ΥΠΟΛΟΓΙΣΤΕΣ Ι

ΥΠΟΡΟΥΤΙΝΕΣ

Τι είναι μια υπορουτίνα;

- Μια ομάδα εντολών, σχεδιασμένη να εκτελεί έναν ή περισότερους υπολογισμούς
  - Ιδανικές για περιπτώσεις που ο υπολογισμός επαναλαμβάνεται πολλές φορές μέσα στο πρόγραμμα
  - Συντελούν σημαντικά στην καθαρότητα ενός προγράμματος
  - Συντελούν σημαντικά στην μεταβατικότητα ενός προγράμματος

Ποιά η διαφορά συναρτήσεων και υπορουτίνων;

- Είναι και οι δύο υποπρογράμματα
- Γράφονται και οι δύο μετά το τέλος του προγράμματος
- Μπορούμε να χρησιμοποιήσουμε όποια απο τις δύο θέλουμε για οποιαδήποτε πράξη

Συναρτήσεις
- είσοδος: λίστα μεταβλητών
- έξοδος: μεταβλητή εξόδου
- κλήση: ανάθεση σε μεταβλητή

Υπορουτίνες
- είσοδος: λίστα μεταβλητών
- έξοδος: λίστα μεταβλητών
- κλήση: χωρίς ανάθεση (με `CALL`)
- δεν χρειάζεται να δηλώσουμε το όνομά της

Παράδειγμα #1: η πράξη SQ

με συνάρτηση

```
PROGRAM TEST_SQ
      IMPLICIT NONE
      DOUBLE PRECISION X, SQ
      WRITE(*,*) 'ΝΟΥΜΕΡΟ;'
      READ(*,*) X


      WRITE(*,*) SQ(X)
END

REAL*8 FUNCTION SQ(X)
      IMPLICIT NONE
      DOUBLE PRECISION X
      SQ = X**2
      RETURN
END
```

με υπορουτίνα

```
PROGRAM TEST_SQ
      IMPLICIT NONE
      DOUBLE PRECISION X
      WRITE(*,*) 'ΝΟΥΜΕΡΟ;'
      READ(*,*) X

      CALL SQ(X)
      WRITE(*,*) X
END

SUBROUTINE SQ(X)
      IMPLICIT NONE
      DOUBLE PRECISION X
      X = X**2
      RETURN
END
```

Παράδειγμα #1Β: η πράξη SQ με υπορουτίνα: με ή χωρίς τροποποίηση

με τροποποίηση

```
PROGRAM TEST_SQ
      IMPLICIT NONE
      DOUBLE PRECISION X
      WRITE(*,*) 'ΝΟΥΜΕΡΟ;'
      READ(*,*) X

      CALL SQ(X)
      WRITE(*,*) X
END

SUBROUTINE SQ(X)
      IMPLICIT NONE
      DOUBLE PRECISION X
      X = X**2
      RETURN
END
```

χωρίς τροποποίηση

```
PROGRAM TEST_SQ
      IMPLICIT NONE
      DOUBLE PRECISION X, Y
      WRITE(*,*) 'ΝΟΥΜΕΡΟ;'
      READ(*,*) X

      CALL SQ(X,Y)
      WRITE(*,*) Y
END

SUBROUTINE SQ(X, Y)
      IMPLICIT NONE
      DOUBLE PRECISION X, Y
      Y = X**2
      RETURN
END
```


Πότε να χρησιμοποιούμε συνάρτηση και πότε υπορουτίνα;

Συνάρτηση
- όταν η πράξη είναι απλή με μία και μόνο έξοδο
- όταν δεν τροποποιούμε τις μεταβλητές εισόδου

Υπορουτίνα
- όταν οι πράξεις είναι πολλές και οι έξοδοι επίσης πολλές
- όταν πρέπει να τροποποιήσουμε τις μεταβλητές εισόδου

