

10.1. Προσδοκώμενα αποτελέσματα



Με το κεφάλαιο αυτό κλείνει η ενότητα του προγραμματισμού με χρήση δομημένης γλώσσας προγραμματισμού. Το κεφάλαιο ασχολείται με την έννοια τμηματικού προγραμματισμού, πως δηλαδή αναλύεται το πρόγραμμα σε υποπρογράμματα και τον τρόπο με τον οποίο η ΓΛΩΣΣΑ χειρίζεται τα **υποπρογράμματα**.

Για να αναλύσεις σωστά ένα σύνθετο πρόγραμμα σε υποπρογράμματα πρέπει αρχικά να αποφασίζεις πότε θα χρησιμοποιήσεις **συναρτήσεις** και **διαδικασίες**, να γνωρίζεις τη δομή καθώς και τα βασικά χαρακτηριστικά αυτών των υποπρογραμμάτων. Κάθε γλώσσα προγραμματισμού έχει ελαφρώς διαφορετικό τρόπο με τον οποίο αντιμετωπίζει τα υποπρογράμματα και ειδικά τον τρόπο με τον οποίο χειρίζεται τις παραμέτρους. Στο βιβλίο σου παρουσιάστηκαν θεωρητικά οι αρχές επικοινωνίας των υποπρογραμμάτων με τη χρήση των παραμέτρων, στο εργαστήριο θα γνωρίσεις το συγκεκριμένο τρόπο που το δικό σου προγραμματιστικό περιβάλλον υλοποιεί αυτές τις αρχές και πως χρησιμοποιεί τις παραμέτρους.

Τέλος σε αυτό το κεφάλαιο θα γνωρίσεις τον τρόπο που υλοποιείται η αναδρομή τα πλεονεκτήματα από τη σύνταξη αναδρομικών προγραμμάτων αλλά και τα μειονεκτήματα της σε σχέση με τα επαναληπτικά προγράμματα.

Οι λυμένες ασκήσεις του κεφαλαίου αυτού παρουσιάζονται στο περιβάλλον της ι-δεατής γλώσσας προγραμματισμού ΓΛΩΣΣΑ και επίσης στα πραγματικά προγραμματιστικά περιβάλλοντα Basic και Pascal.

10.2. Επιπλέον παραδείγματα



Παράδειγμα 1

Πολλά από τα προγράμματα που αναπτύχθηκαν στα προηγούμενα κεφάλαια μπορούν να γραφούν καλύτερα με τη χρήση υποπρογραμμάτων. Εδώ θα δούμε το πρόγραμμα που υπολογίζει τα βασικά στατιστικά μεγέθη τη μέση τιμή, την τυπική απόκλιση και τη διάμεσο τιμή που παρουσιάστηκε στο βιβλίο σου στο κεφάλαιο 9.

Το πρόγραμμα χρησιμοποιεί τις εξής διαδικασίες και συναρτήσεις:

Υπολόγισε_MO_ΤυπΑπ :Υπολογίζει τη μέση τιμή και την τυπική απόκλιση ακεραίων αριθμών. Το τμήμα αυτό θα μπορούσε να υλοποιηθεί και με δύο συναρτήσεις, μία για τον υπολογισμό της μέσης τιμής και μίας δεύτερης για τον υπολογισμό της τυπικής απόκλισης.

Ταξινόμηση: Η διαδικασία αυτή ταξινομεί τα στοιχεία του πίνακα χρησιμοποιώντας μία παραλλαγή του αλγορίθμου που παρουσιάστηκε στο βιβλίο σου.

Υπολογισμός_Διαμέσου: Πραγματική συνάρτηση η οποία υπολογίζει τη διάμεσο τιμή.

ΠΡΟΓΡΑΜΜΑ Στατιστική

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ : I, Πλήθος, Στοιχεία[100], Μέγιστο, Διάμεσος,
Άθροισμα, Άθροισμα_2, Βοηθητική

ΠΡΑΓΜΑΤΙΚΕΣ: MO, Τυπ_Απόκλιση

ΑΡΧΗ

! Εισαγωγή στοιχείων

ΓΡΑΨΕ 'Δώσε το πλήθος των αριθμών (μέγιστο 100)'

ΔΙΑΒΑΣΕ Πλήθος

ΓΙΑ I **ΑΠΟ** 1 **ΜΕΧΡΙ** Πλήθος

ΓΡΑΨΕ 'Δώσε έναν αριθμό'

