Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #4

12 Απριλίου 2024

Παναγιώτης Παύλου

c-programming-24@allos.gr

Άλλες μορφές ελέγχου ροής

Πιο σπάνια χρησιμοποιούμενες μορφές ελέγχου ροής

Παραλλαγές ελέγχου ροής do/while

Κάποιες – λίγες φορές – συμφέρει να **εκτελούνται οι εντολές** του while τουλάχιστον **μία φορά πριν ελεγχθεί η συνθήκη**.

Γι' αυτές τις περιπτώσεις υπάρχει η ακόλουθη μορφή:

```
do {
      // σώμα του do/while
} while( συνθήκη );
```

όπου θέλει προσοχή το σύμβολο του τερματισμού της εντολής (;) που είναι υποχρεωτικό να γραφεί μόνο του μετά την while.

Παραλλαγές ελέγχου ροής switch/case (1/2)

Κάποιες φορές στα πολλαπλά `if/else if/else` οι συγκρίσεις είναι μόνο **ισοτικές** και γίνονται ανάμεσα **στην ίδια παράσταση** σε σχέση με διάφορες, **γνωστές εκ των προτέρων σταθερές τιμές**.

Γι' αυτές τις περιπτώσεις υπάρχει η switch/case η οποία συντάσσεται όπως δίπλα.

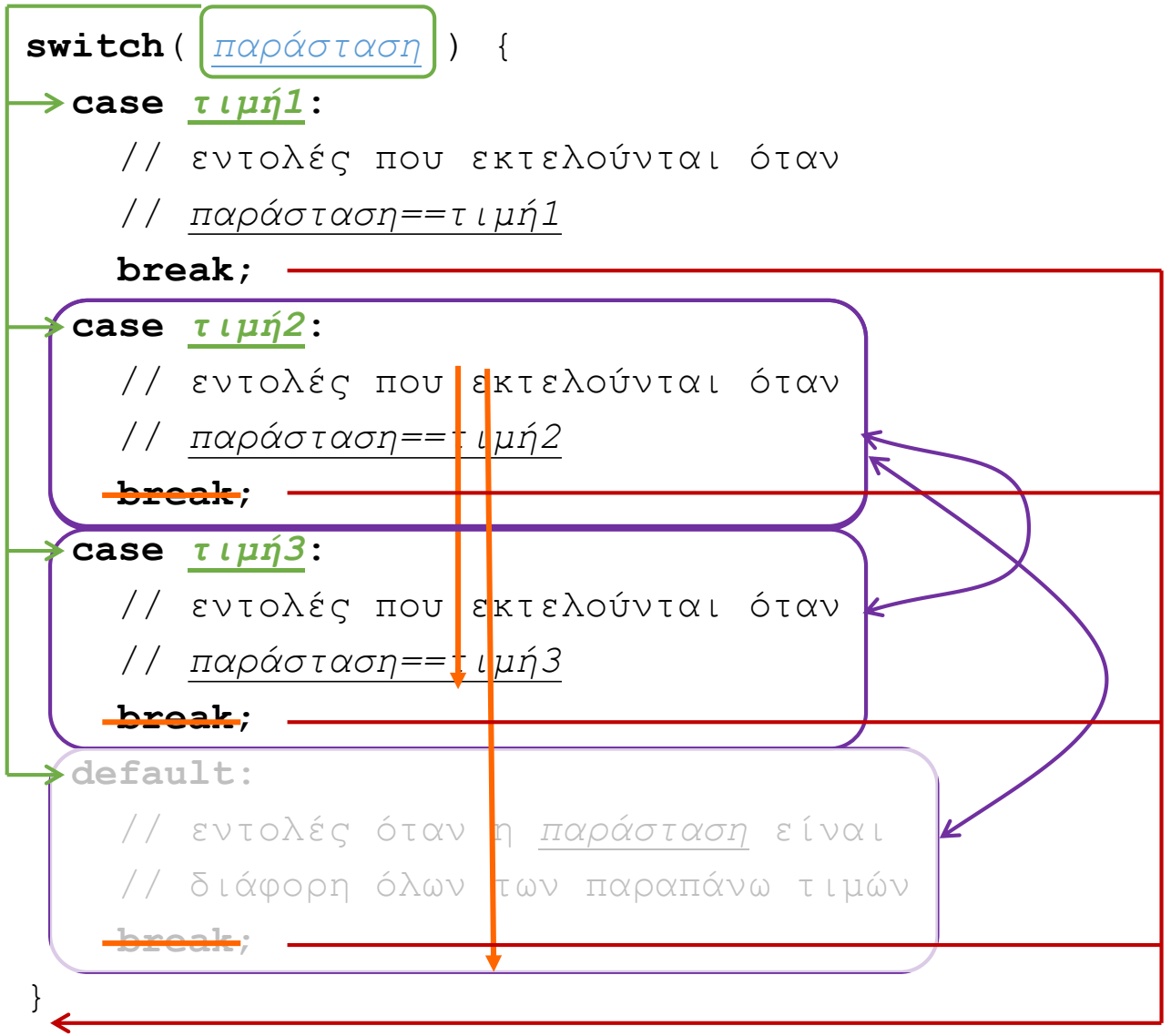
Η διάταξη με την οποία παρουσιάζεται δίπλα η switch/case μοιάζει με τη δομή της `if/else if/else`, όπου το **`default`** αντιστοιχεί στο **`else`**.

```
switch( παράσταση ) {
  case τιμή1:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή1
    break;
  case τιμή2:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή2
    break;
  case τιμή3:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή3
    break;
  default:
    // εντολές όταν η παράσταση είναι
    // διάφορη όλων των παραπάνω τιμών
    break;
}
```

