Podsumowanie

- Wartość wielomianu można obliczyć algorytmem naiwnym, który wykonuje $2n$ operacji mnożenia i n operacji dodawania, gdzie n oznacza stopień wielomianu.
- Schemat Hornera polega na wielokrotnym wyłączaniu argumentu przed nawias we wzorze ogólnym funkcji wielomianowej, aż w najbardziej wewnętrznych nawiasach otrzymamy wielomian pierwszego stopnia.
- Obliczając wartość wielomianu stopnia n z wykorzystaniem schematu Hornera, wykonujemy n operacji mnożenia i n operacji dodawania. Algorytm ten jest optymalny.
- Schemat Hornera stosuje się m.in. do obliczania wartości dziesiętnej liczb zapisanych w systemie o innej podstawie oraz do szybkiego podnoszenia do potęgi.

Zadania

* **1** Napisz program, który wczyta z klawiatury dwa wielomiany, a następnie wyznaczy wielomian będący ich sumą i go wypisze.

* **2** Napisz program, który z pliku tekstowego przekazanego ci przez nauczyciela (np. *wielomian.txt*) wczyta stopień i współczynniki wielomianu, a następnie dla argumentu wczytanego z klawiatury obliczy wartość wielomianu, korzystając ze schematu Hornera. Stopień wielomianu znajduje się w pierwszym wierszu pliku tekstowego, w kolejnych wierszach podane są współczynniki wielomianu.

* **3** Napisz program, który wczyta z klawiatury podstawę systemu pozycyjnego z zakresu od 2 do 9, liczbę cyfr liczby oraz cyfry tej liczby i obliczy jej wartość dziesiętną, korzystając ze schematu Hornera.

** **4** Napisz program obliczający wartość wielomianu zgodnie ze specyfikacją przedstawioną na s. 122, z wykorzystaniem schematu Hornera. Funkcję realizującą schemat Hornera zapisz rekurencyjnie.

** **5** Napisz program, który wczyta z klawiatury dwa wielomiany, a następnie wyznaczy wielomian będący ich iloczynem i go wypisze.
Wskazówka: Stopień wielomianu będącego iloczynem wielomianów to suma stopni wielomianów składowych.

*** **6** Napisz program, który wczyta z klawiatury wielomian oraz liczbę całkowitą c i obliczy iloraz wielomianu przez dwumian $x - c$. Zastosuj schemat Hornera.

*** **7** Napisz program obliczający przybliżoną wartość $\sin x$ ze wzoru:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

dla $0 \leqslant x \leqslant 6{,}28$. Obliczenia zakończ po zsumowaniu sześciu pierwszych wyrazów. Wykorzystaj schemat Hornera, a wartości silni zapamiętaj w tablicy.

8. Metody obliczeń przybliżonych

W wielu sytuacjach wymagających zastosowania obliczeń łatwo jest otrzymać dokładny wynik – np. w wyborach parlamentarnych, w których zliczany jest każdy oddany głos. Istnieją jednak problemy, których nie potrafimy rozwiązać za pomocą znanych algorytmów znajdujących dokładne wyniki lub algorytmy takie są czasochłonne. Można wówczas wykorzystać metody, które dają wyniki przybliżone.

Cele lekcji

- Znajdziesz miejsce zerowe funkcji metodą bisekcji.
- Obliczysz pierwiastek kwadratowy z liczby metodą bisekcji i metodą Newtona–Raphsona.
- Poznasz metodę prostokątów i metodę trapezów, wykorzystywane do obliczania pól obszarów zamkniętych.
- Zastosujesz metodę Monte Carlo do policzenia przybliżonej wartości liczby π oraz do symulacji ruchów Browna.

W poprzednich tematach zajmowaliśmy się m.in. **znajdowaniem pierwiastków równania kwadratowego** oraz **obliczaniem wartości wielomianu**. Choć przedstawione tam algorytmy umożliwiają znalezienie dokładnych wyników, to programy realizujące te algorytmy podają rozwiązania obarczone błędami. Wynikają one jednak z reprezentacji liczb w komputerze, a nie z samych algorytmów. Teraz zajmiemy się problemami i metodami ich rozwiązania, w których przybliżony wynik jest przede wszystkim skutkiem zastosowanego algorytmu.

Algorytmy znajdowania pierwiastków równania kwadratowego, s. 115–119

Obliczanie wartości wielomianu, s. 122–126

8.1. Znajdowanie miejsc zerowych funkcji

Jedną z metod znajdowania miejsc zerowych funkcji, czyli argumentów, dla których funkcja przyjmuje wartość 0, jest **metoda bisekcji (połowienia)**. Postępuje się w niej podobnie jak w **algorytmie przeszukiwania binarnego** – w kolejnych krokach zmniejszamy o połowę zakres przeszukiwanych danych.

Metoda bisekcji (połowienia)

Przeszukiwanie binarne, podręcznik *Informatyka na czasie 2. Zakres rozszerzony*, s. 154

W ogólnym przypadku w metodzie bisekcji konstruujemy ciąg, którego wyrazami są przybliżenia poszukiwanej wartości, a granicą tego ciągu jest poszukiwana wartość. Kolejne wyrazy ciągu to środki coraz mniejszych przedziałów, które przeszukujemy. Zakres poszukiwań zawężamy o połowę, zastępując jeden z końców przedziału środkiem przedziału. Kończymy, gdy wartość kolejnego wyrazu ciągu jest równa szukanej wartości lub gdy długość przeszukiwanego przedziału jest nie większa od przyjętej dokładności obliczeń (wynik jest wówczas przybliżony – jest nim ostatnio wyznaczony środek przedziału).

Aby można było zastosować metodę bisekcji do poszukiwania miejsca zerowego funkcji f w przedziale $[a; b]$, muszą być spełnione warunki:

- funkcja f jest określona i ciągła w przedziale $[a; b]$,
- wartości funkcji dla końców przedziału $[a; b]$ mają różne znaki, czyli $f(a) \cdot f(b) < 0$.

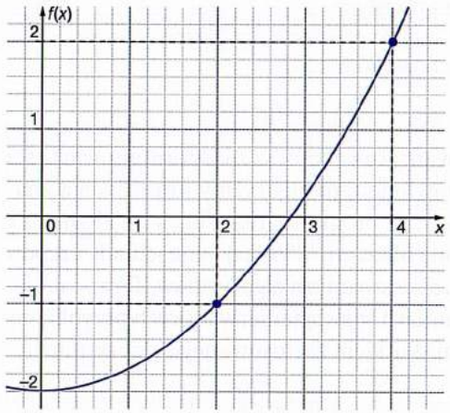
Warto wiedzieć

Istnieją inne metody przybliżonego znajdowania miejsc zerowych funkcji, np. metoda siecznych.

Zauważ, że jeśli funkcja jest ciągła w przedziale $[a; b]$ i wartości tej funkcji dla końców przedziału mają różne znaki, musi istnieć przynajmniej jedno miejsce zerowe. Takiej pewności nie mielibyśmy, gdyby wartości funkcji dla końców przedziału miały jednakowe znaki.

Metodą bisekcji w danym przedziale będziemy szukać jednego miejsca zerowego. Żeby znaleźć inne miejsca zerowe, należałoby wielokrotnie stosować algorytm dla różnych przedziałów.

