Πανεπιστήμιο Αιγαίου

Εισαγωγή στην Πληροφορική

Ενότητα 8: Αλγόριθμοι

Ανδρέας Παπασαλούρος

Τμήμα Μαθηματικών

Σάμος, Μάϊος 2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Η μελέτη των αλγορίθμων αποτελεί το κύριο αντικείμενο της Επιστήμης των Υπολογιστών. Παραδείγματα προβλημάτων των οποίων λύσεις είναι αλγόριθμοι είναι τα ακόλουθα:

- Με δεδομένο ένα ακέραιο αριθμό η εύρεση αν αυτός είναι *πρώτος* ή όχι.
- Εύρεση του *αντίστροφου* ενός πίνακα.
- Εύρεση της τετραγωνική ρίζας ενός αριθμού.
- Με δεδομένο έναν χάρτη με τις αποστάσεις μεταξύ πόλεων που συνδέονται οδικώς, η εύρεση της *συντομότερης* διαδρομής μεταξύ δύο πόλεων.
- Με δεδομένη μια εικόνα ενός αντικειμένου σε ψηφιακή μορφή ο προσδιορισμός των ορίων αυτού του αντικειμένου (B detection).
- Με δεδομένη τη δομή μιας πρωτεΐνης, η τρισδιάστατη απεικόνισή της στο χώρο (protein folding).

Όλα τα παραπάνω προβλήματα αυτά σχετίζονται με *επεξεργασία δεδομένων*.
Ένας αλγόριθμος είναι μια περιγραφή μιας λύσης για μια κατηγορία *προβλημάτων* με τη μορφή μιας ακολουθίας *βημάτων*.
Μια τέτοια περιγραφή πρέπει να είναι *αποτελεσματική*.
Ένας αλγόριθμος αποτελεί μια *αφηρημένη λύση*
Η λύση έχει τη μορφή μιας περιγραφής συγκεκριμένων *βημάτων*
Ο συνολικός αριθμός βημάτων κατά την εκτέλεση του αλγορίθμου πρέπει να είναι *πεπερασμένος*, με άλλα λόγια, ο αλγόριθμος πρέπει να *τερματίζει*.
Είναι δυνατόν να υλοποιηθεί σε μια από τις γνωστές γλώσσες προγραμματισμού.

Φυσική γλώσσα

Ένας αλγόριθμος είναι δυνατόν να περιγραφεί με διαφορετικούς τρόπους.
Ως παράδειγμα δίνεται ο αλγόριθμος υπολογισμού των ριζών διωνύμου: $ax + b = 0$ όταν είναι γνωστοί οι συντελεστές a και b.
Η περιγραφή του αλγορίθμου είναι δυνατόν να γίνει σε μη δομημένο κείμενο σε *φυσική γλώσσα*:

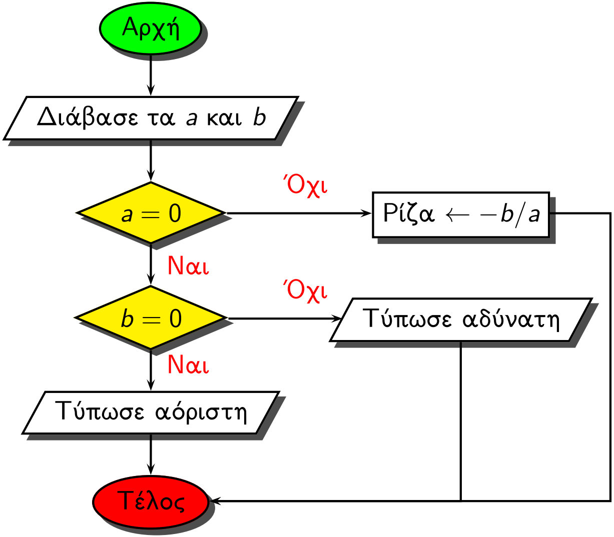
> *Εάν το $a = 0$ τότε αν $b = 0$ η εξίσωση είναι αόριστη αλλιώς είναι αδύνατη. Εάν $a \neq 0$ τότε η εξίσωση έχει μία λύση, $-\frac{b}{a}$.*

Διαγράμματα ροής

συνθήκη	Απόφαση
εντολή	Εντολή
αρχή	Αρχή/Τέλος
Ε/Ε	Είσοδος/Έξοδος

Διαγράμματα ροής

Περιγραφή του αλγορίθμου για τη ρίζα του διωνύμου



Ψευδοκώδικας

Ο ψευδοκώδικας είναι μια μορφή δομημένου κειμένου για την περιγραφή αλγορίθμων.

