Tentamen
C++-programmering

2016–03–19, 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 $16 + 6 + 2 + 18 + 8 = 50$ poäng. För godkänt krävs 25 poäng (3/25, 4/33, 5/42).

---

1. Betrakta följande klass:

```
class A {
public:
    A(size_t sz = 10) : storage(new int[size = sz]) {}
    int operator[](size_t pos) const { return storage[pos]; }
    int& operator[](size_t pos) { return storage[pos]; }
private:
    int* storage; size_t size;
};
```

   a) När man använder klassen i nedanstående funktioner får man ett kompileringsfel. Var, och hur rättar man lämpligen felet? Svara med den kod som saknas samt ange var den ska placeras i programkoden.

```
void print(const string& msg, A a) {
    cout << msg << " " << a << endl;
}
void f() {
    A a1;
    for (size_t i = 0; i != 10; ++i) { a1[i] = i + 1; }
    print("This is a1: ", a1);
    A a2;
    a2 = a1;
    print("This is a2: ", a2);
}
```

   b) När felet ovan har rättats blir det en minnesläcka när man kör programmet. Vilka ytterligare konstruktioner behövs i klassen för att programmet ska fungera korrekt ur minneshanterings-synpunkt, och varför behövs de? Implementera de saknade konstruktionerna och skriv för var och en av dem en kort motivering till varför de behövs.

   c) Före C++11 brukade man tala om en tumregel man kallade ”rule of three". Numera har denna tumregel utvecklats och kallas oftast i stället för ”rule of five”. Vad säger tumregeln ”rule of five” och vad är skälet till att man infört utökningen?

   d) Man brukar rekommendera att destruktorer i C++ ska vara virtuella. Varför? Svara genom att ge ett exempel där det annars blir fel.

2. Betrakta följande klasser och funktioner:

```
struct Foo{
    virtual void print() const {std::cout << "Foo" << std::endl;}
};
struct Bar : Foo {
    virtual void print() const override {std::cout << "Bar" << std::endl;}
};
struct Qux : Bar {
    virtual void print() const override {std::cout << "Qux" << std::endl;}
};
void print1(const Foo* f)
{
    f->print();
}
void print2(const Foo& f)
{
    f.print();
}
void print3(Foo f)
{
    f.print();
}

int main() {
    Foo* a =  new Bar;
    Bar& b = *new Qux;
    Bar  c = *new Qux;

    print1(a);
    print1(&b);
    print1(&c);
    std::cout << std::endl;

    print2(*a);
    print2(b);
    print2(c);
    std::cout << std::endl;

    print3(*a);
    print3(b);
    print3(c);
    std::cout << std::endl;
}
```

Programmet går att kompilera och köra utan exekveringsfel.
Vad skrivs ut när `main()` exekveras?

3. I ett C++-program förekommer det ibland något som kallas för ”virtuellt arv” (virtual inheritance). Förklara vad virtuellt arv är och varför det finns i språket. Visa också hur man i C++ specificerar att en klass ska ärva virtuellt.

4. 1-wire är en s.k. fältbusstandard framtagen av Dallas Semiconductor för att från en dator kommunicera med enkla, billiga enheter som t.ex. temperaturgivare och liknande. Antingen kan ett 1-wire-nätverk anslutas direkt till en dator med en speciell adapter eller också kan man använda en s.k. 1-wire-till-ethernet-brygga via vilken man kan kommunicera med 1-wire-enheterna över ett vanligt ethernetnätverk. Ett exempel på en sådan brygga är OW-SERVER från Embedded Data Systems[1]. Denna brygga skannar kontinuerligt av 1-wire-nätverket efter anslutna sensorer, läser av deras aktuella mätvärden och sparar dem i sitt minne. Via en inbyggd webbserver kan datorer när som helst ladda ner de senast avlästa värdena i form av en XML-fil.

   Appendix A (se slutet av tentamen) visar hur XML-filen kan se ut för en brygga med fyra anslutna temperaturgivare. För varje ansluten temperatursensor finns det en sektion som inleds med ”<owd_XXXXXX” där ”XXXXXX” är sensorns modellbeteckning och avslutas med ett motsvarande ”</owd_XXXXXX>". Däremellan är vi intresserade av två fält. Det första är sensorns unika identifierare. Den hittar man mellan XML-taggarna ”<ROMId>” och ”</ROMId>”. Det andra fältet är sensorns nuvarande värde (temperatur i °C). Det hittar man mellan XML-taggarna ”<Temperature Units=”Centigrade”>” och ”</Temperature>”. Den första sensorn i exemplet har alltså identiteten ”630000041C430228” och anger 5,1875°C som aktuellt värde.

