mapクラス

- **連想配列**と呼ばれるデータ構造
- 配列を添え字番号でなく、**【キー(key)】**というデータでアクセス
- 配列の要素は**【値(value)】**
- **【キー】**と**【値】**のデータ型は、基本データ型(intやstring等)が使用可能
- **【キー】**と**【値】**は一対一で対応

```
std::map <データ型, データ型> インスタンス名
```

mapクラス

- mapクラスのメンバ関数
  - size()　：全要素をカウント
  - clear()　：配列要素を全消去
  - empty()　：配列が空かどうかをチェック
  - erase()　：指定のデータを削除
  - insert()：キーと値のペアを追加
  - find()　：指定したキーのイテレータを返す
  - count()　：指定したキーにマッチする要素数を返す
  - at()　：指定したキーの値を返す

mapクラス

- 教科書P241~242 **Sample608**

- C++作業フォルダ内に**Sample608**フォルダを作成
```
mkdir Sample608
cd Sample608
```

- main.cppを作成してVisualStudioで編集
```
copy nul main.cpp
```

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
   map <string, int> score;//キーがstring、値がintのmapを宣言
   score[“Tom”] = 100;          //キーがTom、値が100
   score["Bob"] = 80;
   score["Mike"] = 76;
   cout << "Tomの点数は" << score["Tom"] << "点" << endl;
   cout << "Bobの点数は" << score["Bob"] << "点" << endl;
   cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
   return 0;
}
```

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);    //キーを指定して削除
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << "Bobの点数は" << score["Bob"] << "点" << endl;
    cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
    return 0;
}
```

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << “Bobの点数は” << score[“Bob”] << “点” << endl;
    if(score.count(“Mike”)) {    //指定キーの要素数から値の有無をチェック
        cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
    }
    return 0;
}
```

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score[
    score[
    score.erase(
    score.insert(make_pair(“John”,88)); //キー:John、値:88のペアを追加
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << “Bobの点数は” << score[“Bob”] << “点” << endl;
    if(score.count(“Mike”)) {
        cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
    }
    return 0;
}
```

//ペアを定義して挿入
　イテレータ不要(キー値により自動ソート)

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);
    score.insert(make_pair(“John”,88));
    auto itr = score.find(“John”); //キー:Johnがあればイテレータを返す
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << “Bobの点数は” << score[“Bob”] << “点” << endl;
    if(score.count(“Mike”)) {
        cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
    }
    return 0;
```

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);
    score.insert(make_pair(“John”,88));
    auto itr = score.find(“John”);
    cout << itr->first << "の点数は" << itr->second << "点" << endl;
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << “Bobの点数は” << score["Bob"] << "点" << endl;
    if(score.count("Mike")){
        cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
    }
```

//イテレータのfirstメンバがキー
secondメンバが値を表す

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);
    score.insert(make_pair(“John”,88));
    score.emplace(“David”, 70);
    au
    co                                    << itr->second << "点" << endl;
    co                                  Tom"] << "点" << endl;
    cout << “Bobの点数は” << score[“Bob”] << “点” << endl;
    if(score.count(“Mike”)) {
        cout << "Mikeの点数は" << score["Mike"] << "点" << endl;
```

//emplaceで追加
　この場合はペアの定義は不要

mapクラス

main.cpp (Sample608)

```
#include <iostream>
#include <string>
#include <map>
using namespace std;
int main() {
    map <string, int> score;
    score[“Tom”] = 100;
    score["Bob"] = 80;
    score[“Mike”] = 76;
    score.erase(“Mike”);
    score.insert(make_pair(“Joh
    score.emplace(“David”, 70);
    if(score.count(“John”)){
     auto itr = score.find(“John”);
     cout << itr->first << "の点数は" << itr->second << "点" << endl;
    }
    cout << "Tomの点数は" << score["Tom"] << "点" << endl;
    cout << “Bobの点数は” << score[“Bob”] << “点” << endl;
```

//Johnのキーがない場合誤動作するため、キーの存在チェックを行う

mapクラス

- mapまとめ

  - **連想配列**を実現するコンテナクラス
  - 添え字番号でなく、【キー】を使ってアクセス可能
  - **【キー(key)】**と**【値(value)】**がペアになる
  - map内部のデータは、キーの値によって昇順でソートされている
  - mapにデータを追加する際はキーと値のペアで追加
  - イテレータを使ってキーや値を取得可能