Παραλλαγές ελέγχου ροής switch/case (2/2)

Η switch/case όμως είναι **γενικότερη** και μερικές φορές πιο **δυνατή**, επειδή:

1. Όταν λείπει η **break**, η εκτέλεση συνεχίζεται στις εντολές που ακολουθούν κανονικά στο επόμενο case ή default, μέχρι να βρεθεί κάποιο break ή να κλείσει η switch. Αυτό δεν μπορούσε να γίνει με τα πολλαπλά `if/else if/else`.
2. Η σειρά των **case** δεν παίζει ρόλο, μπορεί να επιλεγεί βάσει του #1
3. Η θέση της **default** δεν παίζει ρόλο , μπορεί να επιλεγεί βάσει του #1
4. Η χρήση της default είναι προαιρετική, όπως και του else



Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Έλεγχος ροής εντός βρόχου

Έλεγχος με τις break και continue – Ατέρμονες βρόχοι

break & continue

Οι εντολές break και continue μας επιτρέπουν να ελέγχουμε την εκτέλεση ενός βρόχου, από το σώμα του, ανεξάρτητα από τη συνθήκη του. Δείτε πως συντάσσονται και πως λειτουργούν:

```
for (int i=0; i<10; i++) {
    if (i==5) {
        break;
    }
    printf("%d ", i);
}
```

0 1 2 3 4 5 6 7 8 9

```
for (int i=0; i<10; i++) {
    if (i==5) {
        continue;
    }
    printf("%d ", i);
}
```

0 1 2 3 4 5 **6 7 8 9**

Προφανώς αυτές, εντός των βρόχων, θα τις συναντάμε πάντα μέσα σε κάποιο if. Επίσης σε εμφωλευμένους βρόχους, το break και το continue αφορά το loop στο οποίο γράφεται.

Ατέρμονες βρόχοι

Τώρα που υπάρχει η δυνατότητα να σταματά η εκτέλεση ενός βρόχου μέσα από το σώμα του, έχει πλέον νόημα η χρήση ενός βρόχου με συνθήκη που είναι πάντα αληθής. Δηλαδή ενός ατέρμονος βρόχου.

Οπότε πλέον η εκτέλεσή του μπορεί να τερματιστεί από διαφορετικές συνθήκες και σε διαφορετικά σημεία.

```
while (true) {
    ...
    if (...) {
        break;
    }
    ...
    if (...) {
        break;
    }
    ...
}
```