Δεν θα αναφερθούμε λεπτομερώς σε κανόνες για τη μορφή του ψευδοκώδικα.

Ο τρόπος χρήσης του θα φανεί μέσα από τα παραδείγματα που ακολουθούν.

Σε κάθε περίπτωση, σκοπός είναι η κατά το δυνατόν ακριβής περιγραφή του αλγορίθμου ώστε να είναι δυνατή η κωδικοποίησή του σε κάποια γλώσσα προγραμματισμού και όχι η συμμόρφωση με αυστηρούς κανόνες.

Περιγραφή με χρήση μιας γλώσσας προγραμματισμού

Τόσο τα διαγράμματα ροής όσο και ο ψευδοκώδικας αποτελούν *αφηρημένες* περιγραφές αλγορίθμων.

Ο κώδικας ενός προγράμματος σε κάποια γλώσσα προγραμματισμού ο οποίος *υλοποιεί* έναν αλγόριθμο μπορεί επίσης να χρησιμοποιηθεί και για την *περιγραφή* του αλγορίθμου.

Για το σκοπό αυτό απαιτείται ο κώδικας να είναι *ευανάγνωστος* και να περιέχει επαρκή *σχόλια*.

Αλγόριθμος 1 Εύρεση της ρίζας της εξίσωσης $ax + b = 0$

είσοδος: a, b
έξοδος: Η ρίζα της εξίσωσης $ax + b = 0$

```
Διάβασε τα a και b
εάν a = 0 τότε
  εάν b = 0 τότε
    τύπωσε 'Αόριστη'
  αλλιώς
    τύπωσε 'Αδύνατη'
  τέλος εάν
αλλιώς {a ≠ 0}
  ρίζα ← −b/a
  τύπωσε ρίζα
τέλος εάν
```

Ο αλγόριθμος του Ευκλείδη

Ένας από τους αρχαιότερους αλγορίθμους προτάθηκε από τον Ευκλείδη για την εύρεση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ) δύο θετικών ακέραιων αριθμών. Ο ΜΚΔ είναι ο μεγαλύτερος ακέραιος που διαιρεί ακριβώς και τους δύο αριθμούς.

Αφαιρετικός αλγόριθμος του Ευκλείδη

Πρόταση

Ο ΜΚΔ ενός φυσικού αριθμού, $a \neq 0$, και του 0 είναι το a.

Πρόταση

Για δύο φυσικούς, a, b, ο ΜΚΔ των $|a-b|$ και $min(a,b)$ είναι ίσος με τον ΜΚΔ των a και b.

Σύμφωνα με τις παραπάνω προτάσεις, για να βρούμε τον ΜΚΔ των a και b, με $a > b$, αρκεί να βρούμε τον ΜΚΔ του $|a-b|$ και *του μικρότερου από τα a και b.*
Σε κάθε βήμα, αντικαθιστούμε το a με το $a-b$, αν $a > b$ και το b με το $b-a$ αν $b \leq a$. Η διαδικασία αυτή επαναλαμβάνεται μέχρι το b να γίνει 0, οπότε ο a αποτελεί τον ΜΚΔ.

Παράδειγμα εκτέλεσης του αλγορίθμου δίνεται στον πίνακα που ακολουθεί, στον οποίο φαίνονται οι τιμές των a και b *μετά* το τέλος κάθε επανάληψης. Η επανάληψη με αριθμό 0 αναφέρεται στις *αρχικές τιμές* των a και b.

