Tentamen
EDAF30 Programmering i C++

2015–05–06, 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 även att du kan använda C++ standardklasser. Rena ”C-lösningar” på problem som med fördel kan lösas enligt mera objektorienterade principer kan därför ge poängavdrag, även om de är korrekta.

Uppgifterna ger preliminärt $12 + 12 + 12 + 14 = 50$ poäng. För godkänt krävs preliminärt 25 poäng (betygsgränser 3/25, 4/33, 5/42). Observera att ordningsföljden eller poängantalet för uppgifterna inte nödvändigtvis avspeglar deras svårighet.

---

1. Skriv en klass `Urand` men en operation `rnd()` som genererar slumptal i ett intervall `[0, N)`. Slumptalen ska dras *utan* återläggning, det vill säga man ska få varje tal i intervallet bara en gång. När man har fått alla tal i intervallet ska man börja om från början.

   Exempel på användning av klassen:

```
int main() {
        Urand urand(10, time(0)); // random numbers 0-9, current time is random seed
        for (int i = 0; i < 5; ++i) {
                for (int j = 0; j < 10; ++j) {
                        cout << urand.rnd() << " ";
                }
                cout << endl;
        }
}
```

   Programmet genererar fem rader utskrift. Varje rad innehåller talen 0–9 i slumpmässig ordning.

   Följande metod ska användas: lägg in talen `0-N` i en vektor och hämta tal ur vektorn efterhand. Blanda vektorn när det behövs. `rand()` ger ett slumpmässigt heltal i intervallet `[0, RAND_MAX)` (`RAND_MAX` är en fördefinierad konstant), `srand(unsigned int seed)` sätter slumptalsfröet.

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

Programmet ser ut så här:

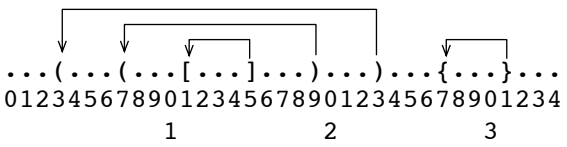
```
#include <iostream>
#include "accumulator.h"

using namespace std;

int main() {
    Accumulator accum;
    char cmd;
    while (cin >> cmd) {
        switch (cmd) {
            case 'p':
                cout << "Sum is now " << 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. Texteditorer som är avsedda för editering av programtext brukar hjälpa till med ”parentesmatchning”: när man skriver en högerparentes, `)`, `]` eller `}`, så markerar editorn motsvarande vänsterparentes. Exempel (`...` står för en godtycklig följd av tecken som inte innehåller några parenteser):



En texteditor beskrivs av följande klass:

```
class Editor {
public:
    Editor() {}
    std::string::size_type findMatchingLeftPar(std::string::size_type pos) const;
    ... // functions to edit the text (insert and delete characters)
private:
    std::string text;
};
```

Funktionen `findMatchingLeftPar` returnerar positionen för den vänsterparentes som motsvarar högerparentesen som finns i position `pos` i texten, eller `string::npos` om det inte finns någon sådan parentes.

Implementera funktionen. Observera att du inte behöver kontrollera att parenteser av annan typ innanför matchningen är korrekt balanserade.

4. I kursboken (J. Skansholm: C++ direkt) hittar man följande funktion som använder sig av binärsökning för att leta upp positionen (returnerar en pekare till elementet) för ett heltal i ett sorterat fält:

```
int *bin_sok(int sokt, int *forsta, int *sista) {
        if (forsta>sista)
                return 0;
        int *mitten = forsta + (sista-forsta)/2;
        if (sokt<*mitten)
                return bin_sok(sokt,forsta,mitten-1);
        else if (sokt>*mitten)
                return bin_sok(sokt,mitten+1,sista);
        else
                return mitten;
}
```

   a) Skriv om funktionen till en funktionsmall så att man i stället för att bara kunna leta i ett int-fält kan söka efter ett element i ett sorterat fält av godtycklig typ (med element som kan jämföras).
   b) Antag att vi vill använda din funktionsmall ovan för att söka efter element av klassen Bil enligt nedan. **OBS!** Sökningen ska ske baserat på *registreringsnumret* för bilen

```
class Bil {
public:
    Bil()  : reg_nr(0), owner(0) { }
    void set(const string& r, const string& o) {
        delete reg_nr;    // delete 0 is ok in C++ so no null check required...
        delete owner;
        reg_nr = new string(r);
        owner = new string(o);
    }
    string get_reg_nr() {
        return *reg_nr;
    }
    string get_owner() {
        return *owner;
    }
private:
    string *reg_nr;
    string *owner;
};
```

   Nedan finns ett litet exempel som visar hur det ska kunna fungera:

```
    Bil bil[4];
    bil[0].set("ABC123","Pelle");
    bil[1].set("BCD234","Lena");
    bil[2].set("CDE345","Oskar");
    bil[3].set("DEF456","Sofia");
    Bil s;
    s.set("BCD234","");
    Bil *b = bin_sok(s,bil,bil+3);
    if (b!=0) {
        cout << b->get_reg_nr() << endl << b->get_owner() << endl;
    } else {
        cout << "Fanns ej!" << endl;
    }
```

   Komplettera koden för klassen `Bil` (dvs ändra ej existerande kod - lägg bara till ny kod) så att koden i exemplet ovan fungerar!

vänd...

c) Ett separat problem med klassen `Bil` i den föregående uppgiften är att man riskerar att råka ut för en minnesläcka när man använder den. Vad innebär en minnesläcka?

d) *Komplettera* koden (ändra ej existerande kod) för klassen `Bil` ovan så att användning av den inte medför en minnesläcka. Var observant så att du inte i och med ändringen introducerar fel som beror på för tidig avallokering av minne.

e) Om du hade fått lov att ändra i den ursprungliga koden för klassen `Bil` skulle det gå att åtgärda problemen med minnesläckor på ett mycket enklare sätt (utan att ändra klassens beteende utåt – det publika gränssnittet). Hur?