

演習:map

- C++作業フォルダ内にPracMapフォルダを作成
`mkdir PracMap`
`cd PracMap`
- PracMapフォルダ内に main.cpp を作成する
`copy nul main.cpp`
- main.cpp をVisualStudioで開く

演習:map

PracMap(main.cpp)

```
#include<map>
#include<vector>
#include<iostream>
using namespace std;
typedef struct {    // 構造体Enemy
    string name;
    int hp, atk, def;
} Enemy;
int main() {

    return 0;
}
```

演習:map

•課題①

構造体Enemyを格納できるvectorの配列
vEneを宣言しなさい

vEneには初期値として以下のデータを格納する

Name	HP	ATK	DEF
Slime,	10,	5,	8
Wolf,	20,	30,	1
Spider,	30,	15,	5

演習:map

PracMap(main.cpp)

```
#include<map>
#include<vector>
#include<iostream>
using namespace std;
typedef struct {    // 構造体Enemy
    string name;
    int hp, atk, def;
} Enemy;
int main() {
    // 構造体Enemyを管理するvector配列を宣言
    vector<Enemy> vEne{ { "Slime",10,5,8 },
    { "Wolf",20,30,1 }, { "Spider",30,15,5 } };
    return 0;
}
```

演習:map

•課題②

キーは文字列(`string`)、
値には構造体(`Enemy`)を格納するmapコンテナ
`mapEn`を宣言しなさい

ただし、`mapEn`は空のmapとする。

演習:map

PracMap(main.cpp)

```
typedef struct {    // 構造体Enemy
    string name;
    int hp, atk, def;
} Enemy;

int main() {
    // 構造体Enemyを管理するvector配列を宣言
    vector<Enemy> vEne{ { "Slime", 10, 5, 8 },
                       { "Wolf", 20, 30, 1 }, { "Spider", 30, 15, 5 } };
    map<string, Enemy> mapEne{};
    return 0;
}
```

mapは基本データ型以外も格納可能

演習:map

- 課題③

mapEneのキーとして、エネミーの名前を格納し、
値にはパラメータ(Name, HP, Atk, Def)を格納する。

vEneに格納されているすべてのエネミーのデータを
mapEneに格納しなさい

演習:map

PracMap(main.cpp)


```
typedef struct {    // 構造体Enemy
    string name;
    int hp, atk, def;
} Enemy;

int main() {
    // 構造体Enemyを管理するvector配列を宣言
    vector<Enemy> vEne{ { "Slime", 10, 5, 8 },
                       { "Wolf", 20, 30, 1 }, { "Spider", 30, 15, 5 } };
    map<string, Enemy> mapEne{};
    for (const auto& d : vEne) { // 構造体を参照でdに格納
        mapEne.emplace(d.name, d);
    }
    return 0;
}
```


演習:map

•課題④

キーボードからエネミーの名前を入力し、**mapEne**から名前をキーとして値(Name,HP,Atk,Def)を画面表示するようにしなさい
(実行例)

エネミー名> Spider 

Name: Spider

HP: 30

Atk: 15

Def: 5

演習:map

PracMap(main.cpp)

```
vector<Enemy> vEne{ { "Slime",10,5,8 },
{ "Wolf",20,30,1 }, { "Spider",30,15,5 } };
map<string, Enemy> mapEne{};
for (const auto& d : vEne) { // 構造体を参照でdに格納
    mapEne.emplace(d.name, d);
}
cout << "エネミーの名前を入力>";
string input;
cin >> input;
if (mapEne.count(input)) {
    cout << "Name: " << mapEne[input].name << endl
        << "   HP: " << mapEne[input].hp << endl
        << "  Atk: " << mapEne[input].atk << endl
        << "  Def: " << mapEne[input].def << endl;
}
return 0;
}
```

演習:map

- C++作業フォルダ内にPracMap2フォルダを作成
`mkdir PracMap2`
`cd PracMap2`
- PracMap2フォルダ内に `main.cpp` を作成する
`copy nul main.cpp`
- `main.cpp` をVisualStudioで開く

