

Lista 3

Minimalny próg zaliczenia listy: 12 punktów.

Rozwiązania należy oddać na ostatnich zajęciach przed 27 kwietnia. Praca nad rozwiązaniami powinna być udokumentowana w repozytorium.

1. (6 pkt.) Napisz funkcję:

bool match(const string& pattern, const string& s),

która sprawdza, czy łańcuch **s** pasuje do wzorca **pattern**. Znak '?' we wzorcu jest zgodny z dowolnym znakiem. Znak '*' oznacza zgodność z dowolnym ciągiem znaków w łańcuchu. Pozostałe znaki są zgodne same ze sobą [1].

2. (3 pkt.) Zapoznaj się z implementacją klasy **Point2d** oraz z programem demonstrującym jej możliwości (<http://tinyurl.com/y4735nfo>). Rozbuduj klasę o metody pozwalające na wykonanie przekształceń:

- jednokładności względem początku układu współrzędnych o skali k ,
- obrót względem początku układu współrzędnych o kąt ϕ .

Zauważ, że implementacja tych przekształceń będzie łatwiejsza po przejściu do układu współrzędnych biegunowych. Zmień sposób przechowywania danych wewnątrz klasy: ze współrzędnych kartezjańskich na współrzędne biegunowe. Zaktualizuj definicje metod klasy **Point2d**. Zauważ, że nie jest konieczna zmiana w interfejsie publicznym klasy.

3. (3 pkt.) Zaprojektuj i zaimplementuj klasę **TicTacToe**, reprezentującą rozgrywkę w Kółko i krzyżyk. Implementacja powinna wykorzystywać typ wyliczeniowy do określenia stanu pól na planszy, aktualnego gracza oraz zwycięzcy. Wykorzystując tę klasę napisz program, który umożliwi przeprowadzenie rozgrywki w Kółko i krzyżyk: gracz przeciwko graczowi.

4. (13 pkt.) Zaimplementuj klasę **VectorInt**, umożliwiającą przechowywanie ciągu liczb całkowitych. Interfejs klasy powinien udostępniać:

- bezargumentowy konstruktor, który alokuje pamięć na 16 liczb całkowitych,
- konstruktor przyjmujący jeden argument – liczbę elementów, na które powinna wystarczyć pierwotna alokacja pamięci,
- konstruktor kopiujący,
- operator przypisania,
- destruktor,
- metodę **at** przyjmującą indeks i zwracającą wartość na tej pozycji,
- metodę **insert** przyjmującą indeks i wartość, którą wstawia na tej pozycji,
- metodę **pushBack** dodającą wskazany element na końcu ciągu (w razie potrzeby powinno nastąpić powiększenie pamięci),
- metodę **popBack** usuwającą ostatni element ciągu,

- metodę `shrinkToFit` dostosowującą zaalokowaną pamięć do jej wypełnienia,
- metodę `clear`, która pozwala na usunięcie wszystkich elementów wektora,
- metodę `size` zwracającą liczbę elementów przechowywanych w wektorze,
- metodę `capacity` zwracającą liczbę elementów, które można przechowywać w wektorze bez realokacji pamięci,
- przeciążony operator `<<` pozwalający na wypisywanie elementów wektora z wykorzystaniem obiektów klasy `ostream`.

Klasa nie może powodować wycieków pamięci. [2]

Zadanie dodatkowe:

(5 pkt.) Rozbuduj klasę `TicTacToe`, aby możliwa była gra: gracz przeciwko komputerowi. Komputer do wyboru ruchu powinien wykorzystywać algorytm min-max.

Karol Tarnowski
Wrocław, 2020

[1] https://cs.pwr.edu.pl/kobylanski/dydaktyka/resources/wip_lab4.pdf

[2] A. Allain, *Przewodnik dla początkujących C++*, rozdział 25.