Programowanie i metody numeryczne

Zadania – seria 3.

Język C++: programowanie obiektowe.

Zadanie 1. `circles` – Porównywanie położeń okręgów.

Niech będzie dany pewien kartezjański układ współrzędnych na płaszczyźnie. Napisz klasę `Circle` reprezentującą okrąg w tym układzie.

Zaimplementuj:

- prywatne pole `x`, przechowujące odciętą środka okręgu w danym układzie współrzędnych,
- prywatne pole `y`, przechowujące rzędną środka okręgu w danym układzie współrzędnych,
- prywatne pole `r`, przechowujące promień okręgu,
- publiczną metodę `Circumference`, zwracającą obwód okręgu,
- publiczną metodę `Scale`, zwiększającą promień okręgu k-krotnie, gdzie wartość k ma być przekazywana metodzie jako argument,
- publiczną metodę `Intersection`, zwracającą liczbę punktów wspólnych okręgu z innym okręgiem, reprezentowanym przez inną instancję klasy `Circle`, przekazaną metodzie `Intersection` jako argument.

Korzystając z tej klasy napisz program `circles` mający na celu porównywanie położeń okręgów na płaszczyźnie. Program powinien wczytywać ze standardowego wejścia sześć liczb zmiennoprzecinkowych opisujących dwa okręgi, oznaczających kolejno: odciętą środka pierwszego okręgu, rzędną środka pierwszego okręgu, promień pierwszego okręgu, odciętą środka drugiego okręgu, rzędną środka drugiego okręgu i promień drugiego okręgu, a następnie wypisywać na standardowe wyjście obwody tych okręgów oraz informację o liczbie ich punktów wspólnych.

Wskazówka. Nieskończoność może być w języku C++ reprezentowana za pomocą obiektu `std::numeric_limits<double>::infinity()` zdefiniowanego w pliku nagłówkowym `limits`. Sprawdzenia, czy zmienna typu `double` ma wartość równą nieskończoności, można dokonać dzięki funkcji `std::isinf(double)` z pliku nagłówkowego `cmath`.

Zadanie 2. `velocity` – Relatywistyczne składanie prędkości.

Niech $\beta = v/c$ będzie prędkością wyrażoną w jednostkach prędkości światła c. Zgodnie z relatywistycznym prawem składania prędkości w przypadku jednowymiarowym, jeżeli β_2 jest prędkością ciała drugiego względem ciała pierwszego, zaś β_1 – prędkością ciała pierwszego w pewnym układzie odniesienia, to prędkość ciała drugiego w tym układzie odniesienia wynosi

$$\beta = \frac{\beta_1 + \beta_2}{1 + \beta_1 \beta_2}.$$

Napisz klasę `Velocity`, reprezentującą prędkość w ruchu jednowymiarowym.

Zaimplementuj:

- konstruktor, który można wywołać bez argumentów lub z jednym argumentem; w pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością zero, w drugim zaś – wartością podaną jako argument,
- publiczną metodę `gamma`, zwracającą wartość czynnika

$$\gamma = \frac{1}{\sqrt{1-\beta^2}},$$

- operator dodawania `+`, działający zgodnie z relatywistycznym prawem składania prędkości,
- operator `+=`, działający zgodnie z relatywistycznym prawem składania prędkości.

Korzystając z tej klasy napisz program `velocity`, który wczytuje ze standardowego wejścia dwie prędkości wyrażone w jednostkach prędkości światła i wypisuje na standardowe wyjście ich relatywistyczną sumę, również wyrażoną w jednostkach prędkości światła, oraz odpowiadający jej czynnik γ.

Przykładowe wykonanie
Wywołanie:
Windows: `velocity.exe`
macOS / Linux: `./velocity`
Wejście:
```
0.5 0.7
```
Wyjście:
```
beta = 0.888889
gamma = 2.18282
```

Zadanie 3. `rational` – Liczby wymierne.

Napisz klasę `RationalNumber` reprezentującą liczbę wymierną. Klasa ma przechowywać liczby całkowite p i q, których iloraz jest równy reprezentowanej liczbie wymiernej, przy czym p i q powinny być względnie pierwsze oraz $q > 0$.

Zaimplementuj:

- konstruktor(y), który(-e) można wywołać bez argumentów lub z jednym albo dwoma argumentami całkowitymi; w pierwszym przypadku tworzony obiekt powinien być inicjalizowany wartością 0, w drugim – zadaną liczbą całkowitą, zaś w trzecim – ilorazem argumentów konstruktora; konstruktor wywołany z dwoma argumentami powinien działać poprawnie również wtedy, gdy wartości jego argumentów nie są względnie pierwsze lub gdy drugi argument jest ujemny,
- publiczne metody `Numerator` i `Denominator`, zwracające odpowiednio licznik p i mianownik q reprezentowanej liczby wymiernej,
- operator konwersji na typ `double`,
- operator konwersji na typ `string`, zapisujący liczbę w postaci `p/q`, np. `-5/7`,
- jednoargumentowy operator zmiany znaku `-`,
- operator porównania `<`,
- operatory preinkrementacji i postinkrementacji `++`,
- operatory dodawania `+` i mnożenia `*`,

- operatory `+=` i `*=`,
- operator `<<`, wypisujący do strumienia typu `ostream` liczbę reprezentowaną przez obiekt, oraz operator `>>`, wczytujący odpowiednią liczbę ze strumienia typu `istream`; w obu przypadkach liczbę zapisujemy w postaci `p/q`, np. `-5/7`; operator `>>` powinien działać poprawnie również wtedy, gdy podane wartości p i q nie są względnie pierwsze lub gdy q jest ujemne.

