# 第8回演習

# 演習 1

- コンテナの中身を出力する print() 関数を書け。
- 資料 p.10 を参考にし、範囲 for 文は用いないこと。

```
#include <iostream>   // prac08.hpp
template<typename T>
void print(const T& x) {
   // ここを考える
}
```

```
// 必要な include を加える                    prac08-1.cpp
#include "prac08.hpp"
int main() {
   std::vector a {1,2,3,4,5};
   std::list   b {7,4,2,1,9};
   std::set    c {6,2,8,6,2};
   print(a); // 1, 2, 3, 4, 5
   print(b); // 7, 4, 2, 1, 9
   print(c); // 2, 6, 8
}
```

# 演習 2

- 資料 p.14 の find と insert の組み合わせを試すテンプレート関数を prac08.hpp に作成せよ。
  - `template<typename T>`
    `void find_and_insert(T& c, int f, int x);`
  - c: コンテナ, f: 探す値, x: 挿入する値

```
#include <vector>          // prac08-2.cpp
#include <list>
#include "prac08.hpp"
int main() {
    std::list x {3,5,2,8,9,6,4};
    find_and_insert(x, 9, 10); // 9
    print(x); // 3, 5, 2, 8, 10, 9, 6, 4

    std::vector y {3,5,2,8,9,6,4};
    find_and_insert(y, 6, 7); // 6
    print(y); // 3, 5, 2, 8, 9, 7, 6, 4
}
```

# 演習 3

- 資料 p.15 を参考に指定値の前すべてに値を挿入するテンプレート関数を prac08.hpp に作成せよ。
  - `template<typename T>`
    `void insert_before(T& c, int f, int x);`
  - c: コンテナ, f: 指定値, x: 挿入する値

```
#include <vector>     // prac08-3.cpp
#include <list>
#include "prac08.hpp"
int main() {
    std::vector x {10,3,10,8,10,10};
    insert_before(x, 10, 5);
    print(x); // 5, 10, 3, 5, 10, 8, 5, 10, 5, 10

    std::list y {10,3,10,8,10,10};
    insert_before(y, 10, 5);
    print(y); // 5, 10, 3, 5, 10, 8, 5, 10, 5, 10
}
```

# 演習 4

- 資料 p.16 を参考に指定値をすべて削除するテンプレート関数を prac08.hpp に作成せよ。
  - `template<typename T>`
    `void remove_all(T& c, int x);`
  - c: コンテナ, x: 削除する値
- list にはテンプレートの特殊化をしても良い (第 3 回)

```
#include <vector>  // prac08-4.cpp
#include <list>
#include "prac08.hpp"
int main() {
    std::vector x {3,5,3,8,3,6,3};
    remove_all(x, 3); // すべての 3 を削除
    print(x); // 5, 8, 6

    std::list y {3,5,3,8,3,6,3};
    remove_all(y, 3); // すべての 3 を削除
    print(y); // 5, 8, 6
}
```

# 演習 5

- 連想コンテナを出力する print() 関数が作成せよ。

```
#include <iostream> // prac08-5.cpp
#include <map>
#include <unordered_map>
template<typename T>void print(const T& c) {
    // ここを考える (範囲 for 文でも良い)
}

int main() {
    std::multimap<std::string,int> x
    { {"xn",10},{"ya",5},{"xn", 5},{"xn",36},{"sa",5}};
    print(x); // [sa,5][xn,10][xn,5][xn,36][ya,5]

    std::unordered_multimap<std::string,int> y
    { {"xn",10},{"ya",5},{"xn", 5},{"xn",36},{"sa",5}};
    print(y); // [ya,5][sa,5][xn,10][xn,36][xn,5]
}  //  二つ目の print() はこの順序でない場合もある
```

二つ目の print() はこの順序でない場合もある

# 演習 6

- 資料 p.18 を参考に指定キーに対応する値をすべて 2 倍するテンプレート関数を作成せよ。

```cpp
#include <iostream> // prac08-6.cpp
#include <map>
#include <unordered_map>
template<typename T>void print(const T& c){/*演習5*/}
template<typename T>
void double_values(T& c, std::string k) {/*埋める*/}

int main() {
    std::multimap<std::string,int> x
    { {"xn",10},{"ya",5},{"xn", 5},{"xn",36},{"sa",5}};
    double_values(x, "xn");
    print(x); // [sa,5][xn,20][xn,10][xn,72][ya,5]
    std::unordered_multimap<std::string,int> y
    { {"xn",10},{"ya",5},{"xn", 5},{"xn",36},{"sa",5}};
    double_values(y, "xn");
    print(y); // [ya,5][sa,5][xn,72][xn,10][xn,20]
}  //  二つ目の print() はこの順序でない場合もある
```

# 演習 7

- 二つのイテレータ範囲の値を 3 倍して三つ目のイテレータの先に出力するテンプレート関数を作成せよ。

```cpp
#include <iostream> // prac08-7.cpp
#include <vector>
#include <iterator>
template<typename T, typename K>
void triple(T begin, T end, K out) { /* 埋める */ }

int main() {
    std::istream_iterator<int> in{std::cin}, eos;
    std::ostream_iterator<int> out{std::cout, ", "};
    triple(in, eos, out);
    std::cout <<"\n";

    std::vector v {10,20,30};
    triple(v.begin(), v.end(), v.begin());
    triple(v.begin(), v.end(), out);
    std::cout <<"\n"; // 90, 180, 270,
}
```

実行例：

```
$ echo 1 2 3 | ./a.out
3, 6, 9,
90, 180, 270,
```