Επανάληψη	a	b
0	28	16
1	$12 = 28 - 16$	16
2	12	$4 = 16 - 12$
3	$8 = 12 - 4$	4
4	$4 = 8 - 4$	4
5	4	$0 = 4 - 4$

Είδαμε ότι ο αλγόριθμος 2 θα δώσει το σωστό αποτέλεσμα, *αν τερματίσει*. Πρέπει να δούμε αν πράγματι τερματίζει την εκτέλεσή του μετά από πεπερασμένο αριθμό βημάτων. Αρκεί να δείξουμε ότι μετά από έναν αριθμό βημάτων θα γίνει $a = b$. Πράγματι, σε *κάθε* βήμα του παραπάνω είναι προφανές ότι η διαφορά $|a - b|$ μειώνεται. Άρα, μετά από πεπερασμένο αριθμό βημάτων το $a - b$ γίνεται 0 και ο αλγόριθμος τερματίζει.

Αλγόριθμος 2 Ο αφαιρετικός αλγόριθμος του Ευκλείδη

είσοδος: Δύο φυσικοί αριθμοί, a, b
έξοδος: Ο ΜΚΔ

```
Διάβασε τα a και b
εάν a = 0 τότε
    ΜΚΔ ← b
αλλιώς
    όσο b ≠ 0 εκτέλεσε
        εάν a > b τότε
            a ← a − b
        αλλιώς εάν a ≤ b τότε
            b ← b − a
        τέλος εάν
    τέλος όσο
    ΜΚΔ ← a
τέλος εάν
τύπωσε ΜΚΔ
```

Για τη μελέτη κάθε αλγορίθμου ενδιαφέρουν τα ακόλουθα:

- η *ορθότητα* του αλγορίθμου, δηλ. η απόδειξη ότι παράγει το σωστό αποτέλεσμα.
- ο *τερματισμός*, δηλ, η απόδειξη ότι ο αλγόριθμος τερματίζει μετά από μια πεπερασμένη ακολουθία βημάτων.
- ο *χρόνος εκτέλεσης*, δηλ. ο υπολογισμός του αριθμού των βημάτων τα οποία εκτελεί ο αλγόριθμος μέχρι να τερματίσει.

```
PROGRAM GCD1
IMPLICIT NONE
  INTEGER a,b, mkd

  PRINT *,"Δώστε έναν ακέραιο:  "
  READ*, a,b
```

```
  IF (a == 0) THEN
    mkd = b
  ELSE
    DO WHILE (b /= 0)
      IF (a > b) THEN
        a = a - b
      ELSE
        b = b - a
      END IF
    END DO
    mkd = a
  END IF
  PRINT *, mkd
END PROGRAM GCD1
```

Στην προηγούμενη ενότητα περιγράφηκε η αρχική εκδοχή του αλγορίθμου του Ευκλείδη, η οποία βασίζεται σε διαδοχικές *αφαιρέσεις* και γι΄ αυτό ονομάζεται *αφαιρετικός αλγόριθμος του Ευκλείδη*. Ο αλγόριθμος αυτός επιδέχεται τροποποιήσεις οι οποίες τον καθιστούν πιο αποδοτικό, από την άποψη του αριθμού των βημάτων εκτέλεσης. Μια τέτοια τροποποίηση παρουσιάζεται στη συνέχεια. Αυτή η μορφή του αλγορίθμου προτάθηκε αρχικά από τον Ευκλείδη.

Ο τροποποιημένος αλγόριθμος χρησιμοποιεί την πράξη του *ακέραιου υπολοίπου* (modulus) αντί για την πράξη της αφαίρεσης. Βασίζεται στην ακόλουθη πρόταση:

Πρόταση

Για κάθε δύο φυσικούς αριθμούς, a και b, ο ΜΚΔ των $\mathrm{mod}\,(a, b)$ και b είναι ίσος με τον ΜΚΔ των a και b.

