



Publication Year	2000
Acceptance in OA	2023-03-09T11:53:59Z
Title	Template Coding for FORTRAN 90 Programs
Authors	MARIS, Michele
Handle	http://hdl.handle.net/20.500.12386/34014
Volume	PL-LFI-OAT-TN-012.2

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 1 of 18
---	------------------------------	--	--

Template Coding for FORTRAN 90 Programs

Document No.: PL-LFI-OAT-TN-012.2

Issue/Rev. No.: 0.0


Date: 2000-June-2

Prepared by: Michele Maris

Authorized by: Fabio Pasian

Abstract

Many lines of FORTRAN 90 code follows a regular pattern + variable substitutions i.e. CODE TEMPLATES. A mechanism to introduce CODE TEMPLATES in F90 through the use of a PERL 5.0 preprocessor is introduced. The advantage of CODE TEMPLATING is a relevant saving of time and errors in code writing once the template is properly defined.

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 2 of 18
---	------------------------------	--	--

Document Approval

Name	Organization	Signature	Date
F.Pasian LFI DPC Manager	OAT		2000 June 2

Document Status Sheet

1. Document Title: An LFI data processor to create a Quick-Look Source Catalogue			
2. Issue	3. Revision	4. Date	5. Reason for change
N	0	2000-June-2	Initial version
	1	2000-June-6	Added new functionalities



	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 3 of 18
---	------------------------------	--	--

Table of Contents

Document Approval.....	2
Document Status Sheet.....	2
Table of Contents.....	3
1. Introduction.....	4
2. Rules.....	4
3. Template Language.....	4
4. Example.....	7
4.1 Example of definition:.....	7
4.2 Example of usage.....	7
4.3 Example of Result:.....	8

	Code Templates in F90	Document No.	PL-LFI-OAT-TN-012.2
		Issue/Rev. No.	0.0
		Date	2 June 2000
		Page	4 of 18

1. Introduction

Many lines of FORTRAN 90 code follows a regular pattern + variable substitutions i.e. CODE TEMPLATES. A mechanism to introduce CODE TEMPLATES in F90 through the use of a PERL 5.0 preprocessor is introduced. The advantage of CODE TEMPLATING is a relevant saving of time and errors in code writing once the template is properly defined. In this document a short discussion of FORTRAN templates and a tool to manage them is presented.

2. Rules


The first effort in developing these rules is to prevent miss compilation of FORTRAN code not properly processed.

1. Templates declarations must be not FORTRAN 90 keywords
2. Templates definitions must be removed from FORTRAN code after their readout
3. Templates definitions and calls must interfere with FORTRAN 90 keywords in order to prevent the compilation of code with missing parts.
4. Template command must hold one line only
5. Templates must be defined at the beginning of the F90 code outside any other F90 code.
6. Template syntax is PERL: variables are individuated by \$<<name>>
7. When requested a manual page may be generated for a given template
8. PERL commands to compose names may be added before the execution of the code generation
9. It is assumed that people creating templates knows few basic PERL programming, in particular:
 - i) scalar variables naming;
 - ii) scalar variables assignment;
 - iii) the meaning of the local attribute;
 - iv) string definition rules;
 - v) the string chaining operator ‘.’;
 - vi) the need of ‘;’ after each line.

Rule (9) is acceptable in the view that templates are written by few persons and applied by many.

3. Template Language

Templates are defined in a at the top of the FORTRAN90 file or in a separated file which is pasted at the top just before preprocessing.

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 5 of 18
---	------------------------------	--	--

The definition part ends with the command

```
#END_F90_TEMPLATES_DEFINITIONS
```

Everything preceding this line and outside the single template definitions is skipped.

