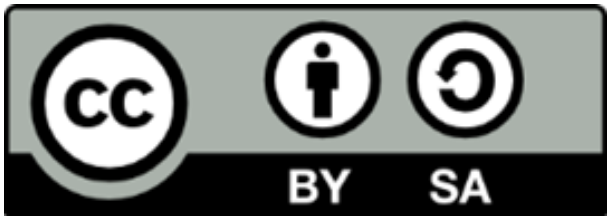
Διαδικαστικός Προγραμματισμός

Ενότητα 2: Εντολές ελέγχου → εντολές υπό συνθήκη

Καθηγήτρια Μαρία Σατρατζέμη
Τμήμα Εφαρμοσμένης Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» 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.

Σκοποί ενότητας

- Να κατανοήσετε τη σχέση μεταξύ εντολών και παραστάσεων.
- Να αναγνωρίζετε και να χρησιμοποιείτε τις σύντομες μορφές εντολών ανάθεσης.
- Να μάθετε να χειρίζεστε λογικά δεδομένα και να εκτιμήσετε τη σπουδαιότητά τους.
- Να κατανοήσετε τη συμπεριφορά των σχεσιακών και λογικών τελεστών.
- Να μάθετε τις λεπτομέρειες των εντολών **if** και **switch** και τις περιπτώσεις στις οποίες χρησιμοποιούμε την κάθε μια.

Κατηγορίες εντολών

- **Απλές εντολές** (simple statements): εντολές που εκτελούν κάποια ενέργεια, όπως για παράδειγμα αναθέσεις τιμών και κλήσεις της συνάρτησης **printf**.

- **Εντολές ελέγχου** (control statements): εντολές που επηρεάζουν τον τρόπο με τον οποίο εκτελούνται άλλες εντολές. Οι εντολές ελέγχου της C χωρίζονται σε δύο βασικές κατηγορίες:
  - **Εντολές υπό συνθήκη**: χρησιμοποιούνται για την επιλογή δύο ή περισσότερων ανεξάρτητων διαδρομών σε ένα πρόγραμμα, ανάλογα με το αποτέλεσμα κάποιου ελέγχου συνθήκης (**if**, **switch**).
  - **Εντολές επανάληψης**: χρησιμοποιούνται όταν χρειάζεται να επαναλαμβάνεται στο πρόγραμμα μια λειτουργία ένα καθορισμένο πλήθος φορών (**for**) ή όσο ισχύει μια συγκεκριμένη συνθήκη (**while**).

Απλές εντολές

- **Κανόνας απλών εντολών** (simple statements): κάθε απλή εντολή αποτελείται από μια παράσταση, η οποία ακολουθείται από ελληνικό ερωτηματικό:

  παράσταση;

- Αν και κάθε παράσταση που ακολουθείται από ελληνικό ερωτηματικό είναι μια **έγκυρη** εντολή της C, αυτό δεν σημαίνει ότι είναι και μια **χρήσιμη** εντολή:

  n1 + n2;

Ενσωματωμένες αναθέσεις τιμών

Ενσωματωμένες αναθέσεις (embedded assignments) ονομάζονται οι αναθέσεις που χρησιμοποιούνται ως μέρος μεγαλύτερων παραστάσεων.

Η παράσταση

(x = 6) + (y = 7)

έχει ως αποτέλεσμα:

- την ανάθεση της τιμής 6 στη μεταβλητή **x**
- την ανάθεση της τιμής 7 στη μεταβλητή **y**
- η παράσταση να πάρει την τιμή 13

Πολλαπλές αναθέσεις

- Ειδική περίπτωση των ενσωματωμένων αναθέσεων
- Χρησιμοποιείται όταν θέλουμε να αναθέσουμε την ίδια τιμή σε περισσότερες από μία μεταβλητές:

n1 = n2 = n3 = 0;

- Η C αποτιμά τους τελεστές ανάθεσης από τα δεξιά προς τα αριστερά:

n1 = (n2 = (n3 = 0));

