwinięcia dla danej dziesiętnej liczby naturalnej *a*.

Zauważmy, że we wzorze (2), czyli w binarnym przedstawieniu liczby *a*, wszystkie składniki z wyjątkiem ostatniego są podzielne przez 2, a zatem ten ostatni składnik, czyli b_0 jest resztą z dzielenia liczby *a* przez 2. Bity b_1, b_2,..., to reszty z dzielenia przez 2 kolejno otrzymywanych ilorazów. Dzielenie kończymy, gdy iloraz wynosi 0, gdyż wtedy kolejne reszty będą już cały czas równe 0, a zera na początku rozwinięcia binarnego nie mają żadnego znaczenia. Prześledź ten proces na przykładzie z rys. 2.

dzielenie	iloraz	reszta
749 | 2	374	**1**
374 | 2	187	**0**
187 | 2	93	**1**
93 | 2	46	**1**
46 | 2	23	**0**
23 | 2	11	**1**
11 | 2	5	**1**
5 | 2	2	**1**
2 | 2	1	**0**
1 | 2	0	**1**

Rysunek 2.
Przykład tworzenia binarnej reprezentacji liczby dziesiętnej 749. Otrzymaliśmy $749 = (1011101101)_2$

Zauważmy, że w powyższym algorytmie binarna reprezentacja liczby jest tworzona od końca, czyli od najmniej znaczącego bitu.

Wprowadzimy teraz dwie operacje, wykonywane na liczbach całkowitych, których wyniki są również liczbami całkowitymi, a które są przydatne przy obliczaniu ilorazu i reszty z dzielenia liczb całkowitych przez siebie.

Dla dwóch liczb całkowitych k i l definiujemy:

$r = k$ **mod** l – r jest resztą z dzielenia k przez l, czyli r spełnia nierówności $0 \le r < l$, gdyż reszta jest nieujemną liczbą mniejszą od dzielnika;

$q = k$ **div** l – q jest ilorazem całkowitym z dzielenia k przez l, czyli q jest wynikiem dzielenia k przez l z pominięciem części ułamkowej.

Z definicji tych dwóch operacji wynika następująca równość:

$$k = l \cdot q + r = l \cdot (k \text{ div } l) + (k \bmod l). \tag{7}$$

Upewnij się, że dobrze rozumiesz te dwie operacje, które często występują w obliczeniach komputerowych na liczbach całkowitych.

Ćwiczenie 11. Dla liczby naturalnej $l = 6$ i dla liczby naturalnej k, zmieniającej się od 0 co 1 do 20 oblicz wartości k div l oraz k mod l i sprawdź prawdziwość równości (7).

Ćwiczenie 12. Przyjmij, że $l = 2$, a więc interesuje nas iloraz i reszta z dzielenia liczby naturalnej k przez 2. Podaj, w zależności od parzystości liczby k, ile wynosi k mod 2 oraz k div 2.

Dotychczasowa dyskusja prowadzi nas do następującego algorytmu:

Algorytm: Zamiana dziesiętnej liczby naturalnej na postać binarną

Dane: Dziesiętna liczba naturalna a.

Wynik: Ciąg bitów, tworzących binarne rozwinięcie liczby a, w kolejności od najmniej znaczącego bitu.

Krok 1. Powtarzaj krok 2 dopóki a jest liczbą większą od zera, w przeciwnym razie zakończ algorytm.

Krok 2. Za kolejny bit (od końca) rozwinięcia przyjmij: a mod 2 i przypisz: $a := a$ div 2.

Poniżej przedstawiamy implementację tego algorytmu w języku Pascal.

```
Program Rozwiniecie_binarne;
 var a:integer;
begin
 read(a);
 while a <> 0 do begin
  write(a mod 2,' ');
  a:=a div 2
 end
end.
```

Ćwiczenie 13. W powyższym programie, kolejne bity rozwinięcia binarnego liczby a są wypisywane w kolejności od najmniej znaczącego, a więc odwrotnie, niż to się przyjmuje. Zmodyfikuj ten program tak, aby binarne rozwinięcie danej liczby było wyprowadzane od najbardziej znaczącego bitu.
Wskazówka. Posłuż się tablicą, w której będą przechowywane kolejno generowane bity.

