Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2012-12-04

Arv och polymorfi

Mål

I den här laborationen skall du skapa ett objektorienterat program som använder arv, polymorfi och flera delar av det du tidigare lärt dig.

Läsanvisningar

- Arv och klasser (inheritance)
  - Polymorfi (polymorphism)
  - Referensmedlemsvariabler (&)
  - Konstanta medlemsvariabler (const)
  - Konstanta medlemsfunktioner (const)
  - (Statiska medlemsvariabler (static))
- Argument till main (kommandoradsargument)
  - C-strängar
  - Inbyggda arrayer (C-arrayer)
- Typkonverteringar (to_string, stod, stoi)
  - Fånga undantag från dessa (try, catch, exception)
- Filseparering
  - Implementationsfiler kompileras
  - Deklarationsfiler inkluderas (#include “my_header.h”)
  - Dubbelinkluderingsskydd (header guards)
    #ifndef MY_HEADER_H
    #define MY_HEADER_H
    #endif // MY_HEADER_H
- Repetition
  - “Allt” om klasser
  - Objektorientering
  - Standard Template Libray (STL)

Uppgift: Förenklad kretssimulator

Du skall skriva ett program som utför en förenklad simulering av en elektrisk krets. En (icke förenklad) elektrisk krets består av Resistorer, Kondensatorer, Spolar, Dioder, Transistorer, Spänningskällor och Strömkällor.
I våra förenklade kretsar förekommer Resistorer (som har en viss resistans), Kondensatorer (som har en viss kapacitans och en nuvarande laddning) och Batterier (som har en outtömlig laddning). Dessa komponenter är sammankopplade via kopplingspunkter. Varje komponent måste vara kopplad till två kopplingspunkter. Varje kopplingspunkt representeras av en viss laddning och kan vara kopplad till

oändligt många komponenter. Observera att det räcker att komponenterna vet hur de är kopplade. Kopplingspunkterna behöver bara veta "sin" laddning, inte vad som är anslutet.
Du skall skapa klasser för att representera de olika komponenterna och göra det möjligt att bygga upp olika "kretskort" i huvudprogrammet. Målet är att det skall vara relativt enkelt att bygga upp en ny krets med dina klasser. Därefter skall alla komponenter i kretsen simuleras i små tidsintervall.

Simulering av komponenter

Simuleringen går till så att varje komponent utför sitt arbete under ett litet tidsintervall, och detta upprepas under ett stort antal iterationer. Varje komponent har ett speciellt beteende i hur den överför laddningspartiklar från kopplingspunkt till kopplingspunkt.
OBS! I detta program frångår vi för enkelhets skull hur det fungerar i verkligheten på flera punkter. Istället gör vi enligt följande specifikation:

Ett batteri ser till att kopplingspunkten på plus-sidan får samma laddning som batteriet och kopplingen på minussidan får laddningen noll.

En resistor flyttar laddningspartiklar från sin mest positiva kopplingspunkt till sin minst positiva kopplingspunkt. Mängden flyttade partiklar beror på hur stor skillnaden är och hur stor resistansen är, samt hur lång tid som förflyter.

Exempel: Antag att resistansen är 2.0 Ohm och vi simulerar i steg om 0.1 tidsenheter. Om laddningen på sida a är 5.0 och laddnigen på sida b är 9.0 så är spänningen över resistorn $9.0-5.0=4.0$. Nu flyttas $4.0/2.0*0.1$ laddningspartiklar från sida b (som ju har högst laddning) till sida a.

En kondensator "suger upp" positiva laddningspartiklar från sin mest positiva sida och lagrar dessa internt. Samtidigt suger den upp lika många negativa laddningspartiklar från sin minst positiva sida (vilket i praktiken innebär att den lägger till positiva laddningspartiklar där). Mängden laddningspartiklar som sugs upp beror på hur stor skillnad det är mellan sidorna, hur mycket som tidigare sugits upp, och kapacitansen.

Exempel: Antag att kapacitancen är 0.8 Fahrad och vi simulerar i steg om 0.1 tidsenheter. Om laddningen på sida a är 5.0 och laddnigen på sida b är 9.0 så är skillnaden $9.0-5.0=4.0$. Om vi har 3.0 lagrat sedan tidigare så kommer vi nu lagra ytterligare $0.8*(4.0-3.0)*0.1$. Dessa tas från sidan med högst laddning och läggs till både internt och på andra sidan.

Spänningen över en komponent mäts genom att ta skillnaden på den största laddningsnivån och den minsta laddningsnivån.

