Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. B. Gärtner

14. Dezember 2004

Informatik für Mathematiker und Physiker Serie 9 WS 04/05

URL: `http://www.ti.inf.ethz.ch/ew/courses/Info1_04/`

Aufgabe 1 (6 Punkte)

Ein Polynom p mit `int` Koeffizienten kann als Variable v vom Typ `std::vector<int>` repräsentiert werden. Hierbei entspricht `v[i]` dem Koeffizienten von x^i, $0 \leq i \leq \text{grad}(p)$. Zum Beispiel ist $p(x) = x^3 + 2x^2 - 1$ durch $v = (-1, 0, 2, 1)$ repräsentiert.

Im folgenden verwenden wir `Nt` als Synonym für `int` und `Polynom` als Synonym für `std::vector<Nt>`.

a) Schreiben Sie eine Funktion

```
Nt eval(const Polynom& a, Nt x)
```

welche das Polynom `a` an der Stelle `x` auswertet.

b) Schreiben Sie eine Funktion

```
void add(Polynom& s, const Polynom& a, const Polynom& b)
```

welche die Summe `s` der Polynome `a` und `b` berechnet.

c) Schreiben Sie eine Funktion

```
void multiply(Polynom& p, const Polynom& a, const Polynom& b)
```

welche das Produkt `p` der Polynome `a` und `b` berechnet.

Schreiben Sie auch eine passende Testumgebung, in der Sie die einzelnen Funktionen an verschiedenen Beispielen ausprobieren.

Aufgabe 2 (6 Punkte)

Im folgenden wird `It` als Synonym für `std::vector<int>::iterator` verwendet. Schreiben Sie eine Funktion

```
void random_shuffle(It b, It e)
```

welche die Elemente des range $[b, e)$ zufällig permutiert. Gehen Sie dabei analog zum Sortieralgorithmus aus der Vorlesung vor: Im i-ten Schritt, $0 \leq i < e - b$, wählen Sie das Element an der Position $b + i$ zufällig gleichverteilt aus den Elementen von $[b + i, e)$. Verwenden Sie hierzu die Funktion `ifm::random` aus Serie 5 sowie die in der Bibliothek `<algorithm>` definierte Funktion `std::iter_swap`.

Testen Sie die Funktion im Zusammenhang mit dem Sortierverfahren aus der Vorlesung.

Abgabe: Aufgaben 1 und 2: bis 20. Dezember 2004, 16.00 Uhr, per Email.

Eidgenössische Technische Hochschule Zürich — Ecole polytechnique fédérale de Zurich — Politecnico federale di Zurigo — Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. B. Gärtner

14. Dezember 2004

Informatik I: Material aus der Vorlesung

Programm: eratosthenes.C

```
// Programm: eratosthenes.C
// Primzahlenberechnung - Sieb des Eratosthenes.

#include <iostream>
#include <vector>

typedef std::vector<bool>     Vec;

int main()
{
  std::cout << "Berechne Primzahlen bis: ";
  unsigned int n;
  std::cin >> n;

  std::cout << "Primzahlen in [0," << n << "): ";
  Vec is_prime (n, true);
  for (unsigned int i = 2; i < n; ++i)
    // Invariante: Fuer alle j in [2,n):
    //   (1) j Primzahl => is_prime[j]
    //   (2) j % k == 0 fuer ein k in [2,i) => !is_prime[j]
    if (is_prime[i]) {
      std::cout << i << " ";
      // markiere alle echten Vielfachen von i als nicht-prim
      for (unsigned int m = 2*i; m < n; m += i)
        is_prime[m] = false;
    }
  std::cout << std::endl;

  return 0;
}
```

Programm: eratosthenes-it.C

```
// Programm: eratosthenes-it.C
// Primzahlenberechnung - Sieb des Eratosthenes.

#include <iostream>
#include <vector>

typedef std::vector<bool>     Vec;
typedef Vec::iterator         It;
typedef Vec::difference_type  Diff;

int main()
{
  std::cout << "Berechne Primzahlen bis: ";
  unsigned int n;
  std::cin >> n;

  std::cout << "Primzahlen in [0," << n << "): ";
  Vec is_prime (n, true);
  for (It i = is_prime.begin() + 2; i != is_prime.end(); ++i)
    // Invariante: Fuer alle j in [2,n):
    //   (1) j Primzahl => *(is_prime.begin() + j)
    //   (2) j % k == 0 fuer ein k in [2,i-is_prime.begin())
    //         => !*(is_prime.begin() + j)
    if (*i) {
      Diff j = i - is_prime.begin();
      std::cout << j << " ";
      // markiere alle echten Vielfachen von j als nicht-prim
      for (It m = i + j; m < is_prime.end(); m += j)
        *m = false;
    }
  std::cout << std::endl;

  return 0;
}
```

Programm: minsort.C

```
// Programm: minsort.C
// Sortieren durch Minimum-Suche.

#include <iostream>
#include <vector>
#include <algorithm>

typedef std::vector<int>     Vec;
typedef Vec::iterator        It;
typedef Vec::const_iterator  Cit;

namespace ifm {

  It min_element(It b, It e)
  // PRE: [b,e) ist gueltiger range.
  // POST: Rueckgabewert ist der erste Iterator i in [b,e),
  //   fuer den gilt: fuer alle j in [b,e) ist *i <= *j.
  {
    if (b == e) return e;
    It m = b++;
    for (; b != e; ++b)
      if (*b < *m) m = b;
    return m;
  }

  void min_sort(It b, It e)
  // PRE: [b,e) ist gueltiger range.
  // POST: [b,e) ist aufsteigend sortiert, d.h., fuer alle
  //   Paare i,j aus [b,e) mit j aus [i,e) gilt *i <= *j.
  {
    for (; b != e; ++b)
      std::iter_swap(b, ifm::min_element(b, e));
  }

} // namespace ifm

int main()
{
  Vec v;
  for (int i = 0; i < 5; ++i) {
    v.push_back(2*i);
    v.push_back(9-2*i);
  }
  // v == (0,9,2,7,4,5,6,3,8,1)

  // Ausgabe von v
  std::cout << "v = (";
  for (Cit i = v.begin(); i != v.end(); ++i)
    std::cout << *i << " ";
  std::cout << ")" << std::endl;

  // Sortieren von v
  ifm::min_sort(v.begin(), v.end());

  // Ausgabe von v
  std::cout << "v = (";
  for (Cit i = v.begin(); i != v.end(); ++i)
    std::cout << *i << " ";
  std::cout << ")" << std::endl;

  return 0;
}
```