- Όταν γράφετε πολλαπλές αναθέσεις πρέπει να προσέχετε όλες οι μεταβλητές να είναι του ίδιου τύπου, προκειμένου να αποφεύγετε ενδεχόμενη αυτόματη μετατροπή που μπορεί να οδηγήσει σε απρόσμενα αποτελέσματα:

```
double d;
int i;
d = i = 1.5;
```

} **i = 1 & d = 1.0**

Ιδιωματισμοί σύντομης ανάθεσης

Για οποιοδήποτε διμελή τελεστή **τελ**, η εντολή

μεταβλητή = μεταβλητή τελ παράσταση;

μπορεί να αντικατασταθεί από τη

μεταβλητή τελ= παράσταση;

Παραδείγματα:

balance = balance + deposit;	⟷	**balance += deposit;**
balance = balance - surcharge;	⟷	**balance -= surcharge;**
x = x / 10;	⟷	**x /= 10;**
x = x * 10;	⟷	**x *= 10;**

Μια εντολή ανάθεσης δεν είναι μια μαθηματική εξίσωση, αλλά μια λειτουργία που αποθηκεύει την τιμή της παράστασης που βρίσκεται στα δεξιά του συμβόλου ίσον στη μεταβλητή που βρίσκεται στα αριστερά του.

Τελεστές αύξησης και μείωσης

Η πρόσθεση του 1 σε μια μεταβλητή ονομάζεται **αύξηση** (increment) αυτής της μεταβλητής και μπορεί να επιτευχθεί με τον τελεστή **++**.
Οι παρακάτω εντολές έχουν το ίδιο αποτέλεσμα:

```
x++;
x += 1;
x = x + 1;
```

Η αφαίρεση του 1 από μια μεταβλητή ονομάζεται **μείωση** (decrement) αυτής της μεταβλητής και μπορεί να επιτευχθεί με τον τελεστή **--**.
Οι παρακάτω εντολές έχουν το ίδιο αποτέλεσμα:

```
x--;
x -= 1;
x = x - 1;
```

Λογικά δεδομένα

- Ο τύπος δεδομένων – για τον οποίο οι μόνες έγκυρες τιμές είναι αληθής είτε ψευδής – ονομάζεται **λογικός** ή αλλιώς μιλάμε για **Boolean δεδομένα**.
- Οι περισσότερες σύγχρονες γλώσσες προγραμματισμού διαθέτουν έναν ειδικό λογικό τύπο του οποίου το πεδίο ορισμού περιλαμβάνει τις δύο προαναφερθείσες τιμές.
- Στη C δεν ορίζεται τέτοιος τύπος.
- Ωστόσο, στη βιβλιοθήκη **genlib** ορίζεται ο ειδικός τύπος **bool** και τα ονόματα σταθερών **TRUE** και **FALSE**.
- Η C διαθέτει δύο κατηγορίες τελεστών που μπορούν να χρησιμοποιηθούν με λογικές τιμές:
  - Σχεσιακοί τελεστές
  - Λογικοί τελεστές

Σχεσιακοί τελεστές

Οι **σχεσιακοί τελεστές** (relational operators) χρησιμοποιούνται για τη σύγκριση δύο τιμών και το αποτέλεσμα της σύγκρισης είναι πάντα **TRUE** ή **FALSE**.

Αν υποθέσουμε ότι **x = 10** τότε:

Τελεστής	Όνομα	Παράδειγμα	Αποτέλεσμα
<	Μικρότερο από	x < 4	FALSE
>	Μεγαλύτερο από	x > 4	TRUE
<=	Μικρότερο από ή ίσο με	x <= 4	FALSE
>=	Μεγαλύτερο από ή ίσο με	x >= 10	TRUE
==	Ίσο	x == 5	FALSE
!=	Όχι ίσο – Διάφορο	x != 5	TRUE

Οι σχεσιακοί τελεστές χρησιμοποιούνται μόνο για τη σύγκριση τιμών ατομικών δεδομένων (atomic data), δηλαδή τιμών δεδομένων που δεν αποτελούνται από μικρότερα συστατικά μέρη (όπως τα αλφαριθμητικά που αποτελούνται από μεμονωμένους χαρακτήρες).

