第9回演習

演習 1:count,ex09-1.cpp

- 1,2,3,1,2,2,3,4 を vector に保持する。
- キーボードから繰り返し入力した数値が何個あるかを出力するプログラムを作成せよ。

実行例:

```
$ ./a.out
2
count(2): 3
1
count(1): 2
3
count(3): 2
4
count(4): 1
5
count(5): 0
```

演習 2:count_if, ex09-2.cpp

- 2,3,5,7,11 を std::list a に保持する。
- キーボードから繰り返し入力した数値に対して、リスト a に含まれる約数とその約数の個数を出力するプログラムを作成せよ。
- ヒント：
  1. 入力を大域変数に入れ、述語となる関数が使う。
  2. 述語となる関数でも出力を行う

実行例: 括弧内が約数でコロンの後に個数

```
$ ./a.out
10
10 => (2)(5):2
12
12 => (2)(3):2
33
33 => (3)(11):2
21
21 => (3)(7):2
```

演習 3:minmax_element, ex09-3.cpp

- `std::deque d` に `4,2,1,5,9,8,9` を保持する。
- 最大最小値を除いて平均値を計算するプログラムを作成せよ。最大最小がそれぞれ複数ある場合には、それらはすべて除く。この例では $(4+2+5+8)/4 = 4.75$ となる。
  - Excel 関数とはちょっと違うので知っている人は注意

```
int main()
{
    std::deque d {4,2,1,5,9,8,9};
    auto [min,max] { std::minmax_element(d.begin(),
                                         d.end()) };
    double sum{0};
    int num{0};
    // この後を埋める
    // 出力：trimmean = 4.75
}
```

演習 4:find_if_not, ex09-4.cpp

- 整数の並びの中から最初に見つかる 2 でも 3 でも割り切れない数を出力するプログラムを作れ。
- `std::find_if_not` を使うこと。

```
// 適切なヘッダファイルを指定する

// 関数 pred を定義する

int main()
{
    std::vector a {2,6,18,17,12,3};
    auto it { std::find_if_not(a.begin(),
                               a.end(), pred) };
    std::cout << *it <<"\n"; // 17
}
```

演習 5:search_n, ex09-5.cpp

- 同一整数の指定回の並びを探すプログラムを作成せよ。
- 回数と対象となる整数はコマンド引数で指定する。

```
int main(int argc, char *argv[]) {
    if (argc < 3) {
        cout <<"usage:"<< argv[0]<<" count value\n";
        return 1;
    }
    int c{std::stoi(argv[1])}, v{std::stoi(argv[2])};
    std::vector a {1,4,3,3,4,4,2,2,2,2,3};
    // 以降を埋める
}
```

```
$ ./a.out
usage:./a.out count value
$ ./a.out 2 3
found at 2
$ ./a.out 2 4
found at 4
```

```
$ ./a.out 5 2
not found
$ ./a.out 4 2
found at 6
$ ./a.out 3 2
found at 6
```

演習 6:search, ex09-6.cpp

- 整数の並び a の部分列 s の各要素の値と、もう一つの整数の並び b の順序の対応する各要素の値との間で、差の絶対値が偶数という関係を持つ s を探すプログラムを作れ。

```
std::vector a{2,3,4,7,8,1,2,3,4}, b{1,2,3};
auto it { std::search(a.begin(), a.end(),
                      b.begin(), b.end(),  pred) };
// it が s の先頭のイテレータ
// 3 4 7 を出力
```

演習 7:find_end, ex09-7.cpp

- 4 つ引数の `std::find_end()` と同機能の関数を作れ。
- ヒント:
  - `std::search` を 1 回以上呼び、末尾の部分列を探す。
  - `std::next` を使っても良い
  - `std::find_end` はもちろん使用しない。

```
template<typename T, typename K>
T myfind_end(T b1, T e1, K b2, K e2){
  // std::search() を 1 回以上呼び出す
}

int main() {
   std::vector a{3,1,2,3,8,1,2,3,5,1,2,3,6},
               s{1,2,3};
   auto it { myfind_end(a.begin(), a.end(),
                        s.begin(), s.end()) };
   if (it != a.end())
      std::cout <<"found at "
                << it-a.begin() <<"\n"; // 9
}
```