Με βάση το παραπάνω, ο αλγόριθμος διαμορφώνεται ως εξής:

Αλγόριθμος 3 Δεύτερη εκδοχή του αλγορίθμου του Ευκλείδη

είσοδος: Δύο φυσικοί αριθμοί, a, b
έξοδος: Ο ΜΚΔ
  Διάβασε τα a και b
  όσο $b \neq 0$ **εκτέλεσε**
    $temp \leftarrow b$
    $b \leftarrow \mod(a, b)$
    $a \leftarrow temp$
  τέλος όσο
  ΜΚΔ $\leftarrow a$
  τύπωσε ΜΚΔ

Ο αλγόριθμος δίνει σε κάθε επανάληψη στο a την τιμή του b και στο b την τιμή του $\mod (a, b)$. Η μεταβλητή *temp* χρειάζεται για την προσωρινή αποθήκευση της τιμής του b. Χωρίς αυτή την αποθήκευση η προηγούμενη τιμή του b θα χανόταν κατά την ανάθεση $b \leftarrow \mod (a, b)$. Το ακέραιο υπόλοιπο της διαίρεσης δύο αριθμών είναι μικρότερο και από τους δύο, άρα σε κάθε επανάληψη η τιμή του b *μειώνεται*. Αυτό εξασφαλίζει ότι μετά από πεπερασμένο αριθμό βημάτων ο αλγόριθμος τερματίζει αφήνοντας τον ΜΚΔ στην μεταβλητή a.

Επανάληψη	a	b
0	28	16
1	16	$12 = \mod(28, 16)$
2	12	$4 = \mod(16, 12)$
3	4	$0 = \mod(4, 4)$

Παρατηρήστε ότι ο αλγόριθμος τερματίζει μετά από 3 βήματα, σε σύγκριση με τα 5 βήματα του αλγορίθμου 2.

Ένα πρόγραμμα Fortran το οποίο υλοποιεί τον αλγόριθμο 3 είναι το ακόλουθο:

```
PROGRAM GCD2
IMPLICIT NONE
  INTEGER a,b, temp

  PRINT *,"Δώστε δύο φυσικούς αριθμούς:  "
  READ*, a,b
  DO  WHILE (b /= 0)
    temp = b
    b = mod(a,b)
    a = temp
  END DO
  PRINT *, a
END PROGRAM GCD2
```

Αριθμητικοί αλγόριθμοι

Στη συνέχεια περιγράφονται μερικοί αλγόριθμοι για την διενέργεια αριθμητικών υπολογισμών. Οι αλγόριθμοι αυτοί κυρίως αναφέρονται στον υπολογισμό *αθροισμάτων*. Να σημειωθεί ότι για πολλά από τα προβλήματα που παρουσιάζονται η λύση είναι δυνατόν να δοθεί με εναλλακτικούς τρόπους, είτε αναλυτικά είτε με χρήση έτοιμων συναρτήσεων βιβλιοθήκης. Εδώ συζητούνται για εκπαιδευτικούς λόγους και για την εξοικείωση με τη λύση παρόμοιων προβλημάτων.

Η τετραγωνική ρίζα ενός αριθμού x είναι δυνατόν να υπολογιστεί σύμφωνα με τη μέθοδο του Newton χρησιμοποιώντας τον τύπο:

$$r_{\text{νέο}} = \frac{1}{2}\left(r + \frac{x}{r}\right)$$

Η παραπάνω σχέση εφαρμόζεται *επαναληπτικά* δίνοντας σε κάθε επανάληψη μια νέα προσέγγιση για τη ρίζα, $r_{\text{νέο}}$, με βάση την προηγούμενη προσέγγιση. r. Ως αρχική προσέγγιση, r, είναι δυνατό να δοθεί οποιαδήποτε τιμή. Σε κάθε επανάληψη η νέα προσέγγιση βρίσκεται πιο κοντά στην πραγματική τιμή της τετραγωνικής ρίζας. Επίσης, σε κάθε επανάληψη, η διαφορά τιμής της νέας προσέγγισης από την προηγούμενη μειώνεται κατ΄ απόλυτη τιμή. Η διαδικασία τερματίζεται όταν η διαφορά της παλιάς από την νέα τιμή, κατά απόλυτη τιμή, είναι μικρότερη από μια μικρή σταθερά, η οποία ορίζει και την ακρίβεια του υπολογισμού.