ΔΙΑΒΑΣΕ Στοιχεία[I]

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΚΑΛΕΣΕ Υπολόγισε_MO_ΤυπΑπ(Στοιχεία, Πλήθος, MO, Τυπ_Απόκλιση)

ΚΑΛΕΣΕ Ταξινόμηση(Στοιχεία, Πλήθος)

Διάμεσος <- Υπολογισμός_Διαμέσου(Στοιχεία)

! Εκτύπωση αποτελεσμάτων

ΓΡΑΨΕ 'ΑΠΟΤΕΛΕΣΜΑΤΑ ΓΙΑ ΤΟΥΣ \', Πλήθος, 'ΑΡΙΘΜΟΥΣ'

ΓΡΑΨΕ 'ΜΕΣΟΣ ΟΡΟΣ:', MO

ΓΡΑΨΕ 'ΤΥΠΙΚΗ ΑΠΟΚΛΙΣΗ: \', Τυπ_Απόκλιση

ΓΡΑΨΕ 'ΔΙΑΜΕΣΟΣ:', Διάμεσος

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Στατιστική

ΔΙΑΔΙΚΑΣΙΑ Υπολόγισε_MO_ΤυπΑπ(Πίνακας, N, MO, ΤυπΑποκλ)

! Υπολογισμός μέσου όρου

!Υπολογισμός τυπικής απόκλισης

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: Πίνακας[100], N, I, Άθροισμα, Άθροισμα_2

ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ, ΤυπΑποκλ

ΑΡΧΗ

Άθροισμα <- 0

Άθροισμα_2 <- 0

ΓΙΑ I **ΑΠΟ** 1 **ΜΕΧΡΙ** N

 Άθροισμα <- Άθροισμα+ Πίνακας[I]

 Άθροισμα_2 <- Άθροισμα_2+ Πίνακας[I]^2

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΜΟ <- Άθροισμα/N

ΤυπΑποκλ <- T_P((N* Άθροισμα_2-Άθροισμα^2)/(N*(N-1)))

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Υπολόγισε_ΜΟ_ΤυπΑπ

ΔΙΑΔΙΚΑΣΙΑ Ταξινόμηση(Πίνακας, N)

!Ταξινόμηση των στοιχείων του πίνακα

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: I, N1, T, Βοηθητική

ΑΡΧΗ

N1 <- N

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

T <- 0

ΓΙΑ I **ΑΠΟ** 1 **ΜΕΧΡΙ** N1-1

ΑΝ Πίνακας[I] > Πίνακας[I+1] **ΤΟΤΕ**

 Βοηθητική <- Πίνακας[I]

 Πίνακας[I] <- Πίνακας[I+1]

 Πίνακας[I+1] <- Βοηθητική

 T <- I

ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

N1 <- T

ΜΕΧΡΙΣ_ΟΤΟΥ T=0

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Ταξινόμηση

ΣΥΝΑΡΤΗΣΗ Υπολογισμός_Διαμέσου(A, N) : **ΑΚΕΡΑΙΑ**

ΑΚΕΡΑΙΕΣ: A[100], N

ΑΡΧΗ

ΑΝ N MOD 2 = 0 **ΤΟΤΕ**

 Υπολογισμός_Διαμέσου <- (A[N/2]+A[N/2+1])/2

ΑΛΛΙΩΣ

 Υπολογισμός_Διαμέσου <- A[(N+1)/2]

ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ Υπολογισμός_Διαμέσου

Προγραμματιστικό περιβάλλον Pascal.

```
program Statistiki;
```

```
type list=array[1..100] of integer;
```

```

var
  avg, st_dev:real;
  i,median,n:integer;
  a:list;
{-----}
procedure sort(var a:list;n:integer);
{Ταξινόμηση των στοιχείων του πίνακα A}
var
  i,n1,t,temp:integer;
begin
  n1:=n;
  repeat
    t:=0;
    for i:=1 to n1-1 do
      if a[i]>a[i+1] then
        begin
          temp:=a[i];
          a[i]:=a[i+1];
          a[i+1]:=temp;
          t:=i;
        end;
    n1:=t;
  until t=0;
end;
{-----}
procedure ave_stdev(a:list;n:integer;var average,stddv:real);
{υπολογισμός μέσου όρου και τυπικής απόκλισης}
var
  i:integer;
  sum, sum_2:real;

begin
  sum:=0; sum_2:=0;
  for i:=1 to n do
    begin
      sum:=sum+a[i]; sum_2:=sum_2+sqr(a[i]);
    end;
  average:=sum/n;
  stddv:=sqrt((n*sum_2-sqr(sum))/(n*(n-1)));
end;
{-----}
function med(a:list; n:integer):real;
{υπολογισμός της διαμέσου τιμής}
begin
  if n mod 2=0 then
    med:=(a[n div 2]+a[(n div 2)+1])/2
  else
    med:=a[(n+1) div 2];
end;

