C++ のテンプレート

01ca0125 鈴木 藍
2002 年 5 月 30 日

目 次

概 要

C++ のテンプレートについて調べる。また、実際にテンプレートを使っていくつかのサンプルプログラムを作成し、動作を確認する。

レポートの目的

C++ のテンプレートに触れて ソースに目をならす。C++ で、本以外の文書の検索方法になれる。

1 C++ のテンプレートとは

C++にはテンプレートという機構が導入され、これによりクラスや関数内で使われている型を置き換える形で、 新たな型をコンパイル時に自動的に生成させることができる。C のマクロのさらに拡張版のようなものである。具体的には、以下のようにクラスのテンプレートを作成した場合、クラスの型を引数とすることで、新しい型を生成することが出来る。

```
#include <iostream>
#include <string>

template <class T>
class Tekito {
  public:
    void show() { cout << "test" << endl; }
};

main()
{
  Tekito <string> tekito;
  tekito.show();
}
```

また、クラスに限らず、関数にも定義する事が出来る。

```
template<class T>
#include <iostream>

T inc(T t)
{
  return ++t;
}

main()
{
  int a = 10;
  double b = 2.2;

  a = inc(a);
  b = inc(b);
}
```

こういった機構を使用する事により、より汎用的な実装が可能になる。

2 iterator を使用したプログラム

2.1 iterator テンプレート

micro soft の Visual C++ のサポートページでは、STL の iterator は以下のように定義されている。

```
template<class C, class T, class Dist = ptrdiff_t>
    struct iterator {
    typedef C iterator_category;
    typedef T value_type;
    typedef Dist distance_type;
    };
```

このテンプレート クラスは、すべての反復子の基本型になる。また、メンバ型 iterator_category (テンプレート パラメータ C のシノニム)、value_type (テンプレート パラメータ T のシノニム)、および distance_type (テンプレート パラメータ Dist のシノニム) を定義する。

2.2 vector テンプレート

今回は、配列を操作するために vector テンプレートを使用している。

2.3 iterator を使ったプログラム

iterator そのものを テンプレートとして定義しようと考えていたが、C++ の開発環境が無いために言語使用を調べられなかった。今回は、vector クラスの iterator を使用してサンプルプログラムを作成した。

2.3.1 サンプルプログラムの概要

以下は、整数型のデータ格納済の配列を vector コンテナに格納して、iterator を使って操作をするプログラムである。

```
#include <vector>
#include <iostream>

main()
{
  int array[] = {1,3,5,7,9,11,13,15};
  vector<int> v(array,&array[8]);
  vector<int>::iterator Ite;

  // initalize
  Ite = v.begin();
  // put array
  Ite ++;

  // test for vector class iterator
  cout << "test printing \"pointer\"" << endl;
  cout << *Ite << endl;

  // delete an element
  v.erase(v.begin());
  cout << *Ite << endl;

  // test 'for' word
  cout << "test printing \"for\"" << endl;
  for (Ite = v.begin(); Ite != v.end(); Ite++) {
    cout << *Ite << endl;
  }
}
```

実行結果は以下の通りである。

```
[spiral: ~/oop/c++]$ g++ sample.cpp
[spiral: ~/oop/c++]$ a.out
test printing "pointer"
3
5
test printing "for"
3
5
7
9
11
13
15
```

iterator の動作などは確認済であるが、その実装については C ならば可能であると思う。また、iterator は あるクラス (コンテナ) が生成する方が自然ではないかと思う。

まとめ・結論

テンプレートを使用すれば、プログラムの部品化と汎用化が可能である。この機能は実際に社会に出て仕事をする際に必要だと感じる。

感想

C++ については、このテンプレートの機能など 利用価値の高い物があることは認めているが、まずこれらを開発する環境がないことには始まらない。UNIX そのものが開発環境であるから、C に関しては 文書も豊富で整った環境であるが、C++ はそうではない。man にも info にも情報は無いに等しい。 GNU が C++ の開発環境に無頓着であることが原因だと考えられるが、このように 言語が標準で用意している関数やクラスなどを調べるのに時間がかかるような生産性の低い言語 (あるいは環境) は使うべきではない。その点においては Visual C++ や Wide studio は正しいと思う。

参考文献

[1] MSDN online
http://www.microsoft.com/japan/developer/library/
2002 年 5 月 29 日 参照

[2] イテレータ
http://homepage2.nifty.com/aito/cpp/node21.html#SECTION00042000000000000000
2002 年 5 月 29 日 参照

[3] クラステンプレート
http://www.njk.co.jp/otg/Study/CPP2/template.html
2002 年 5 月 29 日 参照

[4] イテレータ (反復子)
http://www.wakhok.ac.jp/ sumi/stl/ex_iterator.html#example
2002 年 5 月 29 日 参照

[5] STL メモ
http://oita.cool.ne.jp/ja6hfa/ja6hfa/cpp/stlmemo.html
2002 年 5 月 29 日 参照

[6] Skelton of GOF's Design Pattern
http://www11.u-page.so-net.ne.jp/tk9/ys_oota/mdp/Iterator/index.htm
2002 年 5 月 29 日 参照

[7] STL(Standard Template Library)
http://www.wakhok.ac.jp/ sumi/stl/index.html
2002 年 5 月 29 日 参照