Με βάση τον παραπάνω αλγόριθμο, ένα πρόγραμμα για τον υπολογισμό της τετραγωνικής ρίζας είναι το ακόλουθο:

```
PROGRAM SquareRoot
IMPLICIT NONE

   REAL    :: X, R, Rnew, Tolerance
   INTEGER :: Count

   READ*, X, Tolerance
```

```
Count = 0
R = X
DO
   Count = Count + 1          ! Αύξηση του μετρητή
   Rnew = 0.5 * (R + X / R) ! Υπολογισμός νέας προσέγγισ

   IF (ABS(R - Rnew) < Tolerance) EXIT ! Αν βρίσκονται
αρκετά κοντά, τερματισμός του βρόχου
   R = Rnew                               ! αλλιώς κρατάμε τη
νέα προσέγγιση
END DO
```

```
   PRINT *, 'Μετά από ', Count, 'επαναλήψεις:'
   PRINT *, 'Η τετραγωνική ρίζα είναι ', Rnew
   PRINT *, 'Με χρήση της συνάρτησης SQRT() η ρίζα είναι
', SQRT(X)

END PROGRAM SquareRoot
```

Η σειρά του Ζήνωνα

Να υπολογιστεί το άθροισμα

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^{i} = 1 + \frac{1}{2} + \frac{1}{4} + \cdots$$

Η σειρά του Ζήνωνα

Παρατηρούμε ότι ισχύει

$$\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = \sum_{i=0}^{\infty} term_i$$

όπου

$term_0 = 1$

$term_{i+1} = term_i \times (\frac{1}{2})$

Με βάση τα παραπάνω μια υλοποίηση είναι η ακόλουθη

```
program zeno1
  implicit none
  real(8) :: sum, term
  real(8), parameter :: e = 1e-5

  sum = 0.
  term = 1.
  do while (term > e)
     sum = sum + term
     term = .5 * term
  end do
  print*, term
end program zeno1
```

Μια δεύτερη προσέγγιση για τον τερματισμό

Η επανάληψη συνεχίζεται για όσο ο υπολογιστής μπορεί να διακρίνει εσωτερικά τις ποσότητες sum και sum + term.
Σε αυτή την περίπτωση το term έχει αρκετά μικρή τιμή.
Η συνθήκη τερματισμού γίνεται:

```
sum = 0; term = 1
do while (sum /= sum + term)
   i = i + 1
   sum = sum + term
   term = .5 * term
end do
```

Το νέο πρόγραμμα γίνεται:

```
program zeno2
  implicit none
  real(8) :: sum, term

  sum = 0.
  term = 1.
  do while (sum /= sum + term)
     sum = sum + term
     term = .5 * term
  end do
  print*, term
end program zeno2
```

Υπολογισμός αναπτυγμάτων Taylor και Maclaurin

Η τιμή μιας συνάρτησης $f(x)$ μπορεί να υπολογιστεί ως το άθροισμα απείρων όρων με βάση τον τύπο Maclaurin:

$$\begin{aligned} f(x) &= f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \ldots + \frac{f^{(n)}(0)}{n!}x^n + \ldots \\ &= \sum_{i=0}^{\infty} \frac{f^{(i)}(0)}{i!}x^i \end{aligned}$$

Η συνάρτηση $ln(x)$

Για τη συνάρτηση $f(x) = ln(x)$ η σειρά Maclaurin είναι η ακόλουθη:

$$ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots - \frac{x^n}{n} - \cdots, |x| < 1$$

Ο $n-$οστός όρος, *term*, υπολογίζεται στη $n-$οστή επανάληψη ως εξής: $term = \frac{x^n}{n}$. Σε κάθε επανάληψη, ο όρος αυτός προστίθεται στην (ή αφαιρείται από την) προηγούμενη τιμή του αθροίσματος των όρων, *sum*. Το άθροισμα των όρων αρχικά είναι μηδέν.

Σε περιπτώσεις απείρων σειρών οι οποίες συγκλίνουν, ο *τερματισμός* της επαναληπτικής διαδικασίας γίνεται όταν ο τελευταίος όρος που υπολογίζεται γίνεται μικρότερος από μια πολύ μικρή τιμή, έστω e. Με βάση τα παραπάνω, ο αλγόριθμος διαμορφώνεται ως εξής:

Προσέξτε ότι στον αλγόριθμο 4 ο βρόχος θα εκτελεστεί *τουλάχιστον* μία φορά. Ένα πρόγραμμα Fortran το οποίο υλοποιεί τον παραπάνω αλγόριθμο δίνεται στη συνέχεια. Το πρόγραμμα ελέγχει αν ο αριθμός x βρίσκεται μέσα στο διάστημα $(-1, 1)$. Το πρόγραμμα υπολογίζει το άθροισμα με ακρίβεια 5 δεκαδικών ψηφίων.

