POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

PJC Zadania 12

Rozwiązania należy przesłać w postaci odpowiednio podzielonych plików o rozszerzeniach .hpp i .cpp

Zadanie 1

Napisz szablon klasy `matrix` przechowującej dowolny typ liczbowy jako swoje elementy. Szablon ten winien udostępniać:

- Konstruktor umożliwiający utworzenie obiektu macierzy z dwuwymiarowego wektora tych samych elementów.
- Gettery do wymiarów macierzy.
- Metodę `transpose()` zwracającą nowy obiekt będący transpozycją macierzy, na której wywoływana jest metoda.
- Operatory indeksowania (`operator[]`) tak, aby można było za pomocą składni `m[x][y]` dostać się do elementu pod rzędem `x` i kolumną `y` macierzy `m`.
- Operator mnożenia, który będzie mógł pomnożyć dwie macierze dowolnego typu (np. jedną przechowującą `int`y, a drugą przechowującą `float`y). W celu uzyskania /wydedukowania typu elementu przechowywanego przez macierz wynikową, skorzystaj z *type trait*a o nazwie std::common_type. W przypadku braku możliwości przemnożenia takich macierzy (na przykład przez niekompatybilne wymiary) podnieś własnoręcznie stworzony wyjątek `matrix_exception` z dokładną informacją o tym, jaka operacja się nie powiodła (podając też niezgodne wymiary macierzy), który będzie można uzyskać za pomocą wywołania metody `what()`, odziedziczonej z odpowiedniej nadklasy.

Filip Kwiatkowski, fk@pjwstk.edu.pl

Za to zadanie można uzyskać maksymalnie 50% punktów, jeżeli wewnętrzna reprezentacja macierzy będzie realizowana inaczej niż przez jednowymiarową tablicę / jednowymiarowy wektor.

Zadanie 2

Stwórz przestrzeń nazw `pjc`, a w niej zagnieżdżoną przestrzeń `string_operators`. W tej przestrzeni dodaj:

- `operator*`, który przemnoży `std::string`a przez `int`a, czego wynikiem będzie nowy `std::string` powtórzony odpowiednią liczbę razy. Zadbaj o to, aby można było wywołać ten operator zarówno podając `int`a z prawej i stringa z lewej strony jak i odwrotnie.
- `operator-`, który od jednego stringa odejmie drugiego w taki sposób, że wszystkie wystąpienia prawego stringa w lewym zostaną usunięcie.
- Operatory `*=` oraz `-=` odpowiadające w dziale operatorom podanym powyżej, lecz modyfikujące stringi, na których są wywoływane zamiast tworzyć nowe obiekty.
- `operator/`, który dostając z prawej strony liczbę podzieli string na równe części i zwróci go w postaci wektora stringów. Jeżeli długość stringa nie jest podzielna przez te liczbę, to ostatni element wynikowego wektora może być krótszy.

Uwaga: Tak stworzony operatory (zamknięte w przestrzeni nazw) są niewidoczne do użytku "na zewnątrz", o ile nie skorzystamy z dyrektywy `using namespace` tam, gdzie chcemy ich używać.

Zadanie 3

Zaimplementuj odpowiedni zestaw szablonów operatorów, który umożliwi dodawanie do dowolnego wektora (`std::vector` z każdym typem) dowolnej liczby elementów typu, który przechowuje, za pomocą następującej składni:

```
auto vec = std::vector<int>{1, 2};
vec += 3, 4, 5;
vec.push_back(6);
```

Listing 1: wymagane wspierane operacje dodawania elementów do wektora

Po wykonaniu powyższego kodu, wektor `vec` będzie przechowywał elementy `[1, 2, 3, 4, 5, 6]`.