Tentamen
C++-programmering

2014–04–29, 14.00–19.00

Hjälpmedel: En valfri C++-bok. OH-bilderna från föreläsningarna är *inte* tillåtna.

Du ska i dina lösningar visa att du behärskar C++ och att du kan använda C++ standardklasser. ”C-lösningar” ger inga poäng, även om de är korrekta.

Uppgifterna ger preliminärt $18 + 16 + 8 + 3 + 5 = 50$ poäng. För godkänt krävs 25 poäng (3/25, 4/33, 5/42).

---

1. I en bok är ett sakregister (ett index) en sorterad lista med viktiga ord med hänvisningar till de sidor där ordet förklaras eller används. Du ska skriva ett program som producerar något som liknar ett sakregister för en text.

   - Texten ska läsas från standard input, registret ska skrivas på standard output.
   - *Alla* orden i texten ska vara med i registret. Hänvisningarna ska vara till *rader* i indata. Om ett ord förekommer flera gånger på samma rad ska det bara finnas en hänvisning till raden.
   - Ord är följder av tecken åtskilda med whitespace. Skiljetecknen `,.;:!?()` ska tas bort ur orden.
   - Orden ska skrivas ut med ett ord per rad följt av radhänvisningarna i växande ordning. Små och stora bokstäver betraktas som olika, men orden ska sorteras oberoende av skiftläge.

   Exempel på indata (från en sång av Art Paul Schlosser) och utskrift:

   ```
   Mary had a little lamb
   Little lamb, little lamb
   Mary had a little lamb and some olives on the side.

   a 1 3
   and 3
   had 1 3
   lamb 1 2 3
   Little 2
   little 1 2 3
   Mary 1 3
   olives 3
   on 3
   side 3
   some 3
   the 3
   ```

   Observera att `Mary` kommer mellan `little` och `olives`, att både `Little` och `little` finns med och att inga skiljetecken finns med.

2. Här är ett utkast till en klass som beskriver heltalsvektorer (det hade naturligtvis varit bättre att använda `std::vector<int>`):

```
class Vector {
public:
    Vector(size_t n_max); // creates a vector with space for n_max elements
    Vector(const Vector&);
    Vector& operator=(Vector);
    Vector& operator+=(Vector);
    size_t size() { return n; }
    int operator[](size_t i) { return v[i]; }
    int operator[](size_t i) const { return v[i]; }
private:
    int* v;   // the numbers
    size_t n; // the size
};

Vector operator+(Vector, Vector);
```

a) Skriv en funktion där samtliga konstruktorer, funktioner och operatorer utnyttjas. Skriv kommentarer som visar när respektive konstruktion utnyttjas, i stil med följande:

```
Vector v(10); // the constructor Vector(size_t)
```

b) I klassen saknas en väsentlig deklaration. Vilken? Vad inträffar om deklarationen och tillhörande definition inte finns med?

c) Funktionen `size()` går inte alltid att använda. När? Korrigera funktionen.

d) Implementera funktionerna `operator+=` och `operator+`. Du kan förutsätta att vektorerna som adderas är lika långa.

e) I ett testprogram förekommer följande programavsnitt:

```
Vector v(10);
for (size_t i = 0; i != v.size(); ++i) {
    v[i] = i;
}
for (size_t i = 0; i != v.size(); ++i) {
    cout << v[i] << " ";
}
cout << endl;
```

Kompilatorn ger ett felmeddelande på rad 3: ”expression is not assignable”. Vad beror felet på? Korrigera felet.

f) I testprogrammet finns också följande sats:

```
Vector v2 = 100;
```

Vad händer när satsen utförs? Om vi inte vill att detta ska hända, utan att satsen i stället ska ge ett kompileringsfel, hur ska vi då ändra deklarationen av klassen?

g) Allt verkar nu fungera. Någon påpekar dock att definitionen av `operator=` (se nedan) innehåller två fel. När och på vilket sätt skulle felen ha visat sig? Rätta till felen.

```
Vector& Vector::operator=(Vector rhs) {
    v = new int[n = rhs.n];
    for (size_t i = 0; i != n; ++i) {
        v[i] = rhs.v[i];
    }
    return *this;
}
```

h) Programmet fungerar nu, men det är väldigt långsamt när man arbetar med stora vektorer. Vad beror detta på?

3. Om egenskaperna hos `ostream`-iteratorer, från Lippmans bok:

`ostream_iterator<T> out(os);`	`out` writes values of type `T` to output stream `os`.
`ostream_iterator<T> out(os, d);`	`out` writes values of type `T` followed by `d` to output stream `os`. `d` points to a null-terminated character array.
`out = val`	Writes `val` to the `ostream` to which `out` is bound using the `<<` operator. `val` must have a type that is compatible with the type that `out` can write.
`*out, ++out, out++`	These operations exist but do nothing to `out`. Each operator returns `out`.

We can use an `ostream_iterator` to write a sequence of values:

```
vector<int> vec = ...;
ostream_iterator<int> out_iter(cout, " ");
for (auto e : vec) {
    *out_iter++ = e; // the assignment writes this element to out
}
cout << endl;
```

[...] Rather than writing the loop ourselves, we can more easily print the elements in `vec` by calling `copy`:

```
copy(vec.begin(), vec.end(), out_iter);
cout << endl;
```

Implementera klassen `ostream_iterator`. Kalla klassen `OS` så slipper du skriva så mycket.

4. Ersätt kommentarerna i följande program med något som gör att programmet skriver ut `true`:

```
#include <iostream>
using namespace std;
// ...
int main() {
    // ...
    cout << boolalpha << (x == 1 && x == 2) << endl;
}
```

5. I en vektor finns ett stort antal studenter:

```
class Student {
public:
    Student(const string& name, int grade);
    string get_name() const;
    int get_grade() const;
    // ...
};

bool passed_exam(const Student& s) {
    return s.get_grade() >= 3;
}

vector<Student> students = { ... };
sort(students.begin(), students.end(), [](const Student& s1, const Student& s2)
        { return s1.get_name() < s2.get_name(); });
```

... fortsätter nästa blad

I följande programavsnitt flyttas de underkända studenterna till en annan vektor. Ordningen mellan objekten bibehålls:

```
vector<Student> failed;
auto it = students.begin();
while (it != students.end()) {
    if (!passed_exam(*it)) {
        failed.push_back(*it);
        students.erase(it);
    } else {
        ++it;
    }
}
```

a) Programmet innehåller ett fel. Det är mycket möjligt att felet inte märks när man kör programmet, men om man ändrar vektorn `students` till en länkad lista (`std::list`) så får man troligen ett adresseringsfel. Korrigera felet.

b) Programmet är ineffektivt. Varför?

c) Skriv om programmet så att det blir effektivare. Elegantast blir det med hjälp av en standardalgoritm.