Καλές προγραμματιστικές συνήθειες:
- Συνάρτηση: μία πράξη, ένα αποτέλεσμα (ένα νούμερο)
- Υπορουτίνα: γενικευμένο σύνολο πράξεων, πολλά αποτελέσματα (π.χ. τροποποίηση ολόκληρου πίνακα)

Παράδειγμα #2: τροποποίηση πίνακα

Γράψτε υπορουτίνα που να δέχεται μονοδιάστατο πίνακα και να τροποποιεί το κάθε στοιχείο του στο τετράγωνό του

```
SUBROUTINE SQN(X, N)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N)

      DO I = 1, N
            X(I) = X(I)**2
      END DO

      RETURN
END
```


Παράδειγμα #2Β: τροποποίηση πίνακα

Γράψτε πρόγραμμα που να διαβάζει μονοδιάστατο πίνακα και με την **SQN** να τροποποιεί το κάθε στοιχείο του στο τετράγωνό του

```
PROGRAM TEST_SQN
      IMPLICIT NONE
      INTEGER NMAX, N, I
      PARAMETER(NMAX = 1000)
      DOUBLE PRECISION X(NMAX)

      WRITE(*,*) 'ΠΟΣΑ ΣΗΜΕΙΑ ΘΑ ΕΙΣΑΓΕΤΕ;'
      READ(*,*) N
      IF (N > NMAX .OR. N <= 0) STOP
      READ(*,*) (X(I), I = 1, N)

      CALL SQN(X, N)

      WRITE(*,*) 'ΤΑ ΝΕΑ ΣΤΟΙΧΕΙΑ ΤΟΥ ΠΙΝΑΚΑ'
      WRITE(*,*) (X(I), I = 1, N)
END
```

Παράδειγμα #3: υπορουτίνα για ταξινόμηση με μέθοδο φυσαλίδας

```
SUBROUTINE BUBBLE_SORT(X, N)
      IMPLICIT NONE
      INTEGER N, I, J
      DOUBLE PRECISION X(N)

!.....ΓΙΑ ΚΑΘΕ ΘΕΣΗ J
      DO J = N, 2, -1
!............ΒΡΙΣΚΟΥΜΕ ΤΟ ΑΝΤΙΣΤΟΙΧΟ ΜΕΓΙΣΤΟ
            DO I = 1, J-1
                  IF(X(I) > X(I+1))THEN
                        CALL SWAP( X(I) , X(I+1) )
                  END IF
            END DO
      END DO

      RETURN
END
```


Παράδειγμα #3: υπορουτίνα SWAP

```
SUBROUTINE SWAP (X, Y)
      DOUBLE PRECISION X, Y, TEMP

      TEMP = X
      X = Y
      Y = TEMP

      RETURN
END
```


Παράδειγμα #4: πολλαπλασιασμός πινάκων

Γράψτε υπορουτίνα που δέχεται 3 δισδιάστατους τετραγωνικούς πίνακες και τροποποιεί τον τρίτο ώστε να είναι το γινόμενο των δύο πρώτων

```
SUBROUTINE MATRIX_PRODUCT(A, B, C, N)
      IMPLICIT NONE
      INTEGER N, I, J, K
      DOUBLE PRECISION A(N,N), B(N,N), C(N,N)

      DO J = 1, N
            DO I = 1, N
                  C(I,J) = 0
                  DO K = 1, N
                        C(I,J) = C(I,J) + A(I,K) * B(K,J)
                  END DO
            END DO
      END DO

      RETURN
END
```


Παράδειγμα #5: απόσταση διανυόμενη απο σώμα με μεταβλητή ταχύτητα

(ή με άλλα λόγια:
αριθμητική ολοκλήρωση γραμμικής διαφορικής εξίσωσης)

- Έστω ένα σώμα ξεκινάει απο την θέση x=0, και η ταχύτητά του δίνεται απο την παρακάτω σχέση, όπου ο χρόνος *t* μετριέται σε δευτερόλεπτα

$$v(t) = \frac{0.03t - 0.001t^2}{1 + 0.0001t^3}$$

Ποιά απόσταση έχει διανύσει μετά απο Μ δευτερόλεπτα;

