Czy ta liczba jest pierwsza

- Zrozumienie, czym są liczby pierwsze i ich znaczenie.
- Nauczenie się, jak rozpoznać liczbę pierwszą.
- Praktyka w pisaniu algorytmu sprawdzającego, czy liczba jest pierwsza.

- Wyjaśnienie pojęcia liczby pierwszej: **liczba pierwsza** to taka liczba, która ma dokładnie dwa dzielniki – 1 oraz samą siebie.
- Przykłady liczb pierwszych (2, 3, 5, 7, 11) i złożonych (4, 6, 8, 9).
- Zastosowanie w kryptografii

Przedstawienie sposobu na sprawdzenie, czy liczba jest pierwsza:

1. **Sprawdź, czy liczba jest większa od 1.**
2. **Sprawdź dzielniki do pierwiastka kwadratowego danej liczby**

Algorytm:

Lista kroków dla algorytmu sprawdzającego, czy liczba nnn jest liczbą pierwszą:

1. **Sprawdź, czy $n \leq 1$:**
   - Jeśli tak, liczba n nie jest liczbą pierwszą. Zakończ działanie algorytmu.
2. **Sprawdź, czy $n = 2$:**
   - Jeśli tak, liczba n jest liczbą pierwszą, ponieważ 2 jest jedyną parzystą liczbą pierwszą.
3. **Sprawdź, czy n jest parzyste:**
   - Jeśli n jest podzielne przez 2 (czyli $n \mod 2 = 0$), to nie jest liczbą pierwszą, ponieważ liczby pierwsze są podzielne tylko przez 1 i samą siebie.
4. **Sprawdź podzielność przez liczby od 3 do $\sqrt{n}$:**
   - Iteruj po liczbach od 3 do $\sqrt{n}$, sprawdzając, czy n jest podzielne przez którąkolwiek z nich.
   - Jeśli znajdziesz liczbę, przez którą n jest podzielne, oznacza to, że n nie jest liczbą pierwszą.

Wnioski:

- Jeśli żadna liczba w powyższych krokach nie dzieli nnn bez reszty, to liczba nnn jest liczbą pierwszą.

```
#include <bits/stdc++.h>
using namespace std;

bool isPrime(int n) {
    if (n <= 1)
        return false;
    if (n == 2)
        return true;
    if (n % 2 == 0)
        return false;

    for (int i = 3; i <= sqrt(n); i += 2) {
        if (n % i == 0)
            return false;
    }
    return true;
}

int main() {
    int n;
    cout << "Podaj liczbę: ";  //opcjonalnie
    cin >> n;

    if (isPrime(n))
        cout << n << " jest liczbą pierwszą.\n";
    else
        cout << n << " nie jest liczbą pierwszą.\n";

    return 0;
}
```

Jak to działa:

1. **Sprawdzamy przypadki brzegowe:**
   - Jeśli $n \leq 1$, zwracamy `false`, ponieważ liczby mniejsze lub równe 1 nie są pierwsze.
   - Jeśli $n = 2$, zwracamy `true`, ponieważ 2 jest liczbą pierwszą.
   - Jeśli n jest parzyste i większe niż 2, zwracamy `false`, ponieważ jest podzielne przez 2.
2. **Pętla od 3 do $\sqrt{n}$:**
   - Iterujemy po nieparzystych liczbach od 3 do $\sqrt{n}$. Jeśli znajdziemy dzielnik, zwracamy `false`.
   - Jeśli pętla zakończy się bez znalezienia dzielnika, zwracamy `true`, co oznacza, że n jest liczbą pierwszą.

- `using namespace std;` pozwala pominąć prefiks `std::` przy wszystkich standardowych funkcjach.
- `<bits/stdc++.h>` to rozbudowana biblioteka, która automatycznie dołącza wiele standardowych bibliotek C++, co przydaje się przy tworzeniu prototypów czy testów, chociaż nie jest zalecane do używania w kodzie produkcyjnym ze względu na wydajność.

Daj znać, jeśli chcesz dodać kolejne funkcje lub modyfikacje!

Zadania

1. Podstawowe zadania praktyczne:

- **Zadanie 1:** Sprawdź, czy liczby 29, 47 i 51 są liczbami pierwszymi, stosując krok po kroku algorytm oparty na dzielnikach do pierwiastka kwadratowego.
- **Zadanie 2:** Napisz pseudokod algorytmu sprawdzającego, czy liczba jest pierwsza, bez używania komputera.
- **Zadanie 3:** Znajdź wszystkie liczby pierwsze w zakresie od 1 do 50, stosując algorytm ręcznie lub w kodzie.

2. Zadania na modyfikacje algorytmu:

- **Zadanie 4:** Zmień algorytm tak, aby wypisywał dzielnik, na którym algorytm przerywa działanie (np. jeśli sprawdzana liczba to 51, program powinien wypisać, że 51 jest podzielne przez 3).
- **Zadanie 5:** Zmodyfikuj algorytm tak, aby zwracał ilość sprawdzeń (ilość wykonanych iteracji pętli) do ustalenia, czy liczba jest pierwsza. Przetestuj algorytm na liczbach 29, 47 i 51 i zobacz, jak różni się liczba iteracji.

