# 第4回演習

# 演習 1

- 以下のコードを実行して結果を確認せよ。

```
// ex04-1.cpp
void print(const std::vector<int>& a) {
    cout << a.size() <<", "<< a.capacity() <<": ";
    for (auto e : a)
        cout << e <<" ";
    cout <<"\n";
}

int main() {
    std::vector<int> a {1,2,3};
    print(a);     // 3, 3:  1 2 3
    a.pop_back();
    print(a);     // 2, 3:  1 2
    a.pop_back();
    print(a);     // 1, 3:  1
    a.clear();
    print(a);     // 0, 3:
}
```

# 演習 2

- 以下のコードを実行して結果を確認せよ。

```
// ex04-2.cpp
int main() {
    std::vector a {3,2,3,4,8};
    a.front() = 1;
    a.back() = 5;
    cout << a.front() <<" "<< a.back() <<"\n"; // 1 5
    for (auto e : a)
        cout << e <<" ";
    cout <<"\n"; // 1 2 3 4 5
}
```

# 演習 3

- 以下のコードを実行して結果を確認せよ。

```
// ex04-3.cpp
int main() {
  std::vector a {1,2,3,4};
  auto it { a.begin() };   // イテレータ
  cout << *it <<" "<< *(it + 1)<<"\n"; // 1 2
  ++it;
  cout << *it <<"\n";                  // 2
  cout << a.end() - a.begin() <<"\n";  // 4
}
```

# 演習 4

- 以下のコードを実行して結果を確認せよ。

```
// ex04-4.cpp
void print(const std::vector<int>& a) {
    for (auto e : a) cout << e <<" ";
    cout <<"\n";
}

int main() {
    std::vector a {8,6,5,3,2,1};
    a.insert( a.begin(), 9 );    // push_front() に相当
    print(a); // 9 8 6 5 3 2 1
    a.insert( a.begin()+2, 7 ); // a[2] に 7 を挿入
    print(a); // 9 8 7 6 5 3 2 1
    a.erase( a.begin()+3 );      // a[3] を削除
    print(a); // 9 8 7 5 3 2 1
}
```

# 演習 5

- 以下のコードを実行して結果を確認せよ。

```
// ex04-5.cpp
int main() {
  std::vector a {1,2,3,4,5};

  for (auto it=a.begin(); it!=a.end(); ++it) {
     auto e { *it };
     std::cout << e <<"\n";
  }

  for (auto e : a)
     std::cout << e <<"\n";
}
```

# これ以降の演習

- これまでと違って講義資料のソースコードの確認が必要です。
- まず、`myvec.hpp` を作ります。
- 資料の `p.18, 24, 25, 26, 27, 28, 29` を見てください。

# 演習 6

- myvec.hpp を作成し以下が動くことを確認せよ。

```
#include <iostream>    // ex04-6.cpp
#include "myvec.hpp"
void print(const MyVec<int>& a) {
   std::cout <<a.capacity()<<" [";
   for (size_t i = 0; i < a.size(); i++)
      std::cout <<(i==0?"":" ")<< a[i];
   std::cout <<"]\n";
}
int main() {
   MyVec<int> x;
   for (int i = 0; i < 5; i++) {
      x.push_back(i);
      print(x);
   }
   x.insert(x.begin()+3, 10);
   print(x); // 8 [0 1 2 10 3 4]
   x.erase(x.begin()+2);
   print(x); // 8 [0 1 10 3 4]
}
```

# 演習 7

- test 関数テンプレートを作成せよ。
  - 前の演習の main の局所変数 x を関数の仮引数にする。
  - 仮引数の型はテンプレート引数で、void 関数とする。
  - 変数の宣言以外では main と test は同じ処理。
- 前の演習の print も関数テンプレートに変更せよ。
- main の内容を以下として全体を完成させよ。

```
#include <iostream>  // ex04-7.cpp
#include "myvec.hpp"
//
// ここをうめる
//
int main() {
    MyVec<int> x;
    test(x);
    cout <<"\n";
    std::vector<int> y;
    test(y);
}
```

# 演習 8

- MyVec に範囲指定の要素削除メンバ関数を追加せよ。
- void erase(size_t first, size_t last);
- 添字が (first, last] の範囲の要素を削除する。
- last > cursize ならば last=cursize とする。
- 範囲が無効ならば何もしない。

```
int main() {  // ex04-8.cpp
    // これを試せるようにする
   MyVec<int> x, y;
   for (int i = 0; i <5; i++) x.push_back(i);
   print(x); // 8 [0 1 2 3 4]
   y = x;
   y.erase(y.begin(), y.begin()+2);
   print(y); // 8 [2 3 4]
   y = x;
   y.erase(y.begin()+2, y.end());
   print(y); // 8 [0 1]
}
```

# 演習 9

- MyVec に範囲指定の同一要素挿入機能を追加せよ。
- `void insert(size_t pos, size_t count, const T& v);`
- pos の場所から count 個の v を挿入する。
- 実装の条件：
  - pos > cursize ならば pos=cursize とする。
  - 1 要素挿入の insert を count 回呼ぶ実装は不可。
  - 挿入後の capsize は要素数と同じにする。

```
int main() {   // ex04-9.cpp
  MyVec<int> x;
  x.insert(x.begin(), 5, 1);
  print(x); // 5 [1 1 1 1 1]
  x.insert(x.begin()+3, 5, 2);
  print(x); // 10 [1 1 1 2 2 2 2 2 1 1]
  x.insert(x.begin()+5, 4, 3);
  print(x); // 14 [1 1 1 2 2 3 3 3 3 2 2 2 1 1]
}
```

# 演習 10

- std::vector には以下の挿入メンバ関数がある
- void insert(iterator pos,
  iterator first, iterator last);
- オブジェクトの pos の場所に、別のオブジェクトの (first,last] の範囲の要素を挿入する。
- print 関数と空欄を考えてプログラムを完成させよ。

```
#include <iostream> // ex04-10.cpp
#include <vector>
// print テンプレート関数
int main() {
  std::vector a{1,2,3,4}, b{5,6,7,8,9};
                // a に b の一部を挿入する
  print(a); // 8 [1 6 7 8 9 2 3 4]
}
```

- MyVec クラスには、この機能提供するための同じ引数のメンバ関数を追加できない理由を考えよ。