Strömmen genom en resistor fås genom att dela spänningen över resistorn med dess resistans. Strömmen "genom" en kondensator approximerar vi med $C(V-L)$ där C är kapacitancen, V är spänningen över kondensatorn och L dess laddning. Ett batteri låtsas vi alltid har strömmen noll.

OBS! Det är förmodligen bra att gå igenom de klasser ni tänker er och hur simuleringen skall gå till med assistent eller mail till examinator för att få till en bra design som fungerar.

För kretsar med enbart ett batteri, resistorer och kondensatorer skall spänningen över varje komponent stabiliseras efter många iterationer. Hur många iterationer som krävs beror på storlek av komponenternas värden och hur stort varje tidsintervall är. Det stabila värdet skall stämma med det resultat som kan räknas fram enligt elläran. Beroende på hur man kopplat positiva och negativa sidan av komponenter kan det tänkas skilja på tecken.
Huvudprogrammet skall från kommandoraden ta in:

- hur många iterationer som skall simuleras
- hur många iterationer (rader) som skall skrivas ut
- hur stort tidsintervall som simuleras i varje steg
- spänningen på batteriet

T.ex. 1000000 iterationer i steg om 0.01 tidsenheter med spänningen 10 Volt och 10 utskrifter (d.v.s. 100000 iterationer mellan varje utskrift).
KRAV: Om något kommandoradsargument saknas eller är ogiltigt skall programmet avsluta med ett instruktivt felmeddelande. Ogiltiga värden skall detekteras genom att fånga de undantag som slängs av std::stoi, eller fånga egendefinerade undantag som slängs inifrån egna underfunktioner för felkontroll.

I huvudprogrammet skall du bygga upp tre olika kretsar enligt nedan. Klassdesignen skall vara sådan att detta är enkelt. Dessa kretsar skall simuleras och spänningen över varje komponent skrivas ut i en tabell med så många rader som efterfrågats. Resultatet skall stämma överens med exemplen. OBS! Vi bryr oss inte om mycket små avvikelser, rådgör med assistenten om du är osäker.
Ett exempel på hur en krets med två parallellkopplade resistorer kan byggas upp och simuleras:

```
Connection p, n;
vector<Component*> net;
net.push_back(new Battery("Bat", 24.0, p, n));
net.push_back(new Resistor("R1",  6.0, p, n));
net.push_back(new Resistor("R2",  8.0, p, n));
simulate(net, 10000, 10, 0.1);
```

