

Using the Macro Facility

Natural provides a facility with which source can be generated automatically, relieving you of much routine editing work, and eliminating editing errors. This feature is useful when writing programs with a similar structure but with different content.

The macro feature is an extension of the Natural language and consists of special processing statements executed when the program is compiled, as well as variables in source lines substituted by valid values at compilation time. These special processing lines and variables are distinguished in the source by typing the macro character as first character in the line.

As a simple example of macro processing lines and variables, if you define the following macro:

```
0010 § MOVE 'PERSONNEL' to #FILE-NAME (A32)
0020 § MOVE 'NAME'      to #KEY      (A32)
0030 READ §#FILE-NAME by §#KEY
```

the following source line is generated at compilation time:

```
READ PERSONNEL BY NAME
```

By varying the values, a variety of source lines can be created from the same 'skeleton' macro. You can use this macro feature in Natural in a variety of ways:

1. Macro objects:

Macro objects are a special type of Natural object and can be accessed and maintained as any other Natural object (specify TYPE=MACRO). They contain macro processing lines and macro variables and can be used to generate Natural programs, or they can be referenced from other Natural objects to generate text lines;

2. Inline macros:

Other Natural objects (e.g. Natural programs, PDS members, VSE/ESA members, etc.) can use macro variables in their source. They can also reference a macro object using the special INCLUDE-MACRO <name> statement. The variables are substituted and the specified macro object is executed, and the generated lines are included in the source when the object is compiled (Natural) or submitted (non-Natural);

3. Edit macros:

When starting an edit session with a Natural object (Natural program, PDS member, VSE/ESA member, LMS element etc.), you can specify a macro object to be used as a model for the new edit session. The specified macro is executed and the generated lines written to the new edit session. The lines generated in this way are protected, but you can reserve some places in the macro object in which you can add your own code after the macro is invoked as an edit macro.

The generated output of objects that use the macro facility is written to the user workpool, where it can be checked and further handled (for full details and examples, see the section Macro Facility in the Natural ISPF Programmer's Guide).