Αλγόριθμος 4 Υπολογισμός αν. Taylor της $ln(1-x)$

είσοδος: Ένας πραγματικός x με $|x| < 1$
έξοδος: Η τιμή $ln(1-x)$

$sum \leftarrow 0$
$n \leftarrow 1$
επανάλαβε
  $term \leftarrow \frac{x^n}{n}$
  $sum \leftarrow sum + \frac{x^n}{n}$
  $n \leftarrow n + 1$
μέχρι $term < e$
$sum \leftarrow -sum$
τύπωσε sum

```
program taylorln
implicit none

  real(8) :: sum, term, x
  real(8), parameter :: E = 1.0D-5
  integer :: n

  print *, ' Δώσε το  x:'; read *, x

  if (abs(x) >= 1 ) then
    print *, ' Πρέπει  |x| < 1'
    stop
  end if
...
```

```
...
  sum = 0.
  n = 1
  do
    term =  x ** n / n
    sum = sum - term
    n = n + 1
    if (term < E) then
      exit
    end if
  end do
  print *, ' ln(1  - ', x, ') = ', sum
 end program taylorln
```

Η συνάρτηση $cos(x)$

Το ανάπτυγμα Taylor της συνάρτησης *cos* δίνεται από τον ακόλουθο τύπο:

$$cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots, \text{για κάθε } x$$

Είναι φανερό ότι το άθροισμα μπορεί να υπολογιστεί με τον ίδιο τρόπο όπως στον αλγόριθμο 4:
Ο $n-$οστός όρος, *term*, υπολογίζεται στη $n-$οστή επανάληψη ως εξής:

$$term_n = \frac{(-1)^n}{(2n)!}x^{2n}$$

Σε κάθε επανάληψη, ο όρος αυτός προστίθεται στην προηγούμενη τιμή του αθροίσματος. Το άθροισμα των όρων αρχικά είναι μηδέν.
Παρ' όλα αυτά, αυτός ο τρόπος υπολογισμού είναι προβληματικός τόσο ως προς την επίδοση όσο και ως προς την ορθότητά του.

Ένας αποτελεσματικότερος αλγόριθμος προκύπτει αν παρατηρήσουμε ότι κάθε όρος, $term_n$, του αθροίσματος, προκύπτει από τον προηγούμενο ως εξής:

$$term_n = term_{n-1}(-1)\frac{x^2}{(2n)(2n-1)}, n > 0$$

με $term_0 = 1$. Ο αλγόριθμος 5 υπολογίζει το ανάπτυγμα της $cosx$ με βάση τα παραπάνω.

Προσέξτε ότι ο όρος *term* αρχικοποιείται στην τιμή 1, ο οποίος είναι ο πρώτος όρος του αθροίσματος, για $n = 0$, ενώ η επανάληψη ξεκινάει από το $n = 1$.
Ένα πρόγραμμα Fortran για τον υπολογισμό του συνημιτόνου δίνεται στη συνέχεια:

Αλγόριθμος 5 Υπολογισμός αναπτύγματος Taylor της *cosx*

είσοδος: Ένας πραγματικός x

έξοδος: Η τιμή *cosx*

$sum \leftarrow 0$

$term \leftarrow 1$

$n \leftarrow 1$

όσο $term \geq e$ **εκτέλεσε**

$sum \leftarrow sum + term$

$term \leftarrow term(-1)\frac{x^2}{(2n)(2n-1)}$

$n \leftarrow n + 1$

τέλος όσο

τύπωσε *sum*

```
program cosine
implicit none
 real (8) :: sum, term, x
 real(8), parameter :: e = 1.D-5
 integer :: n

 print *, "Give a real number: "
 read*, x
...
```

```
...
 sum = 0.0
 term = 1.
 n = 1
 do while (term >= e)
   sum = sum + term
   term = term * (-1) * (x ** 2 / (2 * n * (2 * n - 1)))
   n = n + 1
 end do

 print *, sum

end program cosine
```