Programowanie I

Zajęcia nr 9

Rafał Masełek

5. maja 2022r.

Zadanie 1. – Okrąg (autorstwa B. Zglinickiego)

Niech będzie dany pewien kartezjański układ współrzędnych na płaszczyźnie. Napisz klasę Circle reprezentującą okrąg. Klasa powinna zawierać

- pole x, przechowujące odciętą środka okręgu w tym układzie współrzędnych,
- pole y, przechowujące rzędną środka okręgu w tym układzie współrzędnych,
- pole r, przechowujące promień okręgu,
- metodę circumference, zwracającą obwód okręgu,
- metodę intersection, zwracającą liczbę punktów wspólnych okręgu z innym okręgiem, reprezento- wanym przez inną instancję klasy Circle, przekazaną metodzie intersection jako argument.

Korzystając z tej klasy, napisz program circles, przyjmujący jako argumenty wywołania sześć liczb zmien- noprzecinkowych opisujących dwa okręgi, oznaczających kolejno: odciętą środka pierwszego okręgu, rzędną środka pierwszego okręgu, promień pierwszego okręgu, odciętą środka drugiego okręgu, rzędną środka dru- giego okręgu i promień drugiego okręgu, i wypisujący na ekranie informację o liczbie punktów wspólnych tych okręgów.

Wskazówka. Nieskończoność może być w języku Python reprezentowana np. jako float("inf") lub za pomocą stałej inf z modułu math. Sprawdzenia, czy zmienna ma wartość równą nieskończoności, można dokonać, korzystając z funkcji isinf z modułu math.

Zadanie 2 – stos (autorstwa B. Zglinickiego)

Napisz klasę Stack, implementującą strukturę danych nazywaną stosem. W razie potrzeby możesz też napisać dodatkową klasę Node, reprezentującą element stosu. Korzystając z tej klasy, napisz program stack, który przyjmuje jako argumenty wywołania dowolną ilość liczb zmiennoprzecinkowych i wypisuje na ekranie te liczby w kolejności odwrotnej do tej, w której zostały podane.

Zadanie 3 – czteropęd

Czterowektor pędu dla cząstki o masie m, całkowitej energii E oraz pędzie $\vec{p}$ zdefiniowany jest następująco ($c = 1$):

$$p^\mu = (E, \vec{p}) \tag{1}$$

Odpowiednikiem mnożenia skalarnego jest *kontrakcja/zwężenie* zdefiniowana z użyciem metryki Minkowskiego $\eta_{\mu\nu} = \mathrm{diag}(+1, -1, -1, -1)$ w następujący sposób:

$$p^\mu k_\mu = \eta_{\mu\nu} p^\mu k_\nu = E_p E_k - \vec{p} \cdot \vec{k}. \tag{2}$$

W szczególności:

$$p_\mu p^\mu = E^2 - |\vec{p}|^2 = m^2 \tag{3}$$

Napisz klasę *FourMomentum*, która przechowuje informację o 4 składowych czteropędu. Klasa powinna mieć konstruktor umożliwiający podanie energii i 3 składowych pędu i domyślnie ustawiający zera w razie nie podania którejś ze składowych. Zaimplementuj następujące funkcjonalności:

- estetyczne drukowanie składowych w postaci (E, p_x, p_y, p_z),
- dodawanie i odejmowanie 4-wektorów za pomocą operatorów "+" i "-",
- zwężanie 4-wektorów za pomocą operatora "*",
- liczenie masy niezmienniczej cząstki zdefiniowanej jako $m^2 = p_\mu p^\mu$.

W głównej części programu stwórz kilka czterowektorów i przetestuj działanie zaimplementowanych metod. Użyj do tego funkcji print().

Zadanie dla chętnych – Symulacja rozprzestrzeniania się epidemii (autorstwa dra T. Kazimierczuka)

Napisz program epidemic implementujący prosty model rozprzestrzeniania się epidemii. Program ma opisywać niewielką społeczność, której członkowie rozmieszczeni są na planszy o zadanych wymiarach: szerokości w i wysokości h. Na planszy wprowadzamy kartezjański układ współrzędnych (x, y) w taki sposób, aby jego początek pokrywał się z lewym dolnym rogiem planszy, wówczas $x \in [0, w]$ i $y \in [0, h]$.
Członkowie społeczności mogą być zdrowi, chorować lub być bezobjawowymi nosicielami choroby. W kolejnych krokach symulacji każdy z nich zmienia swoje położenie, przesuwając się w losowym kierunku o losowy dystans, którego maksymalna wartość powinna być w przypadku osób chorych mniejsza niż w przypadku osób zdrowych i nosicieli. Gdy któraś z osób w wyniku zmiany położenia znajdzie się poza planszą, powinna pojawić się w odpowiednim położeniu z drugiej strony planszy. Jeśli osoba zdrowa znajdzie się odpowiednio blisko osoby chorej lub nosiciela, może się od niej zarazić z pewnym prawdopodobieństwem.
Program powinien definiować dwie klasy:

