**RPG** is a [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [business applications](https://en.wikipedia.org/wiki/Business_software), introduced in 1959 for the [IBM 1401](https://en.wikipedia.org/wiki/IBM_1401). It is most well known as the primary programming language of IBM's [midrange computer](https://en.wikipedia.org/wiki/Midrange_computer) product line, including the [IBM i](https://en.wikipedia.org/wiki/IBM_i) [operating system](https://en.wikipedia.org/wiki/Operating_system).[[1]](https://en.wikipedia.org/wiki/IBM_RPG#cite_note-1) RPG has traditionally featured a number of distinctive concepts, such as the program cycle, and the column-oriented syntax.[[2]](https://en.wikipedia.org/wiki/IBM_RPG#cite_note-2) The most recent version is **RPG IV**, which includes a number of modernization features, including free-form syntax

RPG 3 had below specs

**H :** Control specification . This is also present in next version of RPG which is RPGLE/RPGIV

**F :** File specification. This is also present in next version of RPG which is RPGLE/RPGIV

**E :** Extension spec(s) are next, and describe arrays and tables, which may be prefetched from disk files (an Input table), drawn from constants placed at the end of the source between \*\* and /\* symbols, or built from calculations. This is not present in RPGLE/RPGIV. Arrays are defined in D-SPEC

**L:** Line Counter spec(s) are next, and if present, describe the form to be printed. It defines the number of lines in a page and the positions where printing begins and ends. This is not present in RPGLE/RPGIV

I : Input specs are next, and describe the data areas within files. RPG II permits redefinition of data areas so that a field named FLDA might occupy the same area as an array AR that contains 8 elements of 1 character each. Non-record areas such as data structures can be described. Depending on the values of the input record, indicators may be conditioned.

This is not present in RPGLE/RPGIV

C  Calculation spec(s) are next. Total fields may be described and accumulated. Complex computations and string manipulations are possible. Indicators may be conditioned.

This is present in RPGLE/RPGIV

O : Output specifications, which describe the output record in terms of fields and output positions.

This is present in RPGLE/RPGIV

In rpgle/rpgiv new specification for field definition got added which is D-SPEC. E/L/I spec got dropped in RPGLE

**Z-ADD/Z-SUB :** This is basically used to initialize numeric variables.

Z-ADD opcode will make variable as 0 and then add the value in factor 2 to that.

**Z-ADD 2 A** (Here A will contain value of 2 ).

**Z-SUB** opcode will make variable as 0 and then Subtract the value in factor 2 to that.

**Z-SUB 2 A** (Here A will contain value of -2 )

**ADD/SUB/DIV/MULT :** These opcodes are used for arithmetic addition/subtraction/division/multiplication.

**MOVEL**

The MOVEL operation transfers characters from factor 2 to the result field. Moving begins with the leftmost character in factor 2. You cannot specify resulting indicators if the result field is an array. You can specify them if the result field is an array element, or a non-array field.

[**MOVE (Move)**](http://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/c092508402.htm#ToC_855)

·         The MOVE operation transfers characters from factor 2 to the result field.

·         Moving starts with the rightmost character of factor 2.

·         When moving Date, Time or Timestamp data, factor 1 must be blank unless either the source or the target is a character or numeric field.

·         If factor 2 is longer than the result field, the excess leftmost characters or digits of factor 2 are not moved.

·         If the result field is longer than factor 2, the excess leftmost characters or digits in the result field are unchanged, unless padding is specified.

·         If factor 2 is shorter than the length of the result field, a P specified in the operation extender position causes the result field to be padded on the left after the move occurs.

·         A MOVE operation does not change the length of a variable-length result field.

**ADDUR :-**

The ADDDUR operation adds the duration specified in factor 2 to a date or time and places the resulting Date, Time or Timestamp in the result field.

Duration can be defined as \*M/\*D/\*Y along with factor 2 or extended names like \*MONTHS/\*DAYS/\*YEARS

**Syntax is**

DATE ADDDUR VALUE:DURATION RESULT

Duration can be defined as \*M/\*D/\*Y along with factor 2 or extended names like \*MONTHS/\*DAYS/\*YEARS

re 1. ADDDUR Operations

\*...1....+....2....+....3....+....4....+....5....+....6....+....7...+....

HKeywords+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

H TIMFMT(\*USA) DATFMT(\*MDY&)

DName+++++++++++ETDsFrom+++To/L+++IDc.Keywords+++++++++++++++++++++++++++++

\*

DDateconst C CONST(D'12 31 92')

\*

\* Define a Date field and initialize

\*

DLoandate S D DATFMT(\*EUR) INZ(D'12 31 92')

DDuedate S D DATFMT(\*ISO)

Dtimestamp S Z

Danswer S T

CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq....

\* Determine a DUEDATE which is xx years, yy months, zz days later

\* than LOANDATE.