Παράδειγμα #5: απόσταση διανυόμενη απο σώμα με μεταβλητή ταχύτητα

Μέθοδος του Euler

- Διαφορική εξίσωση: $\frac{dx(t)}{dt} = v(t)$
- Διακριτοποίηση: $\frac{x(t+\Delta t) - x(t)}{\Delta t} = v(t)$
- Ολοκλήρωση στον χρόνο:

$$x(t+\Delta t) = x(t) + \Delta t \cdot v(t)$$

- Ή σε μορφή αλγόριθμου (για Δt=1sec) $x_{i+1} = x_i + v_i$

Παράδειγμα #5: απόσταση διανυόμενη απο σώμα με μεταβλητή ταχύτητα (1/2)

Γράψτε πρόγραμμα που υπολογίζει την απόσταση σαν συνάρτηση του χρόνου, όταν η ταχύτητα δίνεται από (t σε δευτερόλεπτα):

$$v(t) = \frac{0.03t - 0.001t^2}{1 + 0.0001t^3}$$

```
PROGRAM DISTANCE1
      IMPLICIT NONE
      INTEGER M, T
      DOUBLE PRECISION X, V, VELOCITY

      WRITE(*,*) 'ΠΟΣΑ ΒΗΜΑΤΑ ΘΕΛΕΤΕ;'
      READ(*,*) M
      X = 0
      DO T = 1, M
            V = VELOCITY(T)
            CALL DISTANCE(X, V)
      END DO

      WRITE(*,*) 'Η ΑΠΟΣΤΑΣΗ ΕΙΝΑΙ', X
END
```


Παράδειγμα #5: απόσταση διανυόμενη απο σώμα με μεταβλητή ταχύτητα (2/2)

```
DOUBLE PRECISION FUNCTION VELOCITY ( T )
      IMPLICIT NONE
      INTEGER T

      VELOCITY = (0.03*T - 0.001* T**2)/(1 + 0.0001* T**3)

      RETURN
END
!-------------------------------------------------------
SUBROUTINE DISTANCE (X, V)
      IMPLICIT NONE
      DOUBLE PRECISION X, V

      X = X + V

      RETURN
END
```


Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα

- Έστω Ν σώματα ξεκινάνε απο την θέση x=0 με αρχική ταχύτητα (σε m/sec) που δίνεται απο την σχέση (i=1, N)

$$v(i) = 100\left|\cos(100i)\right|$$

Εάν η κίνηση είναι με σταθερή ταχύτητα, μετά απο Μ δευτερόλεπτα, ποιό είναι μακρύτερα και ποιό κοντύτερα στην αρχή, και ποιές θα είναι αυτές οι αποστάσεις;

- **Τι θα χρειαστούμε;**
  - Δύο πίνακες για απόσταση και ταχύτητα
  - Μια συνάρτηση για να δημιουργήσουμε αρχικές ταχύτητες
  - Μια υπορουτίνα για να εκτελεί ένα βήμα
  - Μια συνάρτηση που γυρνάει την θέση του μεγίστου ενός πίνακα
  - Μια συνάρτηση που γυρνάει την θέση του ελαχίστου ενός πίνακα

Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα (1/5)

- Συνάρτηση για την δημιουργία αρχικών ταχυτήτων

```
DOUBLE PRECISION FUNCTION VELOCITY ( I )
      IMPLICIT NONE
      INTEGER I

      VELOCITY = 100 * ABS ( COS (100 * I) ) )

      RETURN
END
```


Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα (2/5)

- Υπορουτίνα για εκτέλεση ενός βήματος

```
SUBROUTINE RUN_DISTANCE (X, V, N)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N), V(N)

      DO I = 1, N
            X(I) = X(I) + V(I)
      END DO

      RETURN
END
```


Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα (3/5)

- Συνάρτηση για εύρεση θέσης μέγιστου πίνακα

```
INTEGER FUNCTION IMAX (X, N)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N), XMAX

      XMAX = X(1)
      IMAX = 1
      DO I = 2, N
            IF (X(I) > XMAX) THEN
                  XMAX = X(I)
                  IMAX = I
            END IF
      END DO

      RETURN
END
```


Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα (4/5)

- Συνολικό πρόγραμμα

```
PROGRAM RACE
      IMPLICIT NONE
      INTEGER NMAX, M, N, I, IMIN, IMAX
      PARAMETER(NMAX=1000)
      DOUBLE PRECISION DIST(NMAX), VEL(NMAX), VELOCITY

      WRITE(*,*) 'ΠΟΣΑ ΣΩΜΑΤΑ ΤΡΕΧΟΥΝ ΚΑΙ ΓΙΑ ΠΟΣΑ ΒΗΜΑΤΑ;'
      READ(*,*) N, M

      IF (N > NMAX .OR. N <= 0) STOP

      DO I = 1, N
            DIST(I) = 0
            VEL(I) = VELOCITY(I)
      END DO
```


ΣΥΝΕΧΙΖΕΤΑΙ...

Παράδειγμα #6: απόσταση διανυόμενη απο Ν σώματα με σταθερή ταχύτητα (5/5)

- Συνολικό πρόγραμμα

```
      DO I = 1, M
            CALL RUN_DISTANCE(DIST, VEL, N)
      END DO

      WRITE(*,*) 'ΠΡΩΤΟ ΤΟ' , IMAX(DIST,N)
      WRITE(*,*) 'ΜΕ ΑΠΟΣΤΑΣΗ', DIST(IMAX(DIST,N))
      WRITE(*,*) 'ΚΑΙ ΤΑΧΥΤΗΤΑ', VEL(IMAX(DIST,N))
      RETURN
END
```


Παράδειγμα #7: απόσταση διανυόμενη απο Ν σώματα, με κανόνες ταχύτητας

Κίνηση Ν σωμάτων με κανόνες ταχύτητας: σε κάθε βήμα όποιο σώμα έχει ταχύτητα πάνω απο τον μέσο όρο ταχυτήτων, πρέπει να την μειώσει κατά 10%, ενώ όποιο έχει ταχύτητα κάτω απο τον μέσο όρο ταχυτήτων πρέπει να την αυξήσει κατά 10%.

- **Τι θα χρειαστούμε;**
  - Μια συνάρτηση να υπολογίζει τον μέσο όρο
  - Μια υπορουτίνα που να εφαρμόζει τον κανόνα

Παράδειγμα #7: απόσταση διανυόμενη απο Ν σώματα, με κανόνες ταχύτητας (1/2)

- Συνάρτηση για μέσο όρο

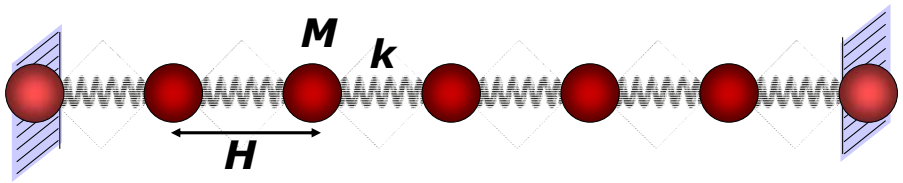
```
DOUBLE PRECISION FUNCTION AVERAGE (X, N)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N)
      AVERAGE = 0
      DO I = 1, N
            AVERAGE = AVERAGE + X(I)
      END DO

      AVERAGE = AVERAGE / N

      RETURN
END
```


Παράδειγμα #7: απόσταση διανυόμενη απο Ν σώματα, με κανόνες ταχύτητας (2/2)

