

Laboration 2

Menysystem

Inledning

Syftet med denna laboration är att visa hur virtuella funktioner fungerar i C++ och hur objekt kan samarbeta via abstrakta gränssnitt.

Förberedelser

Lägg dina filer i en separat katalog och se till att den är lässkyddad för alla utom dina två laborationskamerater. Om du vill att en assistent ska kunna titta på dina filer, ge AFS-gruppen `oopk_e00:assistenter` läsrättigheter i katalogen.

Ladda alltid kursmodulen innan du börjar arbeta:

```
module load oopk_e00
```

Kopiera en makefile från kurskatalogen:

```
cp $OOPKHOME/Makefile .
```

och redigera filen. Då kommer kursbiblioteket `oopk` automatiskt med vid kompileringen.

Uppgiften

Du ska skriva några klasser som gör det lätt att skapa menyer i menyorienterade program. En meny kan se ut på följande sätt på skärmen:

```
--- Huvudmeny ---
1. Läs indata.
2. Skriv utdata.
3. Bearbeta data.
4. Avsluta program.
Välj alternativ:
```

Först visas menyens namn (här "Huvudmeny"), därefter ett antal numrerade alternativ. När användaren matar in en siffra efter "Välj alternativ:" och trycker RETURN ska motsvarande operation utföras. Detta upprepas tills alternativet "Avsluta program" väljs.

En sådan meny ska i ditt C++-program representeras av en instans av en klass `Menu`. Alternativerna i menyn ska representeras av instanser av subclasser till en abstrakt basclass `MenuItem`. `MenuItem`-objekten ska dels hålla reda på vilken text som ska skrivas ut på motsvarande rad i menyn (t.ex. "Läs indata"), dels utföra rätt operation när användaren väljer motsvarande siffra i menyn. För varje ny typ av operation kan man då definiera en ny subclass till `MenuItem`, instansiera ett objekt av denna klass, och associera objektet med menyn. Självklart ska man inte behöva ändra i `Menu`-koden; den ska skrivas en gång för alla. Menyobjektet ska automatiskt numrera alternativen. Menyn ovan ska kunna skapas på följande sätt i `main`:

```
int main() {
    Menu menu("Huvudmeny");
    ReadItem read_item(menu, "Läs indata.");
    WriteItem write_item(menu, "Skriv utdata.");
    ProcessItem process_item(menu, "Bearbeta data.");
    QuitItem quit_item(menu, "Avsluta program.");

    menu.prompt();
    return 0;
}
```

(Denna kod kunde lika gärna definiera en pop-up-meny i ett fönsterorienterat gränssnitt, men för enkelhetens skull antar vi här att `Menu` bara arbetar med standard input och standard output.)

`ReadItem`, `WriteItem`, `ProcessItem` och `QuitItem` är alla subclasser till `MenuItem` som utför olika saker då de aktiveras från `menu` (låt dem skriva ut lite olika text på skärmen för att simulera detta). `Menu::prompt()` ska skriva ut menyn och vänta på inmatning, beordra vald `MenuItem` att utföra opera-

tionen genom att anropa en funktion (säg, `MenuItem::invoke()`), skriva ut menyn på nytt etc.

Instanser av subklassen `QuitItem` måste ha ett sätt att tala om för menu att det är dags att returnera från `Menu::prompt`. De kan de göra t.ex. genom att returnera ett speciellt värde till menu, eller anropa en funktion, säg, `Menu::quit` i menu. Exakt hur det går till kommer inte att synas i main, så det får du bestämma själv.

Om menyalternativen är många vill man antagligen införa undermenyer. T.ex. kanske man vill flytta ned alternativen "Läs indata" och "Skriv utdata" till en undermeny "Filhantering". Definiera en subklass `SubmenuItem` som knyter ihop en `Menu` med en annan:

```
int main() {
    Menu submeny("Filhanteringsmeny");
    ReadItem read_item(submeny, "Läs indata.");
    WriteItem write_item(submeny, "Skriv utdata.");
    QuitItem return_item(submeny, "Återvänd till huvudmenyn.");

    Menu menu("Huvudmeny");
    SubmenuItem submenu_item(menu, "Filhantering...", submeny);
    ProcessItem process_item(menu, "Bearbeta data.");
    QuitItem quit_item(menu, "Avsluta program.");

    menu.prompt();
    return 0;
}
```

På skärmen ska det då se ut på följande sätt. Notera att `QuitItem` får undermenyn att återvända till närmast högra menynivå. (alv> är promptern)

```
alv> ./a.out
--- Huvudmeny ---
1. Filhantering...
2. Bearbeta data.
3. Avsluta program.
Välj alternativ: 1
--- Filhanteringsmeny ---
1. Läs indata.
2. Skriv utdata.
3. Återvänd till huvudmenyn.
Välj alternativ: 3
--- Huvudmeny ---
1. Filhantering...
2. Bearbeta data.
3. Avsluta program.
Välj alternativ: 3
alv>
```

Tips

I kursbiblioteket finns två funktioner som omvandlar mellan strängar och heltal:

```
bool string2int(const string& str, int& i);
string int2string(int i);
```

Dessa är deklarerade i filen `<oopk/convert.hh>` som du alltså måste inkludera först. `string2int` returnerar `true` om `str` kan tolkas som ett heltal. I så fall läggs resultatet i parametern `i`. Annars returneras `false`, och `i` modifieras inte. `int2string` kan inte misslyckas, så den returnerar strängen direkt.