Rysunek 8.1 przedstawia fragment wykresu funkcji $f = \frac{1}{4}x^2 - 2$. Jest ona określona i ciągła w przedziale [2; 4]. Ponadto wartości funkcji dla końców przedziału mają różne znaki. Można zatem zastosować metodę bisekcji do poszukiwania miejsca zerowego tej funkcji w przedziale [2; 4].



Rys. 8.1. Fragment wykresu funkcji $f = \frac{1}{4}x^2 - 2$, określonej i ciągłej w przedziale [2; 4], $f(2) \cdot f(4) < 0$

Na początku poszukiwania miejsca zerowego musimy ustalić dokładność obliczeń. W kolejnych krokach najpierw sprawdzamy, czy wartość funkcji dla środka przedziału jest równa 0 – jeśli tak jest, znaleźliśmy miejsce zerowe i algorytm kończy działanie. W przeciwnym przypadku dzielimy przedział na pół, zastępując jeden z jego końców wyznaczonym środkiem (wybieramy połowę spełniającą warunki metody bisekcji). Jeśli długość przedziału jest większa od przyjętej dokładności obliczeń, ponawiamy poszukiwanie miejsca zerowego w tym przedziale, jeśli nie – algorytm kończy działanie, a za miejsce zerowe przyjmujemy ostatnio wyznaczony środek przedziału (wynik jest przybliżony).

Tabela 8.1 zawiera informacje o przedziałach i wyliczanych wartościach w sześciu początkowych krokach algorytmu dla funkcji z rysunku 8.1.

Krok algorytmu	Rozpatrywany przedział	Wartość funkcji dla lewego końca przedziału	Wartość funkcji dla prawego końca przedziału	Środek przedziału	Wartość funkcji dla środka przedziału
1	[2; 4]	−1	2	3	0,25
2	[2; 3]	−1	0,25	2,5	−0,4375
3	[2,5; 3]	−0,4375	0,25	2,75	−0,109375
4	[2,75; 3]	−0,109375	0,25	2,875	0,06640625
5	[2,75; 2,875]	−0,109375	0,06640625	2,8125	−0,0224609375
6	[2,8125; 2,875]	−0,0224609375	0,06640625	2,84375	0,021728515625

Tabela 8.1. Informacje o przedziałach oraz wyliczanych wartościach w sześciu pierwszych krokach algorytmu poszukującego miejsca zerowego funkcji $f = 0{,}25x^2 - 2$ w przedziale [2; 4] metodą bisekcji

Gdybyśmy za dokładność obliczeń przyjęli 0,01, otrzymalibyśmy miejsce zerowe równe 2,82812. Dla dokładności 0,00001 wyznaczonym miejscem zerowym byłaby liczba 2,82843. Dokładną wartością miejsca zerowego jest $2\sqrt{2}$, czyli w przybliżeniu 2,828427.

Oto specyfikacja problemu poszukiwania miejsca zerowego funkcji w danym przedziale:

Specyfikacja

Dane: a, b – liczby rzeczywiste, a < b,
f(x) – funkcja określona i ciągła w przedziale [a; b], f(a) i f(b) mają różne znaki, czyli f(a) · f(b) < 0,
eps – dokładność obliczeń.

Wynik: x0 – miejsce zerowe funkcji f(x) w przedziale [a; b] znalezione z dokładnością eps.

Najpierw obliczymy wartość x0 = (a + b)/2. Jeśli f(x0) = 0, to szukanym miejscem zerowym jest x0. W przeciwnym przypadku poszukujemy dalej miejsca zerowego: w przedziale [a; x0], gdy f(a) · f(x0) < 0, albo w przedziale [x0; b], gdy f(x0) · f(b) < 0. Obliczenia kończymy, gdy otrzymamy f(x0) = 0 lub długość przedziału jest nie większa od dokładności obliczeń eps. Oto zapis algorytmu w pseudokodzie:

```
x0 ← (a + b)/2
dopóki f(x0) ≠ 0 oraz b − a > eps wykonuj
    jeśli f(a) * f(x0) < 0 to b ← x0
    w przeciwnym przypadku a ← x0
    x0 ← (a + b)/2
```

Dobra rada

Przy ustalaniu dokładności obliczeń w programie znajdującym miejsce zerowe funkcji pamiętaj o ograniczeniach komputera – przechowuje on liczby z określoną precyzją. Ponadto zbyt duża liczba iteracji w algorytmie może znacznie spowolnić działanie programu.

Implementując powyższy algorytm, założymy, że mamy zdefiniowane stałą EPS, informującą o dokładności obliczeń, oraz funkcję f, znajdującą wartość funkcji dla parametru x.

Definicja stałej oraz nagłówek funkcji mogą wyglądać następująco:

```
const float EPS=0.00001;
float f(float x);
```

Oto kod źródłowy funkcji `main` programu znajdującego miejsce zerowe funkcji `f` metodą bisekcji:

Fragment kodu źródłowego programu znajdującego miejsce zerowe funkcji `f` metodą bisekcji – funkcja `main`

```
1.  int main()
2.  {
3.      float a, b, x0, fa, fb, fx0;
4.      cout<<"a = "; cin>>a;
5.      cout<<"b = "; cin>>b;
6.      fa=f(a); fb=f(b);
7.      if (fa*fb>=0)
8.      {
9.          cout<<"Wartosci funkcji dla koncow przedzialu ";
10.         cout<<"nie maja roznych znakow"<<endl;
11.         return 0;
12.     }
13.     x0=(a+b)/2; fx0=f(x0);
14.     while (abs(fx0)>EPS && b-a>EPS)
15.     {
16.         if (fa*fx0<0) b=x0;
17.         else
18.         {
19.             a=x0; fa=fx0;
20.         }
21.         x0=(a+b)/2; fx0=f(x0);
22.     }
23.     cout<<"x0 = "<<x0;
24.     return 0;
25. }
```

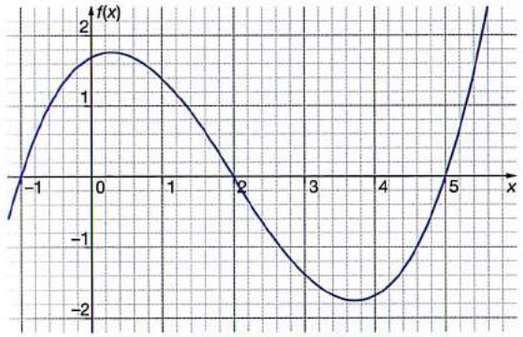
Dobra rada

Pamiętaj, że z funkcji abs możesz korzystać w programie, jeśli dołączysz bibliotekę `cmath`.

W linii 3 oprócz zmiennych `a`, `b`, `x0` są zadeklarowane zmienne `fa`, `fb` i `fx0`, służące do zapamiętania wartości funkcji `f` dla parametrów odpowiednio `a`, `b`, `x0`. Dzięki temu nie musimy parokrotnie obliczać wartości funkcji dla tych samych parametrów. W liniach 4–5 wczytywane są końce przedziału, a w linii 6 obliczane są wartości funkcji dla tych końców. Dodatkowo w liniach 7–12 sprawdzamy, czy wartości funkcji dla końców przedziału mają różne znaki. Jeśli znaki nie są różne, to program kończy działanie – metodą bisekcji nie znajdziemy miejsca zerowego w tym przedziale (co nie oznacza, że go nie ma).