Korzystając z tej klasy napisz program `rational`, który wczytuje ze standardowego wejścia dwie liczby wymierne, a następnie wypisuje na standardowe wyjście w kolejnych liniach:

- podane liczby w postaci dziesiętnej,
- liczby przeciwne do podanych,
- podane liczby w kolejności niemalejącej,
- sumę i iloczyn podanych liczb.

Przykładowe wykonanie
Wywołanie:
Windows: `rational.exe`
macOS / Linux: `./rational`
Wejście:
```
2/4 1/-3
```
Wyjście:
```
0.5 -0.3333333
-1/2 1/3
-1/3 1/2
1/6 -1/6
```

Zadanie 4. `pimc` – Liniowy generator kongruentny i metody Monte Carlo.

Napisz klasę `LCG` implementującą generator liczb pseudolosowych określany jako *liniowy generator kongruentny*, stanowiący ciąg dodatnich liczb całkowitych $(x_n)_{n=0}^{\infty}$ zadany wzorem rekurencyjnym

$$x_{n+1} = (ax_n + b) \bmod m$$

dla pewnej wartości początkowej $x_0 \in \{0, 1, \ldots, m-1\} \subset \mathbb{Z}$, nazywanej *ziarnem*, gdzie m, $a \in \mathbb{N}$ oraz $b \in \mathbb{Z}_+ \cup \{0\}$ są parametrami spełniającymi a, $b < m$.

Zaimplementuj:

- publiczny alias `result_type` reprezentujący typ `size_t`,
- konstruktor przyjmujący jako argumenty wartości ziarna oraz parametrów generatora,
- operator wywołania `()`, zwracający nowo wygenerowaną liczbę pseudolosową,
- publiczną metodę `Min`, zwracającą najmniejszą możliwą do wygenerowania liczbę (czyli 1, gdy $m = 0$, lub 0, gdy $m \neq 0$),
- publiczną metodę `Max`, zwracającą największą możliwą do wygenerowania liczbę (czyli $m - 1$).

Korzystając z tej klasy napisz program `pimc`, którego zadaniem jest obliczenie przybliżonej wartości liczby π z wykorzystaniem prostej metody Monte Carlo. Program ma przyjmować jako argument wywołania dodatnią liczbę całkowitą N. Po uruchomieniu program powinien wylosować N punktów leżących w kwadracie $[0, 1]^2$ korzystając z instancji klasy `LNG` oraz określić liczbę K tych spośród wylosowanych punktów, które leżą

w ćwiartce koła $\left\{ (x, y) \in \mathbb{R}^2 \,\middle|\, x^2 + y^2 \leqslant 1,\ x \geqslant 0,\ y \geqslant 0 \right\}$. Następnie program ma obliczyć wartość liczby π na podstawie przybliżonej zależności

$$\pi \approx \frac{4K}{N}$$

i wypisać ją na standardowe wyjście.

Zadanie 5. `points` – Punkty na sferze jednostkowej.

Napisz strukturę `Point` reprezentującą punkt w trójwymiarowej przestrzeni euklidesowej. Struktura ta powinna przechowywać współrzędne kartezjańskie punktu oraz dodatkowo liczbę całkowitą reprezentującą identyfikator punktu.

Napisz ponadto szablon funkcji

```
template<typename RNG>
Point NewPoint(int id)
```

która wykorzystując generator liczb pseudolosowych `RNG` zwraca instancję struktury `Point` reprezentującą losowy punkt na sferze jednostkowej o identyfikatorze `id`.

Napisz również funkcję

```
Point MeanPoint(const std::vector<Point> &v)
```

zwracającą instancję struktury `Point` reprezentującą punkt o współrzędnych będących średnimi odpowiednich współrzędnych punktów z wektora `v` i identyfikatorze -1.

Korzystając z napisanych przez Ciebie klasy i funkcji napisz program `points`. Program ten powinien przyjmować jako argument wywołania dodatnią liczbę całkowtią N. Po uruchomieniu program powinien losować N punktów leżących na sferze jednostkowej, przypisując im kolejne identyfikatory całkowite dodatnie, oraz wypisywać na standardowe wyjście identyfikatory i współrzędne tych punktów. Następnie program powinien znaleźć punkt, którego współrzędne są średnimi odpowiednich współrzędnych wylosowanych punktów, i wypisać jego współrzędne na standardowe wyjście.

Opracowanie: Bartłomiej Zglinicki.