## Ćwiczenie 0

Napisz funkcję

```
int power(int x, unsigned n)
```

która podnosi liczbę x do potęgi naturalnej n i zwraca wynik. Użyj przeciążania i napisz funkcje, które zamiast int używają: long int, unsigned, float i double. Przetestuj na poniższym mainie.

```
#include <iostream>
#include <typeinfo>

using namespace std;

int main()
{
        cout << "[int] -3^2=" << power(-3, 2) << "\ttype is: "
         << typeid(power(-3, 2)).name() << endl;
        cout << "[int] 5^0=" << power(5, 0) << "\ttype is: "
        << typeid(power(5, 0)).name() << endl;
        cout << "[unsigned] 4^2=" << power(4U, 2) << "\ttype is: "
         << typeid(power(4U, 2)).name() << endl;
        cout << "[unsigned] 5^0=" << power(5U, 0) << "\ttype is: "
         << typeid(power(5U, 0)).name() << endl;
        cout << "[long int] -3^2=" << power(-3L, 2) << "\ttype is: "
        << typeid(power(-3L, 2)).name() << endl;
        cout << "[long int] 5^0=" << power(5L, 0) << "\ttype is: "
         << typeid(power(5L, 0)).name() << endl;
        cout << "[float] 4^2=" << power(4.0f, 2) << "\ttype is: "
         << typeid(power(4.0f, 2)).name() << endl;
        cout << "[float] 5^0=" << power(5.0f, 0) << "\ttype is: "
         << typeid(power(5.0f, 0)).name() << endl;
        cout << "[double] 4^2=" << power(4.0, 2) << "\ttype is: "
         << typeid(power(4.0, 2)).name() << endl;
        cout << "[double] 5^0=" << power(5.0, 0) << "\ttype is: "
         << typeid(power(5.0, 0)).name() << endl;
        return 0;
}
```

## Ćwiczenie 1

Używając szablonów funkcji zrealizuj to samo zadanie co w ćwiczeniu 1. Tym razem niech n będzie typu int i w przypadku gdy $n < 0$ szablon powinien zwrócić nan. Napisz specjalizacje szablonu dla typów float i double, które obsłużą podnoszenie do ujemnej potęgi. Przetestuj na pozniższym mainie:

```
#include <iostream>
#include <typeinfo>
#include <math.h>
using namespace std;

int main()
{
```

```
        cout << "[int] -3^2=" << power(-3, 2) << "\ttype is: "
         << typeid(power(-3, 2)).name() << endl;
        cout << "[int] 5^0=" << power(5, 0) << "\ttype is: "
         << typeid(power(5, 0)).name() << endl;
        cout << "[unsigned] 4^2=" << power(4U, 2) << "\ttype is: "
         << typeid(power(4U, 2)).name() << endl;
        cout << "[unsigned] 5^0=" << power(5U, 0) << "\ttype is: "
         << typeid(power(5U, 0)).name() << endl;
        cout << "[long int] -3^2=" << power(-3L, 2) << "\ttype is: "
         << typeid(power(-3L, 2)).name() << endl;
        cout << "[long int] 5^0=" << power(5L, 0) << "\ttype is: "
         << typeid(power(5L, 0)).name() << endl;
        cout << "[float] 4^2=" << power(4.0f, 2) << "\ttype is: "
        << typeid(power(4.0f, 2)).name() << endl;
        cout << "[float] 2^-2=" << power(2.0f, -2) << "\ttype is: "
         << typeid(power(2.0f, -2)).name() << endl;
        cout << "[float] 5^0=" << power(5.0f, 0) << "\ttype is: "
         << typeid(power(5.0f, 0)).name() << endl;
        cout << "[double] 4^2=" << power(4.0, 2) << "\ttype is: "
         << typeid(power(4.0, 2)).name() << endl;
        cout << "[double] 5^0=" << power(5.0, 0) << "\ttype is: "
         << typeid(power(5.0, 0)).name() << endl;
        cout << "[double] 2^-2=" << power(2.0, -2) << "\ttype is: "
         << typeid(power(2.0, -2)).name() << endl;

        return 0;
}
```

## Ćwiczenie 2

Napisz szablon funkcji o nazwie "concatenate", który przyjmie dwa wektory A i B przez referencje, utworzy na stosie nowy wektor tego samego typu, wpisze do niego wpierw elementy z A a później z B. Szablon powienien zwracać wskaźnik na utworzony wektor. Przetestuj działanie na poniższym mainie.

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
        const vector<char> Achar = {'a', 'b', 'c'};
        const vector<char> Bchar = {'z', 'z', 'z', 'z'};
        vector<char>* Vchar = concatenate(Achar, Bchar);
        cout << "Vchar = [ ";
        for(auto em : *Vchar)
        {
                cout << em << " ";
        }
        cout << "]" << endl;
```

```
        delete Vchar; // <---- zwolnienie pamieci!

        const vector<int> Aint = {1, 2, 3};
        const vector<int> Bint = {0, 1, 5, 6, 7};
        vector<int>* Vint = concatenate(Aint, Bint);
        cout << "Vint = [ ";
        for(auto em : *Vint)
        {
                cout << em << " ";
        }
        cout << "]" << endl;
        delete Vint; // <---- zwolnienie pamieci!

        return 0;
}
```