Linie 13–22 realizują algorytm zapisany w pseudokodzie. Zwróć uwagę na linię 14 – nie porównujemy w niej wartości `fx0` z zerem, lecz ze stałą `EPS`. Należy tak zrobić, ponieważ zmienna `fx0` jest reprezentowana jako liczba zmiennopozycyjna, czyli wartość pamiętana przez komputer jest zaokrąglana. Instrukcja z linii 23 wypisuje otrzymane miejsce zerowe.

Rysunek 8.2 przedstawia wykres funkcji $f(x) = \frac{1}{6}x^3 - x^2 + \frac{1}{2}x + \frac{5}{3}$ oraz efekty wykonania programu dla tej funkcji i różnych przedziałów.




```
a = -3
b = 0
x0 = -1
```

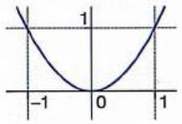
```
a = 0
b = 3
x0 = 2
```

```
a = 3
b = 10
x0 = 5
```

Rys. 8.2. Wykres funkcji $f(x) = \frac{1}{6}x^3 - x^2 + \frac{1}{2}x + \frac{5}{3}$ oraz efekty wykonania programu dla przedziałów [–3; 0], [0; 3] i [3; 10]

Warto wiedzieć

Istnieją funkcje ciągłe, które mają miejsce zerowe, ale nie mają wartości o różnych znakach dla końców jakiegokolwiek przedziału. Dlatego metodą bisekcji nie znajdziemy ich miejsca zerowego. Taką funkcją jest np. $f(x) = x^2$.



Oto definicja funkcji `f` dla przykładu z rysunku 8.2:

Definicja funkcji obliczającej wartość funkcji $f(x) = \frac{1}{6}x^3 - x^2 + \frac{1}{2}x + \frac{5}{3}$ dla danego parametru

```
1.  float f(float x)
2.  {
3.      return 1.0/6*x*x*x-x*x+0.5*x+5.0/3;
4.  }
```

Ćwiczenie 1

Napisz program znajdujący miejsce zerowe funkcji metodą bisekcji.

Zapamiętaj

Metoda bisekcji (połowienia) umożliwia znalezienie miejsca zerowego funkcji określonej i ciągłej w przedziale $[a; b]$, jeśli wartości funkcji dla końców przedziału mają różne znaki, czyli $f(a) \cdot f(b) < 0$. Metoda polega na zawężaniu w każdym kroku przedziału poszukiwań o połowę do momentu, gdy środek przedziału jest miejscem zerowym lub długość przedziału jest nie większa od dokładności obliczeń (wówczas wynikiem jest ostatnio wyznaczony środek przedziału). Zawężając przedział, wybieramy tę połowę, dla której zachowany jest warunek $f(a) \cdot f(b) < 0$.

Dobra rada

Pamiętaj, że dla argumentów całkowitych znak / oznacza część całkowitą z dzielenia. Jeśli chcesz, żeby program potraktował znak / jako dzielenie z częścią ułamkową, przynajmniej jeden z argumentów zapisz z użyciem kropki. Na przykład dla `5.0/3` otrzymasz wynik rzeczywisty, a dla `5/3` – całkowity.

8.2. Obliczanie pierwiastka kwadratowego

Pierwiastek kwadratowy z nieujemnej liczby rzeczywistej obliczaliśmy za pomocą **funkcji** `sqrt`. Pokażemy, jak to zrobić bez użycia tej funkcji.

Funkcja `sqrt`, s. 117

Obliczanie pierwiastka kwadratowego metodą bisekcji

Na początku będziemy poszukiwać pierwiastka kwadratowego z liczby rzeczywistej większej od 1. Oto specyfikacja problemu:

Specyfikacja

Dane: c – liczba rzeczywista, c > 1,
eps – dokładność obliczeń.

Wynik: pkw – pierwiastek kwadratowy z liczby c wyliczony z dokładnością eps.

Metoda bisekcji (połowienia), s. 129

Poszukiwanie pierwiastka kwadratowego z $c > 1$ **metodą bisekcji** sprowadza się do poszukiwania miejsca zerowego funkcji $f(x) = x^2 - c$ w przedziale $[1; c]$ – w tym przedziale jest ono równe $\sqrt{c}$.

Oto algorytm znajdowania pierwiastka kwadratowego z liczby rzeczywistej $c > 1$ metodą bisekcji, zapisany w pseudokodzie:

```
a ← 1
b ← c
dopóki b - a > eps wykonuj
    pkw ← (a + b)/2
    jeśli pkw * pkw < c to a ← pkw
    w przeciwnym przypadku b ← pkw
pkw ← (a + b)/2
```

Warto wiedzieć

Metodą bisekcji można także znaleźć pierwiastek kwadratowy z dodatniej liczby rzeczywistej $0 < c < 1$. Należy wówczas poszukiwać miejsca zerowego funkcji $f(x) = x^2 - c$ w przedziale $[c; 1]$.

Najpierw obliczamy środek przedziału (pkw). Jeśli jego kwadrat jest mniejszy od c, poszukiwania kontynuujemy w lewej połowie przedziału, w przeciwnym przypadku – w prawej. Obliczenia kończymy, gdy długość przedziału jest mniejsza od dokładności obliczeń eps lub jej równa.

Ćwiczenie 2

Napisz program znajdujący przybliżoną wartość pierwiastka kwadratowego z liczby rzeczywistej większej od 1 metodą bisekcji.

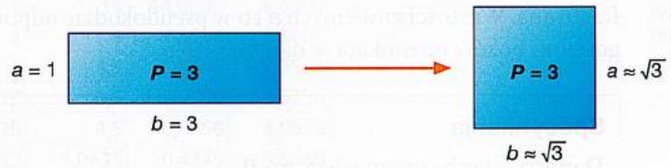
Obliczanie pierwiastka kwadratowego metodą Newtona–Raphsona

Zauważ, że $\sqrt{c}$, gdzie c jest dodatnią liczbą rzeczywistą, odpowiada długości boku kwadratu o polu c. Poszukiwanie takiego kwadratu można rozpocząć od dowolnego prostokąta o bokach a i b oraz polu c. Potem zmieniamy boki prostokąta w następujący sposób: bok a zastępujemy średnią arytmetyczną boków z poprzedniego kroku, bok b wyliczamy, dzieląc pole prostokąta przez bok a. Działania powtarzamy do momentu, gdy różnica długości boków prostokąta będzie nie większa od założonej dokładności obliczeń (boki będą takiej samej lub prawie takiej samej długości). Opisany sposób postępowania nazywa się **metodą Newtona–Raphsona**. Jej działanie pokażemy na przykładzie.

Metoda Newtona–Raphsona

Algorytm obliczania pierwiastka kwadratowego metodą Newtona–Raphsona

Znajdziemy przybliżoną wartość $\sqrt{3}$. Odpowiada ona długości boku kwadratu o polu 3. Poszukiwanie kwadratu o takim polu można rozpocząć od dowolnego prostokąta o iloczynie długości boków równym 3. My zaczniemy od prostokąta o bokach $a = 1$ i $b = 3$. Przyjmiemy, że kolejne przybliżenia $\sqrt{3}$ to długości boku a wyznaczane w poszczególnych krokach. Wartość pierwiastka znajdziemy z dokładnością do 0,00001, zatem przybliżenia wyznaczamy, dopóki $|a - b| > 0{,}00001$. Wyjściowy prostokąt oraz wynik, do którego dążymy, ilustruje poniższy rysunek.



