Ludwig-Maximilians-Universität München
Mathematisches Institut
Gregor Kleen

Programmieren II für Studierende der Mathematik

Blatt 4 – Lösungsvorschlag

Aufgabe 4 Sei $N \in \mathbb{N}$, $N \geq 2$. Vereinbaren Sie $N \leq 2^{32}$ als Konstante für das gesamte Programm am besten durch Zuweisung eines Literals in hexadezimaler Schreibweise. Kennt man die Primzahlen kleiner als N, so lässt sich die Primfaktorzerlegung einer Zahl $n \in N$ mit $n < N^2$ durch fortgesetztes Abdividieren der Primteiler bestimmen.

Erstellen Sie zunächst eine Klasse `Primzahlen` mit einer internen statischen Datenkomponente vom Typ `bitset<N>`, die für jede Zahl $k \in N$ mit $k < N$ speichert ob sie eine Primzahl ist, oder nicht. Implementieren Sie eine statische Komponentenfunktion die das Sieb des Eratosthenes verwendet um die statische Datenkomponente korrekt zu befüllen. Rufen Sie diese statische Komponentenfunktion zu Beginn des Hautprogramms einmal auf.

Hinweis (Sieb des Eratosthenes). Um alle Primzahlen kleiner als N zu bestimmen können wir ausgehend von $k = 2$ alle Vielfachen von Primzahlen $i \cdot k$ streichen mit $i \geq k$ und $i \cdot k < N$. Die erste noch nicht gestrichene Zahl ist dann die nächste Primzahl k. Iterieren Sie dieses Verfahren solange $k^2 < N$.

Primzahlen

```
class Primzahlen {
  private:
    static bitset<N> primes;

  public:
    static void init() {
      primes.set();
      primes.reset(0); primes.reset(1);

      for (unsigned long k = 2; k*k < N; k++)
        if (primes[k])
          for (unsigned long i = k; i * k < N; i++)
            primes.reset(i*k);
    }

    Primzahlen Ist Primzahl
};

bitset<N> Primzahlen::primes{};
```

Fügen Sie zur Klasse `Primzahlen` eine statische Komponentenfunktion `ist_primzahl` hinzu die für ein als Parameter gegebenes n mit $0 \leq n < N$ vom Typ `unsigned long` als `bool` zurück gibt ob n eine Primzahl ist.

Primzahlen Ist Primzahl

```
static bool ist_primzahl(unsigned long n) {
  if (n < N)
    return primes[n];
  else if (N * N > n) {
```

```
    for (unsigned long k = 2; k < N; k++)
      if (primes[k] && n % k == 0)
        return false;
    return true;
  } else {
    ostringstream err;
    err << n << " lies outside of acceptable range for ist_primzahl [0.." << N*N << ")";
    throw domain_error(err.str());
  }
}
```

Erstellen Sie eine Klasse `Primfaktor`. Objekte der Klasse `Primfaktor` sollen zur Modellierung einer Primzahlpotenz der Form p^k für p eine Primzahl und sowohl p wie auch k vom Typ `int` dienen.

Primfaktor

```
class Primfaktor {
  public:
    unsigned long p, k;

    Primfaktor(unsigned long p_ = 0, unsigned long k_ = 1): p(p_), k(k_) {}

    Ausgabe Primfaktor
};
```

Erstellen Sie eine Klasse `Primfaktorzerlegung`. Objekte der Klasse `Primfaktorzerlegung` modellieren eine natürliche Zahl indem intern eine Liste von Primfaktoren vom Datentyp `list<Primfaktor>` gespeichert wird. Als Invariante der Klasse soll sichergestellt werden, dass die Liste von Primfaktoren stets aufsteigend nach Primzahl sortiert vorliegt und für alle gespeicherten Primfaktoren p^k gilt, dass $k \geq 1$.