Λογικοί τελεστές (1)

- Οι **λογικοί τελεστές** (logical operators) εφαρμόζονται σε λογικούς τελεστέους και το αποτέλεσμα τους είναι μια λογική τιμή:

!	Λογική άρνηση – ΟΧΙ
&&	Λογική σύζευξη – ΚΑΙ
||	Λογική διάζευξη – Ή

- Στον παρακάτω πίνακα που ονομάζεται **πίνακας αλήθειας** (truth table) φαίνεται το αποτέλεσμα κάθε μιας από τις τρεις λογικές πράξεις καθώς μεταβάλλονται οι τιμές των τελεστέων της.

p	q	p && q	p || q	!p
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE

Λογικοί τελεστές (2)

Συνηθισμένες παγίδες: η περίπτωση που φαίνεται να προκαλεί σύγχυση είναι όταν χρησιμοποιείται ένας από τους τελεστές **!** ή **!=** σε συνδυασμό με κάποιον από τους **&&** και **||**, με βασική αιτία το γεγονός ότι η ελληνική γλώσσα έρχεται μερικές φορές σε αντίθεση με τη μαθηματική λογική.

Για παράδειγμα, αν σε κάποιο πρόγραμμα πρέπει να εκφράσετε ότι
“η x δεν ίση είτε με 2 είτε με 3”
και μεταφράσετε απλά τη διατύπωση αυτής της συνθήκης ελέγχου από τα ελληνικά τότε θα προκύψει η έκφραση
x != 2 || x != 3
η οποία έχει πάντα ως αποτέλεσμα **TRUE!!!**

Βοήθεια σε τέτοιες περιπτώσεις παρέχουν κάποιοι κανόνες, γνωστοί ως νόμοι του De Morgan:

!(p || q)	ισοδυναμεί με	**!p && !q**
!(p && q)	ισοδυναμεί με	**!p || !q**

για οποιεσδήποτε λογικές παραστάσεις **p** και **q**.

Πρόωρη αποτίμηση (1)

- Η C ερμηνεύει τους τελεστές **&&** και **||** με διαφορετικό τρόπο απ' ότι άλλες γλώσσες προγραμματισμού.

- Κάθε φορά που ένα πρόγραμμα C αποτιμά οποιαδήποτε παράσταση της μορφής
  - ***παρ1 && παρ2***
    ή
  - ***παρ1 || παρ2***
- οι μεμονωμένες υποπαραστάσεις αποτιμώνται πάντα από τα αριστερά προς τα δεξιά, και η αποτίμηση διακόπτεται τη στιγμή που μπορεί να προσδιοριστεί η απάντηση.

- Αυτό το στυλ αποτίμησης, το οποίο διακόπτεται μόλις γίνει γνωστή η απάντηση, ονομάζεται **πρόωρη αποτίμηση** (short-circuit evaluation).

- Ένα βασικό πλεονέκτημα της πρόωρης αποτίμησης είναι ότι επιτρέπει σε μια συνθήκη να ελέγχει την εκτέλεση μιας δεύτερης.

Πρόωρη αποτίμηση (2)

Ένα βασικό πλεονέκτημα της πρόωρης αποτίμησης είναι ότι επιτρέπει σε μια συνθήκη να ελέγχει την εκτέλεση μιας δεύτερης.

Για παράδειγμα, υποθέστε ότι θέλετε να εκφράσετε τη συνδυασμένη συνθήκη ότι:

(1) η τιμή της ακέραιης μεταβλητής **x** είναι μη μηδενική
(2) η **x** διαιρεί ακριβώς την **y**.

Στη C, μπορείτε να εκφράσετε αυτή τη συνθήκη ελέγχου ως εξής:

(x != 0) && (y % x == 0)

επειδή η παράσταση **y % x** θα αποτιμηθεί μόνο αν η **x** είναι διάφορη του μηδενός.

Η αντίστοιχη παράσταση στην Pascal δεν θα έδινε το σωστό αποτέλεσμα γιατί στην Pascal αποτιμώνται πάντα και τα δύο μέρη της συνθήκης.