Krok 1

Zmieniamy długości boków prostokąta tak, aby długość boku a była średnią arytmetyczną długości boków wyjściowego prostokąta, czyli:

$a = \frac{(1+3)}{2} = 2$. Zatem $b = \frac{3}{2} = 1{,}5$.

$|a - b| > 0{,}00001$, więc przechodzimy do kolejnego kroku.

Krok 2

Obliczamy długości boków a i b:

$a = \frac{(2+1{,}5)}{2} = 1{,}75$

$b = \frac{3}{1{,}75} \approx 1{,}71429$

$|a - b| > 0{,}00001$, więc szukamy dalej.

Krok 3

Obliczamy długości boków a i b:

$a \approx \frac{(1{,}75 + 1{,}71429)}{2} \approx 1{,}73215$

$b \approx \frac{3}{1{,}73215} \approx 1{,}73195$

$|a - b| > 0{,}00001$, więc przechodzimy do następnego kroku.

Krok 4

Obliczamy długości boków a i b:

$a \approx \frac{(1{,}73215 + 1{,}73195)}{2} \approx 1{,}73205$

$b \approx \frac{3}{1{,}73205} \approx 1{,}73205$

$|a - b| < 0{,}00001$, zatem algorytm kończy działanie.

Wynikiem działania algorytmu jest przybliżona wartość $\sqrt{3}$ równa 1,73205.

Warto wiedzieć

Inny ciąg kolejnych przybliżeń, zbieżny do poszukiwanego pierwiastka kwadratowego, tworzą kolejne wartości boku b prostokąta.

Wyznaczane w poszczególnych krokach długości boku a tworzą ciąg złożony z kolejnych przybliżeń $\sqrt{c}$, a granicą tego ciągu jest $\sqrt{c}$. Jeśli pierwszym wyrazem ciągu będzie 1 – długość boku a pierwszego prostokąta – to n-ty wyraz tego ciągu, gdzie n jest liczbą całkowitą dodatnią, można wyrazić wzorem rekurencyjnym:

$$a_n = \begin{cases} 1 & \text{dla } n = 1 \\ \dfrac{a_{n-1} + \frac{c}{a_{n-1}}}{2} & \text{dla } n > 1 \end{cases}$$

Warto wiedzieć

Metodę Newtona–Raphsona można wykorzystać także do znalezienia miejsca zerowego funkcji.

Oto zmodyfikowana specyfikacja problemu znajdowania przybliżonej wartości pierwiastka kwadratowego oraz zapisany w pseudokodzie algorytm znajdujący wartość pierwiastka kwadratowego metodą Newtona–Raphsona. Wartości zmiennych a i b w pseudokodzie odpowiadają długościom boków prostokąta w danym kroku.

Specyfikacja

Dane: c – liczba rzeczywista, c > 0,
eps – dokładność obliczeń.

Wynik: pkw – pierwiastek kwadratowy z liczby c wyliczony z dokładnością eps.

Warto wiedzieć

W podanym algorytmie wyznaczania pierwiastka kwadratowego metodą Newtona–Raphsona pierwszym wyrazem ciągu (wartością początkową zmiennej a) może być dowolna liczba dodatnia. Wówczas wartością początkową zmiennej b będzie c/a.

```
a ← 1
b ← c
dopóki |b − a| > eps wykonuj
    a ← (a + b)/2
    b ← c/a
pkw ← (a + b)/2
```

Ćwiczenie 3

Napisz program znajdujący przybliżoną wartość pierwiastka kwadratowego z dodatniej liczby rzeczywistej metodą Newtona–Raphsona.

Zapamiętaj

Metoda Newtona–Raphsona służy do znajdowania pierwiastka kwadratowego z dodatniej liczby rzeczywistej c i polega na obliczeniu przybliżonej długości boku kwadratu o polu c z wykorzystaniem prostokątów o tym samym polu. Za wartości początkowe długości boków prostokąta można przyjąć dowolne liczby, których iloczyn jest równy c, np. 1 i c. Długości boków w każdym kolejnym kroku to średnia arytmetyczna dotychczasowych długości oraz iloraz wartości c i policzonej średniej. Obliczenia kończymy, gdy długości boków prostokąta różnią się o nie więcej niż dokładność obliczeń, a więc otrzymaliśmy w przybliżeniu kwadrat.

Porównanie metod bisekcji i Newtona–Raphsona

Zastanówmy się, za pomocą której z dwóch omówionych metod szybciej otrzymamy przybliżenie wartości pierwiastka kwadratowego. Łatwo zauważyć, że szybsza jest metoda Newtona–Raphsona, ponieważ w jednym kroku zmieniamy obydwa końce przedziału poszukiwań (długości boków prostokąta). W przypadku metody bisekcji zmienialiśmy w jednym kroku tylko jeden koniec. Ilustruje to dobrze implementacja obu algorytmów w arkuszu kalkulacyjnym (rys. 8.3).

Warto wiedzieć

Przybliżenie pierwiastka trzeciego stopnia z dodatniej liczby rzeczywistej c odpowiada długości krawędzi sześcianu o objętości c. Długość tej krawędzi można wyznaczyć, wykorzystując prostopadłościany o tej samej objętości. Za długości krawędzi wyjściowego prostopadłościanu można przyjąć np. liczby 1, 1 i c.

1	c = 15			eps = 0,001
2	a	b	b - a	pkw
3	1	15	14,0000	8,0000
4	1	8	7,0000	4,5000
5	1	4,5	3,5000	2,7500
6	2,75	4,5	1,7500	3,6250
7	3,625	4,5	0,8750	4,0625
8	3,625	4,0625	0,4375	3,8438
9	3,84375	4,0625	0,2188	3,9531
10	3,84375	3,953125	0,1094	3,8984
11	3,84375	3,898438	0,0547	3,8711
12	3,871094	3,898438	0,0273	3,8848
13	3,871094	3,884766	0,0137	3,8779
14	3,871094	3,87793	0,0068	3,8745
15	3,871094	3,874512	0,0034	3,8728
16	3,872803	3,874512	0,0017	3,8737
17	3,872803	3,873657	0,0009	3,8732

1	c = 15		eps = 0,001	
2	a	b	|b - a|	pkw
3	1	15	14,0000	8,0000
4	8	1,875	6,1250	4,9375
5	4,9375	3,037975	1,8995	3,9877
6	3,987737	3,761532	0,2262	3,8746
7	3,874634	3,871333	0,0033	3,8730
8	3,872984	3,872983	0,0000	3,8730

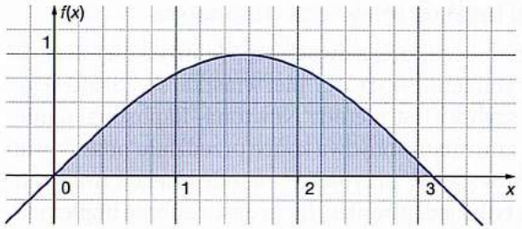
Rys. 8.3. Implementacja w arkuszu kalkulacyjnym algorytmów znajdowania pierwiastka kwadratowego metodą bisekcji (z lewej) oraz metodą Newtona–Raphsona (z prawej)

Ćwiczenie 4

Przygotuj w arkuszu kalkulacyjnym implementację algorytmów obliczania przybliżonej wartości pierwiastka kwadratowego:
a) metodą bisekcji,
b) metodą Newtona–Raphsona.