Primfaktorzerlegung

```
class Primfaktorzerlegung {
  private:
    list<Primfaktor> factors;

  public:
    Primfaktorzerlegung Konstruktor

    Typumwandlung

    Primfaktorzerlegung Ist Primzahl

    Ausgabe Primfaktorzerlegung
};
```

Erstellen Sie einen Konstruktor für die Klasse `Primfaktorzerlegung` mit einem Argument $0 < n < N^2$ vom Typ `unsigned long`. Für $n = 1$ soll die interne Liste leer angelegt werden. Verwenden Sie zur Berechnung der Primfaktorzerlegung die Funktion `ist_primzahl` aus der Klasse `Primzahlen`.

Primfaktorzerlegung Konstruktor

```
Primfaktorzerlegung(unsigned long n = 1) {
  for (unsigned long p = 2; p * p <= n && p * p < N; p++) {
    if (!Primzahlen::ist_primzahl(p))
      continue;

    unsigned long k = 0;
    while (n % p == 0) {
      k++;
      n /= p;
    }

    if (k >= 1)
      factors.push_back(Primfaktor{p, k});
  }

  if (n != 1)
    factors.push_back(Primfaktor{n});
}
```

Überladen Sie den Ausgabeoperator für die Klasse `Primfaktorzerlegung` sodass die Ausgabe in der Form p_1^k_1*...*p_n^k_n erfolgt. Für die leere Liste soll `1` ausgegeben werden.

Ausgabe Primfaktor

```
friend ostream& operator<<(ostream& stream, Primfaktor pf) {
  stream << pf.p;
  if (pf.k != 1)
    stream << "^" << pf.k;
  return stream;
}
```

Ausgabe Primfaktorzerlegung

```
friend ostream& operator<<(ostream& stream, Primfaktorzerlegung pfz) {
  if (pfz.factors.empty())
    return stream << "1";

  for (list<Primfaktor>::iterator i = pfz.factors.begin();
    i != pfz.factors.end();
    i++) {
    if (i != pfz.factors.begin())
      stream << "*";
    stream << *i;
  }

  return stream;
}
```

Fügen Sie einen Typumwandlungsoperator hinzu um Objekte der Klasse `Primfaktorzerlegung` zu Werten vom Typ `unsigned long` implizit konvertieren zu können.

Typumwandlung

```
operator unsigned long() {
  unsigned long res = 1;
  for (Primfaktor pf: factors)
    while (pf.k-- > 0)
      res *= pf.p;
  return res;
}
```

Erstellen Sie eine mit der Klasse `Primfaktorzerlegung` befreundete Funktion `ist_primzahl` die für ein Argument vom Typ `Primfaktorzerlegung` unter direkter Verwendung der internen Darstellung als Liste von `Primfaktoren` bestimmt und als `bool` zurückgibt ob das Argument eine Primzahl ist.

Primfaktorzerlegung Ist Primzahl

```
friend bool ist_primzahl(const Primfaktorzerlegung& pfz) {
  if (pfz.factors.empty())
    return false;

  list<Primfaktor>::const_iterator it = pfz.factors.begin();
  return it->k == 1 && ++it == pfz.factors.end();
}
```

Erstellen Sie ein Hauptprogramm, das für Zahlen der Form $2^j - 1$ mit $j = 1, \ldots, 64$ und $2^j - 1 \leq N^2 - 1$ jeweils die Primfaktorzerlegung ausgibt. Sofern es sich um eine Primzahl handelt soll jeweils zusätzlich der Text `Mersenne-Primzahl` ausgegeben werden. Überprüfen Sie zudem ob das Ergebnis der Typumwandlung von der Primfaktorzerlegung zurück zu `unsigned int` identisch ist mit der zu faktorisierenden Zahl und geben Sie eine geeignete Warnung aus, sollte dem nicht so sein.

factorisation.cpp

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <bitset>
#include <list>

using namespace std;

const unsigned long N = 0x100000000;
```

Primzahlen

Primfaktor

Primfaktorzerlegung