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Βιβλιοθήκη του μαθήματος

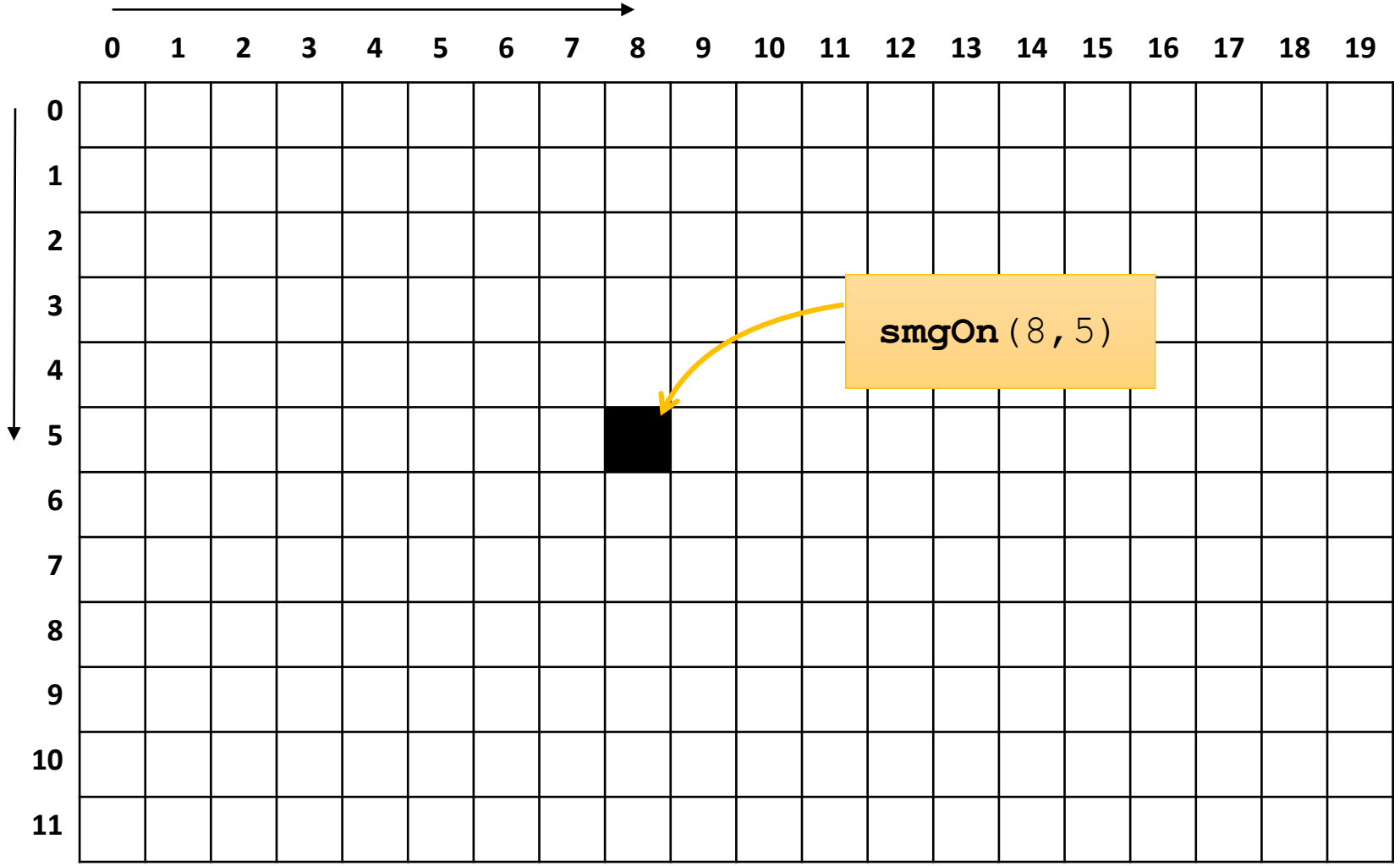
smLib : Η βιβλιοθήκη που θα μας βοηθά στο μάθημα

Εκπαιδευτικό πλέγμα / Grid

Το πλέγμα είναι μία διάταξη πίνακα στην οποία μπορούμε να «ανάβουμε» και να «σβήνουμε» τα στοιχεία του. Οι συντεταγμένες μετράνε από πάνω αριστερά, όπου βρίσκεται το κελί 0,0. Επίσης αυξάνονται προς τα δεξιά και προς τα κάτω, δηλαδή η κατακόρυφη διεύθυνση είναι ανάποδη από το κλασσικό x,y σύστημα συντεταγμένων.

Οπτική αναπαράσταση grid

Οπτικά το πλέγμα είναι όπως ο παρακάτω πίνακας.