演習:map

PracMap2(main.cpp)

```
#include <iostream>
#include <map>
#include <random>
using namespace std;
int main() {
    // 英単語の和訳と英訳のペアをdicに格納
    map<string, string> dic{ {"活動", "activity"},
        {"雰囲気", "atmosphere"}, {"血液", "blood"},
        {"環境", "environment"}, {"温度", "temperature"} };

    return 0;
}
```

演習:map

- 課題①

`dic`の先頭要素のキー(日本語訳)を表示しなさい

表示にはイテレータを用いることとする

演習:map

PracMap2(main.cpp)

```
#include <iostream>
#include <map>
#include <random>
using namespace std;
int main() {
    // 英単語の和訳と英訳のペアをdicに格納
    map<string, string> dic{ {"活動", "activity"},
        {"雰囲気", "atmosphere"}, {"血液", "blood"},
        {"環境", "environment"}, {"温度", "temperature"} };

    auto it = dic.begin();
    cout << "dicの先頭キー:" << it->first << endl;
    // イテレータ用の変数を宣言せずに書くパターン
    // cout << dic.begin()->first << endl;

    return 0;
}
```

演習:map

•課題②

`dic`のキー(日本語訳)をすべて表示しなさい

ただし、表示に用いるループ処理は、

- イテレータを用いたループ
- 範囲for

のどちらでもかまわないものとする

演習:map

PracMap2(main.cpp)

```
using namespace std;
int main() {
    // 英単語の和訳と英訳のペアをdicに格納
    map<string, string> dic{ {"活動", "activity"},
        {"雰囲気", "atmosphere"}, {"血液", "blood"},
        {"環境", "environment"}, {"温度", "temperature"} };

    for (auto it = dic.begin(); it != dic.end(); it++){
        cout << it->first << endl;
    }
    // 範囲for
    // for (auto p : dic){
    //     cout << p.first << endl;
    // }

    return 0;
}
```


演習:map

•課題③

乱数生成用の処理をプログラムに追加しなさい

```
//乱数生成器を作成
random_device rand_dev{};
//乱数アルゴリズムにメルセンヌツイスターを指定
mt19937 rand_engine(rand_dev());
// 0～(mapの要素数)-1 を均等な確率で得る分布生成器作成
uniform_int_distribution<int>
    dist(0, dic.size() - 1);
int rnd = dist(rand_engine); //rndには0~4が格納
```

演習:map

PracMap2(main.cpp)

```
using namespace std;
int main() {
    // 英単語の和訳と英訳のペアをdicに格納
    map<string, string> dic{ {"活動", "activity"},
        {"雰囲気", "atmosphere"}, {"血液", "blood"},
        {"環境", "environment"}, {"温度", "temperature"} };

    // 乱数生成器を作成
    random_device rand_dev{};
    // 乱数アルゴリズムにメルセンヌツイスターを指定
    mt19937 rand_engine(rand_dev());
    // 0～(mapの要素数)-1 を均等な確率で得る分布生成器を作成
    uniform_int_distribution<int>
        dist(0, dic.size() - 1);

    return 0;
}
```

演習:map

•課題④

乱数生成処理によって、0～4までの乱数を取得し、
dicの先頭イテレータから乱数ぶんだけ進めた
キーの値を表示しなさい
ただし、このとき**next**関数を使うものとする

`std::next`(イテレータ, イテレータを進める量)

例) `auto it = dic.begin();` // 先頭イテレータ取得
 `it = next(it, 3);` // +3してイテレータ更新

演習:map

PracMap2(main.cpp)

```
random_device rand_dev{};
mt19937 rand_engine(rand_dev());
uniform_int_distribution<int>
    dist(0, dic.size() - 1);

auto it = dic.begin();
it = next(it, dist(rand_engine));
cout << it->first << endl;

return 0;
}
```

演習:map

•課題⑤

乱数により、dicからひとつのキーを表示し、それに対応する英単語をキーボードから入力させる

キーボードから入力したものと、英単語が一致していれば「正解」、一致していなければ「不正解」と表示する

演習:map

PracMap2(main.cpp)

```
auto it = dic.begin();  
it = next(it, dist(rand_engine));  
cout << it->first << “の英単語は?”;
```

```
string input  
cin >> input;  
if ( it->second == input ) {  
    cout << “正解” << endl;  
} else {  
    cout << “不正解” << endl;  
}  
return 0;  
}
```