```

```

begin {κυρίως πρόγραμμα}
  write('\ΔΩΣΕ ΤΟΝ ΠΛΗΘΟΣ ΤΩΝ ΑΡΙΘΜΩΝ (<100):');
  readln (n);
  for i:=1 to n do
  begin
    write('\ΔΩΣΕ ΤΟΝ \, i:3, 'ο ΑΡΙΘΜΟ :'); readln (a[i]);
  end;
  ave_stdev (a,n,avg,st_dev);
  qsort(a,1,n);
  median:= med(a,n);
  writeln ('Μέση τιμή :', avg, '\Τυπική απόκλιση :',st_dev);
  writeln('\Διάμεσος τιμή :',median);
end.

```

Προγραμματιστικό περιβάλλον Basic.

```

DECLARE FUNCTION Median! (x!(), n!)
DECLARE SUB Sort (k!(), n!)
DECLARE SUB MeanAndStdDev (z!(), m!, s!)
\ Στατιστική
DIM x(100)
DATA 7
DATA 1,5,7,12,9,13,6
READ n
FOR i = 1 TO n: READ x(i): NEXT i
\
CLS
CALL MeanAndStdDev(x(), mx, sx)
CALL Sort(x(), n)
med = Median(x(), n)
\
CLS
PRINT "**** Αποτελέσματα ****"
PRINT "_____ "
PRINT "mx="; mx, "sx="; sx
PRINT "median="; med
END

SUB MeanAndStdDev (z(), m, s)
\ Υπολογισμός μέσης τιμής και τυπικής απόκλισης
s1 = 0: s2 = 0
n = UBOUND(z)
FOR i = 1 TO n
  s1 = s1 + z(i)
  s2 = s2 + z(i) * z(i)
NEXT i
m = s1 / n
s = SQR(s2 / n - m * m)
END SUB

```

```

FUNCTION Median (x(), n)
IF n MOD 2 = 0 THEN
    Median = (x(n / 2) + x (n/2 + 1 ) ) /2
ELSE
    Median = x((n + 1) / 2)
END IF
END FUNCTION

SUB Sort (x(), n) STATIC
k = n
DO
    t = 0
    FOR i = 1 TO k - 1
        IF x(i) > x(i + 1) THEN
            SWAP x(i), x(i + 1)
            t = i
        END IF
    NEXT i
    k = t
LOOP UNTIL t = 0
END SUB

```

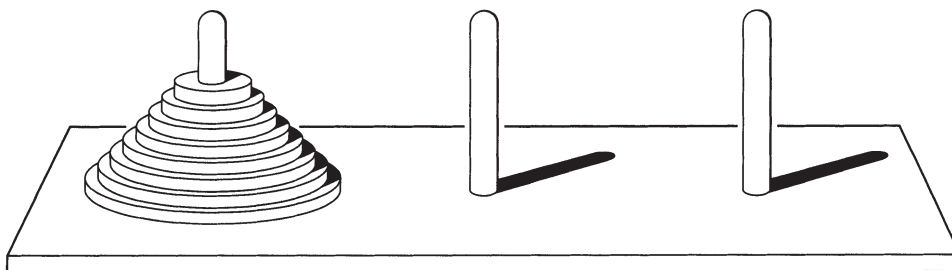
Παράδειγμα 2

Ένα χαρακτηριστικό πρόβλημα το οποίο λύνεται εύκολα με τη χρήση αναδρομής, ενώ είναι πολύ δύσκολο με επαναληπτική διαδικασία είναι οι πύργοι του **Ανόι**.