```
int main() {
  Primzahlen::init();

  unsigned long zweip = 1;
```

```
  while (true) {
    Primfaktorzerlegung pfz{zweip};
    cout << setw(20) << zweip;
    if (ist_primzahl(pfz))
      cout << " Mersenne-Primzahl: ";
    else
      cout << ":                   ";
   cout << pfz << endl;
   if (zweip != static_cast<unsigned long>(pfz))
     cout << "Typumwandlung (" << static_cast<unsigned long>(pfz) << ") entspricht nicht
     ↪  Argument (" << zweip << ")" << endl;

    unsigned long old_zweip = zweip;
    zweip = (zweip << 1) | 1;
    if (zweip > N * (N - 1) + (N - 1) || old_zweip == zweip)
      break;
  }
}
```

```
                   1:                   1
                   3 Mersenne-Primzahl: 3
                   7 Mersenne-Primzahl: 7
                  15:                   3*5
                  31 Mersenne-Primzahl: 31
                  63:                   3^2*7
                 127 Mersenne-Primzahl: 127
                 255:                   3*5*17
                 511:                   7*73
                1023:                   3*11*31
                2047:                   23*89
                4095:                   3^2*5*7*13
                8191 Mersenne-Primzahl: 8191
               16383:                   3*43*127
               32767:                   7*31*151
               65535:                   3*5*17*257
              131071 Mersenne-Primzahl: 131071
              262143:                   3^3*7*19*73
              524287 Mersenne-Primzahl: 524287
             1048575:                   3*5^2*11*31*41
             2097151:                   7^2*127*337
             4194303:                   3*23*89*683
             8388607:                   47*178481
            16777215:                   3^2*5*7*13*17*241
            33554431:                   31*601*1801
            67108863:                   3*2731*8191
           134217727:                   7*73*262657
           268435455:                   3*5*29*43*113*127
           536870911:                   233*1103*2089
          1073741823:                   3^2*7*11*31*151*331
          2147483647 Mersenne-Primzahl: 2147483647
          4294967295:                   3*5*17*257*65537
          8589934591:                   7*23*89*599479
         17179869183:                   3*43691*131071
         34359738367:                   31*71*127*122921
         68719476735:                   3^3*5*7*13*19*37*73*109
```

```
         137438953471:                   223*616318177
         274877906943:                   3*91625968981
         549755813887:                   7*79*8191*121369
        1099511627775:                   3*5^2*11*17*31*41*61681
        2199023255551:                   13367*164511353
        4398046511103:                   3^2*7^2*43*127*337*5419
        8796093022207:                   431*9719*2099863
       17592186044415:                   3*5*23*89*397*683*2113
       35184372088831:                   7*31*73*151*631*23311
       70368744177663:                   3*47*499069107643
      140737488355327:                   2351*4513*13264529
      281474976710655:                   3^2*5*7*13*17*97*241*257*673
      562949953421311:                   127*4432676798593
     1125899906842623:                   3*11*31*251*601*1801*4051
     2251799813685247:                   7*103*2143*11119*131071
     4503599627370495:                   3*5*53*157*1613*2731*8191
     9007199254740991:                   6361*1416003655831
    18014398509481983:                   3^4*7*19*73*22906579627
    36028797018963967:                   23*31*89*881*3191*201961
    72057594037927935:                   3*5*17*29*43*113*127*15790321
   144115188075855871:                   7*32377*635879915089
   288230376151711743:                   3*59*233*1103*2089*3033169
   576460752303423487 Mersenne-Primzahl: 576460752303423487
  1152921504606846975:                   3^2*5^2*7*11*13*31*41*61*151*331*1321
  2305843009213693951 Mersenne-Primzahl: 2305843009213693951
  4611686018427387903:                   3*1537228672809129301
  9223372036854775807:                   7^2*73*127*337*60247241209
 18446744073709551615:                   3*5*17*257*641*439125228929
```