- Υπορουτίνα για εφαρμογή κανόνα

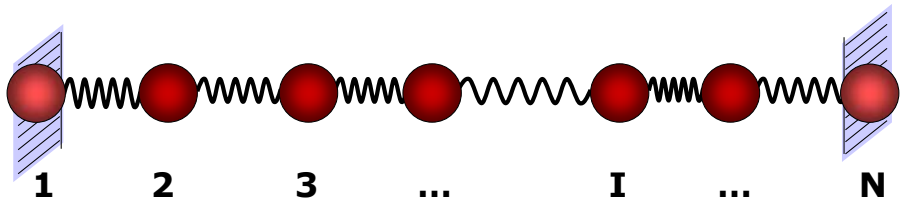
```
SUBROUTINE SCALE_VELOCITY (V, N)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION V(N), AVERAGE, VM

      VM = AVERAGE(V, N)
      DO I = 1, N
            IF (V(I) > VM) THEN
                  V(I) = V(I) * 0.9
            ELSE IF (V(I) < VM) THEN
                  V(I) = V(I) * 1.1
            END IF
      END DO

      RETURN
END
```

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια



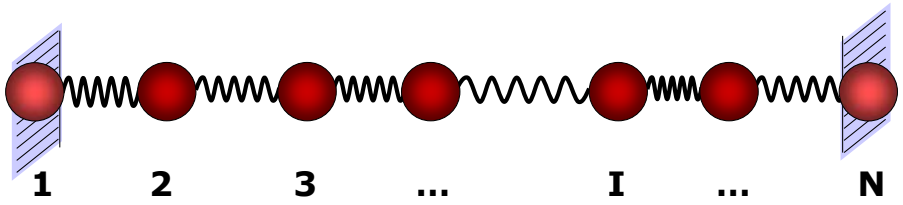
- Γράψτε πρόγραμμα που να λύνει την χρονική εξέλιξη του γραμμικού συστήματος Ν ελατηρίων
- Συνθήκες:
  - Αρχικές θέσεις τυχαίες
  - Αρχικές ταχύτητες μηδέν
  - Πρώτη και τελευταία είναι ακίνητες

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια



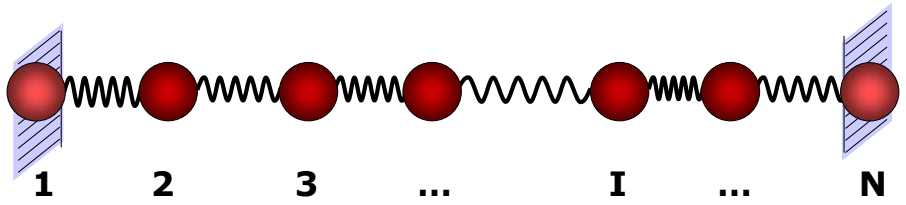
- Κάθε σφαίρα Ι (εκτός πρώτης και τελευταίας), περιγράφεται απο:
  - Την θέση της
  - Την ταχύτητά της
  - Την δύναμη πού νοιώθει εξαιτίας των δύο ελατηρίων της

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια



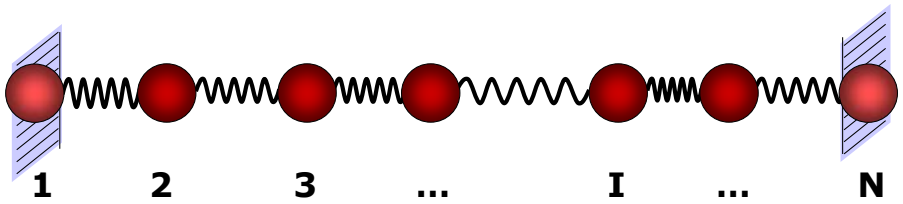
- Αρχικές θέσεις:
  - **X(1) = 0**
  - **X(N) = H * (N - 1)**
  - Οι υπόλοιπες δίνονται απο τον χρήστη
- Αρχικές ταχύτητες
  - Όλες μηδέν

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια



- Δύναμη στη σφαίρα Ι
  - Εάν το **δεξί** της ελατήριο είναι τεντωμένο, η δύναμη είναι πρός τα δεξιά (θετική): `F = K * (X(I+1) - X(I) - H)`
  - Εάν το **δεξί** της ελατήριο είναι συμπιεσμένο, η δύναμη είναι πρός τα αριστερά (αρνητική): `F = K * (X(I+1) - X(I) - H)`
  - Εάν το **αριστερό** της ελατήριο είναι τεντωμένο, η δύναμη είναι πρός τα αριστερά (αρνητική): `F = K * (X(I-1) - X(I) + H)`
  - Εάν το **αριστερό** της ελατήριο είναι συμπιεσμένο, η δύναμη είναι πρός τα δεξιά (θετική): `F = K * (X(I-1) - X(I) + H)`