Εντολές για το grid

Για να ορίσετε τις **διαστάσεις** του πλέγματος καλείτε την συνάρτηση

`smGrid(πλατος, ύψος)`

Συνήθως θα είναι η 1η εντολή από αυτές που σχετίζονται με το grid. Εάν την εκτελέσετε 2η φορά αργότερα, τότε αλλάζει το μέγεθος του grid, αλλά σβήνει όλο το προηγούμενο περιεχόμενο.

Εάν δεν κληθεί η συνάρτηση αυτή, τότε δεν υπάρχει πλέγμα.

Για να **ανάψετε** ή να **σβήσετε** ένα κελί καλείτε την

`smgOn(x, y)` *ή* **`smgOff(x, y)`**

Όπου x και y είναι οι οριζόντια και κατακόρυφη συντεταγμένη αντίστοιχα. Η αρίθμηση ξεκινά από το 0.

Εντολές για το grid

Για να δείτε την κατάσταση ενός κελιού καλέστε την

`smgTest(x, y)`

η οποία επιστρέφει **αληθές** εάν το κελί είναι αναμένο ή **ψευδές** εάν είναι σβηστό. Εάν κληθεί για σημείο εκτός πλέγματος, τότε επιστρέφεται πάντα ψευδές.

Επίσης για αν δείτε τις διαστάσεις του grid, χρησιμοποιείστε τις

`smgWidth()`

και

`smgHeight()`

που επιστρέφουν την αντίστοιχη διάσταση.

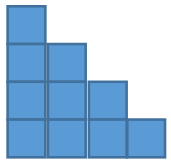
Το grid και τα tests της C*

- Επειδή στη σελίδα εκτέλεσης της απλοποιημένης C, εκτελούνται και δοκιμές μετά την εκτέλεση της main, όταν οι δοκιμές αυτές καλούν την smGrid για να δημιουργήσουν ένα πλέγμα δοκιμών, τότε ό,τι πλέγμα έχετε σχεδιάσει εσείς στη main διαγράφεται. Γι'αυτό όταν θέλετε να δείτε μόνο το αποτέλεσμα της main ως προς το πλέγμα, να χρησιμοποιείτε το κουμπί "Run Main Only".
- Κάποια tests ελέγχουν το «οπτικό» αποτέλεσμα που υπάρχει στο grid, όπως αυτό διαμορφώνεται από τον κώδικά σας, σε σχέση με το πώς θα έπρεπε να είναι. Γι'αυτό εάν το αποτέλεσμα αυτό δεν είναι σωστό. Σταματά σε εκείνο το σημείο η εκτέλεση των ελέγχων ώστε να μπορείτε να δείτε (με τη βοήθεια του υπομνήματος που υπάρχει κάτω από το πλέγμα) τι πήγε λάθος.

Πρακτική 1

Δημιουργήστε τα ακόλουθα σχήματα στο πλέγμα.

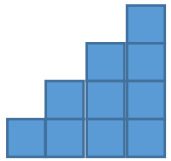
Πρώτα το:

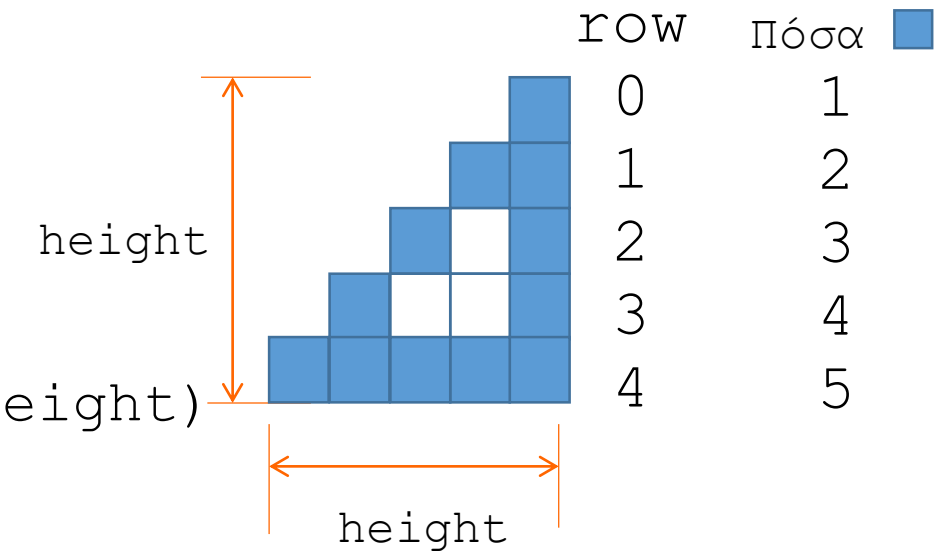


μέσω μιας συνάρτησης

```
function drawTriangle(x, y, height)
```

και ύστερα το:



