Tentamen
EDAF30 Programmering i C++

2017–04–20, 14:00–19:00

Hjälpmedel: En valfri C++-bok. Andra papper med anteckningar eller utskrifter är *inte* tillåtna.

Du ska i dina lösningar visa att du behärskar C++ och även att du kan använda C++ standardklasser och algoritmer. Rena ”C-lösningar” på problem som med fördel kan lösas enligt mera objektorienterade principer, eller egen implementering av sådant som finns i standardbiblioteket, kan därför ge poängavdrag, även om de är korrekta. I bedömningen av en lösning kan även minneshantering beaktas där det är relevant. Om du använder saker ur standardbiblioteket i din kod, var tydlig med att du gör det. Använd fullt kvalificerade namn eller `using`-direktiv för de namn du använder.

Uppgifterna ger preliminärt $5+18+8+13+6=50$ 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. Vad skrivs ut när följande program exekveras? Motivera ditt svar med en beskrivning av varför programmet tolkas som det gör.

```
#include<iostream>
using std::cout;

class B {
public:
    int f(int i) {return i+1; }
};

class D : public B {
public:
    double f(double d) {return d+1.3; }
};

int main()
{
    D* pd = new D;

    cout << pd->f(2) << '\n';
    cout << pd->f(2.3) << '\n';

    delete pd;
}
```

Om du inte vet, kan du på denna uppgift i stället välja att svara “Jag vet inte.”. Detta svar (d v s ett svar med endast meningen “Jag vet inte.”) ger två poäng.

---

2. I den nya C++-standarden, C++11, finns en klassmall `std::array` som är en ”wrapper” runt en inbygggd array. Klassen fungerar nästan som en ”riktig” standard-container, med iteratorer, överlagrad indexeringsoperator och `size`-funktion. Men det är fortfarande inte någon dynamisk struktur: den har fix storlek, så det finns inte `push_back`, `resize` eller liknande funktioner.

   En sådan klassmall kan (delvis) implementeras enligt följande exempel:

```
template <typename T, size_t N>
class array {
public:
    using size_type = size_t;
    T& operator[](size_type i) {return arr[i];}
    size_type size() const {return N;}
private:
    T arr[N];
};
```

   och användas som i detta huvudprogram

```
int main() {
    array<int, 3> a; // <element type, size>
    a[0] = 1;
    a[1] = 4;
    a[2] = 9;
    for (array<int, 3>::size_type i = 0; i != a.size(); ++i) {
        cout << a[i] << endl;
    }
}
```

   vilket ger utskriften

```
4
9
```

   a) Man vill i stället för att skriva ut arrayen direkt i `for`-satsen ovan använda en funktionsmall

```
template <typename Cont>
void print1(const Cont& c) {
    for (typename Cont::size_type i = 0; i != c.size(); ++i) {
        cout << c[i] << endl;
    }
}
```

   så att huvudprogrammet kan bli

```
int main()
{
    array<int, 3> a; // <element type, size>
    a[0] = 1;
    a[1] = 4;
    a[2] = 9;

    print1(a);
}
```

   När man försöker kompilera detta program får man kompileringsfelet

```
In instantiation of 'void print1(const Cont&) [with Cont = array<int, 3ul>]':
error: passing 'const array<int, 3ul>' as 'this' argument discards qualifiers [-fpermissive]
note: in call to 'T& array<T, N>::operator[](array<T, N>::size_type)
[with T = int; long unsigned int N = 3ul; array<T, N>::size_type = long unsigned int]'
```

   Vad beror felet på? Korrigera *klassmallen* `array` så att huvudprogrammet med `print1` fungerar. *Svara med en förklaring av problemet och kod som visar hur* `array` *ska ändras.*

b) Lägg till funktioner så att också följande utskriftsmetod fungerar (d v s att man kan byta ut anropet av `print1` mot ett anrop av nedanstående `print2` i ovanstående huvudprogram):

```
template <typename Cont>
void print2(const Cont& c) {
    for (typename Cont::const_iterator it = c.begin(); it != c.end(); ++it) {
        cout << *it << endl;
    }
}
```

c) Kan man, med din klassmall `array` (d v s inklusive dina svar till uppgift a och b), ersätta anropen av `print1` respektive `print2` med en *range for*-loop? D v s fungerar följande program?

```
int main()
{
    array<int, 3> a; // <element type, size>
    a[0] = 1;
    a[1] = 4;
    a[2] = 9;

    for(auto x : a) {
        cout << x << endl;
    }
```

Om ja, förklara varför (d v s ange vilka funktioner som anropas, motivera att typerna stämmer överens, etc.). Om nej, lägg till det som krävs för att programmet ska fungera.