Długość rozwinięcia binarnego liczby

Osoby, które nie znają logarytmu, mogą opuścić ten podpunkt.

Liczby w komputerze są zapisywane w postaci binarnej. Interesujące jest więc pytanie, ile miejsca w komputerze, czyli ile bitów, zajmuje liczba naturalna a w postaci binarnej. Odpowiedź na to pytanie jest bardzo ważna w informatyce, nawet dzisiaj, kiedy można korzystać z niemal nieograniczonej pamięci komputerów i nie trzeba się obawiać, że jej zabraknie.

Najpierw rozważymy odwrotne pytanie: Jaką największą liczbę naturalną można zapisać na n bitach? Liczba taka ma wszystkie bity równe 1 w swoim rozwinięciu binarnym $(11...11)_2$, a więc jej wartość, jako suma n początkowych wyrazów ciągu geometrycznego z ilorazem równym 2, wynosi:

$$2^{n-1} + 2^{n-2} + ... + 2^1 + 2^0 = (1 - 2^n)/(1 - 2) = 2^n - 1$$

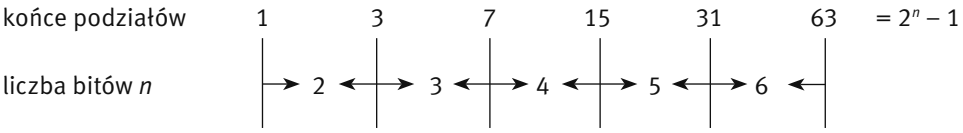
Ta równość ma ciekawą interpretację. Wartość sumy po lewej stronie jest liczbą o jeden mniejszą od liczby $(100...00)_2$, która w rozwinięciu binarnym składa się z n+1 bitów i ma tylko jeden bit równy 1 – najbardziej znaczący. Tą liczbą jest 2^n. Na rys. 3 pokazano, ile bitów potrzeba do przedstawienia w postaci binarnej liczb z poszczególnych przedziałów. Z ostatniej równości wynika, że liczba a może być zapisana na n bitach, jeśli spełnia nierówność:

$$2^n - 1 \geq a, \qquad \text{czyli} \qquad 2^n \geq a + 1.$$

Dla danej liczby a wybieramy najmniejsze n spełniające tę nierówność, gdyż początkowe zera w rozwinięciu liczby nie mają znaczenia. Logarytmując obie strony nierówności można dokładnie określić wartość n:

$$n = \lceil \log_2(a + 1) \rceil$$

Użyliśmy funkcji powała, której wartością jest najmniejsza liczba całkowita większa od liczby stojącej pod powałą, gdyż liczba bitów w rozwinięciu liczby a jest zawsze liczbą naturalną, a wartość logarytmu może nie być liczbą całkowitą,



Rysunek 3.
Liczba bitów potrzebnych do zapamiętania liczb a z poszczególnych przedziałów

Z powyższych rozważań należy zapamiętać, że liczba bitów potrzebnych do zapamiętania w komputerze liczby naturalnej jest w przybliżeniu równa logarytmowi przy podstawie 2 z wartości tej liczby. na przykład, liczba 1000 jest pamiętana na $\lceil \log_2(1000 + 1) \rceil = 10$ bitach.

Przy tej okazji porównaj szybkość wzrastania funkcji liniowej i funkcji logarytmicznej.

Ćwiczenie 14. Narysuj w jednym układzie współrzędnych wykresy dwóch funkcji, logarytmicznej i liniowej. Utwórz także tabelę wartości obu funkcji dla liczb wybranych z przedziału $[1, 10^9]$. Zauważ, jak wolno rośnie funkcja logarytmiczna w porównaniu z funkcją liniową.

Funkcja logarytmiczna odgrywa bardzo ważną rolę w informatyce.

5 SZYBKIE OBLICZANIE WARTOŚCI POTĘGI

Binarna reprezentacja liczb naturalnych jest źródłem szybkich metod obliczania wartości potęgi x^m, gdzie m jest liczbą naturalną, a x może być dowolną liczbą rzeczywistą (wartość podstawy x nie ma znaczenia dla naszych rozważań). Metody te znajdują zastosowanie w algorytmach kryptograficznych, w których są obliczane wartości potęg o bardzo dużych wykładnikach.