Στο πρόβλημα των πύργων του Ανόι υπάρχουν τρεις στύλοι και στον πρώτο από αυτούς βρίσκονται περασμένοι δίσκοι διαφορετικής διαμέτρου, έτσι ώστε οι διάμετροι των δίσκων να μικραίνουν από κάτω προς τα πάνω.

Όλοι οι δίσκοι, που βρίσκονται στον πρώτο στύλο, πρέπει να μεταφερθούν στο τρίτο ακολουθώντας τους εξής κανόνες:



- ⇒ Όταν ένας δίσκος μεταφέρεται πρέπει να τοποθετηθεί σε έναν από τους τρεις στύλους.
- ⇒ Μόνο ένας δίσκος μπορεί να μεταφερθεί κάθε φορά και πρέπει να βρίσκεται στην κορυφή του στύλου.
- ⇒ Ένας μεγαλύτερος δίσκος δεν πρέπει να τοποθετηθεί πάνω από ένα μικρότερο.



Σύμφωνα με το μύθο το πρόβλημα δόθηκε στους μοναχούς του ιερού ναού του Μπενάρες. Στους μοναχούς δόθηκε μια χρυσή βάση με τρεις χρυσές βελόνες και εξήντα τέσσερις μικροί χρυσοί δίσκοι. Όταν οι μοναχοί κατορθώσουν να λύσουν το πρόβλημα, δηλαδή να μεταφέρουν τους εξήντα τέσσερις δίσκους από την πρώτη βελόνα στην τρίτη ακολουθώντας τους τρεις κανόνες που αναφέρθηκαν τότε θα έρθει η συντέλεια του Κόσμου (όπως θα δούμε στη συνέχεια δεν διατρέχουμε κανένα κίνδυνο).

Το παιγνίδι είναι σχετικά εύκολο να λυθεί για μικρό αριθμό δίσκων, τρεις-τέσσερις, αλλά δυσκολεύει εξαιρετικά όσο ο αριθμός των δίσκων αυξάνεται.

Η γενική διατύπωση της λύσης όμως με χρήση αναδρομικής διαδικασίας είναι αρκετά απλή και περιγράφεται από τα παρακάτω βήματα:

- ⇒ Αν υπάρχει μόνο ένας δίσκος τότε μεταφέρεται από τον Στύλο1 στο Στύλο3. Το πρόβλημα λοιπόν έχει λύση για $N=1$.
- ⇒ Αν υπάρχουν δύο δίσκοι τότε χρειάζονται τρεις απλές κινήσεις:
 - Ο πρώτος δίσκος από το Στυλο1 μεταφέρεται στο Στύλο2.
 - Ο δεύτερος δίσκος από τον Στύλο2 μεταφέρεται στο Στύλο3.
 - Ο δίσκος από το Στύλο2 μεταφέρεται στο Στύλο3.

Υποθέτουμε ότι η λύση υπάρχει για $N-1$ δίσκους, τότε για N δίσκους η λύση δίνεται αναδρομικά:

- ⇒ Οι $N-1$ δίσκοι μεταφέρονται από τον Στύλο1 στο Στύλο2, χρησιμοποιώντας το Στύλο3 ως βοηθητικό.
- ⇒ Ο τελευταίος δίσκος, που είναι ο τελευταίος άρα και ο μεγαλύτερος, μεταφέρεται από το Στύλο1 στο Στύλο3.
- ⇒ Οι $N-1$ δίσκοι μεταφέρονται από το Στύλο2 στο Στύλο3, χρησιμοποιώντας το Στύλο1 ως βοηθητικό.

Το πρόγραμμα που λύνει τους Πύργους του Ανόι είναι το ακόλουθο:

ΠΡΟΓΡΑΜΜΑ Πύργοι_του_Ανόι

ΣΤΘΕΡΕΣ

Στύλος1='Α'

Στύλος2='Β'

Στύλος3='Γ'

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: N

ΑΡΧΗ

ΓΡΑΨΕ 'δώσε τον αριθμό των δίσκων'

ΔΙΑΒΑΣΕ N

ΚΑΛΕΣΕ Μετακίνηση(N, Στύλος1, Στύλος2, Στύλος3)

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ Μετακίνηση(N, ΣτύλοςΑ, ΣτύλοςΒ, ΣτύλοςΓ)