- klasę Person, opisującą pojedynczą osobę, zawierającą:
  - pola x i y, przechowujące liczby zmiennoprzecinkowe reprezentujące współrzędne osoby na planszy,
  - pole status, przechowujące informację o stanie zdrowia osoby (zdrowy, chory, nosiciel), np. w for- macie łańcucha tekstowego lub odpowiednio interpretowanej liczby całkowitej,
  - pole statyczne maxDistance, przechowujące maksymalny dystans, jaki może w jednym kroku symulacji przebyć osoba zdrowa lub nosiciel; na początek przypisz mu wartość 1.,
  - pole statyczne maxIllDistance, przechowujące maksymalny dystans, jaki może w jednym kroku symulacji przebyć osoba chora; na początek przypisz mu wartość 0.1,
  - metodę move, zmieniającą położenie osoby na planszy o losowy dystans; dystans ten nie powinien być większy od maxDistance lub maxIllDistance, w zależności od wartości pola status,
  - metodę info, zwracającą łańcuch tekstowy prezentujący informacje o osobie – jej stan zdrowia i położenie na planszy – w przejrzystym formacie,
  - metodę pozwalającą wypisywać informacje o osobie – jej stan zdrowia i położenie na planszy – w przejrzystym formacie za pomocą funkcji print,
- klasę Population, reprezentującą całą modelowaną populację, zawierającą:
  - listę people, zawierającą obiekty klasy Person reprezentujące członków populacji,
  - pola h i w, opisujące, odpowiednio, wysokość i szerokość planszy, na której rozmieszczona jest populacja; na początek przyjmij, że plansza jest kwadratem o boku 100.,
  - pole statyczne infectionProbability, określające prawdopodobieństwo tego, że na samym po- czątku symulacji konkretna osoba będzie zakażona (tzn. będzie osobą chorą lub nosicielem); na początek przyjmij, że prawdopodobieństwo to wynosi 20

- pole statyczne infectionDistance, określające dystans, na jaki osoba zdrowa musi się zbliżyć do osoby chorej lub nosiciela, by się zarazić; na początek przyjmij, że dystans ten wynosi 1.,
- konstruktor przyjmujący jako argumenty wysokość i szerokość planszy oraz liczebność populacji; konstruktor ten powinien umieszczać na liście people obiekty klasy Person w ilości odpowiada- jącej liczebności populacji, przypisując każdemu z nich losowe położenie na planszy (tzn. losując wartości pól x i y każdego z obiektów) oraz losowy stan zdrowia (tzn. losując wartość pola status każdego z obiektów); każdy pacjent może otrzymać status zakażonego (a więc nosiciela lub chorego) z prawdopodobieństwem określonym przez wartość pola statycznego infectionProbability, a to, czy osoba zakażona jest nosicielem, czy chorym, powinno być losowane z prawdopodobieństwem 50%.
- metodę move, zmieniającą położenia wszystkich członków populacji, wykorzystując do tego metodę move klasy People (czyli wywołując metodę move dla każdego z obiektów na liście people); metoda ta musi dbać o to, by osoby, które wyjdą poza planszę, wracały na nią z drugiej strony; po dokonaniu zmiany położeń wszystkich osób metoda ta powinna sprawdzać dystans każdej pary osób, i jeśli jest on mniejszy od infectionDistance, a jedna z osób w parze jest nosicielem lub choruje, druga osoba powinna zachorować lub zostać nosicielem z prawdopodobieństwem 50
- metodę paint, wizualizującą rozmieszczenie członków populacji na planszy z wykorzystaniem pakietu Matplotlib; osoby zdrowe powinny być zaznaczone na rysunku zielonymi markerami, osoby chore – czerwonymi, zaś nosiciele - żółtymi,
- metodę info, zwracającą łańcuch tekstowy prezentujący informacje o wszystkich członkach po- pulacji – ich stan zdrowia i położenie na planszy – w przejrzystym formacie,
- metodę pozwalającą wypisywać łańcuch tekstowy zwracany przez metodę info za pomocą funkcji print,
- iterator pozwalający na iterowanie po członkach populacji. Celem programu jest wyświetlenie animacji, której kolejne klatki rysowane będą za pomocą funkcji paint obiektu reprezentującego populację.