Two structures are used to define begin and end a template definition block:

```
#DEF_F90_TEMPLATE <<NAME>> <<ARGUMENTS>> (one line)
```

```
... Manual Lines
```

```
#END_MANUAL
```

```
... Preamble lines
```

```
#END_PREAMBLE
```

```
...
```

```
... F90 code + SUBSTITUTION MARKS
```

```
...
```

```
#END_F90_TEMPLATE <<NAME>> (one line)
```

Preamble are lines which has to be executed after parameters assignment but before to perform the execution of the template substitution. A typical usage is in case of ambiguous substitutions. As an example, let be you have to introduce a list of strings with the form: <<object>>_Temperature,

<<object>>_Current, <<object>>_Voltage (as an example is <<object>> = FEU then:

FEU_Temperature, FEU_Current, FEU_Voltage). One may define

```
#DEF_F90_TEMPLATE AMBIGUOUS OBJECT
```


```
! Variables for object $OBJECT
```

```
    $OBJECT_Temperature = ...
```

```
    $OBJECT_Current = ...
```

```
    $OBJECT_Voltage = ...
```

```
#END_F90_TEMPLATE AMBIGUOUS OBJECT
```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 6 of 18
---	------------------------------	--	--

Unluckily this template will not work, since PERL will look not for the \$OBJECT keyword to be replaced, but for the \$OBJECT_Temperature, \$OBJECT_Current, \$OBJECT_Voltage to be replaced, since '_' is a valid character for a variable name. The solution to prevent this case is to introduce a preamble in the definition where these names are defined starting from \$OBJECT.

```
#DEF_F90_TEMPLATE NOT_AMBIGUOUS OBJECT

    local $OBJECT_Temperature = $OBJECT.'_Temperature';

    local $OBJECT_Voltage = $OBJECT.'_Voltage';

    local $OBJECT_Current = $OBJECT.'_Current';

#END_PREAMBLE

! Variables for object $OBJECT

    $OBJECT_Temperature = ...

    $OBJECT_Current = ...

    $OBJECT_Voltage = ...

#END_F90_TEMPLATE NOT_AMBIGUOUS OBJECT
```

note that PREAMBLE is a list of valid PERL lines, including ';' at the end and note also the use of the local statement before each definition.

Manual are lines displayed when the -U option is given. They are detailed explanations about, as example, the template usage. As example:

```
#DEF_F90_TEMPLATE NOT_AMBIGUOUS OBJECT

    OBJECT = a valid OBJECT Name (ex.: FED)


#END_MANUAL

    local $OBJECT_Temperature = $OBJECT.'_Temperature';

    local $OBJECT_Voltage = $OBJECT.'_Voltage';

    local $OBJECT_Current = $OBJECT.'_Current';

#END_PREAMBLE
```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 7 of 18
---	------------------------------	--	--

```
! Variables for object $OBJECT
```

```
    $OBJECT_Temperature = ...
```

```
    $OBJECT_Current = ...
```

```
    $OBJECT_Voltage = ...
```

```
#END_F90_TEMPLATE NOT_AMBIGUOUS OBJECT
```

After definitions are processed, the preprocessor scans the remaining part of the listing in order to perform template substitutions. The command to order a template substitution is:

```
#APPLY <<NAME>> <<ARGUMENTS>> (one line)
```

As an example:

```
#APPLY NOT_AMBIGUOUS FED
```

will generate the F90 code:

```
! Variables for object FED
```

```
    FED_Temperature = ...
```


```
    FED_Current = ...
```

```
    FED_Voltage = ...
```

4. Example

4.1 Example of definition:

```
#DEF_F90_TEMPLATE OPTIONAL_OPEN UNIT FLAG_I_WISH OBJECT FNAME
    if ($FLAG_I_WISH.eq.1) then
        open (unit=$UNIT,file=$FNAME,status='new',      &
            &          form='unformatted')
        write(16,'(a)') ' U: $UNIT $OBJECT=', $FNAME
    endif
#END_F90_TEMPLATE OPTIONAL_OPEN
#END_F90_TEMPLATES_DEFINITIONS
```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 8 of 18
---	------------------------------	--	--

4.2 Example of usage

```
#APPLY OPTIONAL_OPEN 12 i_wish_print_pix datapix_map datapix_map
#APPLY OPTIONAL_OPEN 13 i_wish_print_noise datanoise datanoise
```

4.3 Example of Result:

```
if (i_wish_print_pix.eq.1) then
  open (unit=12,file=datapix_map,status='new',      &
        &      form='unformatted')
  write(16,'(a)') ' U: 12 datapix_map=',datapix_map
endif

if (i_wish_print_noise.eq.1) then
  open (unit=13,file=datanoise,status='new',      &
        &      form='unformatted')
  write(16,'(a)') ' U: 12 datanoise=',datanoise
```


5. The f90tcrep.pl tool

The preprocessor for template coding is named `f90tcrep.pl` and it is written in PERL 5.0.