Οι συνθήκες που αποτρέπουν σφάλματα αποτίμησης στα επόμενα τμήματα μιας σύνθετης συνθήκης, όπως η συνθήκη

(x != 0)

στο παραπάνω παράδειγμα, ονομάζονται **φύλακες** (guards).

Μεταβλητές σημαίες

Οι μεταβλητές τύπου **bool** χρησιμοποιούνται συχνά στα προγράμματα ως **σημαίες** (flags), για να σηματοδοτήσουν δηλαδή αν έχετε ολοκληρώσει ή όχι κάποια φάση της λειτουργίας.

Στις μεταβλητές σημαίες μπορούμε να αναθέσουμε οποιαδήποτε παράσταση που έχει λογική τιμή. Για παράδειγμα, η εντολή:

done = (itemsRemaining == 0)

«λέει»:

Υπολόγισε την τιμή της **(itemsRemaining == 0)**, η οποία θα είναι είτε **TRUE** είτε **FALSE**, και αποθήκευσε αυτό το αποτέλεσμα στη μεταβλητή **done.**

Εντολές if μίας ή πολλών γραμμών (1)

Η γενική μορφή της εντολής **if** είναι η εξής:

```
if (συνθήκη-ελέγχου)
    εντολή
```

ή

```
if (συνθήκη-ελέγχου)
{
   εντολές
}
```

Λειτουργία/σημασία της εντολής:

- Αν η συνθήκη αποτιμηθεί σε true, δηλαδή αν είναι αληθής, **τότε και μόνο** τότε εκτελείται η εντολή ή οι εντολές που περιλαμβάνονται στο μπλοκ εντολών της if.
- Αν η συνθήκη αποτιμηθεί σε false, δηλαδή αν είναι ψευδής, τότε η εντολή ή οι εντολές που περιλαμβάνονται στο μπλοκ εντολών της if αγνοούνται (δεν εκτελούνται).
- Εκτελείται η εντολή που ακολουθεί μετά την if, είτε η έκφραση ήταν αληθής είτε ψευδής.

Εντολές if μίας ή πολλών γραμμών (2)

```
/* τμήμα κώδικα που ελέγχει-συγκρίνει τις τιμές 2 μεταβλητών (number1 και number2)
   και εμφανίζει σχετικό μήνυμα για τη σχέση που τις συνδέει.
   Για τον έλεγχο χρησιμοποιείται μια αλληλουχία εντολών if. */

int number1, number2;

number1 = 10;
number2 = 20;

if ( number1 > number2 )
     printf("O arithmos %d einai megalyteros apo ton %d.\n", number1, number2);

if ( number1 < number2 )
     printf("O arithmos %d einai mikroteros apo ton %d.\n", number1, number2);

if ( number1 = = number2 )
          printf("Oi arithmoi %d kai %d einai isoi.\n", number1, number2);
```

Εντολές if-else (1)

Η γενική μορφή της εντολής **if/else** είναι η εξής:

```
if (συνθήκη-ελέγχου)
{
        ομάδα_εντολών_1        //εκτελείται αν η συνθήκη είναι αληθής
 }
 else
{
        ομάδα_εντολών_2        //εκτελείται αν η συνθήκη είναι ψευδής
 }
```

Λειτουργία/σημασία της εντολής:

1. Αν η συνθήκη αποτιμηθεί σε true, δηλαδή αν είναι αληθής, τότε εκτελείται η ομάδα_εντολών_1.
   Αν η συνθήκη αποτιμηθεί σε false, δηλαδή αν είναι ψευδής, τότε εκτελείται η ομάδα_εντολών_2.
2. Εκτελείται η εντολή που ακολουθεί μετά την if, είτε η έκφραση ήταν αληθής είτε ψευδής.