8.3. Obliczanie pól obszarów zamkniętych

Umiesz zapewne policzyć pola wybranych obszarów zamkniętych ograniczonych odcinkami, takich jak trójkąty, prostokąty czy trapezy. Jak policzyć pole obszaru zamkniętego ograniczonego wybraną krzywą i osią układu współrzędnych? Weźmy np. obszar ograniczony przez wykres funkcji $f(x) = \sin x$ i oś OX w przedziale od 0 do π. Na rysunku 8.4 na s. 138 obszar ten jest zaznaczony niebieskim kolorem.



Rys. 8.4. Obszar zamknięty ograniczony wykresem funkcji $f(x) = \sin x$ i osią OX w przedziale $[0; \pi]$

Ogólnie problem możemy sformułować następująco: chcemy obliczyć przybliżone pole obszaru zamkniętego ograniczonego prostymi $x = a$, $x = b$, osią OX oraz funkcją $f(x)$, która jest ciągła i ma nieujemne wartości dla argumentów z przedziału $[a; b]$. Niebieski obszar z rysunku 8.4 jest ograniczony przez proste $x = 0$, $x = \pi$, oś OX oraz funkcję $f(x) = \sin x$. Oto specyfikacja problemu:

Specyfikacja

Dane: a, b – liczby rzeczywiste, a < b,
f(x) – funkcja ciągła o nieujemnych wartościach dla argumentów z przedziału [a; b].

Wynik: s – przybliżona wartość pola obszaru zamkniętego ograniczonego prostymi x = a, x = b oraz funkcją f(x) i osią OX.

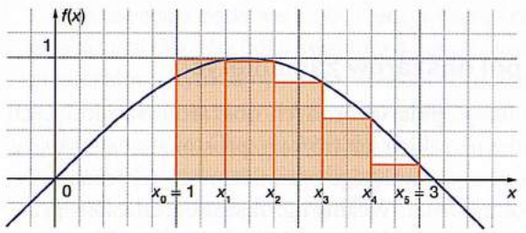
Metoda prostokątów

Jedną z metod jest podzielenie przedziału $[a; b]$ na n części tej samej długości, czyli ciąg przedziałów $[x_0, x_1]$, $[x_1, x_2]$, ..., $[x_{n-1}, x_n]$, $x_0 = a$ i $x_n = b$. Następnie oblicza się pola prostokątów o wierzchołkach w punktach o współrzędnych $(x_{i-1}, 0)$, $(x_i, 0)$, $(x_{i-1}, f(x_i))$, $(x_i, f(x_i))$, gdzie $i = 1, ..., n$, i sumuje te pola. Otrzymamy w ten sposób przybliżoną wartość pola obszaru zamkniętego. Metodę tę nazywamy **metodą prostokątów**. Rysunek 8.5 ilustruje ją dla funkcji $f(x) = \sin x$ oraz przedziału $[1; 3]$ (prostych $x = 1$ i $x = 3$) i dla $n = 5$. Prostokąty, których pola są sumowane, zaznaczono pomarańczowym kolorem.

Metoda prostokątów

Warto wiedzieć

W przypadku górnych wierzchołków prostokątów drugie współrzędne mogłyby być też równe $f(x_{i-1})$. Na pewno jednak muszą być takie same.



Rys. 8.5. Szukanie metodą prostokątów przybliżenia pola obszaru ograniczonego funkcją $f(x) = \sin x$ i osią OX w przedziale $[1; 3]$ dla $n = 5$

Niektóre prostokąty będą obejmować obszar z nadmiarem (wykraczać poza wykres funkcji), inne – obszar z niedomiarem. Im większa będzie wartość n, tym niedomiar lub nadmiar będzie mniejszy, a przybliżenie wartości pola dokładniejsze. Oto zapis algorytmu w pseudokodzie:

```
dx ← (b − a)/n
x ← a
s ← 0
dla i ← 1, 2, ..., n wykonuj
    x ← x + dx
    s ← s + dx * f(x)
```

Zmienna dx pamięta długość przedziału, czyli szerokość prostokąta. W zmiennej s zapiszemy sumę pól prostokątów. Pole pojedynczego prostokąta to iloczyn jego szerokości (dx) i wysokości (wartość funkcji dla argumentu x). Kod źródłowy funkcji main programu realizującego opisany algorytm jest następujący:

```
1.  int main()
2.  {
3.      float a, b, x, dx, s=0;
4.      int n;
5.      cout<<"a = "; cin>>a;
6.      cout<<"b = "; cin>>b;
7.      cout<<"n = "; cin>>n;
8.      dx=(b-a)/n;
9.      x=a;
10.     for (int i=1;i<=n;i++)
11.     {
12.         x=x+dx;
13.         s=s+dx*f(x);
14.     }
15.     cout<<"Pole obszaru: "<<s;
16.     return 0;
17. }
```

Fragment kodu źródłowego programu obliczającego metodą prostokątów przybliżoną wartość pola obszaru ograniczonego funkcją $f(x)$ i osią OX w przedziale $[a; b]$ – funkcja main

Rysunek 8.6 przedstawia efekty wykonania programu obliczającego metodą prostokątów pole obszaru ograniczonego przez funkcję $f(x) = \sin x$ oraz oś OX w przedziale $[0; \pi]$, odpowiednio dla $n = 10$ oraz $n = 100$. Dokładna wartość pola tego obszaru wynosi 2. Widać, że dla $n = 100$ otrzymaliśmy dokładniejsze przybliżenie.

```
a = 0
b = 3.14
n = 10
Pole obszaru: 1.98379
```

```
a = 0
b = 3.14
n = 100
Pole obszaru: 1.99986
```

Rys. 8.6. Przybliżenia pola obszaru ograniczonego funkcją $f(x) = \sin x$ i osią OX w przedziale $[0; \pi]$ dla $n = 10$ i $n = 100$ wyliczone metodą prostokątów

Dobra rada

Jeśli chcesz obliczyć metodą prostokątów pole obszaru znajdującego się pod osią OX, wystarczy, że dla wartości funkcji zastosujesz wartość bezwzględną.

Ćwiczenie 5

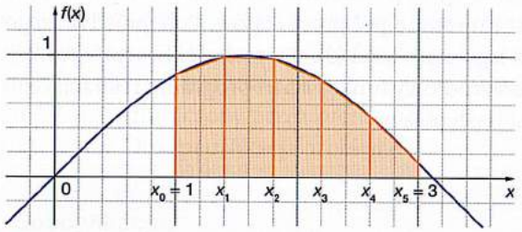
Napisz program obliczający metodą prostokątów przybliżoną wartość pola obszaru zamkniętego, zgodnie ze specyfikacją na s. 138.

Metoda trapezów

Metoda trapezów

Metoda prostokątów, s. 138

Podczas obliczania przybliżonej wartości pola obszaru zamkniętego **metodą trapezów** postępuje się bardzo podobnie jak w wypadku **metody prostokątów**. Zamiast prostokątów do przybliżania pola używa się trapezów. Ilustruje to rysunek 8.7.



