

Erklärung zum Programm „Sieb des Eratosthenes“

Erstmal die Aufgabe:

Der griechische Philosoph Eratosthenes (276 – 195 v. Chr.) hat ein Verfahren entwickelt, mit dem sich Primzahlen im Intervall $[1, n]$ mit der Laufzeitkomplexität $O(n \log n)$ berechnen lassen. Dieses sogenannte *Sieb des Eratosthenes* funktioniert wie folgt:

- (a) Schreibe alle Zahlen von 1 bis n hin und streiche die Zahl 1 durch.
- (b) Sei i die kleinste noch nicht durchgestrichene und nicht eingerahmte Zahl. Rahme i ein und streiche alle Vielfachen von i durch. (\Rightarrow die Vielfachen von i werden “ausgesiebt”.)
- (c) Wiederhole (b) solange bis $i^2 > n$ ist.
- (d) Die eingerahmten und die nicht durchgestrichenen Zahlen sind die Primzahlen von 1 bis n .

Entwickeln Sie eine C-Funktion, die dieses Verfahren anwendet, um die Primzahlen im Intervall $[1, n]$ zu berechnen.

Und hier der Quelltext:

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

void eratosthenes(int zahlen[], int n)
{
    // Aufgabe a)
    for(int i=2; i<=n; i++)
    {
        zahlen[i] = 0; // 0 = nicht durchgestrichen
    }
    zahlen[1] = 1; // Zahl 1 durchstreichen

    int i = 2; // Sei i die kleinste, noch nicht durchgestrichene und nicht eingerahmte Zahl...
    while(i*i <= n) // Aufgabe c) (Wiederhole b) so lange, bis i2 größer n ist)
    {
        if(zahlen[i] == 0) // Wenn i nicht durchgestrichen ist
        {
            int j = 2*i; // j ist jetzt das erste Vielfache von i
            while(j <= n) // So lange j im Intervall von 1 bis n ist...
            {
                zahlen[j] = 1; // Streiche die Vielfachen von j...
                j = j+i; // Erhöhe j um den nächsten Vielfachen
            }
        }
        i++; // Geh zur nächsten Zahl und prüfe (im Schleifenkopf) ob sie durchgestrichen ist
    }
}
```

```

// Die eingerahmten und die nicht durchgestrichenen Zahlen
//(also alle Elemente, in denen eine 0 steht) sind Primzahlen von 1 bis n
// Also ausgeben

// Ausgabe der Primzahlen
for(int i=2; i<=n; i++)
{
    if(zahlen[i] == 0)
    {
        cout << i << endl;
    }
}

main()
{
    // Überschrift
    cout << "Programm zur Primzahlberechnung " << endl << endl;
    // Eingabeaufforderung
    cout << "Geben Sie n ein: ";
    int n;
    cin >> n;

    // Array mit n+1 Plätzen anlegen, damit von 0 bis n alle Plätze da sind
    // (0 brauchen wir nicht, aber n)
    int zahlen[n+1];

    cout << "Primzahlen im Intervall von 1 bis " << n << ": " << endl;
    eratosthenes(zahlen, n);
    getch();
}

```

Erklärungstext:

Beginnen wir, wie immer, mit der main-Funktion:

```

main()
{

```

Aus Gründen der Ästhetik erst einmal die Überschrift...

```

    cout << "Programm zur Primzahlberechnung " << endl << endl;

```

Nun soll der Benutzer die Größe des Intervalls, also n, eingeben...

```

    cout << "Geben Sie n ein: ";
    int n;
    cin >> n;

```

Jetzt legen wir ein Array an, das die **Elemente von 1 bis n** anbietet. Wenn ein Array mit n Plätzen anlegen würden, hätten wir Elemente von 0 bis n-1, das wollen wir nicht, denn wir brauchen ja auch

das nte Element. Also müssen wir ein Array mit n+1 Plätzen anlegen...

```
int zahlen[n+1];
```

Dann schreiben wir schon mal, dass gleich die Primzahlen von n bis n ausgegeben werden:

```
cout << "Primzahlen im Intervall von 1 bis " << n << ": " << endl;
```

Und rufen die Funktion, die wir schreiben werden, auf...

```
eratosthenes(zahlen, n);
```

Zum Ende des main-Programmes soll der Compiler wie immer auf ein Zeichen warten...