Posłużmy się najpierw przykładem. Przypuśćmy, że chcemy obliczyć wartość potęgi. x^{22}. Proste wymnożenie podstawy przez siebie $x^{22} = x \cdot x \cdot \ldots \cdot x$ to 21 mnożeń. A skorzystajmy z binarnej reprezentacji wykładnika potęgi. Można go przedstawić jako sumę potęg liczby 2:

$$22 = 2 + 4 + 16 = 2^1 + 2^2 + 2^4$$

wtedy nasza potęga przyjmuje postać $x^{22} = x^{2+4+16} = x^2 \cdot x^4 \cdot x^{16}$ i można ją obliczyć wielokrotnie podnosząc do kwadratu podstawę x i mnożąc przez siebie odpowiednie czynniki. Dla potęgi 22 obliczanie potęgi przebiega następująco: x^2, $x^4 = (x^2)^2$, $x^8 = (x^4)^2$, $x^{16} = (x^8)^2$ i mnożymy przez siebie $x^2 \cdot x^4 \cdot x^{16}$. W sumie wykonujemy 6 mnożeń (podniesienie do kwadratu wymaga wykonania jednego mnożenia).

Korzystają z przedstawienia wykładnika w reprezentacji binarnej $22 = (10110)_2$ w postaci schematu Hornera, wykładnik można zapisać $22 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (((2 + 0)2 + 1)2 + 1)2 + 0$, a stąd otrzymujemy:

$$x^{(((2+0)2+1)2+1)2+0} = (x^{((2+0)2+1)2+1})^2 = (x^{((2+0)2+1)2}x)^2 = (x^{(2+0)2+1})^2x)^2$$
$$= (x^{(2+0)^2}x)^2x)^2 = (x^{(2+0)})^2x)^2x)^2 = (((x^2)^2x)^2x)^2.$$

Korzystając z tego zapisu, potęgę x^{22} obliczamy wykonując kolejno następujące mnożenia: x^2, $x^4 = (x^2)^2$, $x^5 = x^4 \cdot x$, $x^{10} = (x^5)^2$, $x^{11} = x^{10} \cdot x$, $x^{22} = (x^{11})^2$. W sumie wykonujemy również 6 mnożeń.

Ćwiczenie 15. W oparciu o jedną i drugą metodę naszkicowaną powyżej, podaj kolejność wykonywania mnożeń podczas obliczania wartości potęgi x^{313}. Ile mnożeń jest wykonywanych w obu przypadkach?

Obie metody potęgowania korzystają z binarnego rozwinięcia wykładnika, ale różnią się tym, że w pierwszym przypadku to rozwinięcie jest przeglądane od najmniej znaczącego bitu (mówimy o metodzie **od prawej do lewej**), a w drugim – od najbardziej znaczącego (czyli metodą **od lewej do prawej**). Ponieważ reprezentacja binarna liczby naturalnej jest tworzona od najmniej znaczącego bitu (zob. rozdz. 4), podamy teraz algorytm obliczania wartości potęgi, który wykonuje potęgowanie rozkładając wykładnik na postać binarną (postać ta jednak nie jest zachowywana w algorytmie). Szybkie potęgowanie, wykorzystujące schemat Hornera pozostawiamy do samodzielnego wykonania.

Binarny algorytm potęgowania „od prawej do lewej”

Dane: Liczba naturalna m i dowolna liczba x.
Wynik: Wartość potęgi $y = x^m$.
Krok 0. {Ustalenie początkowych wartości potęgi y i zmiennych pomocniczych z i l.}
$y := 1$: $l := m$; $z := x$;
Krok 1. Jeśli $l \bmod 2 = 1$ (czyli l jest liczbą nieparzystą), to $y := y \cdot z$;
Krok 2. $l := l \operatorname{div} 2$; jeśli $l = 0$, to zakończ algorytm – wynikiem jest bieżąca wartość y.
Krok 3. $z := z \cdot z$; wróć do kroku 1.

Ćwiczenie 16. Wykonaj powyższy algorytm dla $m = 22$ i $m = 313$. Wypisz kolejne wartości zmiennych y, l, z i porównaj kolejne wartości zmiennej y z wcześniej otrzymanymi wartościami potęgi dla tych wartości wykładnika.