Εντολές if-else (2)

```
/* τμήμα κώδικα που ελέγχει-συγκρίνει τις τιμές 2 μεταβλητών (number1 και number2)
   και εμφανίζει σχετικό μήνυμα ανάλογα με το αν είναι ίσοι ή όχι.
   Για τον έλεγχο χρησιμοποιείται μια εντολή if/else. */

int number1, number2;

number1 = 10;
number2 = 20;

if ( number1 = = number2 )
{
    printf("Oi arithmoi %d kai %d einai isoi.\n", number1, number2);
}
else
{
    printf("Oi arithmoi %d kai %d den einai isoi.\n", number1, number2);
}
```

Εντολές if σε μορφή στοίβας (1)

Η γενική μορφή της εντολής **if/else if…else** είναι η εξής:

```
if (συνθήκη_1)
{
        ομάδα_εντολών_1            //εκτελείται αν η συνθήκη_1 είναι αληθής
 }
 else if (συνθήκη_2)     //ελέγχεται αν η συνθήκη_1 είναι ψευδής
        {
                ομάδα_εντολών_2            //εκτελείται αν η συνθήκη_2 είναι αληθής
        }
          …
        else  // σε αυτό το σημείο φτάνουμε αν όλες οι συνθήκες είναι ψευδής
        {
                ομάδα_εντολών_n            /*εκτελείται αν όλες οι συνθήκες -
                                             συνθήκη_1, συνθήκη_2, … - είναι
    ψευδής*/
        }
```

Εντολές if σε μορφή στοίβας (2)

Λειτουργία/σημασία της εντολής:

Αν η συνθήκη_1 αποτιμηθεί σε true, δηλαδή αν είναι αληθής, τότε εκτελείται η ομάδα_εντολών_1 και ο έλεγχος μεταβαίνει μετά την εντολή if/else if…else,

αλλιώς

ελέγχεται η συνθήκη_2 και αν αποτιμηθεί σε true, τότε εκτελείται η ομάδα_εντολών_2 και ο έλεγχος μεταβαίνει μετά την εντολή **if/else if…else**,

αλλιώς

συνεχίζεται ο έλεγχος των εκφράσεων μέχρι να αποτιμηθεί κάποια έκφραση σε true, οπότε εκτελείται η αντίστοιχη ομάδα_εντολών και ο έλεγχος μεταβαίνει μετά την εντολή **if/else if…else**,

αλλιώς

δηλαδή αν όλες οι συνθήκες αποτιμηθούν σε false, τότε εκτελείται η ομάδα εντολών του τμήματος else και ο έλεγχος μεταβαίνει μετά την εντολή **if/else if…else**.

Εντολές if σε μορφή στοίβας (3)

```
/*τμήμα κώδικα που ελέγχει-συγκρίνει τις τιμές 2 μεταβλητών (number1 και
  number2)και εμφανίζει σχετικό μήνυμα για τη σχέση που τις συνδέει.
  Για τον έλεγχο χρησιμοποιείται μια εντολή if/else…if/else. */

int number1, number2;

number1 = 10;
number2 = 20;

if (number1 > number2)
{
    printf("O arithmos %d einai megalyteros apo ton %d.\n", number1, number2);
}
else if (number1 < number2)
    {
            printf("O arithmos %d einai mikroteros apo ton %d.\n", number1, number2);
    }
    else
    {
            printf("Oi arithmoi %d kai %d einai isoi.\n", number1, number2);
    }
```

Αποφυγή πλεονασμού στις λογικές παραστάσεις

Συνηθισμένοι πλεονασμοί:

```
if (done == TRUE) …
```

αντί για

```
if (done) …
```

```
if (itemsRemaining == 0){
        done = TRUE;
}
else {
        done = FALSE;
}
```

αντί για

```
done = (itemsRemaining == 0);
```

Ο τελεστής ?: (1)

Ο τελεστής **?:**, γνωστός ως **ερωτηματικό/άνω και κάτω τελεία** (question-mark colon) σε αντίθεση με οποιονδήποτε άλλο τελεστή της C γράφεται σε δύο μέρη και απαιτεί τρεις τελεστέους:

(συνθήκη) ? παράσταση1 : παράσταση2

Λειτουργία/σημασία της εντολής:

- Αν η συνθήκη αποτιμηθεί σε **TRUE**, δηλαδή αν είναι αληθής, τότε αποτιμάται η παράσταση1 και αποδίδεται ως τιμή σε ολόκληρη την παράσταση.
- Αν η συνθήκη αποτιμηθεί σε **FALSE**, δηλαδή αν είναι ψευδής, τότε αποτιμάται η παράσταση2 και αποδίδεται ως τιμή σε ολόκληρη την παράσταση.