Rys. 8.7. Szukanie metodą trapezów przybliżenia pola obszaru ograniczonego funkcją $f(x) = \sin x$ i osią OX w przedziale $[1; 3]$ dla $n = 5$

Pole pojedynczego trapezu możemy obliczyć, przyjmując za wysokość poziomy odcinek pomiędzy punktami x_{i-1} i x_i o długości dx, a za podstawy – odcinki o długościach $f(x_{i-1})$ i $f(x_i)$, łączące odpowiednio punkty o współrzędnych $(x_{i-1}, 0)$ z $(x_{i-1}, f(x_{i-1}))$ oraz $(x_i, 0)$ z $(x_i, f(x_i))$, gdzie $i = 1, \ldots, n$. Pole pojedynczego trapezu wyraża się więc wzorem:

$$P = \frac{1}{2} \cdot \left(f(x_{i-1}) + f(x_i)\right) \cdot (x_i - x_{i-1}) = \frac{1}{2} \cdot \left(f(x_{i-1}) + f(x_i)\right) \cdot dx$$

Oto zapis algorytmu w pseudokodzie:

```
dx ← (b - a)/n
x ← a
s ← 0
f1 ← f(x)
dla i ← 1, 2, ..., n wykonuj
    x ← x + dx
    f2 ← f(x)
    s ← s + dx * (f1 + f2)/2
    f1 ← f2
```

Wprowadziliśmy dwie dodatkowe zmienne `f1` i `f2`, w których są pamiętane wartości funkcji dla końców rozpatrywanego przedziału $[x_{i-1}; x_i]$, gdzie $i = 1, \ldots, n$.

Fragment kodu źródłowego funkcji `main` programu, który realizuje algorytm obliczania przybliżonej wartości pola obszaru zamkniętego metodą trapezów, jest następujący.


```
1.      dx=(b-a)/n;
2.      x=a;
3.      f1=f(x);
4.      for (int i=1;i<=n;i++)
5.      {
6.          x=x+dx;
7.          f2=f(x);
8.          s=s+dx*(f1+f2)/2;
9.          f1=f2;
10.     }
```

Fragment kodu źródłowego funkcji `main` programu, który oblicza metodą trapezów przybliżoną wartość obszaru ograniczonego funkcją $f(x)$ i osią OX w przedziale $[a; b]$

Ćwiczenie 6

Napisz program obliczający metodą trapezów przybliżoną wartość pola obszaru zamkniętego, zgodnie ze specyfikacją na s. 138.

Rysunek 8.8 przedstawia porównanie metody prostokątów i metody trapezów na przykładzie funkcji $f(x) = \sin x$ w przedziale $[0; \frac{\pi}{2}]$ i $n = 10$. W tym przedziale funkcja jest rosnąca, więc wszystkie prostokąty zawyżają wartość pola. Metoda trapezów w tym przypadku daje zdecydowanie lepsze przybliżenie.

```
a = 0
b = 1.57
n = 10
Pole obszaru: 1.07565
```

```
a = 0
b = 1.57
n = 10
Pole obszaru: 0.99715
```

Rys. 8.8. Przybliżenie pola obszaru zamkniętego ograniczonego funkcją $f(x) = \sin x$ i osią OX w przedziale $[0; \frac{\pi}{2}]$ dla $n = 10$ wyliczone metodą prostokątów (z lewej) i metodą trapezów (z prawej)

Zapamiętaj

Metoda prostokątów, służąca do obliczania pola obszarów zamkniętych ograniczonych wykresem funkcji w określonym przedziale, polega na wyznaczaniu przybliżonej wartości tego pola za pomocą sumy pól odpowiednio dobranych prostokątów. Natomiast metoda trapezów korzysta w podobny sposób z odpowiednio dobranych trapezów.

8.4. Znajdowanie przybliżenia liczby π

Do poszukiwania przybliżonej wartości liczby π można wykorzystać **metodę prostokątów** lub **metodę trapezów**. Można też zastosować metodę Monte Carlo.

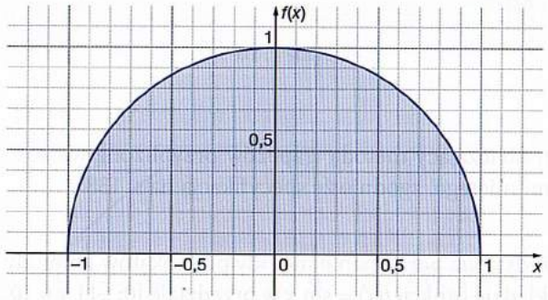
Metoda prostokątów, s. 138
Metoda trapezów, s. 140

Przybliżenie liczby π – metoda prostokątów i metoda trapezów

Dobra rada

Liczba π to iloraz długości okręgu do długości jego średnicy.

Równanie okręgu o środku w punkcie (0, 0) i promieniu r to: $x^2 + y^2 = r^2$. Przyjmijmy, że $r = 1$. Zauważ, że pole obszaru ograniczonego okręgiem o równaniu $x^2 + y^2 = 1$ jest równe liczbie π. Aby obliczyć to pole, najpierw trzeba policzyć pole obszaru ograniczonego krzywą $f(x) = \sqrt{1 - x^2}$ w przedziale [–1; 1] i znajdującego się nad osią OX (rys. 8.9). Następnie otrzymany wynik należy pomnożyć przez 2.



Rys. 8.9. Obszar ograniczony funkcją $f(x) = \sqrt{1 - x^2}$ w przedziale [–1; 1] znajdujący się nad osią OX

Ćwiczenie 7

Napisz program obliczający przybliżoną wartość liczby π. Wykorzystaj:

a. metodę prostokątów,

b. metodę trapezów.

Przybliżenie liczby π – metoda Monte Carlo

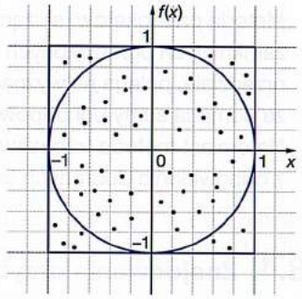
Czy wykorzystując liczby losowe, można obliczać przybliżenia różnych wartości lub symulować pewne zjawiska? Okazuje się, że tak. Metoda używająca w tym celu liczb losowych nosi nazwę **metody Monte Carlo**.

Metoda Monte Carlo

Dobra rada

Do wylosowania współrzędnych punktów możesz wykorzystać arkusz kalkulacyjny.

Przeprowadźmy następujące doświadczenie. Wylosujmy n punktów w obrębie kwadratu o środku w początku układu współrzędnych i długości boku $a = 2$ (a więc punktów o współrzędnych z zakresu od –1 do 1). Policzymy, ile z wylosowanych punktów znajdzie się w kole o środku w punkcie (0, 0) i promieniu $r = 1$, czyli w kole wpisanym w kwadrat. Liczbę punktów w kole oznaczmy przez n_0. Przykładowy wynik doświadczenia ilustruje rysunek 8.10.



Rys. 8.10. Przykładowy wynik losowania punktów w obrębie kwadratu o środku w punkcie (0, 0) i długości boku 2

Iloraz pola koła do pola kwadratu jest w przybliżeniu równy ilorazowi liczby punktów w kole do liczby wszystkich punktów:

$$\frac{\pi \cdot r^2}{a^2} \approx \frac{n_0}{n}$$

gdzie r – promień koła, a – długość boku kwadratu, n – liczba wszystkich punktów w kwadracie, n_0 – liczba punktów w kole.

W opisanym doświadczeniu $a = 2r$, czyli $a^2 = 4r^2$. Otrzymujemy:

$$\frac{\pi \cdot r^2}{4 \cdot r^2} = \frac{\pi}{4} \approx \frac{n_0}{n}, \text{ zatem } \pi \approx \frac{4 \cdot n_0}{n}$$

Daną, która wpływa na dokładność przybliżenia liczby π, jest liczba punktów losowanych w obrębie kwadratu. Im będzie ona większa, tym dokładniejsze przybliżenie otrzymamy. Oto specyfikacja problemu:

Warto wiedzieć

Metodę Monte Carlo można zastosować do obliczania pola obszarów zamkniętych.

