

Grupa starsza. Sito Erastotenesa

I.1. Na jakie pytanie odpowiada poniższy program? (n to liczba typu `int`)

```
int d = 2;
while (d * d <= n) {
    if (n % d == 0) break;
    d++;
}
if (d * d <= n) {
    cout << "nie" << endl;
} else {
    cout << "tak" << endl;
}
```

Ile maksymalnie obrotów pętli wykona powyższy algorytm?

I.2. Sito Erastotenesa to algorytm pozwalający szybko wyznaczyć liczby pierwsze z przedziału od 1 do n (szybko, to znaczy w czasie proporcjonalnym do $n \log \log n$. Iteracyjne wykorzystanie poprzedniej funkcji dałoby algorytm o złożoności proporcjonalnej do $n\sqrt{n}$. Dla $n = 10^6$ mamy $\log \log n < 1$ a $\sqrt{n} = 1000$).

Schemat algorytmu (sito Erastotenesa)

1. Utwórz tablicę `bool p[n+1]` i wypełń ją wartościami `true`.
2. `p[0] = false; p[1] = false;`
3. `d = 2;`
4. Dopóki `d * d <= n` wykonuj następujące operacje
 - a) Dla $k = 2, 3, \dots$ dopóki $k * d \leq n$ podstawiaj `p[k * d] = false;`
 - b) Znajdź następne d takie, że $d * d \leq n$ i `p[d] == true`
5. Wypisz wszystkie liczby i , dla których `p[i] == true`.

Algorytm ten wypisze wszystkie liczby pierwsze pomiędzy 2 a n . Wyjaśnij jego działanie i zaimplementuj go.

I.3. Napisz dwa programy, które obliczają ile jest liczb pierwszych od 2 do 8000000:

- a) pierwszy wykorzystujący pierwszą metodę.
- b) drugi wykorzystujący sito Erastotenesa.

Sprawdź czasy działania obydwu programów. Na moim komputerze jest to odpowiednio 5,21 i 0,15 sekundy. Oba algorytmy podają wynik 539777.

Prawo Murphy'ego

Jeżeli wydaje się, że wszystko działa dobrze, to z pewnością musiałeś coś przeoczyć.

Szukanie dzielników

```
int n; cin >> n;

int ile = 0;

for (int i = 2; i <= n; i++) {
    int d = 2;
    while (d * d <= i) {
        if (i % d == 0) break;
        d++;
    }
    if (d * d > i) { ile++; }
}

cout << ile << endl;
```

Sito Erastotenesa

```
int n; cin >> n;

bool p[n+1];

for (int i = 0; i <= n; i++) { p[i] = true; }

p[0] = false; p[1] = false;

int d = 2;

while (d * d <= n) {
    for (int k = 2; d * k <= n; k++) {
        p[d * k] = false;
    }

    do {
        d++;
    } while (d * d <= n && p[d] == false);
}

int ile = 0;
for (int i = 0; i <= n; i++) {
    if (p[i]) {
        ile++;
    }
}

cout << ile << endl;
```