Zadanie:

1. **Funkcje przeciążone:** Utwórz klasę `baza`, która będzie zawierać dwie tablice dynamiczne (wykorzystaj * i `new`) dla różnych typów liczbowych (np. `float` i `int`). Rozmiary i wartości zawarte w tablicach należy wprowadzić do obiektu poprzez konstruktor (można kopiować wprowadzane tablice do obiektu z użyciem pętli `for`). Następnie utwórz metodę `wypisz()`, która wyświetli na ekranie zawartość tych tablic.
2. **Wykorzystanie szablonów zamiast przeciążania:** Utwórz szablon klasy `b_dowolna`, gdzie parametrem będzie jeden typ `t1`. Obiekt powinien zawierać chronione: rozmiar tablicy i tablicę dynamiczną typu `t1`. Tak jak w klasie `baza`, wprowadź wartości dotyczące zawartości tablicy i jej rozmiaru, a także stwórz metodę `wypisz()`, która wyświetli zawartość tablicy na ekranie.
3. **Szablony i dziedziczenie klasy:** Utwórz kolejny szablon klasy `b_dowolna2`, który będzie zawierał dwa parametry `t1` i `t2` , klasa powinna dziedziczyć w trybie chronionym po `b_dowolna` (przypisz jej jeden z parametrów szablonu). Do zmiennych prywatnych należą: rozmiar tablicy i tablica dynamiczna typu `t2`, zainicjuj wartości odpowiednio za pomocą konstruktora i utwórz publiczną metodę `wypisz_wszystko()`, która wyświetli zawartość tablicy klasy bazowej i pochodnej (wykorzystaj `wypisz()`).
4. **Szablon metody:** Utwórz szablon klasy `b_dowolna_mod` dziedziczącej publicznie po `b_dowolna`, jako parametr przyjmij `t1` (wykorzystaj go analogicznie do poprzednich przykładów przy definiowaniu dziedziczenia dla `b_dowolna`). Wewnątrz klasy stwórz konstruktor, który zainicjuje wartości klasy bazowej, oraz szablon publicznej metody `pobierz(…)`, która przyjmie jako parametry zmienną typu `t2` (uprzednio zdefiniowanego). Metoda ta powinna porównywać `t2` i `t1` , ich zgodność umożliwia wprowadzenie wartości podanej, jako parametr funkcji, zmiennej, która powinna trafić na pierwsza pozycję (`[0]`) dziedziczonej tablicy (zamiana wartości).
5. **Definiowanie kontenera testowego:** Stwórz klasę `kalkulator`, która za pomocą zdefiniowanego konstruktora wczyta liczbę i za pomocą jednoparametrowej (liczba n) publicznej metody `mnożenie(…)` wykona prymitywne mnożenie tej liczby i przechowa jej wartość wewnątrz obiektu (za pomocą pętli `for`, dodającej n razy wczytaną liczbę).
6. **Szablon funkcji:** Stwórz szablon funkcji `suma_wypisz(…)` z jednym parametrem `wsk`, gdzie funkcja jako swoje dwa parametry przyjmie wskaźniki na zmienną typu `wsk`. Jeśli typ będzie zgodny z `kalkulator` (wykorzystaj `typeid()`), wyświetl sumę wymnożonych liczb z dwóch różnych obiektów typu `kalkulator`.
7. **Testowanie:** Dla wybranych danych utwórz odpowiednie obiekty i wywołaj zdefiniowane wewnątrz nich funkcje, porównaj rezultaty jakie można uzyskać za pomocą funkcji przeciążonych klasy `baza` i różnych typów dla obiektów `b_dowolna`. Na koniec dla dwóch zdefiniowanych obiektów typu `kalkulator` wykonaj `mnożenie(…)` przez wybraną liczbę i przetestuj działanie funkcji `suma_wypisz()` dla wspomnianych kontenerów.