Kretsexempel

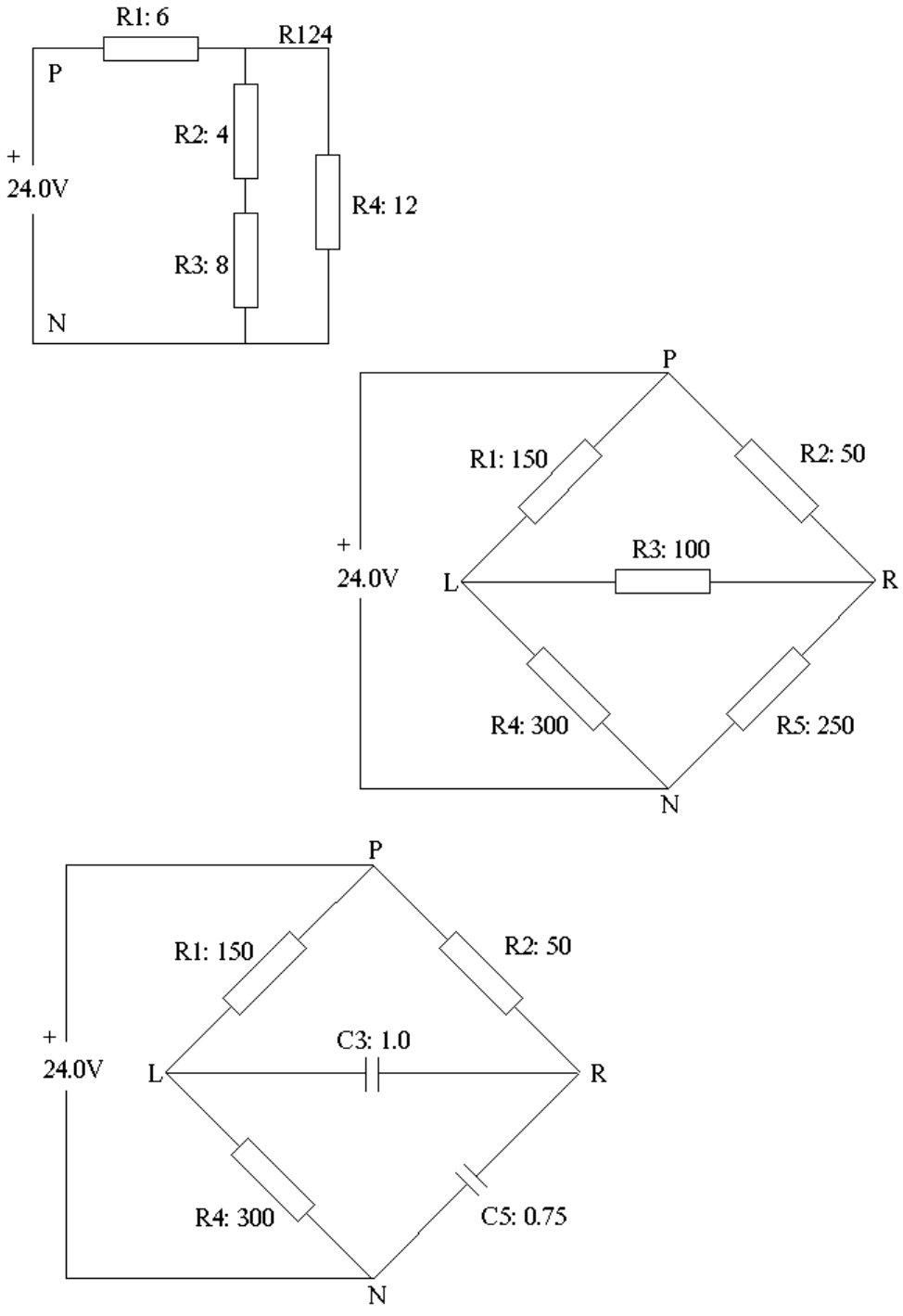
Den översta kretsen i figur 1 har 4 resistorer och ett batteri som är sammankopplade i kopplingspunkterna P, N, R124 och R23.
Nästa krets har 5 resistorer som är ganska besvärliga att räkna ut spänningen över “för hand”, faktiskt tror jag det är snabbare att skriva programmet. Vi kan se att R1 går mellan P och L, R3 mellan R och L och så vidare.
Sista kretsen har även några kondensatorer. För denna krets är det intressant hur den beter sig när spänningen ändras, och att rita grafer över detta, dock fungerar inte vår enkla simulering bra för detta och C++ saknar enkelt grafritningsverktyg. Att skapa växelströmkällor eller transistorer lämnas därför till den intresserade. Vi tittar bara att slutvärdena då kondensatorerna är fullt laddade är rimliga.

```
./a.out  200000 10 0.01 24

         Bat           R1           R2           R3           R4
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00
 24.00  0.00 11.99  2.00  3.99  1.00  7.98  1.00 11.97  1.00

         Bat           R1           R2           R3           R4           R5
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00  7.47  0.05  5.28  0.11  2.19  0.02 16.52  0.06 18.72  0.07
 24.00  0.00  6.40  0.04  4.72  0.09  1.68  0.02 17.60  0.06 19.28  0.08
 24.00  0.00  6.35  0.04  4.69  0.09  1.66  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08
 24.00  0.00  6.34  0.04  4.69  0.09  1.65  0.02 17.65  0.06 19.31  0.08

         Bat           R1           R2           C3           R4           C5
  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr  Volt  Curr
 24.00  0.00  8.50  0.06  4.10  0.08  4.40  0.01 15.50  0.05 19.90  0.03
 24.00  0.00  7.31  0.05  0.94  0.02  6.38  0.01 16.69  0.06 23.06  0.01
 24.00  0.00  7.57  0.05  0.30  0.01  7.27  0.00 16.43  0.05 23.70  0.00
 24.00  0.00  7.79  0.05  0.12  0.00  7.67  0.00 16.21  0.05 23.88  0.00
 24.00  0.00  7.90  0.05  0.05  0.00  7.85  0.00 16.10  0.05 23.95  0.00
 24.00  0.00  7.96  0.05  0.02  0.00  7.93  0.00 16.04  0.05 23.98  0.00
 24.00  0.00  7.98  0.05  0.01  0.00  7.97  0.00 16.02  0.05 23.99  0.00
 24.00  0.00  7.99  0.05  0.00  0.00  7.99  0.00 16.01  0.05 23.99  0.00
 24.00  0.00  8.00  0.05  0.00  0.00  7.99  0.00 16.00  0.05 24.00  0.00
 24.00  0.00  8.00  0.05  0.00  0.00  8.00  0.00 16.00  0.05 24.00  0.00
```



Figur 1: Tre exempelkretsar