In appendix the preprocessor listing plus one example of file to be preprocessed are given.

Several options to help the template operations are added. They are:

- h** : print one help
- vn** : sets the verbosity level: 0 quiet work, 1 only input displayed, 2 Input/Output displayed, 3 template lines are printed, 4 perl lines are printed after execution
- l** : list full templates definitions
- t** : generates a test file for macro definitions
- c** : list calls to template definitions
- u** : Usage for a given template whose name replaces <<OutPutFile>>
example: `f90tcrep -u library STANDARD_OPEN`
- U** : as -u but print the Help section (if any) of a given template
example: `f90tcrep -? library STANDARD_OPEN`
- p** : as -U but print the preamble section of a given template

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 9 of 18
---	------------------------------	--	--

Appendix A: The Preprocessor

```

#
# f90tccprep.pl - 0.0 - By M. Maris - 2 June 2000 -
#
# usage f90tccprep.pl <<InputFile>> <<OutputFile>> [<<Verbosity>>]
#
# Template preprocessor for F90
#
# Assumes definitions at the beginning of the input file
#
# WARNING: DISK SIDE-EFFECT
# The program needs to generate (OVERWRITE) two dump files
# the names of these two files are parametrized in the Variables:
#
#     $DumpPerl $DumpF90
#
# change their content if the program interfere with something in
# your directory.
#
# We hope in the future to be able to remove this flaw
#

# Dump File Names
$DumpPerl = '_f90tccprep_dump_perl.pl';
$DumpF90  = '_f90tccprep_dump_f90.f90';

# Starting header
$Verbosity = 1;

$Today = gmtime(time);

$operator = '';

$Version = "f90tccprep.pl - 0.0 - By M. Maris - 23 May 2000 -";

print << "EOH";
$Version

usage f90tccprep.pl <<InputFile>> <<OutputFile>> [<<Verbosity>>]

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 10 of 18
---	------------------------------	--	---

```

Verbosity : 0 None
           : 1 Only input displayed
           : 2 Input/Output displayed
           : 3 Template lines are printed
           default = $verbosity

```

See added notes and comments about the GENERATION OF TEMPORARY FILES

EOH

```

# Parameters consistecy check
$InputFile  = $ARGV[0]; if (length($InputFile) == 0) { die "Missing input name" };
$OutputFile = $ARGV[1]; if (length($OutputFile) == 0) { die "Missing output name" };

if ($InputFile eq $OutputFile) { die "Input file can not be equal to output file"};

if ($#ARGV == 2) {
    $Verbosity = $ARGV[2];
};

# Source scan
open FID, '<'.$InputFile;
@Listing = <FID>;
close(FID);


# resets statistical counters
$NTemplates = 0;
$NReplacements = 0;
$NLinesGenerated = 0;

$DefinitionStack = {};

