Laboratorium 10A

Programowanie 3: zaawansowane (zadanie wymyślał: Maciej Bartoszuk)

9 grudnia 2016

Contents

Zarys zadania

Poszczególne etapy są niepowiązane ze sobą. Można je robić w dowolnej kolejności, chociaż powinny być one ułożone ode najłatwiejszego do najtrudniejszego.

Wszystkie metody powinny być statyczne.

Etapy

Etap 1 (1 punkt)

Część 1

Napisz funkcję `LukasiewiczTnormGeneration()`, która sama nie przyjmuje parametrów, za to zwraca następującą funkcję:

- funkcja przyjmuje 2 argumenty typu double,
- funkcja zwraca double,
- funkcja zwraca maksimum z następujących dwóch wartości: 0 i a+b-1.

Część 2

Napisz funkcję `ReduceManyArguments()`, która przyjmuje funkcję f o następujących cechach:

- przyjmuje 2 argumenty typu T,
- zwraca wartość typu T,
- jest łączna i symetryczna (tego nie da się w żaden sposób wymusić, po prostu my sami to wiemy),

a także **dowolną liczbę** argumentów typu T.

Funkcja `ReduceList()` działa w taki sposób, że:

- jeśli mamy 0 argumentów typu T, zwracamy wartość domyślną dla T
- dla 1 argumentu zwracamy ten argument
- dla więcej niż 1 argumentu oznaczmy argumenty przez $(x_1, x_2, \ldots, x_n)$. Zwracamy wartość $f(\ldots f(f(x_1, x_2), x_3), \ldots), x_n)$.

Etap 2 (1 punkt)

Część 1

Napisz funkcję `QuadraticFunction()`, która sama przyjmuje:

- funkcję f, która przyjmuje i zwraca double,
- 3 parametry typu double: a, b, c,

i zwraca następującą funkcję (przyjmującą jeden parametr x typu double): $af(x)^2 + bf(x) + c$.

`Main()` uzupełnij tak, aby wywołać `QuadraticFunction()` z funkcją $f(x) = \sin(x + \pi) + 1$ jako parametr (użyj wyrażenia lambda w `Main()`).

Część 2

Napisz funkcję `IsMonotonic()`, która przyjmuje parametry:

- double a,
- double b,
- double n,
- funkcję, która przyjmuje double i zwraca double.

Funkcja `IsMonotonic()` działa w taki sposób, że dla n równomiernie rozłożonych punktów na przedziale $[a, b]$ sprawdza, czy funkcja jest monotoniczna (cały czas rosnąca, cały czas malejąca lub cały czas stała). Zwraca wartość `true`, jeśli funkcja jest monotoniczna i `false` w przeciwnym wypadku.

`Main()` uzupełnij tak, aby wywołać `QuadraticFunction()` z funkcją $f(x) = \sin(x + \pi) + 1$ jako parametr (użyj wyrażenia lambda w `Main()`).

Etap 3 (1 punkt)

Część 1

Napisz funkcję `ManhattanDistance()`, która nie przyjmuje żadnego argumentu, za to zwraca funkcję, która przyjmuje dwie instancje klasy Point2D, a zwraca double. Funkcja ta działa tak, że sumuje wartości bezwzględne różnic na kolejnych współrzędnych punktów.

Część 2

Napisz funkcję `Medoid()`, która przyjmuje parametry:

- listę punktów Point2D,
- funkcję, takiego typu, jak wyprodukowana w części 1.

Funkcja `Medoid()` działa w ten sposób, że wybiera z listy punktów taki punkt, który minimalizuje sumę dystansów do reszty punktów. Do obliczania dystansu używa funkcji danej jako parametr.

Etap 4 (2 punkty)

Część 1

Napisz funkcję `PascalTriangleGeneration()`, która przyjmuje jako parametr trzy liczby typu całkowitego (a, b, c). Funkcja zwraca funkcję, która, co do idei, zwraca kolejny wiersz trójkąta Pascala na podstawie poprzedniego. Innymi słowy, dwa zewnętrzne elementy są przepisywane, a reszta jest tworzona jako suma odpowiednich elementów z poprzedniego wiersza.

Funkcja produkowana w `PascalTriangleGeneration()` powinna przyjmować jako parametr tablicę typu int, a także zwracać tablicę typu int (o rozmiarze o 1 większym niż tablica wejściowa). Pytanie pojawia się, jak wyprodukować pierwsze dwa wiersze trójkąta Pascala, ten o 1 elemencie i ten o 2 elementach. Do tego służą parametry a,b i c. Pierwszy wiersz ma się składać tylko z elementu a, a drugi z elementów b i c. Każdy kolejny da się policzyć na podstawie poprzedniego.

Część 2

Napisz funkcję `TriangleGeneration()`, która przyjmuje parametry:

- funkcję taką, jak wyprodukowana w części 1,
- liczbę całkowitą n.

Funkcja powinna wypisać na ekran pierwsze n wierszy trójkąta generowanego przez funkcję podaną jako argument.