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: N

ΧΑΡΑΚΤΗΡΕΣ: ΣτύλοςΑ, ΣτύλοςΒ, ΣτύλοςΓ

ΑΡΧΗ

ΑΝ N=1 **ΤΟΤΕ**

ΓΡΑΨΕ 'Μετακίνηση από τον', ΣτύλοςΑ, ' στον' , ΣτύλοςΓ

ΑΛΛΙΩΣ

ΚΑΛΕΣΕ Μετακίνηση(N-1, ΣτύλοςΑ, ΣτύλοςΓ, ΣτύλοςΒ)

ΓΡΑΨΕ 'Μετακίνηση από τον', ΣτύλοςΑ, ' στον' , ΣτύλοςΓ

ΚΑΛΕΣΕ Μετακίνηση(N-1, ΣτύλοςΒ, ΣτύλοςΑ, ΣτύλοςΓ)

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Προγραμματιστικό περιβάλλον Pascal.

```

programanoi;
const
  st1='Α';
  st2='Β';
  st3='Γ';
var
  n:integer;

proceduremove(n:integer;sta, stb, stc:char);
begin
  ifn=1then
    begin
      writeln('ΜΕΤΑΚΙΝΗΣΕ ΑΠΟ ',sta,' ΣΤΟ ',stc)
    end
  else
    begin
      move(n-1,sta, stc, stb);
      writeln('ΜΕΤΑΚΙΝΗΣΕ ΑΠΟ ', sta, ' ΣΤΟ ',stc);
    end
  end
end

```



```

        move (n-1, stb, sta, stc)
    end;
end;

begin

    write ('ΑΡΙΘΜΟΣ ΔΙΣΚΩΝ :'); readln(n);
    move (n, st1, st2, st3);
end.

```

Προγραμματιστικό περιβάλλον Basic.

```

DECLARE SUB Move (n!, a$, b$, c$)
` Anoi
INPUT "N=", n
CALL Move(n, "A", "B", "C")
END

SUB Move (n, a$, b$, c$)
IF n = 1 THEN
    PRINT "Move "; a$; " to "; c$
ELSE
    CALL Move(n - 1, a$, c$, b$)
    PRINT "Move "; a$; " to "; c$
    CALL Move(n - 1, b$, a$, c$)
END IF
END SUB

```

Η εκτέλεση του προγράμματος για 4 δίσκους δίνει τα εξής αποτελέσματα

```

Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Γ στον Α
Μετακίνησε από τον Γ στον Β
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Β στον Α
Μετακίνησε από τον Γ στον Α
Μετακίνησε από τον Β στον Γ
Μετακίνησε από τον Α στον Β
Μετακίνησε από τον Α στον Γ
Μετακίνησε από τον Β στον Γ

```

Η λύση που δόθηκε θεωρητικά επιλύει το πρόβλημα για οποιοδήποτε αριθμό δίσκων. Ας μελετήσουμε όμως τον αριθμό των κινήσεων που απαιτείται.

Όπως είδαμε χρειάζονται 15 κινήσεις για την επίλυση του προβλήματος με 4 δίσκους. Για 5 δίσκους οι κινήσεις που χρειάζονται είναι 31.

Γενικά για N δίσκους η λύση δίνεται μετά από $2^N - 1$ κινήσεις. Αυτό σημαίνει ότι για 10 δίσκους χρειάζονται 1023 κινήσεις και για 20 δίσκους οι κινήσεις ξεπερνούν το εκατομμύριο.

Για τους 64 δίσκους που απασχολεί τους μοναχούς χρειάζονται $1,845 \times 10^{19}$, δηλαδή ένας αριθμός με 20 ψηφία. Πόσο μεγάλος είναι ένας τέτοιος αριθμός; Αν υποθέσουμε ότι εκτελούμε χίλιες κινήσεις το δευτερόλεπτο (όσες περίπου μπορεί να εκτελέσει ένας γρήγορος υπολογιστής), τότε χρειάζονται περίπου μισό εκατομμύριο χρόνια!!!

Η επίλυση του προβλήματος του πύργου του Ανόι αποτελεί χαρακτηριστική περίπτωση αλγορίθμων εκθετικής πολυπλοκότητας ($O(2^N)$).

Οι αλγόριθμοι της μορφής αυτής όπως αναφέρθηκε στο βιβλίο σου στο κεφάλαιο 5 "ανάλυση αλγορίθμων" είναι ουσιαστικά ακατάλληλοι για την πρακτική επίλυση προβλημάτων και χρήσιμοι μόνο για θεωρητικά προβλήματα, αφού κάθε αύξηση του N κατά μία μονάδα διπλασιάζει τον απαιτούμενο χρόνο εκτέλεσης.

