

# RPGFREE

This repository is meant to be installable on IBM i in a *source-level-distribution* way:

```
PASERIE/INSTALL REPO_OWNER(AndreaRibuoli) REPOSITORY(RPGFREE)
```

It will contain simple RPG programs that are commented here in the **README.md**.

RPG **full** free format is adopted in these examples.

## RPG01

```
**FREE
Dcl-DS DataLog DtaAra(*AUTO : *USRCTL : 'RPGFREE/DATALOG');
  Giorno   Zoned(2 :0);
  Mese     Zoned(2 :0);
  Anno     Zoned(4 :0);
End-DS;
Dcl-S DataISO DATE(*ISO) Inz(d'2023-12-31');
Anno  = %subdt(DataISO:*YEARS);
Mese  = %subdt(DataISO:*MONTHS);
Giorno = %subdt(DataISO:*DAYS);
Dsply ('La data impostata è ' + %char(DataISO) + '.');
*InLR = *ON;
Return;
```

With **Dcl-DS** we are declaring a *Data Structure*: this one is specialized to be based on a *Data Area* (**\*DTAARA**). On its turn this data area is qualified to be **\*AUTO**: this means the data area **RPGFREE/DATALOG** will be created (if not already existing) and will be updated consistently with the values the fields *Giorno*, *Mese* and *Anno* have when the program is ending.

The values are extracted from a *Stand-alone* variable we declare (**Dcl-S**) as **DATE** in the **\*ISO** format. We initialize the variable while declaring it with a **date literal**: **d'2023-12-31'**.

A DATE type can be manipulated via the **%subdt** *built-in-function* (**BIF**). The second parameter possible values are limited by the type of the first one:

type	*MSECONDS	*SECONDS	*MINUTES	*HOURS	*DAYS	*MONTHS	*YEARS
DATE					yes	yes	yes
TIME		yes	yes	yes			
TIMESTAMP	yes	yes	yes	yes	yes	yes	yes

Please note that **\*MSECONDS** stands for **micro**-seconds (not *milli*) and is only available if the type is a **TIMESTAMP** (not a **TIME**!).

The **DSPLY** is an operation code: it accepts two factors and a result field under the following free-form syntax:

```
DSPLY{(E)} {message {message-queue {response}}}
```

```
Dsply ('La data impostata è ' + %char(DataISO) + '.');
```

The parentheses ( ) have nothing to do with the operation code: **factor 1** ( actually the **message** ) will be the result of the concatenation of three elements:

- 'La data impostata è '
- %char(DataISO)
- '.'

A DATE type needs to be first converted into a character one: the **%char** BIF does the job for us assuming our goal is to adopt the format for the date specified in the job setting.

We can programmatically force a specific date format, e.g. **%char(DataISO : \*iso)**

## RPG02

```
**FREE
Dcl-S $data_$$@ DATE(*ISO) Inz(d'2023-12-31');
Dcl-S #data_$$@ DATE(*ISO) Inz(d'2024-01-01');
Dcl-S @data_$$@ DATE(*ISO) Inz(d'2024-01-02');
Dsply ('La prima data impostata è ' + %char($data_$$@) + '.');
Dsply ('La seconda data impostata è ' + %char(#data_$$@) + '.');
Dsply ('La terza data impostata è ' + %char(@data_$$@) + '.');
*InLR = *ON;
Return;
```

This example is only to verify that in addition to alphabetic characters variable names can also begin with \$, #, and @. Subsequent characters can also be numeric and composed of \_.

## RPG03

```
*FREE
Dcl-S CDCLI CHAR(6);
Dcl-S RAGSC CHAR(35);
Dcl-S Codice LIKE(CDCLI);
Dcl-S Nome LIKE(RAGSC);
Codice = '000780';
Nome = 'Pluto Srl';
EXEC SQL
    UPDATE ANCL200F
```

```
SET    RAGSC = :Nome
WHERE  CDCLI = :Codice;
*InLR = *ON;
Return;
```

---

This example (**SQLRPGLE** type) verifies the ability to embed SQL statements. Also uses the **LIKE** option to declare a variable adopting another one. It demonstrates the use of **host variable** that appear in the SQL statement prefixed with **:**.

## RPG04

---

```
**FREE
Dcl-S DataISO DATE(*ISO) Inz(d'2023-12-31');
Dsply ('La data è inizializzata a ' + %char(DataISO) + '.');
EXEC SQL SELECT CURRENT_DATE INTO :DataISO FROM SYSIBM/SYSDUMMY1;
Dsply ('La data ora è impostata a ' + %char(DataISO) + '.');
*InLR = *ON;
Return;
```

---

This example uses the **INTO** SQL keyword to use host variable as receiver of the value returned by a **SELECT** statement.

## RPG05

---

```
**FREE
Dcl-DS Data0raISO;
  #Data DATE(*ISO) Inz(d'2023-12-31');
  #Ora TIME(*ISO);
End-DS;
Dsply ('La data è inizializzata a ' + %char(#Data) + '.');
EXEC SQL SELECT CURRENT_DATE, CURRENT_TIME INTO :Data0raISO FROM
SYSIBM/SYSDUMMY1;
Dsply ('La data ora è impostata a ' + %char(#Data) + '.');
Dsply ('L'ora è stata impostata a ' + %char(#Ora) + '.');
*InLR = *ON;
Return;
```

---

This example uses the **INTO** SQL keyword to use a **DS** host variable as receiver of **the values** returned by a **SELECT** statement. The number and type of fields in the DS need to match those returned by the SELECT statement.