

C++におけるジェネリッククラス

2Q-3

実装に関する一方式

○室伏 健二*, 梅川 竜一*, 長島 千成**, 石川 貴洋**, 田中 啓士郎**

*株式会社富士通静岡エンジニアリング, **富士通株式会社

1. はじめに

オブジェクト指向言語において、クラスを効率的に生成するための機能として一般的にジェネリッククラスがサポートされている。

C++では、この機能をtemplate機能として提供している。このtemplate機能は、クラス構造中の型をパラメタ化することにより、一つのクラス構造から、異なる型の同じ構造を持つクラスの作成を可能にしている。

ここでは、クラスが持つ情報の一部である関数におけるtemplate機能の実現方法について説明する。

2. 実装における問題点

関数におけるtemplate機能は、あらかじめtemplate定義された関数が引用された際に、その引数に対応した関数定義をコンパイラが生成することによって実現する。この場合、引数の型が異なる引用であれば、その都度、関数定義を生成する。また、引数の型が同じ引用であれば、関数定義は一度だけ生成する。

しかし、この関数定義を生成する処理の実装において、以下の問題が発生した。

① 結合時の二重定義エラー

別翻訳単位で同じtemplate定義を同じ型で引用している場合、翻訳単位ごとに関数を生成しかつ、生成した関数のみを分離することができないため、結合時に二重定義エラーが発生する。

② template定義修正の反映もれ

一つのtemplate定義から、複数の関数定義を生成するため、template定義を修正した場合、該当するtemplate定義から生成したすべての関数を再翻訳する必要がある。しかし、翻訳単位をまたがっているため、翻訳時には、その影響範囲を検索できない。

3. 実現方法概要

3.1 問題点の解決

問題点を以下の方法により解決する。

① 結合時の二重定義エラーの解決

template定義から生成した関数定義のコンパイラ出力結

果(以降オブジェクトと呼ぶ)を、翻訳単位ごとに生成せず、結合を行う前に生成する。これにより、関数の二重定義を解決する。

② template定義修正の反映もれの解決

翻訳時にtemplate定義の情報をファイルに収集し、結合を行う前に、この情報を元に、修正されたtemplate定義から生成したすべての関数定義を再度翻訳する。これにより、template定義の修正反映もれを解決する。

3.2 処理概要

処理の流れを以下に示す。

① コンパイラはtemplate定義に関する情報(template情報1)を取り出し、template以外の部分のオブジェクトを生成する。

② 結合の前処理(以降この処理を行うものをプレリンカと呼ぶ)として、翻訳時に収集した情報を利用し、template定義から生成した関数定義のオブジェクトを生成する。このとき、template定義と関数に関する情報(template情報2)を保存する。template定義が修正され、再度プレリンカを動作させた場合、プレリンカは、その修正に影響のある関数のみを翻訳する。

③ プレリンカで必要なオブジェクトを検索し、リンクを呼び出す。

図1に処理の概要を示す。

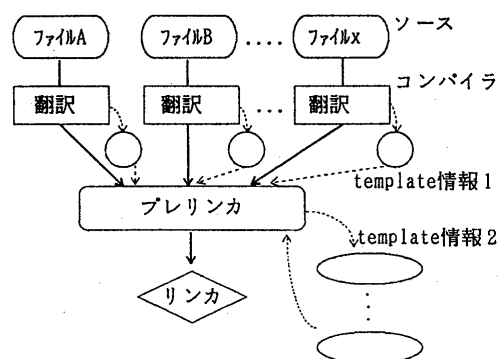


図1 処理概要

A Method Concerning Implementation of Template in Language Processor

Kenji MUROFUSHI¹, Ryuichi UMEKAWA¹, Chinari NAGASHIMA², Takahiro ISHIKAWA², Keishiro TANAKA²

¹FUJITSU SHIZUOKA ENGINEERING, Ltd., ²FUJITSU, Ltd.

4. 処理内容

4.1 コンパイラの処理

翻訳単位とtemplate定義との関係を収集する。

- ① template定義が引用された場合、引用時に指定された引数の型と関数名から、生成する関数名を決定する。
- ② template定義の引用による関数定義を除き、オブジェクトを生成する。
- ③ 以下の情報を収集する。(以降この情報をコンパイラ収集情報と呼ぶ)
 - template定義が含まれるインクルードライブラリ名
 - インクルードライブラリの更新日付
 - template定義の引用時に指定された引数がクラス型であれば、そのクラス型が定義されているインクルードライブラリ名

図2にコンパイラ収集情報を示す。

ファイル名.INF	更新日付	template ライブラリ名	型情報 関数名
ファイル名.INF	更新日付	template ライブラリ名	型情報 関数名

図2 コンパイラ収集情報

4.2 プレリンカの処理

まず、template定義の引用による関数のオブジェクトを生成する。

次に、結合に必要なオブジェクトを検索し、リンクに渡す。

- ① コンパイラ収集情報が格納されているファイル名と更新日付をまとめたファイル(以降管理情報と呼ぶ)が存在するか判断する。図3に管理情報を示す。
- ② 管理情報が存在しない場合以下の処理を行う。
 - (a) コンパイラ収集情報をtemplate定義が格納されているインクルードライブラリごとの情報に変更する(以降プレリンカ収集情報と呼ぶ)。
図4にプレリンカ収集情報を示す。
 - (b) プレリンカ収集情報から、template定義の引用による関数のオブジェクトを生成する。
- ③ 管理情報が存在する場合以下の処理を行う。
 - (a) コンパイラ収集情報の更新日付を検査する。
 - (b) コンパイラ収集情報が更新されている場合、その情報に含まれるtemplate定義のインクルードライブラリの更新日付を確認する。
 - (c) template定義が更新されている場合は、プレリンカ収集情報から、template定義の引用による関数を再翻訳する。
- ④ コンパイラ収集情報を元に、結合に必要なオブジェク

トを検索し、リンクを呼び出す。

ファイル名.INF	更新日付
ファイル名.INF	更新日付
...	...

図3 管理情報

templateライブラリ名.INF	更新日付	template	型情報
更新日付	関数名		
...	...		

図4 プレリンカ収集情報

5. 評価

この方式では、最初にプレリンカを動作させるときに、template定義の引用により生成する関数をすべて翻訳するため、通常のリックに比べ時間がかかる。しかし、一度翻訳した関数は、定義が修正されない限り、再翻訳することはないため、総合的にみて効率のよい翻訳が可能となる。

また、template定義を修正することによる影響範囲を意識しなくても、コンパイラ及びプレリンカにより自動的に検索し、再翻訳するため、定義の新旧による問題が発生しない。

このように、template定義とそれから生成される関数についての情報を取り出し、プレリンカを利用することにより、実用的な関数におけるtemplate機能を実現できる。

6. おわりに

今回の方式により、template機能の関数に対する機能が実現可能となる。しかし、template機能には、クラス定義やconstructor, destructor, friend, virtual 関数などが含まれており、コンパイラへの実装においては、これらの考慮が必要となる。今後、template機能を総合的に検討し、コンパイラへ反映していきたい。

【参考文献】

- [1] 『THE ANNOTATED C++ REFERENCE MANUAL』
Margaret A. Ellis, Bjarne Stroustrup
- [2] 『Working Paper for Draft proposed American National Standard for Information Systems Programming Language C++, X3J16/91-0115』
- [3] 『C++ ツールキット』
J. シャピロ著 山中政秋/秋元信彦訳
- [4] 『C MAGAZINE 1992.5』
ソフトバンク