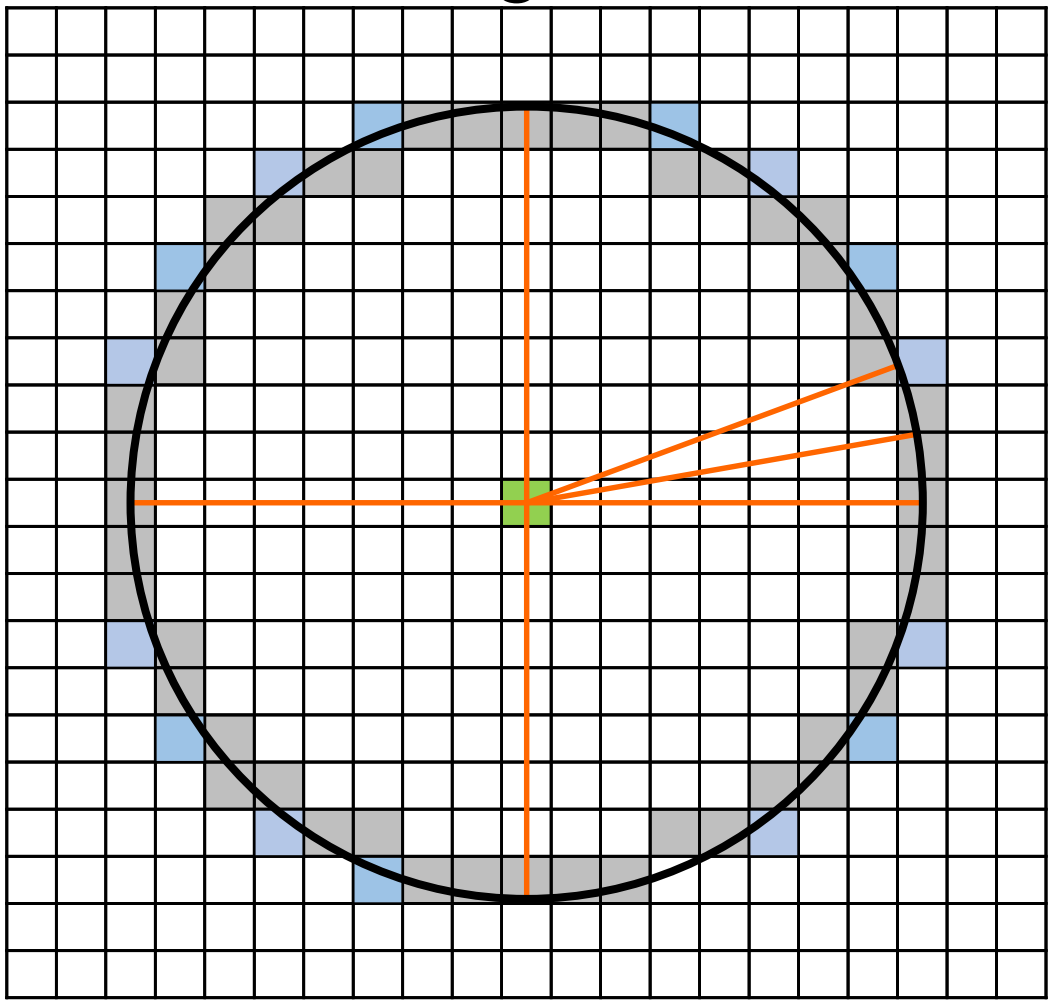


μέσω επέκτασης της που να παίρνει σαν 2η παράμετρο το `right` (true/false).

Επέκταση της συνάρτησης με 3η παράμετρο `outline` που να σχεδιάζει το περίγραμμα του τριγώνου.

