Tentamen
EDAF30 Programmering i C++

2016–05–13, 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 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 $5 + 14 + 9 + 5 + 10 + 7 = 50$ 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. Följande program kan kompileras utan fel.

```
#include<iostream>


int main()
{
    for(unsigned u = 10; u >= 0; --u) {
        std::cout << u << std::endl;
    }

    return 0;
}
```

   Vad händer när programmet exekveras?
   Svara med vad som skrivs ut och/eller förklara vad som händer.

2. I en bok är ett sakregister (ett index) en sorterad lista med viktiga ord med hänvisningar till de sidor där ordet förklaras eller används. Du ska skriva ett program som producerar något som liknar ett sakregister för en text.

   - Texten ska läsas från standard input, registret ska skrivas på standard output.
   - *Alla* orden i texten ska vara med i registret. Hänvisningarna ska vara till *rader* i indata. Om ett ord förekommer flera gånger på samma rad ska det bara finnas en hänvisning till raden.
   - Ord är följder av tecken åtskilda med whitespace. Skiljetecknen `,.;:!?()` ska tas bort ur orden.
   - Orden ska skrivas ut med ett ord per rad följt av radhänvisningarna i växande ordning. Små och stora bokstäver betraktas som olika, men orden ska sorteras oberoende av skiftläge.

   Du ska implementera funktionerna `create_index` och `print_index` samt ett huvudprogram enligt ovan. Dessa ska använda följande givna kod:

```
#include<iostream>
#include<map>
#include<set>
#include<algorithm>

using std::string;

struct Compare {
  bool operator()(const string& s1, const string& s2) const {
    string::size_type sz = std::min(s1.size(), s2.size());
    for(string::size_type i=0; i != sz; ++i) {
      char c1 = tolower(s1[i]);
      char c2 = tolower(s2[i]);
      if (c1 < c2)
        return true;
      else if (c2 < c1)
        return false;
    }
    return s1 < s2;
  }
};

typedef std::map<std::string, std::set<size_t>, Compare> index_t;

index_t create_index(std::istream& in);
void print_index(const index_t& index, std::ostream& out);
```

   Ledning: För att ta bort vissa tecken från en sträng kan man använda algoritmen `std::remove_if` med följande predikat:

```
struct Is_char_from{
    Is_char_from(std::string s) :chars(s) {}
    bool operator()(char c) { return chars.find(c) != string::npos;}
private:
    std::string chars;
};
```

   Notera att `std::remove_if` inte ändrar storleken på containern, utan bara flyttar de element som ska tas bort till slutet, och returnerar en `iterator` som pekar på *det första borttagna elementet* eller containerns `end`. För att faktiskt ta bort de borttagna elementen ur containern kan man använda medlemsfunktionen `erase`. Ett exempel med en `std::string s` och predikatet `my_predicate` är:

```
    s.erase(std::remove_if(s.begin(), s.end(), my_predicate), s.end());
```

uppgiften fortsätter på nästa sida...

Exempel på exekvering av huvudprogrammet: Följande indata (från en sång av Art Paul Schlosser):

```
Mary had a little lamb
Little lamb, little lamb
Mary had a little lamb and some olives on the side.
```

ger följande utskrift:

```
a 1 3
and 3
had 1 3
lamb 1 2 3
Little 2
little 1 2 3
Mary 1 3
olives 3
on 3
side 3
some 3
the 3
```

Observera att `Mary` kommer mellan `little` och `olives`, att både `Little` och `little` finns med och att inga skiljetecken finns med.

Du får lov att förutsätta att indata är normal text, du behöver inte hantera konstiga fall som t ex ord som består enbart av skiljetecken.

Svara med en implementation av funktionerna `create_index(std::istream& in)`, `print_index(const index_t&, std::ostream&)` och `main()`.

3. I C++ använder man ofta templates (mallar) för att skriva generella klasser och funktioner. Ett exempel på en generell funktion är att filtrera en uppsättning värden för att välja ut de som uppfyller ett givet *predikat*. Ett exempel på sådan filtrering är:

```
struct StrWithLen{
    size_t len;
    StrWithLen(size_t l) :len{l} {}
    bool operator()(const std::string& s) const { return s.length() == len;}
};

template <typename C>
void print_result(const C& c)
{
    std::cout << c.size() << " matches: ";
    for(auto x:c)
        std::cout << x << " ";
    std::cout << std::endl;
}


void use1()
{
    std::stringstream ss{"smurf nisse tomte sallad kakor"};

    auto strings = collect_matching(StrWithLen(5), ss);
    print_result(strings);
}
```

Detta program läser strängar från en ström och väljer ut de strängar som har längden 5. Programmet använder en funktions-mall `collect_matching` som tar ett *unärt predikat* och en ström som parametrar. Vid exekvering blir utskriften: `4 matches: smurf nisse tomte kakor`.

Ett annat exempel är att välja ut de strängar som börjar med bokstaven 's', med hjälp av ett predikat `has_initial_s`:

```
bool has_initial_s(const std::string& s)
{
    return s[0] == 's';
}
void use2()
{
    std::stringstream ss{"smurf nisse tomte sallad kakor"};

    auto starts = collect_matching(has_initial_s, ss);
    print_result(starts);
}
```

Vid exekvering blir utskriften: `2 matches: smurf sallad`

Implementera funktionsmallen `collect_matching` enligt ovan. Båda exemplen `use1` och `use2` ska fungera.