10.3. Συμβουλές - υποδείξεις



Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και τις δικές της αρχές για τη σύνταξη και χρήση των υποπρογραμμάτων και πρέπει να μελετήσεις προσεκτικά πως υλοποιούνται τα υποπρογράμματα στο προγραμματιστικό περιβάλλον που χρησιμοποιείς. Υπάρχουν όμως κάποιοι γενικοί κανόνες που πρέπει να ακολουθείς.

- ⇒ Πριν ξεκινήσεις να γράφεις το πρόγραμμα σου να μελετήσεις πώς το πρόγραμμα μπορεί να αναλυθεί σε επί μέρους τμήματα και να αποφασίσεις για τα αντίστοιχα υποπρογράμματα. Χρήσιμο είναι να κάνεις ένα διάγραμμα που θα δείχνει την ιεραρχία ανάμεσα στα υποπρογράμματα. Τότε να αναπτύσσεις τους αλγόριθμους για το κάθε υποπρόγραμμα και στη συνέχεια να γράφεις το πρόγραμμα.
- ⇒ Να μελετάς αν ένα υποπρόγραμμα πρέπει να υλοποιηθεί με διαδικασία ή μπορεί να υλοποιηθεί με συνάρτηση.
- ⇒ Εξέτασε αν κάποια υποπρογράμματα τα οποία έχεις ήδη γράψει ή υπάρχουν σε έτοιμες βιβλιοθήκες προγραμμάτων μπορούν να χρησιμοποιηθούν. Θα γλιτώσεις χρόνο και κόπο.
- ⇒ Κάθε υποπρόγραμμα να προσπαθείς να είναι όσο το δυνατόν πιο ανεξάρτητο από τα άλλα. Αυτό σε προφυλάσσει από λάθη στο πρόγραμμα σου και σου επιτρέπει τη χρήση του σε άλλα προγράμματα αργότερα.

Για να αποφύγεις τα πλέον κοινά λάθη να προσέχεις ιδιαίτερα:

- ⇒ να ορίζεις τον τύπο της συνάρτησης. Οι συναρτήσεις παράγουν μόνο ένα αποτέλεσμα συγκεκριμένου τύπου, ακεραίου, πραγματικού κλπ που πρέπει να ορίζεται.

- ⇒ Να μην υπάρχουν λάθη στην αντιστοίχιση τυπικών και πραγματικών παραμέτρων. Πρόσεξε ότι οι λίστες πρέπει να περιέχουν το ίδιο αριθμό παραμέτρων και κάθε τυπική παράμετρος με την αντίστοιχη πραγματική πρέπει να είναι του ίδιου τύπου.

10.4. Δραστηριότητες - ασκήσεις



Στην τάξη

ΔΤ1. Τι είδους υποπρόγραμμα, διαδικασία ή συνάρτηση, πρέπει να χρησιμοποιήσεις για τα παρακάτω

- A) Εισαγωγή τριών δεδομένων.
- B) Εισαγωγή ενός δεδομένου.
- Γ) Υπολογισμός του μικρότερου από πέντε ακεραίους.
- Δ) Υπολογισμός των δύο μικροτέρων από πέντε ακεραίους.
- E) Έλεγχος αν δύο αριθμοί είναι ίσοι.
- Z) Να ταξινομεί, και να επιστρέφει ταξινομημένους, πέντε αριθμούς.
- H) Έλεγχος αν ένας χαρακτήρας είναι φωνήεν ή σύμφωνο.

ΔΤ2. Να γράψεις τα υποπρόγραμμα που υλοποιούν τα παρακάτω:

- A) Να διαβάσει ένα αριθμό και να επιστρέφει το τετράγωνο του.
- B) Να δέχεται δύο αριθμούς και να επιστρέφει το μικρότερο από δύο αριθμούς.
- Γ) Να δέχεται την τιμή ενός προϊόντος και να υπολογίζει και να τυπώνει την αξία του ΦΠΑ .
- Δ) Να ελέγχει αν ένας αριθμός είναι άρτιος.

