LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Tentamen
C++-programmering

2013–08–24, 8.00–13.00

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 $10 + 12 + 28 = 50$ 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;
    cout << *p << 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) {
        cout << *p << " ";
    }
    cout << endl;
    string y[] = {"Mary", "had", "a", "little", "lamb"};
    for (Ptr<string> p = y; p != y + 5; ++p) {
        cout << *p << " ";
    }
    cout << endl;
}
```

2. I ett program läser man kommandon och heltal och summerar talen. Exempel på konversation med programmet (kommentarerna ingår inte; `p`, `a`, `u`, `c` och `r` är kommandon):

```
p       // print the sum, 0 to begin with
The sum is 0
a 4     // add 4
a 3     // add 3
a 2     // add 2
p
The sum is 9
u       // undo last add
u       // multiple undo's are allowed
p
The sum is 4
c       // commit changes (make them permanent so they cannot be undone)
u       // nothing happens
a 3
a 2
p
The sum is 9
r       // rollback, undo all changes since last commit
p
The sum is 4
```

Programmet ser ut så här:

```
int main() {
    Accumulator accum;
    char cmd;
    while (cin >> cmd) {
        switch (cmd) {
            case 'p':
                cout << "The sum is " << accum << endl; break;
            case 'a': {
                int nbr;
                cin >> nbr;
                accum += nbr;
                break;
            }
            case 'u':
                accum.undo(); break;
            case 'c':
                accum.commit(); break;
            case 'r':
                accum.rollback(); break;
        }
    }
}
```

Implementera klassen `Accumulator`.

3. I LaTeX finns ett stort antal kommandon som bestämmer hur det genererade dokumentet ska se ut. De här kommandona förekommer i denna uppgift:

`\section{text}`	Markerar början på ett nytt avsnitt med rubriken `text`. Avsnitten numreras automatiskt.
`\label{text}`	Definition av en ”etikett” i det aktuella avsnittet. Man kan referera till etiketter.
`\ref{text}`	Referens till en etikett. Byts mot numret på det avsnitt där etiketten finns.

Skriv ett program som går igenom en *.tex*-fil och 1) byter ut `\section`-kommandona mot numrerade rubriker, 2) definierar etiketter och tar bort `\label`-kommandon, 3) byter ut referenser mot avsnittsnummer. Programmet ska också göra en del kontroller. Exempel på infil (till vänster) och resultatfil (till höger):

```
\section{The first section}
This is the first line of
the text.\label{text} It
continues on this line.
And also on this\label{this}
line. References:
"text" found in section \ref{text}.
"lastline" found in section \ref{lastline}.
"start" found in section \ref{start}.

\section{Another section}
\label{text}More text in this
section. This is the
last\label{lastline} line.
```

```
1 The first section
This is the first line of
the text. It
continues on this line.
And also on this
line. References:
"text" found in section 1.
"lastline" found in section 2.
"start" found in section ??.

2 Another section
More text in this
section. This is the
last line.
```

Infilen innehåller två fel. När programmet körs ska felen rapporteras enligt följande:

```
Duplicate label 'text' at input line 12
Undefined reference 'start' at input line 9
```

Regler och förenklingar:

- Infilens namn ges som argument på kommandoraden. Du kan förutsätta att filen existerar och kan öppnas.
- Resultatfilen ska ha samma namn som infilen med tillägget *.res*.
- Referenser till odefinierade etiketter ska bytas mot `??` i stället för avsnittsnummer.
- Andra kommandon än de tre beskrivna ska ignoreras (skrivas ut oförändrade på resultatfilen).
- Det finns högst ett kommando per rad, och kommandon börjar och avslutas på samma rad.
- Allt är syntaktiskt korrekt: till exempel finns till varje `{` ett `}` på samma rad.
- Standardfunktionen `to_string(int)` (ny i C++11) formaterar ett heltal i så få positioner som möjligt.
- Programmet behöver inte vara effektivt.