d) Att först konstruera en array och därefter kopiera in elementen är onödigt klumpigt. Komplettera klassmallen så att man kan ge en array värden när man skapar den. Initieringen behöver även kontrollera att antalet element stämmer. Om inte ska konstruktorn kasta `std::length_error`[1]. Följande program är ett exempel på hur det ska fungera:

```
void test_array_init_and_print()
{
    array<int,5> a{1,3,5,7,9};
    for(auto x : a) {
        cout << x << " ";
    }
    cout << endl;

    array<int,3> b{1,2,3,4,5};
    for(auto x : b) {
        cout << x << " ";
    }
    cout << endl;
}
```

Utmatningen ska vara

```
1 3 5 7 9
terminate called after throwing an instance of 'std::length_error'
  what():  size mismatch in initialization
```

Efter ändringen ska alla ovanstående exempelprogram fungera (d v s ändringarna får inte göra att något av `main`-funktionerna från tidigare i uppgiften slutar fungera).

[1] Notera att `std::array` är implementerad så att man får ett kompileringsfel om längden inte stämmer, men i denna uppgiften ska exception användas.

3. När man försöker kompilera följande program

```
 1  #include <iostream>
 2  using std::cout;
 3  using std::cin;
 4  using std::endl;
 5
 6  struct A {
 7      A(int x) {val = x;}
 8      void print() {cout << "A("<<val<<")\n";}
 9      int val;
10  };
11
12
13  struct B {
14      B(int x) {a=A(x);}
15      void print() {cout << "B("; a.print(); cout << ")\n";}
16      A a;
17  };
18
19  int main()
20  {
21      B b(10);
22      b.print();
23  }
```

får man kompileringsfelet

```
testprogram.cpp|14 col 14 error| no matching function for call to 'A::A()'
|| B(int x) {a=A(x);}
|| ^
testprogram.cpp|7 col 5 info| candidate: A::A(int)
|| A(int x) {val = x;}
|| ^
testprogram.cpp|7 col 5 info| candidate expects 1 argument, 0 provided
testprogram.cpp|6 col 8 info| candidate: constexpr A::A(const A&)
|| struct A {
|| ^
testprogram.cpp|6 col 8 info| candidate expects 1 argument, 0 provided
testprogram.cpp|6 col 8 info| candidate: constexpr A::A(A&&)
testprogram.cpp|6 col 8 info| candidate expects 1 argument, 0 provided
```

a) Förklara varför kompileringsfelet uppstår

b) Visa hur man kan ändra *klassen* `A` för att åtgärda felet.

c) Visa hur man kan ändra *klassen* `B` för att åtgärda felet.

4. Standard-algoritmen `partition` har följande beskrivning:

```
template <typename Iterator, typename UnaryPredicate>
    // Move all elements in [first, last) to the front for which
    // op(elem) yields true. Return the first position for which
    // op yields false. The time complexity is linear.
Iterator partition(Iterator first, Iterator last, UnaryPredicate op);
```

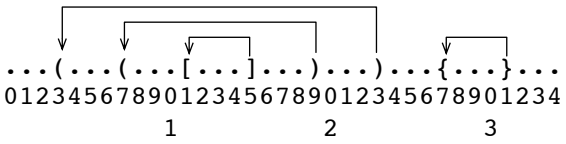
   Exempel: partitionering av sekvensen `{1,2,3,4,5,6}` med villkoret ”jämnt tal” ska ge resultatet `{2,4,6,1,5,3}` (den inbördes ordningen mellan de tre första talen är godtycklig, liksom mellan de tre sista).

   a) Skriv ett fullständigt program som läser in ett antal heltal (avslutade med end-of-file), lagrar talen i en `std::vector`, partitionerar vektorn så att alla jämna tal hamnar först och till slut skriver ut vektorn. Använd standard-algoritmer så mycket du kan (du måste använda `partition`). Du behöver inte formatera utskriften som `{1,2,3,...}`, det är tillräckligt att göra en enkel utskrift, t ex att skriva varje element på en egen rad.

   b) Utöka ditt program så att båda delarna av vektorn sorteras i fallande ordning. Exemplet ovan ska alltså ge resultatet `{6,4,2,5,3,1}`

   c) Implementera funktionsmallen `partition`.

5. 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):



   Givet en klass `Editor`,

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

   implementera funktionen `Editor::find_left_par`: givet en poisition i texten ska positionen för en matchande parentes returneras, om sådan finns, eller `string::npos` om ingen match finns. Du behöver inte kontrollera att parenteser av annan typ innanför matchningen är korrekt balanserade.