Ćwiczenie 17. Napisz program w języku Pascal, będący realizacją algorytmu podnoszenia do potęgi „od prawej do lewej”. Jako wynik, dla danego wykładnika m, wyprowadzaj liczbę mnożeń wykonywanych w tym algorytmie dla obliczenia wartości x^m.

Szybkie algorytmy potęgowania są stosowane w algorytmach szyfrujących, w których wykładniki potęg są bardzo dużymi liczbami.

6 ALGORYTM EUKLIDESA

Przedstawiamy w tym punkcie algorytm, który wśród przepisów opatrywanych tych mianem, został odkryty jako jeden z najwcześniejszych, bo jeszcze w starożytności, przez Euklidesa.

Danymi są dwie nieujemne liczby całkowite m i n. Liczba k jest **największym wspólnym dzielnikiem** m i n, jeśli dzieli m oraz n i jest największą liczbą o tej własności — oznaczamy ją przez NWD(m,n). **Algorytm Euklidesa** jest najstarszą metodą znajdowania największego wspólnego dzielnika dwóch liczb (patrz ramka obok).

Euklides stosował swój algorytm do znajdowania najdłuższego odcinka mieszczącego się całkowitą liczbę razy w dwóch danych odcinkach. Było to około 300 roku p.n.e., na długo przed pojawieniem się określenia algorytm.
Algorytm Euklidesa jest jednym z najczęściej przytaczanych algorytmów w książkach informatycznych i matematycznych. Powodem jest nie tylko szacunek dla jego „wieku", ale to, że ma wiele zastosowań w rozwiązywaniu problemów rachunkowych i posługując się nim, można ilustrować wiele podstawowych pojęć i własności informatycznych.

Ćwiczenie 18. Pierwszą metodą, jaka może przyjść na myśl, gdy chcemy znaleźć największy wspólny dzielnik dwóch danych liczb m i n, jest sprawdzanie podzielności tych liczb przez kolejne liczby naturalne, począwszy od 2, a skończywszy na mniejszej z m i n. Ile dzieleń należy wykonać, aby obliczyć tą metodą NWD(24,48) oraz NWD(46,48)? A dla dowolnych m i n? Opisz tę metodę w postaci algorytmu w wybranej przez siebie reprezentacji. Możesz opisać tę metodę w postaci programu komputerowego.

Nie możesz być chyba zadowolony z efektywności tego algorytmu, zwłaszcza dla przykładowych danych liczbowych — w pierwszym przypadku, gdy jedna z liczb dzieli drugą, od razu widać, że ta mniejsza jest poszukiwanym największym wspólnym dzielnikiem obu liczb. A jeśli żadna z liczb nie dzieli drugiej, tak jak w drugim przypadku?

Załóżmy, że $n \geq m$. Wtedy dzieląc n przez m otrzymujemy następującą równość:

$$n = qm + r, \quad \text{gdzie} \quad 0 \leq r < m \tag{8}$$

Wielkości q i r są odpowiednio **ilorazem** i **resztą** z dzielenia n przez m. Dla danych z ćwicz. 17, równość (8) przyjmuje postać: 48 = 2·24 i 48 = 1·46 + 2. Wynikają stąd następujące wnioski:

- jeśli $r = 0$, to NWD(m,n) = m, czyli jeśli jedna z liczb dzieli drugą bez reszty, to mniejsza z tych liczb jest ich największym wspólnym dzielnikiem;
- jeśli $r \neq 0$, to równość (8) można zapisać w postaci $r = n - qm$; a stąd wynika, że każda liczba dzieląca n i m dzieli całe wyrażenie po prawej stronie tej równości, a więc dzieli również r; zatem, największy wspólny dzielnik m i n dzieli również resztę r.

Te dwa wnioski można zapisać w postaci następującej równości:

$$\text{NWD}(m,n) = \text{NWD}(r,m),$$

w której przyjęliśmy, że NWD(0,m) = m, gdyż 0 jest podzielne przez każdą liczbę różną od zera.
Zastosujmy teraz tę obserwację do obliczenia NWD(25,70). Otrzymamy:

$$\text{NWD}(25,70) = \text{NWD}(20,25) = \text{NWD}(5,20) = \text{NWD}(0,5) = 5,$$

gdyż stosując równość (8) otrzymujemy kolejno:

$$70 = 2\cdot 25 + 20$$
$$25 = 1\cdot 20 + 5$$
$$20 = 4\cdot 5$$

Powyższe postępowanie zawsze się kończy, bo pierwszy element w parze argumentów funkcji NWD maleje, gdyż jest to reszta.