Πρακτική 2

Σχεδιάστε έναν κύκλο στο grid



Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Πρόσθετες δυνατότητες

Global εμβέλεια μεταβλητών – Σταθερές τιμές

Global εμβέλεια μεταβλητών 1/3

Ο πρώτος μηχανισμός είναι η χρήση μεταβλητών έξω από όλες τις συναρτήσεις και – κατά συνέπεια – έξω από όλα τα block εντολών. Αυτές οι μεταβλητές – σε αντίθεση με τις τοπικές (local) – ονομάζονται global (καθολικές) και μπορούν να χρησιμοποιηθούν από οποιοδήποτε σημείο του κώδικα.

Κάποιες εφαρμογές τέτοιων μεταβλητών είναι η αποθήκευση κοινόχρηστης πληροφορίας όπως είναι ρυθμίσεις του προγράμματος, γενικοί μετρητές ή σε άλλη περίπτωση κάποια πληροφορία που θα εξαρτάται από το εκάστοτε πρόβλημα που επιλύεται.

```
let someGlobalCounter = 0;
function func1() {
    someGlobalCounter++;
    ...
}
function func2() {
    someGlobalCounter--;
    ...
}
```

Global εμβέλεια μεταβλητών 2/3

Η χρήση των μεταβλητών επιτρέπεται από το σημείο της δήλωσής τους και κάτω. Αυτό παίζει ρόλο όταν δηλώσετε μια global μεταβλητή ανάμεσα στις συναρτήσεις. (Σημειώστε ότι στην απλοποιημένη C δεν υπάρχει αυτός ο περιορισμός).

Η αρχική τιμή (όταν αναγράφεται) δίνεται πριν την εκτέλεση της πρώτης εντολής της main και πρέπει να είναι τιμή γνωστή κατά την έναρξη της εκτέλεσης του κώδικά σας. Δεν μπορεί να είναι το αποτέλεσμα της κλήσης μιας συνάρτησης. (Και πάλι σημειώστε ότι στην απλοποιημένη C δεν υπάρχει αυτός ο περιορισμός).

Ενναλακτικά η αρχικοποίησή μιας global μεταβλητής μπορεί να γίνεται σε όποιο σημείο του κώδικα κρίνει κατάλληλο ο προγραμματιστής.

Global εμβέλεια μεταβλητών 3/3

Προσοχή!

Η χρήση Global μεταβλητών δεν αποτελεί καλή πρακτική και πρέπει να γίνεται με μέτρο *εκεί που πραγματικά χρειάζεται*!

Αλλιώς σύντομα θα βρεθείτε να ξοδεύετε τον χρόνο σας στο debugging περίπλοκων σφαλμάτων.

Η χρήση της ελάχιστης δυνατής εμβέλειας των μεταβλητών είναι φίλος του κάθε προγραμματιστή!

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Σταθερές τιμές

Κάποιες φορές στον κώδικα πρέπει να χρησιμοποιήσουμε σταθερές τιμές π.χ. το $\boldsymbol{\pi}$ ή το $\boldsymbol{e}$ ή και άλλες – μη μαθηματικές – που εξαρτώνται από το εκάστοτε πρόβλημα που αντιμετωπίζουμε.

Αν και μπορούμε να επαναλαμβάνουμε σε κάθε σημείο του κώδικα τις τιμές αυτές, θα είναι πολύ καλύτερο να έχουμε ονοματισμένη την τιμή ώστε να φαίνεται και τι σημαίνει ο αριθμός αυτός για το πρόβλημά μας.

Η C έχει δύο τρόπους να δημιουργήσει σταθερές, ενώ η C* μόνο τον έναν από αυτούς.

Ο τρόπος αυτός είναι με τη χρήση της λέξης κλειδί **`const`** αντί της **`let`** κατά τη δήλωση μιας μεταβλητής, η οποία ουσιαστικά δεν επιτρέπει άλλη εκχώρηση τιμής στην μεταβλητή αυτή. Π.χ.

```
const N = 10;
```

Δηλαδή δεν έχουμε απλά ονοματίσει τη σταθερή ποσότητα, αλλά έχουμε μία μεταβλητή που την περιέχει και η «μεταβλητή» αυτή είναι «προστατευμένη» από αλλαγές.

Σχεδόν πάντα μια τέτοια σταθερά δηλώνεται έξω από τις συναρτήσεις ως global ώστε όλος ο κώδικάς μας να έχει πρόσβαση σε αυτή.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!