Lösningsförslag, Tentamen i C++

2016–05–13

1. Eftersom en `unsigned`-variabel aldrig kan bli negativ kommer loopen aldrig att terminera. Utskriften blir:

```
10
9
8
7
6
5
4
3
2
1
0
4294967295
4294967294
4294967293
4294967292
4294967291
...
```

2. För att läsa ord för ord från en rad kan man först läsa rader med `getline` och sen använda en `std::istringstream`. Borttagning av skiljetecken kan göras med `std::remove_if`

```
index_t create_index(std::istream& in)
{
  index_t index;

  string separators{",.;:!?()"};
  auto is_separator = Is_char_from{separators};

  size_t line_no{0};
  string line;
  while(std::getline(in, line)){
    ++line_no;
    std::istringstream iss(line);
    string word;
    while(iss >> word) {
      auto it = remove_if(word.begin(), word.end(), is_separator);
      word.erase(it, word.end());
      index[word].insert(line_no);
    }
  }
  return index;
}
int main() {
    auto idx = create_index(std::cin);
    print_index(idx, std::cout);
    return 0;
}
```

För utskriften kan man använda `std::copy`

```
void print_index(const index_t& index, std::ostream& out)
{
  for(const auto& e : index) {
    out << e.first << " ";
    std::copy(e.second.begin(), e.second.end(), std::ostream_iterator<size_t>(out, " "));
    out << std::endl;
  }
}
```

Ett lite klumpigare alternativ att jämföra med är att skriva iterationen över värdena själv,

```
void print_index2(const index_t& index, std::ostream& out)
{
  for(const auto& e : index) {
    out << e.first;
    for(const auto& w : e.second) {
      out << " " << w;
    }
    out << std::endl;
  }
}
```

3.

```
template <typename Pred>
vector<string> collect_matching(Pred p, std::istream& is)
{
    auto res = vector<string>{};
    string s;
    while(is >> s){
        if(p(s))
            res.push_back(s);
    }
    return res;
}
```

4. I `test2` så skickas en referens till en stack-allokerad variabel, och när `consume_number` gör `delete` på denna fås ett fel eftersom man inte får göra `delete` på stack-allokerade objekt.

Felet beror av att i `test2` blandar man modeller för ägarskap — i `test2` så kan man inte överföra ägarskapet för ett stack-allokerat objekt.

5. a) Eftersom `void add(Vektor<T>, Vektor<T>, Vektor<T>)` har värdeparametrar så kommer argumenten att kopieras. När funktionen returnerar så kommer destruktorn att köras på kopiorna, och eftersom `Vektor` inte har någon kopieringskonstruktor så kommer arrayen som `p` pekar på att avallokeras.

b) Ändra till referensanrop

```
template <typename T>
void add(Vektor<T>& c1, Vektor<T>& c2, Vektor<T>& c3)
```

c) Nej, om man implementerar en copy-konstruktor (vilket man bör göra) så kommer programmet inte att längre att ha *undefined behaviour*, men resulatet blir `0 0 0 0 0 0`, eftersom tilldelningarna i `add` då görs till kopian. Om man vill ha en utparameter måste referensanrop användas.

6. a) Felet är på rad 29: `*p = 42;`. Problemet är att `operator*()` returnerar ett `int`-värde, och man kan inte tilldela ett sådant temporärt värde (*rvalue*).

b) ändra till att returnera en referens: `int& operator*() {return *p;}`