ΔΤ3. Να σημειώσεις, στο τετράδιο σου, όλα τα βήματα για τον υπολογισμό του $4!$, τόσο με τη χρήση επαναληπτικής διαδικασίας όσο και με τη χρήση αναδρομικής, σύμφωνα με τα προγράμματα που δίνονται στο βιβλίο σου.

Στο εργαστήριο



Στο προγραμματιστικό περιβάλλον του εργαστηρίου του σχολείου σας:

ΔΕ1. Να γράψεις πρόγραμμα το οποίο θα διαβάσει δύο αριθμούς, θα υπολογίζει το Μέγιστο Κοινό Διαιρέτη (ΜΚΔ) και το Ελάχιστο Κοινό Πολλαπλάσιο (ΕΚΠ) και τέλος θα τυπώνει τα αποτελέσματα.

Υπόδειξη. Για δύο αριθμούς x, y ισχύει: $x \cdot y = ΜΚΔ(x, y) \cdot ΕΚΠ(x, y)$

ΔΕ2. Να εκτελέσεις το πρόγραμμα του παραδείγματος 1.

ΔΕ3. Να γράψεις πρόγραμμα το οποίο να εκτελεί τις τέσσερις πράξεις σε μιγαδικούς αριθμούς.

Για τους μιγαδικούς αριθμούς $\alpha + \beta i$ και $\gamma + \delta i$ έχουμε

$$(\alpha + \beta i) + (\gamma + \delta i) = (\alpha + \gamma) + (\beta + \delta)i$$

$$(\alpha + \beta i) - (\gamma + \delta i) = (\alpha - \gamma) + (\beta - \delta)i$$

$$(\alpha + \beta i) \cdot (\gamma + \delta i) = (\alpha\gamma - \beta\delta) + (\alpha\delta + \beta\gamma)i$$

$$\frac{\alpha + \beta i}{\gamma + \delta i} = \left(\frac{\alpha\gamma + \beta\delta}{\gamma^2 + \delta^2} \right) + \left(\frac{\beta\gamma - \alpha\delta}{\gamma^2 + \delta^2} \right) i$$

Το πρόγραμμα θα οδηγείται από μενού επιλογής όπου ο χρήστης θα επιλέγει το είδος της πράξης.



Στην περίπτωση της διαίρεσης το γ και το δ πρέπει να είναι διάφορα του 0.

ΔΕ4. Να ξαναγράψεις το πρόγραμμα της ΔΕ1 χρησιμοποιώντας αναδρομικές συναρτήσεις.

Σύγκρινε τα δύο προγράμματα.

Στο σπίτι



ΔΣ1. Να γράψεις ένα πρόγραμμα το οποίο διαβάζει τη τιμή βιβλίων σε ΕΥΡΩ και μετατρέπει τις τιμές τους σε Δραχμές, Γερμανικά Μάρκα, Γαλλικά Φράγκα και Ιταλικές λιρέτες. Να χρησιμοποιήσεις για τις μετατροπές τις τρέχουσες ισοτιμίες των νομισμάτων.

ΔΣ2. Να ξαναγράψεις την άσκηση ΔΣ6 του κεφαλαίου 9, τα αποτελέσματα των αγώνων ομίλου του EuroBasket, χρησιμοποιώντας διαδικασίες και συναρτήσεις.



ΔΣ3. Να επεκτείνεις το παράδειγμα 1, ώστε να υπολογίζει την επικρατούσα τιμή, δηλαδή την τιμή που εμφανίζεται περισσότερες φορές.

ΔΣ4. Να γράψεις το πρόγραμμα ΔΕ5 που υπολογίζει τη συνολική χωρητικότητα πυκνωτών και τη συνολική αντίσταση αντιστάσεων με τη χρήση υποπρογραμμάτων.



ΔΣ5. Να γραφεί πρόγραμμα το οποίο να προσθέτει δύο κλάσματα. Το πρόγραμμα δέχεται τέσσερις ακεραίους αριθμούς τους παρανομαστές και τους αριθμητές των δύο κλασμάτων υπολογίζει και εκτυπώνει τον αριθμητή και τον παρανομαστή του αποτελέσματος.

$$A/B + \Gamma/\Delta = E/Z$$

Υπόδειξη

Ενώ το πρόβλημα αρχικά φαίνεται απλό, η υλοποίησή του είναι αρκετά πολύπλοκη. Αρχικά πρέπει να απλοποιηθούν τα κλάσματα, στη συνέχεια να γίνουν ομώνυμα, να προστεθούν οι αριθμητές και τέλος να απλοποιηθεί το αποτέλεσμα.