Możemy teraz przedstawić opis algorytmu Euklidesa. Zauważmy jeszcze, że iloraz q z równości (8) nie występuje w następnych iteracjach tej równości, zatem nie trzeba go wyznaczać.

Algorytm Euklidesa (z dzieleniem) wyznaczania NWD
Dane: Dwie liczby naturalne m i n, .
Wynik: NWD(m,n) — największy wspólny dzielnik m i n.
Krok 1. Jeśli $m = 0$, to n jest szukanym dzielnikiem. Zakończ algorytm.
Krok 2. $r := n \bmod m$, $n := m$, $m := r$. Wróć do kroku 1.

Następne ćwiczenie zwraca uwagę, że dla pewnych liczb algorytm Euklidesa wykonuje dużo iteracji.

Ćwiczenie 19. Zastosuj algorytm Euklidesa do liczb 34 i 55. Co ciekawego możesz powiedzieć o działaniu algorytmu w tym przypadku – ile wynoszą kolejne ilorazy? Wypisz od końca ciąg reszt tworzonych w tym przypadku. Jak inaczej można zdefiniować ten ciąg? Podaj inną parę liczb, dla której algorytm Euklidesa działa podobnie.

Zapiszemy teraz algorytm Euklidesa w postaci programu:

```
program Euklides;
 var m,n,r:integer;
begin
 read(m,n);
 while m>0 do begin
  r:=n mod m;
  n:=m;
  m:=r
 end;
 write(n)
end.
```

Zapiszemy teraz algorytm Euklidesa w postaci wydzielonej funkcji `NWD`, której wartością jest właśnie największy wspólny dzielnik dwóch liczb, będących argumentami tej funkcji.

```
program Euklides_funkcja;
 var m,n:integer;

 function NWD(m,n:integer):integer;
  var r:integer;
 begin
  while m>0 do begin
   r:=n mod m; n:=m; m:=r
  end;
  NWD:=n
 end; {NWD}

begin
 read(m,n);
 writeln(NWD(m,n))
end.{Euklides_funkcja}
```

W nagłówku funkcji, po jej nazwie i parametrach, podaje się typ wyniku funkcji i średnik.

W tresci funkcji występuje instrukcja przypisania nazwie funkcji jej wartości.

Istnieje jeszcze wiele innych implementacji algorytmu Euklidesa, np. wykonujących odejmowanie zamiast dzielenia. Jedna z najciekawszych implementacji wykorzystuje tzw. **rekurencję**, czyli funkcję, która odwołuje się do siebie.

```
program Euklides_rekurencja;
 var m,n:integer;
 function NWD_rek(m,n:integer):integer;
 begin
  if m>n then NWD_rek:=NWD_rek(n,m)
  else if m = 0 then NWD_rek:=n
       else NWD_rek:=NWD_rek(n mod m,m)
 end;
begin
 read(m,n);
 writeln(NWD_rek(m,n))
end.
```

Ćwiczenie 20. Zapoznaj się z przedstawionymi implementacjami algorytmu Euklidesa, uruchom te programy i przetestuj je na wybranych parach liczb naturalnych, w tym m.in. na parze liczb 34 i 55.

7 DODATEK: ALGORYTM, ALGORYTMIKA I ALGORYTMICZNE ROZWIĄZYWANIE PROBLEMÓW

Ten rozdział jest krótkim wprowadzeniem do zajęć w module „Algorytmika i programowanie". Krótko wyjaśniamy w nim podstawowe pojęcia oraz stosowane na zajęciach podejście do rozwiązywania problemów z pomocą komputera.

Algorytm

Powszechnie przyjmuje się, że **algorytm** jest opisem krok po kroku rozwiązania postawionego problemu lub sposobu osiągnięcia jakiegoś celu. To pojęcie wywodzi się z matematyki i informatyki – za pierwszy algorytm uznaje się bowiem algorytm Euklidesa (patrz rozdz. 6), podany ponad 2300 lat temu. W ostatnich latach algorytm stał się bardzo popularnym synonimem przepisu lub instrukcji postępowania.