4. I vissa program använder man ett mönster, "*source/sink*", för resurshantering. Då allokeras objekt av en funktion, en *source*. Objekten skickas därefter vidare till en *sink* som behandlar och *övertar ägarskap* för objekten. Ett minimalt exempel av detta är följande program där `random_int` allokerar en `int` och returnerar en pekare till denna:

```
int* random_int()
{
    int* res=new int;

    *res = rand();
    return res;
}
```

Funktionen `consume_number` skriver ut talet och avallokerar det därefter för att undvika att läcka minne:

```
void consume_number(int* p)
{
    std::cout << "consuming " << *p;
    std::cout << std::endl;

    delete p;
}
```

Ett minimalt testprogram:

```
void test()
{
    int* i = random_int();
    consume_number(i);
}
```

En annan vanlig variant för resurhantering är att man låter *anropande funktion* göra allokeringen, och sen använda referens-anrop för att få en *ut-parameter*:

```
void random_int(int& out)
{
    out = rand();
}
```

Ett testprogram som använder denna tillsammans med `consume_number` från exemplet ovan:

```
void test2()
{
    int i;
    random_int(i);
    consume_number(&i);
}
```

I detta fallet så fungerar `test()` som avsett, men `test2()` ger exekveringsfel. Vilken är skillnaden som gör att `test2()` inte fungerar?

5. Betrakta följande program:

```
#include<iostream>
#include<iomanip>
#include<initializer_list>

template <typename T>
class Vektor{
public:
    Vektor(size_t sz) :size{sz},p{new T[size]{}} {}
    Vektor(std::initializer_list<T> l) :Vektor(l.size()) {assign(l);}
    ~Vektor() {delete p;}
    T& operator[](size_t i) {return p[i];}
    size_t length() const {return size;}
    T* begin() {return p;}
    T* end() {return p+size;}
    const T* cbegin() const {return p;}
    const T* cend() const {return p+size;}

private:
    size_t size;
    T* p;
    void assign(const std::initializer_list<T>& l);
};

template <typename T>
void Vektor<T>::assign(const std::initializer_list<T>& l)
{
    size_t idx{0};

    for(auto e : l) {
        p[idx++] = e;
    }
}

template <typename T>
void add(const Vektor<T> c1, const Vektor<T> c2, Vektor<T> c3)
{
    auto it1 = c1.cbegin();
    auto it2 = c2.cbegin();
    auto it3 = c3.begin();

    while( (it1 != c1.cend() || it2 != c2.cend()) && it3 != c3.end()){
        T tmp1{};
        if(it1 != c1.cend()){
            tmp1 += *it1;
            ++it1;
        }
        T tmp2{};
        if(it2 != c2.cend()){
            tmp2 += *it2;
            ++it2;
        }
        *it3 = tmp1 + tmp2;
        ++it3;
    }
}
```

programmet fortsätter på nästa sida...

```
int main()
{
    Vektor<int> v1{1,2,3,4,5,6};
    Vektor<int> v2{10,20,30,40};
    Vektor<int> v3(v1.length());

    add(v1,v2,v3);

    for(auto e: v3){
        std::cout << e << " ";
    }
    std::cout << std::endl;
}
```

När programmet körs fås utskriften

```
15523872 0 33 44 5 6
```

i stället för det förväntade:

```
11 22 33 44 5 6
```

a) Förklara vad som händer när programmet körs.

b) Kan man genom att ändra *enbart i funktionsmallen* `add` lösa problemet så att programmet ger det förväntade resultatet?
   Om ja, visa hur programmet behöver ändras. Om nej, motivera.

c) Kan man genom att ändra *enbart i klassmallen* `Vektor` lösa problemet så att programmet ger det förväntade resultatet?
   Om ja, visa hur klassen behöver modifieras. Om nej, motivera.

6. Följande program är tänkt att implementera en klass `ptr` som uppför sig som en `int*`, men med lite extra bekvämlighetsfunktioner. Tyvärr ger det ett kompileringsfel i funktionen `use()`:

```
 1  #include<iostream>
 2
 3  class ptr{
 4  public:
 5      ptr() =default;
 6      ptr(int& i) :p{&i} {}
 7      ~ptr() {}
 8      int operator*() {return *p;}
 9      ptr& operator=(int* pi) {p=pi; return *this;}
10      operator int*() {return p;}
11  private:
12      int* p;
13  };
14
15  void print_int(int i)
16  {
17      std::cout << i << std::endl;
18  }
19
20  void use()
21  {
22      int i{17};
23      ptr p{i};
24
25      print_int(*p);
26
27      int j;
28      p = ptr{j};
29      *p = 42;
30
31      print_int(*p);
32
33      p = new int{8};
34      print_int(*p);
35      delete p;
36  }
37
38  int main()
39  {
40      use();
41  }
```

a) På vilken rad fås ett kompileringsfel, och vad är felet?

b) Korrigera klassen `ptr` så att testprogrammet fungerar som avsett. Korrekt utskrift ska vara:

```
17
42
8
```