Specyfikacja

Dane: n – liczba całkowita dodatnia określająca liczbę losowań punktów o współrzędnych (x, y), gdzie x i y są liczbami rzeczywistymi z przedziału [–1; 1].

Wynik: pi – przybliżona wartość liczby π.

Oto zapis algorytmu znajdującego przybliżenie liczby π metodą Monte Carlo (n0 odpowiada liczbie punktów w kole) oraz kod źródłowy funkcji main realizującej ten algorytm:

```
n0 ← 0
dla i ← 0, 1, …, n – 1 wykonuj
    x ← losowa liczba rzeczywista z przedziału [–1; 1]
    y ← losowa liczba rzeczywista z przedziału [–1; 1]
    jeśli x * x + y * y ≤ 1 to n0 ← n0 + 1
pi ← 4 * n0/n
```

```
1.  int main()
2.  {
3.      int n, n0=0;
4.      float x, y;
5.      cout<<"Liczba punktow: "; cin>>n;
6.      srand(time(NULL));
7.      for (int i=0;i<n;i++)
8.      {
9.          x=-1+2*(float(rand())/RAND_MAX);
10.         y=-1+2*(float(rand())/RAND_MAX);
11.         if (x*x+y*y<=1) n0++;
12.     }
13.     cout<<"Przyblizenie pi: "<<4*(float(n0)/n);
14.     return 0;
15. }
```

Fragment kodu źródłowego programu znajdującego przybliżoną wartość liczby π metodą Monte Carlo – funkcja main

Dobra rada

Pamiętaj, że korzystanie z funkcji srand i rand wymaga dołączenia biblioteki cstdlib, a korzystanie z funkcji time – biblioteki ctime.

Warto wiedzieć

Liczby losowane przez komputer nazywa się często pseudolosowymi. Są bowiem generowane według ustalonych algorytmów, czyli tak naprawdę nie są losowane.

Instrukcja w linii 6 inicjuje generator liczb losowych i uzależnia losowanie liczb od czasu wskazywanego przez zegar komputera (dzięki temu po każdym uruchomieniu programu otrzymamy różne wyniki).

Pętla w liniach 7–12 losuje n razy współrzędne punktu (linie 9–10) i bada, czy wylosowany punkt należy do koła o promieniu 1 oraz środku w początku układu współrzędnych (linia 11). Jeśli tak, powiększamy wartość zmiennej `n0`. Funkcja `rand` losuje liczbę całkowitą z zakresu od 0 do `RAND_MAX`. Predefiniowana **stała** `RAND_MAX` określa maksymalną wartość, jaką może zwrócić funkcja `rand`. Po konwersji wylosowanej liczby całkowitej na rzeczywistą (instrukcja `float(rand())`) i podzieleniu jej przez stałą `RAND_MAX` otrzymujemy liczbę rzeczywistą z przedziału [0; 1]. Po pomnożeniu przez 2 i odjęciu 1 otrzymujemy liczbę z przedziału [–1; 1]. W linii 13 wypisywana jest przybliżona wartość liczby π obliczona ze wzoru podanego na s. 143. Ponieważ dzielimy dwie wartości całkowite, należy dokonać konwersji na typ rzeczywisty.

Stała `RAND_MAX`

Warto wiedzieć

Standard języka C++ nie określa dokładnej wartości stałej `RAND_MAX`, gwarantuje jednak, że nie jest mniejsza niż $2^{15} - 1 = 32\,767$. Wartość stałej `RAND_MAX` zależy od kompilatora i bibliotek.

Ćwiczenie 8

Napisz program obliczający przybliżoną wartość liczby π metodą Monte Carlo.

Dobra rada

Dokładniejsze przybliżenie liczby π uzyskasz, gdy wielokrotnie wywołasz program i policzysz średnią arytmetyczną otrzymanych wyników.

Rysunek 8.11 przedstawia przykład wykonania programu obliczającego przybliżoną wartość liczby π metodą Monte Carlo dla $n = 1\,000\,000$. W tym przypadku otrzymaliśmy dokładność jedynie do trzech miejsc części ułamkowej. Dla tej samej wartości n możemy otrzymać różne przybliżenia liczby π.

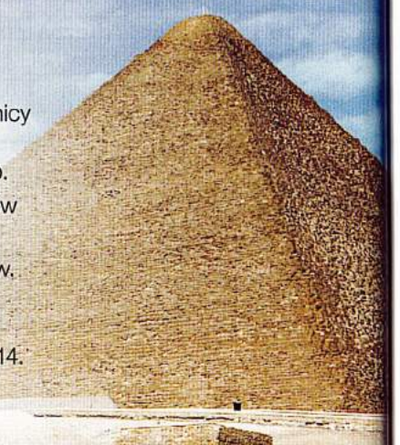
```
Liczba punktow: 1000000
Przyblizenie pi: 3.14162
```

Rys. 8.11. Przykład wykonania programu obliczającego przybliżoną wartość liczby π metodą Monte Carlo dla 1 000 000 wylosowanych punktów

A to ciekawe

Przybliżenia liczby π

Już w starożytności odkryto, że stosunek obwodu koła do jego średnicy jest pewną stałą. Obecnie znamy ją jako liczbę π i wiemy, że jest to liczba niewymierna. Od tysiącleci stosuje się różne jej przybliżenia, np. Archimedes używał do tego ułamka $\frac{22}{7}$. Dzięki rozwojowi komputerów udaje się wyznaczać kolejne cyfry rozwinięcia dziesiętnego liczby π. W 1949 r. wyliczono 2038 cyfr po przecinku, a w 2020 r. – 50 bilionów. Często dla uproszczenia obliczeń przyjmuje się $\pi \approx 3{,}14$. Może to przypadek, ale stosunek sumy długości dwóch dowolnych boków podstawy piramidy Cheopsa do wysokości piramidy wynosi około 3,14.



8.5. Symulacja ruchów Browna

Ruchy Browna to chaotyczny ruch cząstek substancji stałej zawieszonych w cieczy lub gazie. Ponieważ są to ruchy przypadkowe, do ich symulacji można wykorzystać liczby losowe i **metodę Monte Carlo**.

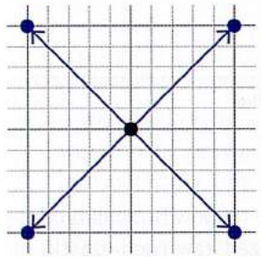
Ruchy Browna

Metoda Monte Carlo, s. 142

Przyjmijmy model, w którym cząstka przemieszcza się na płaszczyźnie w jednym z czterech kierunków. W jednostce czasu zmienia swoje położenie o jednostkową odległość wzdłuż każdej z osi (w lewo lub prawo wzdłuż osi *OX* oraz do góry lub do dołu wzdłuż osi *OY*). Możliwe przesunięcia cząstki pokazuje rysunek 8.12. Czarnym kolorem zaznaczona jest aktualna pozycja cząstki, a niebieskim – jej możliwe położenia po wykonaniu ruchu w jednostce czasu.

Warto wiedzieć

Nazwa „ruchy Browna" pochodzi od nazwiska szkockiego biologa Johna Browna. W 1827 r. obserwował on przez mikroskop przemieszczanie się cząstek pyłku kwiatowego w wodzie i zauważył, że są one w bezustannym, chaotycznym ruchu.