Ο τελεστής ?: (2)

Στην ουσία ο τελεστής ?: είναι μια σύντομη μορφή της εντολής if:

```
if (συνθήκη) {
    value = παράσταση1;
}
else {
    value = παράσταση2;
}
```

Παραδείγματα:

```
max = (x > y) ? x : y
```

αντί για

```
if (x > y){
        max = x;
}
else {
        max = y;
}
```

```
printf("%d item%s found.\n", nItems, (nItems > 1) ? "s" : "");
```

αντί για

```
if (nItems > 1){
} else {
    printf("%d item found.\n", nItems);
}
    printf("%d items found.\n", nItems);
}
```

Πολλαπλή επιλογή: switch (1)

Η γενική μορφή της εντολής **switch** είναι η εξής:

```
switch (παράσταση-ελέγχου)
{
        case τιμή_1:
                ομάδα_εντολών_1     /* εκτελείται αν η τιμή_1 ισούται με την
                                       τιμή την παράσταση ελέγχου */
                break;      /* η εντολή break τερματίζει την εκτέλεση της switch.
                Αν παραληφθεί η break, τότε η ροή εκτέλεσης του προγράμματος θα συνεχίσει
                στην επόμενη case και τελικά θα φτάσει στην default (δίνοντας λάθος
                αποτελέσματα */
        case τιμή_2:
                ομάδα_εντολών_2
                break;
                …
        default:
                ομάδα_εντολών_n
                break;
}
```

Παρατήρηση: η εντολή μπορεί να χρησιμοποιηθεί μόνο για την επιλογή μεταξύ περιπτώσεων οι οποίες προσδιορίζονται με μια ακέραιη σταθερά (ή με μια σταθερά που συμπεριφέρεται ως ακέραιη – όπως ένας χαρακτήρας).

Πολλαπλή επιλογή: switch (2)

Λειτουργία/σημασία της εντολής:

Υπολογίζεται η τιμή της παράστασης που αποτιμάται σε ένα απλό τύπο δεδομένων (ένα αριθμό, ένα αλφαριθμητικό, ή μια τιμή τύπου boolean), και ελέγχεται αν η τιμή_1 ισούται με την τιμή της παράστασης, οπότε εκτελείται η ομάδα_εντολών_1 και εφόσον υπάρχει η εντολή break τερματίζεται η εκτέλεση της **switch**,

αλλιώς

ελέγχεται αν η τιμή_2 ισούται με την τιμή της παράστασης, οπότε εκτελείται η ομάδα_εντολών_2 και εφόσον υπάρχει η εντολή break τερματίζεται η εκτέλεση της **switch**

…

αλλιώς

δηλαδή αν καμία από τις τιμές των εντολών case δεν ισούται με την τιμή της παράστασης, τότε εκτελείται η ομάδα εντολών του τμήματος **default**.

Πολλαπλή επιλογή: switch (3)

```
/* τμήμα κώδικα που ελέγχει την τιμή της μεταβλητής month που αντιπροσωπεύει ένα από
τους 12 μήνες του έτους και εμφανίζει πόσες μέρες έχει ο μήνας αυτός ή το μήνυμα
"Μη έγκυρος αριθμός μήνα" αν η μεταβλητή month δε έχει τιμή στο διάστημα [1..12] */

int month;

printf(“Month ? ”);
month = GetInteger();

switch (month)
{
        case 1: case 3: case 5: case 7: case 8: case 10:
        case 12:
                printf("O %d-os mhnas exei 31 meres.\n", month);
                break;
        case 4: case 6: case 9: case 11:
                printf("O %d -os mhnas exei 30 meres.\n", month);
                break;
        case 2:
                printf("O %d-os mhnas exei 29 meres an to etos einai
                disekto kai 28 an den einai disekto.\n", month);
                break;
        default:
                printf(“Mh egkyros arithmos mhna");
                break;
}
```

Τέλος Ενότητας

Διαδικαστικός Προγραμματισμός
Τμήμα Εφαρμοσμένης Πληροφορικής