   Din uppgift är att implementera en klass `Sensors` med vars hjälp man kan läsa in temperaturmätvärden från en eller flera bryggor (typiskt placerade i olika byggnader) av typen OW-SERVER och kontinuerligt lagra och presentera de senaste mätvärden från alla anslutna sensorer.

   Klassen `Sensors` ska ha följande publika gränssnitt:

```
class Sensors {
public:
    // Constructor
    Sensors();

    // Connects to the OW-SERVER at the address given by url. Reads the current
    // XML data and stores information about the current temperature values.
    // Any previous values for the sensors in the XML file are overwritten.
    // Does not throw any exceptions. Any network errors are ignored (and no
    // data is read).
    void update(std::string url);

    // Returns the latest temperature read from the sensor given by id. Throws a
    // runtime_exception if there is no data available for the given sensor.
    double getTemp(const std::string& id) const;

    // Prints a list of all sensors (id and latest temperature value) sorted by
    // sensor id.
    void print() const;
};
```

   Ett enkelt testprogram för ovanstående klass skulle kunna se ut så här:

```
int main() {
        Sensors sensors;
        sensors.update("http://owserver1.lth.se/detail.xml");
        sensors.update("http://owserver2.lth.se/detail.xml");
        sensors.update("http://nonexistentserver.lth.se/detail.xml");
        std::cout << sensors.getTemp("0B000801848EAA10") << std::endl;
        sensors.print();
        std::cout << sensors.getTemp("XXXXXXXXXXXXXXXX") << std::endl;
}
```

   Motsvarande utdata skulle kunna se ut så här (se nästa sida):

[1] http://www.embeddeddatasystems.com/OW-SERVER-1-Wire-to-Ethernet-Server-Revision-2_p_152.html

```
22.5
06000006AE3F4528 20.1875
0B000801848EAA10 22.5
0C000BE765C89788 5.5
460008018487C610 20.5
480000067BC58721 19.5
630000041C430228 5.1875
libc++abi.dylib: terminating with uncaught exception of type std::runtime_error:
Unknown sensor.
Abort trap: 6
```

Till din hjälp har du en klass `URLstream` som är en subklass till `istream` och med vars hjälp du kan koppla upp en förbindelse till en OW-SERVER och läsa in innehållet från XML-filen. Denna klass ska du *INTE* implementera.

```
// Opens a network connection to a document at a web server. Once open, the
// contents of the document can be read using standard stream operations.
class URLstream : public std::istream {
public:
    // Creates a connection to the web document referenced by url.
    // Throws std::ios_base::failure on network error.
    URLstream(std::string url);
};
```

Använd standardbibliotekets datastrukturer, standardklasser och algoritmer när så är tillämpligt.

5. STL-funktionen `remove_copy_if` kopierar element från ett område till ett annat område, förutom de element som uppfyller ett givet villkor. I följande program kopieras de tal i vektorn `a` som är större än eller lika med ett tal som läses in från standard input (`cin`) till vektorn `b` och de kopierade talen skrivs ut:

```
int c = 0;

bool ltc(int x) { return x < c; }

int main() {
    int a[] = { 5, 10, -3, 2, -8, 11 };
    const size_t SIZE = sizeof(a) / sizeof(int);
    int b[SIZE];
    cin >> c;
    int* last = remove_copy_if(a, a + SIZE, b, ltc);
    int* first = b;
    while (first != last) { // prints 5 10 2 11
        cout << *first++ << endl;
    }
}
```

   a) Tyvärr kräver lösningen att värdet på det inlästa talet lagras i en global variabel (`c`). Man kan klara sig utan funktionen `ltc` och den globala variabeln om man använder sig av en s.k. lambda-funktion i stället. Hur ser anropet av `remove_copy_if` ut om man gör det (vi förutsätter att variabeln `c` nu är en lokal variabel i `main`)?

   b) Om allt man vill göra är att skriva ut talen som är större än eller lika med det inlästa talet är det onödigt att först kopiera dem till vektorn `b`. Man kan genom att modifiera anropet av `remove_copy_if` få talen utskrivna i stället för kopierade. Hur ser anropet av `remove_copy_if` ut i detta fall?

   c) Implementera funktionen `remove_copy_if`. Funktionen ska naturligtvis klara av att elementen som kopieras är av godtycklig typ och att de hämtas från en `vector` eller `deque` eller någon annan STL-behållare. (Det vill säga, de tre första parametrarna ska vara iteratorer.)