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια



- Συνολική δύναμη στη σφαίρα I

```
F = K * (X(I+1) - X(I) - H) + K * (X(I-1) - X(I) + H)
```

```
F = K * (X(I+1) + X(I-1) - 2*X(I))
```

και η επιτάχυνση

```
A = K / M * (X(I+1) + X(I-1) - 2*X(I))
```

- **Η δύναμη στην πρώτη και τελευταία είναι μηδέν**

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Ολοκλήρωση
  - **Ανάπτυγμα Taylor για την απόσταση**

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)(\Delta t)^2$$

  - **Ανάπτυγμα για την ταχύτητα**

$$v(t+\Delta t) = v(t) + a(t)\Delta t$$

  - **ΣΗΜΕΙΩΣΗ: εφαρμόζοντας αυτούς τους τύπους επιτυγχάνουμε λύση με ακρίβεια πρώτης τάξης, όχι ικανοποιητική. Με μικρή τροποποίηση μπορούμε να επιτύχουμε ακρίβεια δεύτερης τάξης, αλλά δεν είναι επι του παρόντος.**

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Υπορουτίνα για υπολογισμό επιτάχυνσης

```
SUBROUTINE ACCELERATION (X, A, N, KM)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N), A(N), KM

      DO I = 2, N-1
            A(I) = KM * (X(I+1) + X(I-1) - 2 * X(I))
      END DO

      RETURN
END
```


Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Υπορουτίνα για υπολογισμό νέας ταχύτητας

```
SUBROUTINE VELOCITY (V, A, N, DT)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION V(N), A(N), DT

      DO I = 2, N-1
            V(I) = V(I) + A(I) * DT
      END DO

      RETURN
END
```

Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Υπορουτίνα για υπολογισμό νέας θέσης

```
SUBROUTINE POSITION (X, V, A, N, DT)
      IMPLICIT NONE
      INTEGER N, I
      DOUBLE PRECISION X(N), V(N), A(N), DT

      DO I = 2, N-1
            X(I) = X(I) + V(I) * DT + 0.5 * A(I) * DT**2
      END DO

      RETURN
END
```


Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Πρόγραμμα

```
PROGRAM SPRINGS
      IMPLICIT NONE
      INTEGER NMAX, N, I, J
      PARAMETER(NMAX=1000)
      DOUBLE PRECISION X(NMAX), V(NMAX), A(NMAX), KM, DT

      WRITE(*,*) 'ΠΟΣΑ ΣΩΜΑΤΑ ΤΡΕΧΟΥΝ ΚΑΙ ΓΙΑ ΠΟΣΑ ΒΗΜΑΤΑ;'
      READ(*,*) N, M
      IF (N > NMAX .OR. N < 0) STOP

      WRITE(*,*) 'ΕΛΑΤΗΡΙΟ/ΜΑΖΑ, ΒΗΜΑ'
      READ(*,*) KM, DT
                                                  ΣΥΝΕΧΙΖΕΤΑΙ...
```


Παράδειγμα #8: Γραμμικό σύστημα μαζών με ελατήρια

- Πρόγραμμα

```
      WRITE(*,*) 'ΔΩΣΕ ΑΡΧΙΚΕΣ ΘΕΣΕΙΣ'
      READ(*,*) (X(I), I = 1, N)
      DO I = 1, N
            V(I) = 0
      END DO

      DO J = 1, M
            CALL ACCELARATION(X, A, N, KM)
            CALL POSITION(X, V, A, N, DT)
            CALL VELOCITY(V, A, N, DT)
            WRITE(*,*) J, (X(I), I = 1, N)
      END DO

END
```