3. Zadania na rozbudowę kodu:

- **Zadanie 6:** Zapisz algorytm w taki sposób, aby dla danej liczby n znajdował wszystkie liczby pierwsze mniejsze niż n.
- **Zadanie 7:** Napisz funkcję w C++, która przyjmuje zakres liczb m i n, gdzie m<n i zwraca wszystkie liczby pierwsze w tym zakresie.
- **Zadanie 8:** Napisz funkcję, która wyświetla pierwsze k liczb pierwszych (np. pierwsze 10 liczb pierwszych to 2, 3, 5, 7, 11, ...).

4. Zadania teoretyczne i analityczne:

- **Zadanie 9:** Wyjaśnij, dlaczego wystarczy sprawdzać dzielniki do pierwiastka kwadratowego liczby, aby określić, czy jest ona liczbą pierwszą.
- **Zadanie 10:** Opisz, jak algorytm zmieniłby się, gdybyśmy szukali liczb pierwszych tylko w zakresie od 2 do 1000. Ile sprawdzeń musiałby wykonać algorytm dla liczby 997?

5. Zadania zaawansowane:

- **Zadanie 11:** Napisz algorytm sprawdzający, czy dana liczba jest liczbą pierwszą, korzystając z metody sita Eratostenesa. Przetestuj go na zakresie 1–100.
- **Zadanie 12:** Napisz funkcję, która generuje wszystkie liczby bliźniacze (pary liczb pierwszych, które różnią się o 2) w zakresie 1-100, np. (3, 5), (11, 13).

Przykłady zadań maturalnych z algorytmem na liczby pierwsze

1. Znajdowanie liczb pierwszych w zakresie

- **Treść zadania**: Napisz program, który wczyta dwie liczby całkowite a i b i wypisze wszystkie liczby pierwsze w zakresie [a,b].
- **Rozwiązanie**: Algorytm sprawdzający liczby pierwsze jest kluczowy do tego zadania. Trzeba zaimplementować pętlę, która przejdzie przez zakres od a do b i sprawdzi każdą liczbę pod kątem pierwszości.

2. Największa liczba pierwsza mniejsza od N

- **Treść zadania**: Napisz program, który dla danej liczby N znajdzie największą liczbę pierwszą mniejszą od N.
- **Rozwiązanie**: Rozwiązanie polega na iteracji w dół od N−1, sprawdzaniu każdej liczby, aż znajdziemy pierwszą liczbę, która jest liczbą pierwszą.

3. Suma liczb pierwszych

- **Treść zadania**: Dla podanego zakresu liczb a i b, znajdź sumę wszystkich liczb pierwszych z tego zakresu.
- **Rozwiązanie**: W tym zadaniu algorytm sprawdzania liczby pierwszej jest stosowany dla każdej liczby w przedziale. Wszystkie liczby pierwsze są sumowane, a wynik jest wypisywany.

4. Sprawdzenie liczby bliźniaczej

- **Treść zadania**: Liczby bliźniacze to liczby pierwsze, które różnią się o 2 (np. 11 i 13). Napisz program, który wypisze wszystkie pary liczb bliźniaczych w przedziale od 1 do N.
- **Rozwiązanie**: Najpierw sprawdzamy, które liczby w zakresie są liczbami pierwszymi, a następnie sprawdzamy dla każdej z nich, czy różnica z kolejną liczbą pierwszą wynosi 2.

5. Najmniejsza liczba pierwsza większa od N

- **Treść zadania**: Napisz program, który znajdzie najmniejszą liczbę pierwszą większą od podanej liczby N.
- **Rozwiązanie**: Rozwiązaniem jest sprawdzanie każdej liczby większej od N, dopóki nie natrafimy na liczbę pierwszą.

6. Liczby pierwsze w liczbach palindromicznych

- **Treść zadania**: Napisz program, który wypisze wszystkie liczby palindromiczne w danym zakresie, które są także liczbami pierwszymi.
- **Rozwiązanie**: Algorytm wymaga połączenia dwóch funkcji: sprawdzającej, czy liczba jest palindromem (czyli jest symetryczna, np. 121), oraz funkcji sprawdzającej, czy liczba jest pierwsza.

7. Rozkład liczby na czynniki pierwsze

- **Treść zadania**: Napisz program, który dla podanej liczby całkowitej N znajdzie jej rozkład na czynniki pierwsze.
- **Rozwiązanie**: Algorytm sprawdza podzielność N przez kolejne liczby pierwsze (od najmniejszej). Jeśli liczba jest podzielna, to dodaje ją jako czynnik i dzieli N przez ten czynnik, kontynuując aż do 1.

Inne przykłady nawiązujące do liczby pierwszej:

8. Liczby pierwsze w tablicy

- **Treść zadania**: Napisz program, który wczyta tablicę n liczb całkowitych i wypisze tylko te liczby, które są liczbami pierwszymi.
- **Rozwiązanie**: Dla każdej liczby w tablicy stosujemy funkcję sprawdzającą pierwszość, a wynikowe liczby są wypisywane.

9. Wielokrotność liczby pierwszej

- **Treść zadania**: Znajdź największą wielokrotność liczby pierwszej w przedziale [a,b].
- **Rozwiązanie**: Najpierw trzeba znaleźć liczby pierwsze w przedziale, a następnie ich wielokrotności, aby ustalić największą z nich.