Appendix A: OW-SERVER XML exempel


```
<?xml version="1.0" encoding="UTF-8"?>
<Devices-Detail-Response xmlns="http://www.embeddeddatasystems.com/schema/owserver"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<PollCount>56610</PollCount>
<DevicesConnected>4</DevicesConnected>
<LoopTime>3.951</LoopTime>
<DevicesConnectedChannel1>1</DevicesConnectedChannel1>
<DevicesConnectedChannel2>2</DevicesConnectedChannel2>
<DevicesConnectedChannel3>1</DevicesConnectedChannel3>
<DataErrorsChannel1>0</DataErrorsChannel1>
<DataErrorsChannel2>2</DataErrorsChannel2>
<DataErrorsChannel3>1</DataErrorsChannel3>
<VoltageChannel1>4.60</VoltageChannel1>
<VoltageChannel2>4.59</VoltageChannel2>
<VoltageChannel3>4.60</VoltageChannel3>
<VoltagePower>4.83</VoltagePower>
<DeviceName>OWServer_v2-Enet</DeviceName>
<HostName>EDSOWSERVER2</HostName>
<MACAddress>00:05:3A:35:9C:4E</MACAddress>
<owd_DS18B20 Description="Programmable resolution thermometer">
<Name>DS18B20</Name>
<Family>28</Family>
<ROMId>630000041C430228</ROMId>
<Health>7</Health>
<Channel>1</Channel>
<RawData>53004B467FFF0D10D4000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000</RawData>
<PrimaryValue>5.1875 Deg C</PrimaryValue>
<Temperature Units="Centigrade">5.1875</Temperature>
<UserByte1 Writable="True">75</UserByte1>
<UserByte2 Writable="True">70</UserByte2>
<Resolution>12</Resolution>
<PowerSource>0</PowerSource>
</owd_DS18B20>
<owd_DS18S20 Description="Parasite power thermometer">
<Name>DS18S20</Name>
<Family>10</Family>
<ROMId>460008018487C610</ROMId>
<Health>7</Health>
<Channel>2</Channel>
<RawData>29004B46FFFF0310C8000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000</RawData>
<PrimaryValue>20.5000 Deg C</PrimaryValue>
<Temperature Units="Centigrade">20.5000</Temperature>
<UserByte1 Writable="True">75</UserByte1>
<UserByte2 Writable="True">70</UserByte2>
</owd_DS18S20>
<owd_DS18B20 Description="Programmable resolution thermometer">
<Name>DS18B20</Name>
<Family>28</Family>
<ROMId>06000006AE3F4528</ROMId>
<Health>7</Health>
<Channel>2</Channel>
```

```
<RawData>43014B467FFF0D10BD0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000</RawData>
<PrimaryValue>20.1875 Deg C</PrimaryValue>
<Temperature Units="Centigrade">20.1875</Temperature>
<UserByte1 Writable="True">75</UserByte1>
<UserByte2 Writable="True">70</UserByte2>
<Resolution>12</Resolution>
<PowerSource>0</PowerSource>
</owd_DS18B20>
<owd_DS18S20 Description="Parasite power thermometer">
<Name>DS18S20</Name>
<Family>10</Family>
<ROMId>0B000801848EAA10</ROMId>
<Health>7</Health>
<Channel>3</Channel>
<RawData>2D004B46FFFF0610220000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000</RawData>
<PrimaryValue>22.5000 Deg C</PrimaryValue>
<Temperature Units="Centigrade">22.5000</Temperature>
<UserByte1 Writable="True">75</UserByte1>
<UserByte2 Writable="True">70</UserByte2>
</owd_DS18S20>
</Devices-Detail-Response>
```