```
    getch();  
}
```

Nun ist es ja so, dass die Arrayelemente, wenn wir sie nicht initialisieren (also ihnen keine Werte zuweisen) irgendwelche Werte zu Anfang haben. Das ist aber nicht Sinn der Sache. Es sollen, laut Aufgabe, Zahlen gestrichen werden und Zahlen geben, die nicht gestrichen werden (das sind dann die eingerahmten).

Also gibt es zwei Zustände. Sozusagen den Zustand 0 und den Zustand 1.

Und das ist jetzt Definitionssache. Sagen wir, die durchgestrichenen Elemente (also Zahlen) sind 1, die nicht durchgestrichenen Elemente erhalten den Eintrag 0.

Jedes Element steht für seine Zahl (also das fünfte Arrayelement steht für die Zahl 5, deswegen brauchen wir auch n Arrayelemente und nicht n-1).

Die Funktion soll eratosthenes heißen und sie braucht zwei Parameter: Einmal ist die Frage, wie groß n ist und einmal soll ihr ein Array übergeben werden, in dem sie dann arbeitet...

Sie schreibt die Primzahlen schließlich auf die Konsole auf, sie gibt aber keinen Wert zurück, also ist der Rückgabetyt leer (sprich: void).

```
void eratosthenes(int zahlen[], int n)  
{
```

In Aufgabe a) hieß es, dass wir alle Zahlen von 1 bis n hinschreiben und die 1 durchstreichen sollen.

```
    // Aufgabe a)  
    for(int i=2; i<=n; i++)  
    {  
        zahlen[i] = 0; // 0 = nicht durchgestrichen  
    }  
    zahlen[1] = 1; // Zahl 1 durchstreichen
```

Das machen wir hier. Wir haben, statt die Zahlen hinzuschreiben, eben solche Elemente angelegt. Nun ist es Definitionssache. Wir legen einfach fest, dass die 0 dafür steht, dass die Zahlen nicht durchgestrichen sind, während die 1 dafür steht, dass sie durchgestrichen sind. Folglich weisen wir dem Element 1, das stellvertretend für die Zahl 1 steht, die 1 zu (denn die soll ja durchgestrichen werden).

Dann heißt es, dass i die kleinste noch nicht durchgestrichene und nicht eingerahmte Zahl sei...

Also...

```
int i = 2; // Sei i die kleinste, noch nicht durchgestrichene und nicht eingerahmte Zahl...
```

In Aufgabe c heißt es, das was wir jetzt machen, sollen wir solange machen, bis i^2 größer als n ist „So lange, bis“ ist ein Hinweis auf eine while-Schleife (solche Aufgaben lassen sich zumindest unkomplizierter mit while lösen)...

```
while(i*i <= n) // Aufgabe c) (Wiederhole b) so lange, bis  $i^2$  größer n ist)
```

```
{
```

Jetzt wollen wir dem Programm sagen dass die Vielfachen von i durchgestrichen werden sollen. Aber nur, wenn i selbst nicht bereits durchgestrichen ist. Diesen Fall müssen wir erstmal prüfen. Also...

```
    if(zahlen[i] == 0) // Wenn i nicht durchgestrichen ist
```

```
    {
```

Was soll er dann machen?

Er soll sich um die Vielfachen von i kümmern. Fangen wir mit dem ersten Vielfachen an...

```
        int j = 2*i; // j ist jetzt das erste Vielfache von i
```

Er soll nur so lange die Vielfachen streichen, wie er sich im Bereich von 1 bis n befindet. Ist klar, müssen wir ihm aber auch sagen... Also:

```
            while(j <= n) // So lange j im Intervall von 1 bis n ist...
```

```
            {
```

Wir hatten festgelegt, dass ein durchgestrichenes Element (das ja stellvertretend für die Zahl steht)

```
                zahlen[j] = 1; // Streiche die Vielfachen von j...
```

```
                j = j+i; // Erhöhe j um den nächsten Vielfachen
```

```
            }
```

Dann schließen wir die Klammer der while-Schleife:

```
    }
```

Nach dem er alle Vielfachen von i gestrichen hat (also die Elemente mit 1 gefüllt hat, ohne Element i selbst zu streichen, denn das Element i hat noch immer den Wert 0, ist also nicht durchgestrichen, somit eingerahmt) soll er sich um die nächste Zahl kümmern, die noch nicht durchgestrichen wurde. Zunächst einmal erhöhen wir dazu das i um den Wert 1. Ob der nächsthöhere Wert bereits durchgestrichen ist oder nicht wird ja im nächsten Durchlauf durch die if-Bedingung geprüft und nur für den Fall, dass die Zahl nicht durchgestrichen ist, werden die Vielfachen durchgestrichen... Also:

```
    i++; // Geh zur nächsten Zahl und prüfe (im Schleifenkopf) ob sie durchgestrichen ist
```

```
}
```

Zum Schluss sollen alle Zahlen, die nicht durchgestrichen wurden, ausgegeben werden, denn diese sind ja nun Primzahlen (so hieß es in der Aufgabe).

Da die Elemente ja für die Zahlen stehen, gehen wir also die Elemente durch und wir wissen, dass die Elemente, in denen eine 0 drin steht, Zahlen repräsentieren, die nicht durchgestrichen wurden.

Also:

```
// Ausgabe der Primzahlen
```

```
for(int i=2; i<=n; i++)
```

```
{
```

```
    if(zahlen[i] == 0)
```

```
    {
```

```
        cout << i << endl;
```

```
    }
```

```
}
```

```
}
```