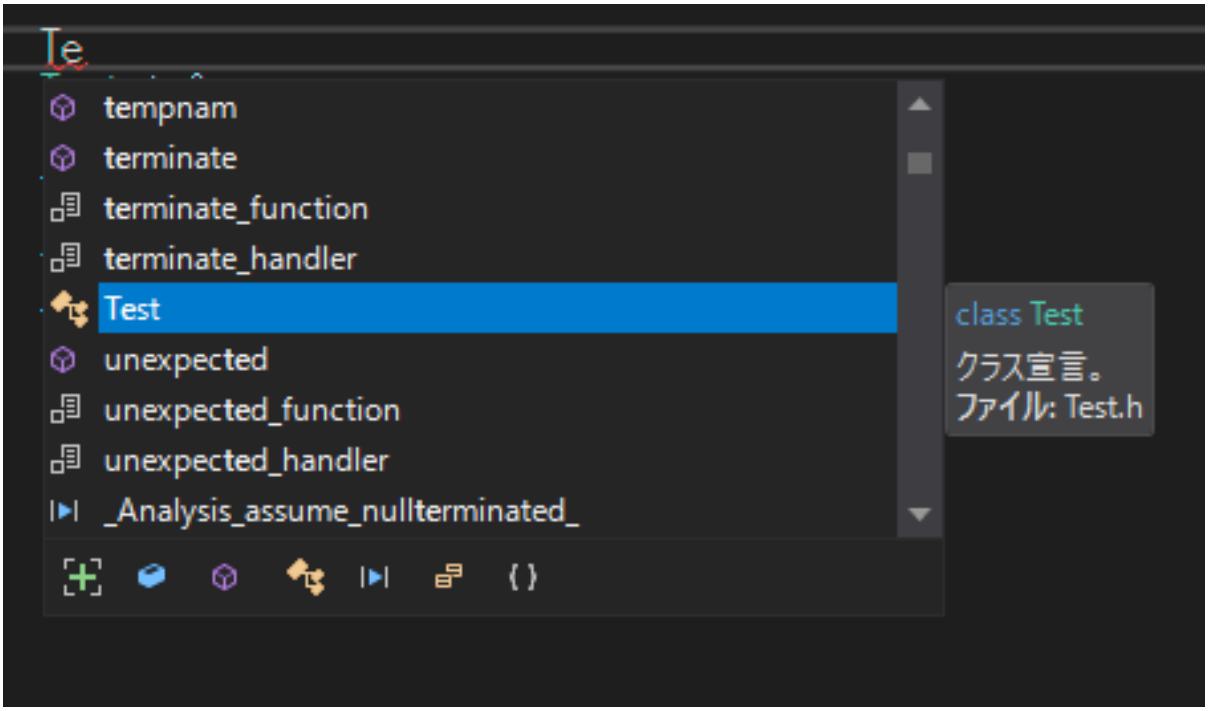
# Chapter 3 C++による大規模開発 ～C++標準テンプレートライブラリの利用(中級)～

## 3.1 検索

この節では次のプログラムを使用します。次のURLからプログラムをダウンロードしておいてください。

連想配列のmapはvectorやmapと違い、std::find()関数を利用することはできません。mapでstd::find()関数を利用すると意味不明なコンパイルエラーが発生します(図3.1)。 **図3.1**




mapでは代わりに、mapのメンバ関数のstd::map::find()関数を利用します。次のコードはstd::map::find()関数を利用するサンプルコードです。

```
std::map< std::string, int > hoge;
hoge.insert({ "豊臣秀吉", 30 });
hoge.insert({ "織田信長", 40 });
hoge.insert({ "武田信玄", 50 });

std::map<std::string, int>::iterator it;
// find関数には検索したい要素のキーを指定する。
it = hoge.find("豊臣秀吉");

if (it == hoge.end()) {
    // 検索対象の値が見つからなかった場合は終端のイテレーターを返してくる。
    std::cout << "見つからなかった" << "\n";
}
else {
    std::cout << "見つかった" << it->second << "\n";
}
```

## 3.2 要素の削除

mapの要素の削除はvector、listと同様にerase()関数を利用することで、要素を削除することができます。erase()関数に削除したいイテレーターを渡すことで削除することができます。次のコードはerase()関数の利用例です。

```
std::map< std::string, int > hoge;
hoge.insert({ "豊臣秀吉", 30 });
hoge.insert({ "織田信長", 40 });
hoge.insert({ "武田信玄", 50 });

std::map<std::string, int>::iterator it;
// find関数には検索したい要素のキーを指定する。
it = hoge.find("豊臣秀吉");

if (it != hoge.end()) {
    hoge.erase(it);
}
```

## 3.3 実習課題

Question_03_03のコメントを読んで課題に取り組みなさい。

## 評価テスト

次の評価テストを行いなさい。
評価テストへジャンプ