Informatik IIa - Praktikum

Oliver Jack
Fachhochschule Jena
Fachbereich Elektrotechnik und Informationstechnik

Sommersemester 2010

Praktikum 4: Zeigerarithmetik, Map-Container

Aufgabe 1 (Zeigerarithmetik): Betrachten Sie das folgende C++-Programm und geben Sie für jede der Zuweisungen an `val` an,

- welchen Wert `val` nach der Zuweisung hat, und
- auf welches Element von `array` der Zeiger `p` nach der Zuweisung zeigt.

Welche Werte haben die Elemente von `array` nach allen Zuweisungen?

```
#include <iostream>
using namespace std;
int main ()
{
        int array[ ] = { 0, 10, 20, 30, 40, 50 };
        int val, *p;

        p = array;
        val = *p;
        val = *p++;
        val = *++p;
        val = (*p)++;
        val = *(p++);
        val = ++*p;
        val = ++(*p);

        return(0);
}
```

Aufgabe 2 (Telefonbuch): Eine `map` ist ein Container, der aus Wertepaaren besteht. Beispiel:

```
map<string, int> telbuch;
```

Eine `map` wird auch als *assoziatives Feld* oder Wörterbuch (englisch: *dictionary*) bezeichnet.

Wenn eine `map` mit einem Wert vom Typ ihres ersten Typs (auch *Schlüssel* genannt) indiziert wird, liefert sie den zugehörigen Wert ihres zweiten Typs (Wert oder *mapped type* genannt) zurück. Beispiel:

```
void eintragAusgeben(const string& s) {
  if (int i = telbuch[s]) cout << s << ' ' << i << endl;
}
```

Wenn kein Schlüssel mit dem Wert von `s` existiert, wird von `telbuch` ein Default-Wert zurückgegeben. Der Default-Wert für einen ganzzahligen Typ in einer `map` ist 0.

Schreiben Sie ein C++-Programm, dass die Eingabe von Namen und zugehörigen Telefonnummern erlaubt. Nach der Eingabe soll es möglich sein, durch Eingabe eines Namens die zugehörige Telefonnummer ausgegeben zu bekommen. Sehen Sie für Ihr Programm vor, dass der Benutzer beliebig viele Abfragen nach Telefonnummern vornehmen kann. Benutzen Sie für Ihr Programm den Standard-Container `map`.

Aufgabe 3 (Werte berechnen): Eine `multimap` ist wie eine `map` mit dem Unterschied, dass für einen Schlüssel auch Duplikate erlaubt sind. Das bedeutet, dass `multimap` nicht wie `map` den Indexoperator für Schlüsselwerte unterstützen kann. Die Operationen `equal_range()`, `lower_bound()` und `upper_bound()` diene dazu, auf mehrere Werte mit dem selben Schlüssel zuzugreifen.

Der Umgang mit einer `multimap` sei in folgenden am Beispiel einer Person, die im Telefonbuch zwei Telefonnummern besitzt, demonstriert.

```
multimap<string, int> test;

// die beiden Telefonnummern von Meyer eingeben
test.insert(make_pair("Meyer", 1234));
test.insert(make_pair("Meyer", 5678));
// Ausgabe aller Telefonnummern von Meyer
eintragAusgeben("Meyer", test);

// ...

// Ausgabefunktion
void eintragAusgeben(const string s,
                     multimap<string, int>& telbuch) {
  typedef multimap<string, int>::const_iterator I;
  pair<I, I> b = telbuch.equal_range(s);
  for (I i=b.first; i!=b.second; ++i)
    cout << i->second << endl;
}
```

Schreiben Sie ein C++-Programm, dass eine Folge von durch Whitespace getrennten (Name, Wert)-Paaren einliest, bei dem Name ein einzelnes durch Whitespace getrenntes Wort und Wert eine ganze Zahl oder eine Gleitkommazahl ist. Berechnen Sie die Summe und den Mittelwert für jeden Namen und für alle Namen und geben Sie sie jeweils aus. Benutzen Sie den Standard-Container `multimap`.