Οι διαδικασίες αυτές απαιτούν τον υπολογισμό του ΜΚΔ (για την απλοποίηση) και του ΕΚΠ για τη μετατροπή των κλασμάτων σε ομώνυμα. Να χρησιμοποιήσετε τις συναρτήσεις της άσκησης ΔΕ1.



ΔΣ6. Δίνεται η εξίσωση $e^x - 2x - 1 = 0$. Να γραφεί πρόγραμμα το οποίο να βρίσκει μια ρίζα της εξίσωσης αυτής στο διάστημα $[1,2]$ με τη μέθοδο της διχοτόμησης, όπως περιγράφηκε στην παράγραφο 4.3 του βιβλίου σου.

Τεστ αυτοαξιολόγησης**Συμπλήρωσε τα κενά με τη σωστή λέξη που λείπει**

1. Η λίστα των παραμέτρων που υπάρχει στη δήλωση μίας διαδικασίας ονομάζεται λίστα _____ παραμέτρων.
2. Τα δύο είδη υποπρογραμμάτων είναι οι _____ και οι _____.
3. Οι μεταβλητές οι οποίες ισχύουν σε όλα τα υποπρογράμματα ενός προγράμματος και όχι μόνο σε αυτό που ορίστηκε λέγεται ότι έχουν _____ εμβέλεια.

Χαρακτήρισε τα παρακάτω σαν σωστό ή λάθος

4. Μια διαδικασία και μια συνάρτηση μπορούν να εκτελούν ακριβώς τις ίδιες λειτουργίες.
5. Το πλήθος των τυπικών και των πραγματικών παραμέτρων πρέπει να είναι ίδιο.
6. Οι αναδρομικές διαδικασίες είναι πάντοτε προτιμότερες από τις αντίστοιχες επαναληπτικές.
7. Η ενεργοποίηση μίας συνάρτησης γίνεται με την εντολή ΚΑΛΕΣΕ.

Διάλεξε ένα μεταξύ των προτεινόμενων

8. Όταν μία μεταβλητή ισχύει μόνο στο υποπρόγραμμα που ορίστηκε ονομάζεται
Α. Τοπική Β. Καθολική Γ. Παράμετρος Δ. Τυπική
9. Τι θα τυπώσουν οι παρακάτω εντολές

.....

- A <- 5
B <- 10

```

Γ <- 0
ΚΑΛΕΣΕ ΔΙΑΔ1 (Α, Β)
ΓΡΑΨΕ Α, Β, Γ
.....
ΔΙΑΔΙΚΑΣΙΑ ΔΙΑΔ1 (Γ, Δ)
.....
ΑΡΧΗ
Γ <- Γ-Δ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ ΔΙΑΔ1

```

A. 5,10,0 B. 5,10, -5 Γ. -5,10,0 Δ. -5,10,-5

10. Τι θα τυπώσουν οι παρακάτω εντολές

```

.....
Α <- 5
Β <- 10
ΚΑΛΕΣΕ ΔΙΑΔ1 (Β, Α)
ΓΡΑΨΕ Α, Β
.....
ΔΙΑΔΙΚΑΣΙΑ ΔΙΑΔ1 (Α, Β)
.....
ΑΡΧΗ
ΓΡΑΨΕ Α, Β
Α <- Α-Β
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ ΔΙΑΔ1

```

A. 5,10 B. 10,5 Γ. 5,10 Δ. 10, 5
5,10 5,5 -5,10 5,10

Διάλεξε όλα όσα χρειάζεται μεταξύ των προτεινόμενων

11. Ο ορισμός κάθε αναδρομικού υποπρογράμματος περιλαμβάνει
- A. Αναδρομική σχέση.
 - B. Τιμή βάσης.
 - Γ. Υπολογισμό παραγοντικού.
 - Δ. Επαναληπτική διαδικασία.
 - E. Καθολικές μεταβλητές.
12. Μερικά από τα πλεονεκτήματα του τμηματικού προγραμματισμού είναι
- A. Λιγότερος χρόνος για την ανάπτυξη του προγράμματος.
 - B. Ευκολότερη διόρθωση.
 - Γ. Ταχύτητα κατά την εκτέλεση.
 - Δ. Χρήση αναδρομικών διαδικασιών.