#
# Begin of Operations
#

# Scans for the definition block and load them
$NLine = 0;
$InSide = 1;
$IsTemplateOpen=0;

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 11 of 18
---	------------------------------	--	---

```

$key = $Listing[$NLine];

while (!( $key =~ m/^\s*#[#]END_F90_TEMPLATES_DEFINITIONS/i))
{
    $key = $Listing[$NLine];

    if ( $key =~ m/^\s*#[#]DEF_F90_TEMPLATE/i)
    {
        @ListObjects = grep(/^\s/,split(/\s+/, $key)); #Tokenize the line

        $IsTemplateOpen = 1;

        $TemplateName = @ListObjects[1];
        @Names = @ListObjects[2..$#ListObjects];

        $NumberParameters = $#Names+1;


        if ($Verbosity >= 2) {
            print "Template:", $TemplateName, "\n";
            print "NParam: ", $NumberParameters, "\n";
            for $IPAR (0..$#Names) {
                print "    : ", @Names[$IPAR], "\n";
            };
            print ",\n";
        };

        @ListLines=();
        $NLine = $NLine + 1;
        $key = $Listing[$NLine];
        while (!( $key =~ m/^\s*#[#]END_F90_TEMPLATE/i) & ($NLine < $#Listing))
        {
            push(@ListLines, $key);
            $NLine = $NLine + 1;
            $key = $Listing[$NLine];
        };

        if ($Verbosity >= 3) {
            print "Macro:\n";
            print @ListLines;
        };

        if (!( $key =~ m/^\s*#[#]END_F90_TEMPLATE/))
        {die "Error: Template definition ended by EOF\n";

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 12 of 18
---	------------------------------	--	---

```

};

$TDictionary{$TemplateName} = {
  parameters => [@Names],
  code       => [@ListLines]
};

$NTemplates = $NTemplates + 1;

$InSide=1;

};

$NLine = $NLine + 1;

if (($NLine > $#Listing) & ($InSide == 1)) {
  die
  "Error: Template definition or TEMPLATE DEFINITION SECTION ended by EOF\n";
};

};

if (!$key =~ m/^\s*#[#]END_F90_TEMPLATES_DEFINITIONS/i)
{
  die
  "Error: Template definition section not ended by END_F90_TEMPLATES_DEFINITIONS\n";
};

#
# Generates Output
#


open FOD, '>'.$OutputFile;

# Please update this constant if you change the number of lines in CARTIGLIO
$NLinesCartiglio = 14;

print FOD << "CARTIGLIO"; # Adds a note at the text beginning

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! This code has been preprocessed with the F90 Macro Code substitutor

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 13 of 18
---	------------------------------	--	---

```
!
! $Version
!
! Original Name : $InputFile
! Final Name : $OutputFile
! Date : $Today
! Operated by : $Operator
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

CARTIGLIO

```
# Scans for evaluations

if ($NLine >= $#Listing) {
die
"Error: Missing the code to be used\n";
};

for $Iline ($NLine + 1 .. $#Listing){

$key = $Listing[$Iline];

if ($key =~ m/^\s*[#]APPLY/i)
{
# Performs the substitution

@ListObjects = grep(/\S/,split(/\s+/, $key)); #Tokenize the line


if ($#ListObjects < 1) {
die "Error: Missing Template Name at line $Iline\n";
};

$TemplateName = @ListObjects[1];

@Values = @ListObjects[2..$#ListObjects];

$NumberValues = $#Values+1;

if ($Verbosity >= 2) {
print "ApplyTo:", $TemplateName, "\n";
print "NValues: ", $NumberValues, "\n";
};
};
};
```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 14 of 18
---	------------------------------	--	---

```

for $IVAL (0..$#Values) {
  print "      : ",@Values[$IVAL],"\\n";
};
print , "\\n";
};

@LLLL = grep(/^$TemplateName/, (keys %TDictionary));

if ($#LLLL < 0) {
  die "Error: Unexistent Template Required at line $ILine\\n";
};

%Object = %{$TDictionary{$TemplateName}};
@Names = @{$Object{parameters}};

if (!$#Names == $#Values) {
  die "Error: Uncorrect Number of Template Parameters Required at line $ILine\\n";
};

# Generates the command list
# Parameters declaration
@CommandList = ("\\n");
for $IPar (0..$#Names) {
  $scratch = 'local $.@Names[$IPar]. = "'.@Values[$IPar].'";'. "\\n";
  push (@CommandList,$scratch);
};

# overstructures to produce @Macro list
push (@CommandList,"\\n");
push (@CommandList,'@Macro = << "EOM"; '. "\\n");


# overstructures the Macro
@ListLines = @{$Object{code}};

for $IPar (0..$#ListLines) {
  push (@CommandList,@ListLines[$IPar]);
};