W szkole, algorytm pojawia się po raz pierwszy na lekcjach matematyki już w szkole podstawowej, na przykład jako algorytm pisemnego dodawania dwóch liczb, wiele klas wcześniej, zanim staje się przedmiotem zajęć informatycznych.

O znaczeniu algorytmów w informatyce może świadczyć następujące określenie, przyjmowane za definicję informatyki:

informatyka jest dziedziną wiedzy i działalności zajmującą się algorytmami

W tej definicji informatyki nie ma dużej przesady, gdyż zawarte są w niej pośrednio inne pojęcia stosowane do definiowania informatyki: **komputery** – jako urządzenia wykonujące odpowiednio dla nich zapisane algorytmy (czyli niejako wprawiane w ruch algorytmami); **informacja** – jako materiał przetwarzany i produkowany przez komputery; **programowanie** – jako zespół metod i środków (np. języków i systemów użytkowych) do zapisywania algorytmów w postaci programów.

Położenie nacisku w poznawaniu informatyki na algorytmy jest jeszcze uzasadnione tym, że zarówno konstrukcje komputerów, jak i ich oprogramowanie bardzo szybko się starzeją, natomiast podstawy stosowania komputerów, które są przedmiotem zainteresowań algorytmiki, zmieniają się bardzo powoli, a niektóre z nich w ogóle nie ulegają zmianie.

Algorytmy, zwłaszcza w swoim popularnym znaczeniu, występują wszędzie wokół nas – niemal każdy ruch człowieka, zarówno angażujący jego mięśnie, jak i będący jedynie działaniem umysłu, jest wykonywany według jakiegoś przepisu postępowania, którego nie zawsze jesteśmy nawet świadomi. Wiele naszych czyn-

ności potrafimy wyabstrahować i podać w postaci precyzyjnego opisu, ale w bardzo wielu przypadkach nie potrafimy nawet powtórzyć, jak to się dzieje lub jak to się stało[3].

Nie wszystkie postępowania z naszego otoczenia, nazywane algorytmami, są ściśle związane z komputerami i nie wszystkie przepisy działań można uznać za algorytmy w znaczeniu informatycznym. Na przykład nie są nimi na ogół przepisy kulinarne, chociaż odwołuje się do nich David Harel w swoim fundamentalnym dziele o algorytmach i algorytmice [3]. Otóż przepis np. na sporządzenie „ciągutki z wiśniami", którą zachwycała się Alicja w Krainie Czarów, nie jest algorytmem, gdyż nie ma dwóch osób, które na jego podstawie, dysponując tymi samymi produktami, zrobiłyby taką samą, czyli jednakowo smakującą ciągutkę. Nie może być bowiem algorytmem przepis, który dla identycznych danych daje różne wyniki w dwóch różnych wykonaniach, jak to najczęściej bywa w przypadku robienia potraw według „algorytmów kulinarnych".

Algorytmika

Algorytmika to dział informatyki, zajmujący się różnymi aspektami tworzenia i analizowania algorytmów, przede wszystkim w odniesieniu do ich roli jako precyzyjnego opisu postępowania, mającego na celu znalezienie rozwiązania postawionego problemu. Algorytm może być wykonywany przez człowieka, przez komputer lub w inny sposób, np. przez specjalnie dla niego zbudowane urządzenie. W ostatnich latach postęp w rozwoju komputerów i informatyki był nierozerwalnie związany z rozwojem coraz doskonalszych algorytmów.

Informatyka jest dziedziną zajmującą się rozwiązywaniem problemów z wykorzystaniem komputerów. O znaczeniu algorytmu w informatyce może świadczyć fakt, że każdy program komputerowy działa zgodnie z jakimś algorytmem, a więc zanim zadamy komputerowi nowe zadanie do wykonania powinniśmy umieć „wytłumaczyć" mu dokładnie, co ma robić. Bardzo trafnie to sformułował Donald E. Knuth, jeden z najznakomitszych, żyjących informatyków:

Mówi się często, że człowiek dotąd nie zrozumie czegoś,
zanim nie nauczy tego – kogoś innego.
W rzeczywistości,
człowiek nie zrozumie czegoś naprawdę,
zanim nie zdoła nauczyć tego – komputera.