Rys. 8.12. Możliwe położenia cząstki po wykonaniu ruchu w jednostce czasu zgodnie z przedstawionym modelem

Specyfikację symulacji ruchów Browna możemy zapisać w następujący sposób:

Warto wiedzieć

Zjawisko ruchów Browna badali niezależnie Albert Einstein oraz polski fizyk Marian Smoluchowski. Na początku XX w. opisali oni matematyczny model zjawiska oraz podali przyczynę ruchów – odbijanie się cząstek substancji stałej od cząsteczek cieczy lub gazu.

Specyfikacja

Dane: n – liczba całkowita dodatnia, liczba ruchów w symulacji odpowiadająca liczbie jednostek czasu.

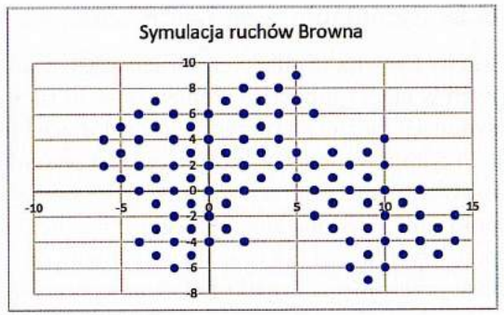
Wynik: x, y – położenie cząstki po n ruchach.

Algorytm w pseudokodzie możemy zapisać następująco. Przyjmujemy, że punktem początkowym cząstki jest punkt (0, 0).

```
x ← 0
y ← 0
dla i ← 1, 2, …, n wykonuj
    dx ← liczba losowa –1 lub 1
    dy ← liczba losowa –1 lub 1
    x ← x + dx
    y ← y + dy
```

Symulację zgodnie z przedstawionym modelem można łatwo przeprowadzić w arkuszu kalkulacyjnym i zobrazować wykresem punktowym. Rysunek 8.13 na s. 146 przedstawia przykład takiej symulacji.

1	x	y	Zmiana lewo/prawo	Zmiana góra/dół
2	0	0	1	-1
3	1	-1	1	1
4	2	0	-1	1
5	1	1	-1	1
6	0	2	-1	1
7	-1	3	-1	-1
8	-2	2	1	-1
9	-1	1	1	-1
10	0	0	-1	1
...	...	...	...	...
197	1	3	-1	1
198	0	4	1	-1
199	1	3	-1	-1
200	0	2	-1	-1
201	-1	1		



Rys. 8.13. Przykład symulacji ruchów Browna dla 200 ruchów cząstki (zaznaczone punkty oznaczają miejsca, w których znalazła się cząstka)

Dobra rada

Jeśli przy otwartym arkuszu kalkulacyjnym naciśniesz klawisz F9, komputer przeliczy ponownie wszystkie zapisane w nim formuły. W arkuszu symulującym ruchy Browna za pomocą tego klawisza ponownie wylosujesz liczby.

Ćwiczenie 9

Przygotuj w arkuszu kalkulacyjnym symulację ruchów Browna zgodnie z modelem omówionym w tym temacie. Przedstaw ruch cząstki na wykresie punktowym. Przeprowadź symulację wielokrotnie.

Zapamiętaj

Metoda Monte Carlo wykorzystuje liczby losowe do modelowania różnych zjawisk oraz obliczania przybliżeń pewnych wartości. Metodą tą można np. obliczyć przybliżoną wartość liczby π oraz symulować ruchy Browna.

A to ciekawe

Skąd pochodzi nazwa metody Monte Carlo?

Za jednego z twórców metody Monte Carlo uważa się polskiego matematyka Stanisława Ulama (1909–1984). Nadał on tej metodzie nazwę, która prawdopodobnie pochodzi od nazwy dzielnicy Monako – Monte Carlo. Niektórym kojarzy się ona z kasynami, gdzie wielu powierza swoje szczęście losowi. Stanisław Ulam w książce *Przygody matematyka* tak wspominał powstanie metody: „Zauważyłem, że znacznie praktyczniejszym sposobem oceniania prawdopodobieństwa ułożenia pasjansa jest wykładanie kart, czyli eksperymentowanie z tym procesem i po prostu zapisywanie procentu wygranych, niż próba obliczenia wszystkich możliwości kombinatorycznych, których liczba rośnie wykładniczo i jest tak wielka, że pominąwszy najprostsze przypadki, jej oszacowanie jest niemożliwe".

Podsumowanie

- Metoda bisekcji (połowienia) polega na zmniejszaniu zakresu przeszukiwanych danych o połowę do znalezienia rozwiązania lub do momentu, gdy długość przedziału poszukiwań jest nie większa niż dokładność obliczeń (wówczas wynikiem jest ostatnio wyznaczony środek przedziału).
- Metodę bisekcji można zastosować do znajdowania miejsca zerowego funkcji $f(x)$ w przedziale $[a; b]$. Aby można było to zrobić, funkcja musi być określona i ciągła w przedziale $[a; b]$ oraz $f(a) \cdot f(b) < 0$.
- Metodę bisekcji można zastosować także do poszukiwania pierwiastka kwadratowego z liczby rzeczywistej c większej od 1. Problem ten sprowadza się do znalezienia miejsca zerowego funkcji $f(x) = x^2 - c$ w przedziale $[1; c]$.
- Metoda Newtona–Raphsona polega na szukaniu przybliżonej długości boku kwadratu o polu x za pomocą prostokątów o tym samym polu.
- Metodą Newtona–Raphsona można znaleźć przybliżoną wartość pierwiastka kwadratowego z dodatniej liczby rzeczywistej x. Za wartości początkowe przedziału poszukiwań można przyjąć 1 i x. Przedział zawężamy, zamieniając jedną z wartości na średnią arytmetyczną końców przedziału z poprzedniego kroku, a drugą – na iloraz wartości x i policzonej średniej.
- Metoda prostokątów służy do obliczania pola obszarów zamkniętych ograniczonych wykresem funkcji i osią układu współrzędnych w określonym przedziale. Polega na znajdowaniu przybliżonej wartości tego pola jako sumy pól odpowiednio dobranych prostokątów.
- Metoda trapezów wykorzystywana jest do obliczania pola obszarów zamkniętych ograniczonych wykresem funkcji i osią układu współrzędnych w określonym przedziale. Polega na znajdowaniu przybliżonej wartości tego pola jako sumy pól odpowiednio dobranych trapezów.
- Metoda Monte Carlo wykorzystuje liczby losowe do modelowania różnych zjawisk oraz wyznaczania niektórych przybliżonych wartości.
- Metodą Monte Carlo można obliczyć przybliżoną wartość liczby π oraz symulować ruchy Browna.

Zadania

1. Wykorzystując program znajdujący miejsce zerowe funkcji metodą bisekcji omówiony w tym temacie, wyznacz kilka miejsc zerowych funkcji $f(x) = x \cdot \sin x$ w przedziale $[-7; 7]$. Znajdź w tym przedziale inne miejsce zerowe, którego metodą bisekcji nie da się wyznaczyć.
2. Napisz program, który wyznaczy metodą bisekcji pierwiastek kwadratowy z dodatniej liczby rzeczywistej $0 < c < 1$.
3. Omówione w temacie programy obliczające pola obszarów zamkniętych metodą prostokątów i metodą trapezów zmodyfikuj tak, aby obliczały prawidłowo pola także dla ujemnych wartości funkcji ograniczającej obszar.