Tentamen
C++-programmering

2017–06–05, 08.00–13.00

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 $10 + 4 + 10 + 10 + 16 = 50$ poäng. För godkänt krävs 25 poäng (3/25, 4/33, 5/42).

---

1. Betrakta följande klass:

```
class Ptr {
public:
    Ptr(int* p) : curr(p) {}
    int operator*() const { return *curr; }
private:
    int* curr;
};
```

Klassen beskriver bara en `int`-pekare och kan inte användas till mycket, men följande fungerar åtminstone:

```
int main() {
    int x[] = {1, 2, 3};
    Ptr p = x;
    std::cout << *p << std::endl;
}
```

a) Det fungerar alltså att läsa via `Ptr`-objektet. Men att skriva (`*p = ...;`) fungerar inte. Varför inte? Korrigera klassen så att också detta fungerar.

b) Skriv om klassen så att följande program fungerar:

```
int main() {
    int x[] = {1, 2, 3};
    for (Ptr<int> p = x; p != x + 3; ++p) {
        std::cout << *p << " ";
    }
    std::cout << std::endl;
    std::string y[] = {"Despite", "the", "constant", "negative", "press",
                       "covfefe"};
    for (Ptr<std::string> p = y; p != y + 6; ++p) {
        std::cout << *p << " ";
    }
    std::cout << std::endl;
}
```

2. Följande program fungerar utmärkt i Java:

```
class strplus {
   public static void main(String[] args) {
      int covfefe = 34567890;
      System.out.println("Covfefe count: " + covfefe);
   }
}
```

   Utskriften blir som förväntat:

```
Covfefe count: 34567890
```

   Man kan skriva på samma sätt i C++:

```
int main() {
   int covfefe_count = 34567890;
   std::cout << "Covfefe count: " + covfefe_count << std::endl;
 }
```

   Programmet är helt legalt och går att kompilera, men när jag (Roger) provkörde programmet på min Mac fick jag följande utskrift från programmet:

```
Segmentation fault: 11
```

   Förklara utskriften från C++-programmet genom att redogöra för vad som händer när programmet körs.

3. Vi vill i en klass hålla reda på namn. Namnen lagrar vi i ett objekt av STL-klassen set. Vi har valt att i mängden lagra pekare på strängarna som innehåller namnen. Klassen ser ut så här:

```
// Reference to covfefe intentially omitted
class NameList {
public:
    NameList() {}
    ~NameList() {}
    void insert(const std::string& name) {
        names.insert(new std::string(name));
    }
    void printSorted() const {
        for (list_type::const_iterator it = names.begin(); it != names.end(); ++it) {
            std::cout << *it << std::endl;
        }
     }
private:
    typedef std::set<std::string*> list_type;
    list_type names;
};
```

   a) Klassen innehåller en uppenbar minnesläcka. Förklara varför klassen läcker minne och visa hur klassen ska ändras för att korrigera felet.

   b) Utskriften i printSorted blir inte alls som förväntat – man får hexadecimala tal i stället för namn. Varför? Korrigera funktionen så att det skrivs ut namn i stället för tal.

   c) Även efter korrigeringen blir det fel – namnen skrivs inte ut i sorterad ordning, trots att ett set ju ska hålla elementen sorterade. Varför inte? Korrigera också detta fel.

4. Skriv en funktion `is_strict_monotonic` som avgör om en sekvens av element är strikt monoton. Med strikt monoton menar vi att antingen kommer elementen i en strikt växande ordning i sekvensen eller så kommer elementen i en strikt minskande ordning. Observera att detta också innebär att dubletter ej får förekomma i en strikt monoton sekvens. Funktionen ska kunna arbeta med alla sekvenser av element som man kan iterera över med hjälp av en iterator och där elementen kan jämföras med varandra parvis med hjälp av operatorn < (mindre än). Funktionen ska returnera `true` om sekvensen är strikt monoton och `false` annars. Du kan förutsätta att sekvensen innehåller åtminstone två element.

   Funktionen ska fungera enligt följande exempel (om man skriver ut ett värde av typen `bool` skrivs antingen 1 eller 0 ut beroende på om värdet var `true` eller `false`):

```
// Prints 1 (true)
int a[] = {1, 2, 3, 4};
std::cout << is_strict_monotonic(a,a+4) << std::endl;

// Prints 1 (true)
int b[] = {4, 3, 2, 1};
std::cout << is_strict_monotonic(b,b+4) << std::endl;

// Prints 0 (false)
int c[] = {1, 2, 7, 3, 4};
std::cout << is_strict_monotonic(c,c+5) << std::endl;

// Prints 0 (false)
std::vector<std::string> v = {"Despite", "the", "constant", "negative", "press",
                              "covfefe"};
std::cout << is_strict_monotonic(v.begin(),v.end()) << std::endl;

// Prints 1 (true)
sort(v.begin(),v.end());
std::cout << is_strict_monotonic(v.begin(),v.end()) << std::endl;
```

5. Skriv ett program som läser ett antal textfiler, vars namn ges på kommandoraden, och skriver ut de 10 vanligaste orden i filerna tillsammans med det totala antalet förekomster i alla filerna. Du får förutsätta att det bara finns bokstäver och ”whitespace” i filerna. Om en fil inte går att öppna ska man få en lämplig felutskrift.

   Exempel:

```
./count_words trump_tweets.txt dracula.txt
fake        9653
news        8965
the     8704
and     6272
to      5048
bad     4976
I       4792
of      4041
covfefe 3891
a       3456
```