# overstructures to produce @Macro list
push (@CommandList,"EOM\\n");

push (@CommandList,'print @Macro;');

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 15 of 18
---	------------------------------	--	---

```

push (@CommandList, "\n};\n");

open TMP, '>'. $DumpPerl;
print (TMP @CommandList);
close TMP;

system('perl ' . $DumpPerl . ' > ' . $DumpF90);

open FID, '< ' . $DumpF90;
@Macro = <FID>;
close FID;

print FOD @Macro;

$NReplacements = $NReplacements + 1;

$NLinesGenerated = $NLinesGenerated+$#Macro+1;

} else {
# Writes the line
print FOD $key;

};

}; # for $ILine

#
# operations end
#

close(FID);


print "\n<< Added a note at the beginning of the output file\n\n";

# bye bye message

$NLinesOfCode = $#Listing - $NLine + 1;
$NLinesOfOutput = $NLinesOfCode + $NLinesGenerated - $NReplacements+$NLinesCartiglio-1;
$NLinesOfInput = $#Listing + 1;

print << "EOH1";
Conversion: $InputFile -> $OutputFile

```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 16 of 18
---	------------------------------	--	---


Relevant Statistics:

```

Lines in Input                : $NLinesOfInput
Of Template Definitions       : $NLine
Of Code                       : $NLinesOfCode
Templates Defined             : $NTemplates
Templates Replaced            : $NReplacements
Lines Generated               : $NLinesGenerated
Total Lines in Output         : $NLinesOfOutput

```

EOH1

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 17 of 18
---	------------------------------	--	---

Appendix B: One Example of F90 Code with Code Templates

```
!!!!
!!!! THIS IS ONE EXAMPLE OF TEMPLATES AND TEMPLATES SUBSTITUTIONS
!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! BLOCK OF TEMPLATES DEFINITIONS          !!!!
!!!! IT MAY BE APPENDED TO THE HEAD TOO      !!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
#DEF_F90_TEMPLATE OPTIONAL_OPEN UNIT FLAG_I_WISH OBJECT FNAME
! Opens $OBJECT
    if ($FLAG_I_WISH.eq.1) then
        open (unit=12,file=$FNAME,status='new',    &
&          form='unformatted')
        write($UNIT,'(a)') ' U: $UNIT $OBJECT=', $FNAME
    endif

#END_F90_TEMPLATE OPTIONAL_OPEN
```


```
#DEF_F90_TEMPLATE OPTIONAL_OPEN2 UNIT FLAG_I_WISH OBJECT FNAME
! Opens $OBJECT but template 2
    if ($FLAG_I_WISH.eq.1) then
        open (unit=12,file=$FNAME,status='new',    &
&          form='unformatted')
        write($UNIT,'(a)') ' U: $UNIT $OBJECT=', $FNAME
    endif

#END_F90_TEMPLATE OPTIONAL_OPEN2
```

```
#END_F90_TEMPLATES_DEFINITIONS
```

```
program template_example.f90
```

```
! some comment line
```

	Code Templates in F90	Document No. Issue/Rev. No. Date Page	PL-LFI-OAT-TN-012.2 0.0 2 June 2000 18 of 18
---	------------------------------	--	---

```

print F0D ! Uncommenting these lines you will be generate errors
! #APPLY
! #APPLY OPTIONAL 12 i_wish_print_pix datapix_map
! #APPLY OPTIONAL_OPEN 12 i_wish_print_pix datapix_map datapix_map
! #APPLY OPTIONAL_OPEN 12 i_wish_print_pix datapix_map datapix_map extra

! This is right
  #APPLY OPTIONAL_OPEN 12 i_wish_print_pix datapix_map datapix_map

! some other comment line

  APPLY = 34

! other application
  #APPLY OPTIONAL_OPEN 13 i_wish_print_noise datanoise datanoise

! other template
  #APPLY OPTIONAL_OPEN2 14 i_wish_print_quantum dataquantum dataquantum

end program template_example.f90

```