TDP004
STL

Eric Ekström

Institutionen för datavetenskap

Övning 1

Givet är ett program som läser in en rad med heltal från terminalen. Hitta det största heltalet och räkna ut antal gånger detta förekommer.

Övning 1

Givet är ett program som läser in en rad med heltal från terminalen. Hitta det största heltalet och räkna ut antal gånger detta förekommer.

```
std::vector<int> v { std::istream_iterator<int>{iss},
                     std::istream_iterator<int>{} };

int max { *std::max_element(v.begin(), v.end()) };
int count { std::count(v.begin(), v.end(), max) };

std::cout << "Det största talet var " << max
          << " och det förekom " << count << " gånger.";
```

Övning 2

Skriv ett program som läser från tangentbordet, räknar antal förekomster av varje unikt ord och skriver ut dessa sorterat i bokstavsordning.

Tips: Fokusera inte på algoritmer. Val av behållare är mycket viktigare här.

Övning 2

Skriv ett program som läser från tangentbordet, räknar antal förekomster av varje unikt ord och skriver ut dessa sorterat i bokstavsordning.

Tips: Fokusera inte på algoritmer. Val av behållare är mycket viktigare här.

```
std::map<string, int> m {};
std::string s {};
while (std::cin >> s)
{
  m[s]++;
}

for (auto [k, v] : m)
{
  std::cout << k << " : " << v;
}
```

Övning 3

1. Läs ett heltal *n* från tangenbordet
2. Läs från tangentbordet till behållaren *vec* fram tills ctrl-D
3. Skriv ut “True” om alla värden i *vec* är jämnt delbra med med *n*, annars “False”

Övning 3

1. Läs ett heltal *n* från tangenbordet
2. Läs från tangentbordet till behållaren *vec* fram tills ctrl-D
3. Skriv ut “True” om alla värden i *vec* är jämnt delbra med med *n*, annars “False”

```
int n {};
std::cin >> n;
std::vector<int> vec { std::istream_iterator<int>{std::cin},
                       std::istream_iterator<int>{} };

bool all_divisible { std::all_of(vec.begin(), vec.end(),
                     [&n](int i)
                     {
                       return i % n == 0;
                     })
};

std::cout << (all_divisible ? "True" : "False") << std::endl;
```

Övning 4

Givet en behållare, sortera alla värden som ligger mellan det första negativa värdet och slutet av behållaren i fallande ordning.

Övning 4

Givet en behållare, sortera alla värden som ligger mellan det första negativa värdet och slutet av behållaren i fallande ordning.

```
auto it = std::find(v.begin(), v.end(),
                    [](int i)
                    {
                      return i < 0;
                    });

std::sort(it, v.end(), std::greater<int>{});
```

www.ida.liu.se/~TDP004/