Staramy się, by prezentowane algorytmy były jak najprostsze i by działały jak najszybciej. To ostatnie żądanie może wydawać się dziwne, przecież dysponujemy już teraz bardzo szybkimi komputerami i szybkość działania procesorów stale rośnie (według prawa Moore'a podwaja się co 18 miesięcy). Mimo to istnieją problemy, których obecnie nie jest w stanie rozwiązać żaden komputer i zwiększenie szybkości komputerów niewiele pomoże, kluczowe więc staje się opracowywanie coraz szybszych algorytmów. Jak to ujął Ralf Gomory, szef ośrodka badawczego IBM:

Najlepszym sposobem przyspieszania komputerów
jest obarczanie ich mniejszą liczbą działań.

Algorytmiczne rozwiązywanie problemów

Komputer jest stosowany do rozwiązywania problemów zarówno przez profesjonalnych informatyków, którzy projektują i tworzą oprogramowanie, jak i przez tych, którzy stosują tylko technologię informacyjno-komunikacyjną, czyli nie wykraczają poza posługiwanie się gotowymi narzędziami informatycznymi. W obu przypadkach ma zastosowanie podejście do **rozwiązywania problemów algorytmicznych**, która polega na systematycznej pracy nad komputerowym rozwiązaniem problemu i obejmuje cały proces projektowania i otrzymania rozwiązania. Celem nadrzędnym tej metodologii jest otrzymanie **dobrego rozwiązania**, czyli takiego, które jest:

- **zrozumiałe dla każdego**, kto zna dziedzinę rozwiązywanego problemu i użyte narzędzia komputerowe,
- **poprawne**, czyli spełnia specyfikację problemu, a więc dokładny opis problemu,
- **efektywne**, czyli niepotrzebnie nie marnuje zasobów komputerowych, czasu i pamięci.

[3] Interesująco ujął to J. Nievergelt – *Jest tak, jakby na przykład stonoga chciała wyjaśnić, w jakiej kolejności wprawia w ruch swoje nogi, ale z przerażeniem stwierdza, że nie może iść dalej.*

Ta metoda składa się z następujących sześciu etapów:

1. **Opis i analiza sytuacji problemowej.** Na podstawie opisu i analizy sytuacji problemowej należy w pełni zrozumieć, na czym polega problem, jakie są dane dla problemu i jakich oczekujemy wyników, oraz jakie są możliwe ograniczenia.
2. **Sporządzenie specyfikacji problemu,** czyli dokładnego opisu problemu na podstawie rezultatów etapu 1. **Specyfikacja problemu** zawiera:
   - opis danych,
   - opis wyników,
   - opis relacji (powiązań, zależności) między danymi i wynikami.

   Specyfikacja jest wykorzystana w następnym etapie jako specyfikacja tworzonego rozwiązania (np. programu).
3. **Zaprojektowanie rozwiązania.** Dla sporządzonej na poprzednim etapie specyfikacji problemu, jest projektowane rozwiązanie komputerowe (np. program), czyli wybierany odpowiedni algorytm i dobierane do niego struktury danych. Wybierane jest także środowisko komputerowe (np. język programowania), w którym będzie realizowane rozwiązanie na komputerze.
4. **Komputerowa realizacja rozwiązania.** Dla projektu rozwiązania, opracowanego na poprzednim etapie, jest budowane kompletne rozwiązanie komputerowe, np. w postaci programu w wybranym języku programowania. Następnie, testowana jest poprawność rozwiązania komputerowego i badana jego efektywność działania na różnych danych.
5. **Testowanie rozwiązania.** Ten etap jest poświęcony na systematyczną weryfikację poprawności rozwiązania i testowanie jego własności, w tym zgodności ze specyfikacją.
6. **Prezentacja rozwiązania.** Dla otrzymanego rozwiązania należy jeszcze opracować dokumentację i pomoc dla (innego) użytkownika. Cały proces rozwiązywania problemu kończy prezentacja innym zainteresowanym osobom (uczniom, nauczycielowi) sposobu otrzymania rozwiązania oraz samego rozwiązania wraz z dokumentacją.