C LOANDATE ADDDUR XX:\*YEARS DUEDATE

C ADDDUR YY:\*MONTHS DUEDATE

C ADDDUR ZZ:\*DAYS DUEDATE

\* Determine the date 23 days later

\*

C ADDDUR 23:\*D DUEDATE

\* Add a 1234 microseconds to a timestamp

\*

C ADDDUR 1234:\*MS timestamp

\* Add 12 HRS and 16 minutes to midnight

\*

C T'00:00 am' ADDDUR 12:\*Hours answer

C ADDDUR 16:\*Minutes answer

\* Subtract 30 days from a loan due date

\*

C ADDDUR -30:\*D LOANDUE

**SUBDUR** : This will subtract duration from date field .

**Note In RPGIV /RPGLE built in functions of %DAYS/%YEARS/%MONTHS can be used to same purpose as ADDDUR/SUBDUR**

**EXAMPLE**

**ADDDUR 20:\*D DATE**

This will add 20 days to date and place result in same field which is date.

Same thing can be achieved by RPGLE by   
**Eval date = date + %days(20)**

**READ/READE/CHAIN in rpg**

We don’t have extended factor 2 in rpg , factor 2 length is limited to 10 characters.

We also don’t have built-in functions like %eof/%equal/%found in rpg.

We need to use indicators in rpg to determine result of file operation

1. Low indicator will always be used for error tracking.

If indicator specified in Low indicator column is ON , this indicates that operation ended in error.

1. READ/READE: End of file condition can be checked by using equal indicator.
2. CHAIN: We will use %found with chain in rpgle to determine if record is present , in rpg we will determine same thing by using HIGH indicator.

Example

...FFilenameIPEAF........L..I........Device+......KExit++Entry+A....U1........ ...+... 9 ..

\*\*\*\*\*\*\*\*\*\*\*\*\* Beginning of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FTESTPRPGUF E K DISK

C MOVE \*BLANKS A 10 A= ' '

C MOVE \*BLANKS B 10 B = ' '

C MOVEL'dup' C 10

\*

C C CHAINTESTPRPG 55

\*

C \*IN55 IFEQ \*OFF

C MOVE 'B' FIELD1

C UPDATTESTREC

C ENDIF

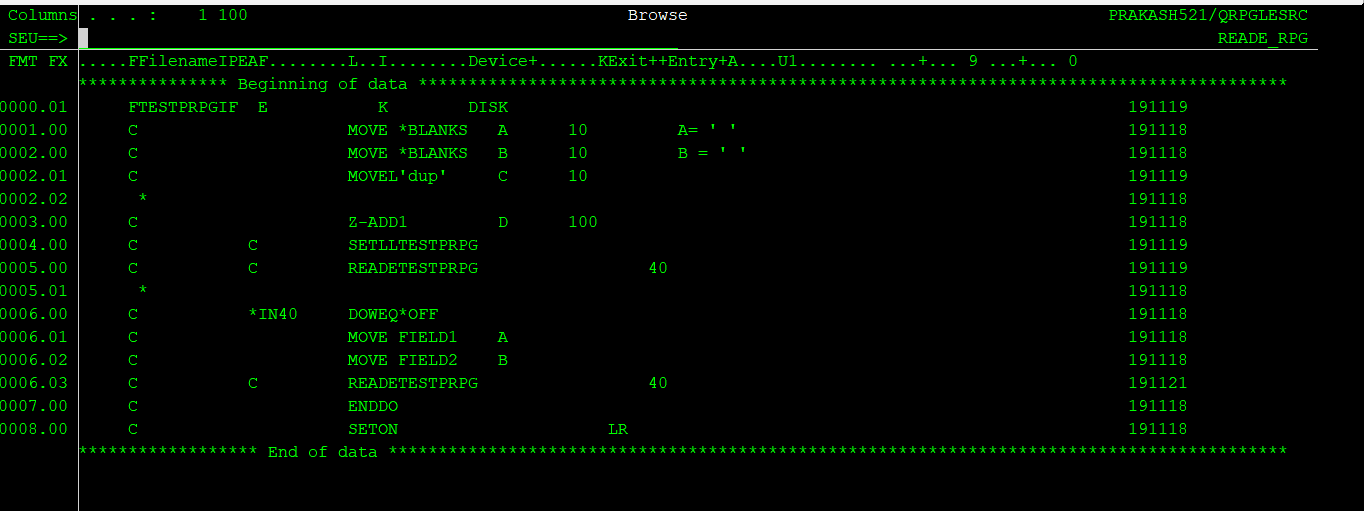
\*

C SETON LR

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Note:-** We don’t have D spec in rpg ,that’s why variable attributes in above program are defined when they are used for first time in MOVE operation

**Below example will show READE operation in rpg**



**For program END set on LR and LR is used in high indicator.**