Chociaż powyższa metodologia jest stosowana głównie do otrzymywania komputerowych rozwiązań, które mają postać programów napisanych w wybranym języku programowania, może być zastosowana również do otrzymywania rozwiązań komputerowych większości problemów z obszaru zastosowań informatyki i posługiwania się technologią informacyjno-komunikacyjną[4], czyli gotowym oprogramowaniem.

Dwie uwagi do powyższych rozważań.

Uwaga 1. Wszyscy, w mniejszym lub większym stopniu, zmagamy się z problemami, pochodzącymi z różnych dziedzin (przedmiotów). W naszych rozważaniach, problem nie jest jednak wyzwaniem nie do pokonania, przyjmujemy bowiem, że **problem** jest sytuacją, w której uczeń ma przedstawić jej rozwiązanie bazując na tym, co wie, ale nie ma powiedziane, jak to ma zrobić. Problem na ogół zawiera pewną trudność, nie jest rutynowym zadaniem. Na takie sytuacje problemowe rozszerzamy pojęcie problemu, wymagającego przedstawienia rozwiązania komputerowego.

Uwaga 2. W tych rozważaniach rozszerzamy także pojęcie **programowania**. Jak powszechnie wiadomo, komputery wykonują tylko programy. Użytkownik komputera może korzystać z istniejących programów (np. za pakietu Office), a może także posługiwać się własnymi programami, napisanymi w języku programowania, który „rozumieją" komputery. W szkole nie ma zbyt wiele czasu, by uczyć programowania, uczniowie też nie są odpowiednio przygotowani do programowania komputerów. Istnieje jednak wiele sposobności, by kształcić zdolność komunikowania się z komputerem za pomocą programów, które powstają w inny sposób niż za pomocą programowania w wybranym języku programowania. Szczególnym przypadkiem takich programów jest oprogramowanie edukacyjne, które służy do wykonywania i śledzenia działania algorytmów. „Programowanie" w przypadku takiego oprogramowania polega na dobieraniu odpowiednich parametrów, które mają wpływ na działanie algorytmów i tym samym umożliwiają lepsze zapoznanie się z nimi.

[4] W najnowszej podstawie programowej dla przedmiotu informatyka, przyjętej do realizacji pod koniec 2008 roku, to podejście jest zalecane jako podstawowa metodologia rozwiązywania problemów z pomocą komputera.

LITERATURA

1. Cormen T.H., Leiserson C.E., Rivest R.L., *Wprowadzenie do algorytmów*, WNT, Warszawa 1997
2. Gurbiel E., Hard-Olejniczak G., Kołczyk E., Krupicka H., Sysło M.M., *Informatyka, Część 1 i 2, Podręcznik dla LO*, WSiP, Warszawa 2002-2003
3. Harel D., *Algorytmika. Rzecz o istocie informatyki*, WNT, Warszawa 1992
4. Knuth D.E., *Sztuka programowania*, Tomy 1 – 3, WNT, Warszawa 2003
5. Sysło M.M., *Algorytmy*, WSiP, Warszawa 1997
6. Sysło M.M., *Piramidy, szyszki i inne konstrukcje algorytmiczne*, WSiP, Warszawa 1998. Kolejne rozdziały tej książki są zamieszczone na stronie: http://www.wsipnet.pl/kluby/informatyka_ekstra.php?k=69
7. Wirth N., *Algorytmy + struktury danych = programy*, WNT, Warszawa 1980

W projekcie **Informatyka +**, poza wykładami i warsztatami, przewidziano następujące działania:

- 24-godzinne kursy dla uczniów w ramach modułów tematycznych
- 24-godzinne kursy metodyczne dla nauczycieli, przygotowujące do pracy z uczniem zdolnym
- nagrania 60 wykładów informatycznych, prowadzonych przez wybitnych specjalistów i nauczycieli akademickich
- konkursy dla uczniów, trzy w ciągu roku
- udział uczniów w pracach kół naukowych
- udział uczniów w konferencjach naukowych
- obozy wypoczynkowo-naukowe.

Szczegółowe informacje znajdują się na stronie projektu

www.informatykaplus.edu.pl