Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

2α

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #2

2 & 3 Μαρτίου 2023

Παναγιώτης Παύλου

c-programming-23@allos.gr

Ροή εκτέλεσης προγράμματος

Πως εκτελούνται οι εντολές ενός προγράμματος σε C

Ροή εκτέλεσης εντολών

Σε όλες τις τυπικές γλώσσες προγραμματισμού, η εκτέλεση των εντολών γίνεται **σειριακά** όπως διαβάζουμε, δηλαδή από πάνω προς τα κάτω και στην ίδια γραμμή, από αριστερά προς τα δεξιά.

Ο όρος σειριακά σημαίνει ότι οι εντολές εκτελούνται:

- μία κάθε φορά
- η μία αμέσως μετά την άλλη
- κάθε στιγμή εκτελείται οπωσδήποτε κάποια εντολή

Τα παραπάνω, ορίζουν μία «ροή» που ονομάζουμε ροή εκτέλεσης του προγράμματος.

Προσοχή! Στη C, δεν επιτρέπεται εντολή γραμμένη έξω από κάποια συνάρτηση. Η συνάρτηση αποτελεί βασικό δομικό στοιχείο της γλώσσας.

Ροή εκτέλεσης εντολών

Η εκτέλεση των εντολών μιας συνάρτησης ξεκινά με την εντολή που ακολουθεί το αρχικό άγκιστρό { της, και τελειώνει είτε με την εντολή που υπάρχει ακριβώς πριν το τελικό άγκιστρό της } , είτε με την εντολή **return** που παρουσιάζεται σε επόμενη διαφάνεια.

Η εκτέλεση ενός προγράμματος ξεκινά με την εκτέλεση της συνάρτησης **main** και όπου καλείται μία συνάρτηση, η ροή της εκτέλεσης του προγράμματος προσωρινά μεταφέρεται στις εντολές της, μέχρι και την ολοκλήρωσή της. Κατόπιν η εκτέλεση συνεχίζεται με την εντολή που ακολουθούσε την κλήση της.

Η εκτέλεση ενός προγράμματος ολοκληρώνεται όταν ολοκληρωθεί η εκτέλεσης της **main**.

```
#include <stdio.h>

void printOranges(int oranges) {
    printf("Exw %d portokalia\n",
           oranges);
    return;
}

int main() {
    printOranges(5);
    return 0;
}
```

Κλήση συναρτήσεων

Πως καλείται μια συνάρτηση

Μια ήδη γνωστή συνάρτηση

Έχουμε ήδη συναντήσει τη συνάρτηση **`printf`**. Αυτή, όπως και όλες οι συναρτήσεις **καλούνται** (=χρησιμοποιούνται) με τον ακόλουθο τρόπο.

```
printf("%f", x);
```

Δηλαδή:

- αρχικά γράφεται το όνομα της συνάρτησης
- ακολουθούν οι παρενθέσεις και
- μέσα τους γράφονται τα **ορίσματα** χωρισμένα με κόμμα.

Σημειώστε εδώ τον όρο ορίσματα. Έτσι ονομάζουμε τις τιμές που δίνουμε (ή περνάμε) σε μία συνάρτηση, καθώς θα τον συναντήσουμε λίγο αργότερα.

Μια άλλη συνάρτηση

Μία άλλη συνάρτηση είναι η **fabs** η οποία υπολογίζει την απόλυτη τιμή του ορίσματος που της δίνεται. Αυτή, επειδή έχει αριθμητικό αποτέλεσμα καλείται ως εξής:

```
newValue = fabs(someValue);
```

ή

```
newValue = someOtherValue + fabs(someValue);
```

Δηλαδή **το αποτέλεσμά** της συνάρτησης λειτουργεί όπως μία σταθερή τιμή ή μία μεταβλητή στην θέση της. Ακριβώς όπως και η αντικατάσταση στα μαθηματικά.

Σημειώστε ότι : Η συγκεκριμένη συνάρτηση ανήκει σε μία άλλη βιβλιοθήκη από ότι η **printf**, την math. Για να χρησιμοποιηθεί στον κώδικα θα πρέπει να περιληφθεί η βιβλιοθήκη αυτή στο πρόγραμμα. Αυτό επιτυγχάνεται γράφοντας στο πάνω μέρος του κώδικα:

```
#include <math.h>
```

Μερικές νέες συναρτήσεις

Δείτε μερικές συναρτήσεις από τις βιβλιοθήκες math και stdlib

Μερικές έτοιμες συναρτήσεις - math

Η βιβλιοθήκη math μεταξύ άλλων περιλαμβάνει και τις σταθερές και συναρτήσεις του πίνακα.

Περισσότερες πληροφορίες θα δείτε στη σχετική ενότητα των σημειώσεων αλλά και online εδώ.

Σταθερές	
`M_E`	Η τιμή του e (2.718...)
`M_PI`	Η τιμή του π (3.14159...)
`M_SQRT2`	Η τιμή του $\sqrt{2}$ (1.414...)
`M_LN2`	Η τιμή του $\ln 2$ (0.6931...)
Γενικά Μαθηματικά	
`double fabs(double x)`	Επιστρέφει την απόλυτη τιμή του x
`double sqrt(double x)`	Επιστρέφει την τετραγωνική ρίζα του x
`double pow( double x, double y )`	Επιστρέφει το x υψωμένο στο y
`double log(double x)`	Ο φυσικός λογάριθμος του x
`double log10(double x)`	Ο λογάριθμος του x ως προς το 10.
`double exp(double x)`	Ο αριθμός e υψωμένος στην x
`double fmod( double x, double y )`	Το υπόλοιπο της διαίρεσης x δια y
Στρογγυλοποίηση	
`double ceil(double x)`	Στρογγυλοποιεί την τιμή του x στο μικρότερο ακέραιο που είναι μεγαλύτερος ή ίσος με το x
`double floor(double x)`	Στρογγυλοποιεί την τιμή του x στο μεγαλύτερο ακέραιο που είναι μικρότερος ή ίσος με το x

Τριγωνομετρικές συναρτήσεις	
`double sin(double x)`	Το ημίτονο της x, όταν η χ είναι εκφρασμένη σε ακτίνια (rad)
`double cos(double x)`	Το συνημίτονο της x, όταν η χ είναι εκφρασμένη σε ακτίνια (rad)
`double tan(double x)`	Η εφαπτομένη της x, όταν η χ είναι εκφρασμένη σε ακτίνια (rad)
Αντίστροφες τριγωνομετρικές συναρτήσεις	
`double asin(double x)`	Το τόξο ημιτόνου του x, εκφρασμένο σε ακτίνια (rad) στο διάστημα [-π/2 , π/2]
`double acos(double x)`	Το τόξο συνημιτόνου του x, εκφρασμένο σε ακτίνια (rad) στο διάστημα [0, π]
`double atan(double x)`	Η τόξο εφαπτομένης του x, εκφρασμένη σε ακτίνια (rad) στο διάστημα [-π/2 , π/2]
`double atan2( double y, double x )`	Η τόξο εφαπτομένης που προκύπτει από τη διαίρεση y/x, λαμβάνοντας υπόψη και τα πρόσημα ώστε να υπολογιστεί η γωνία στο σωστό τεταρτημόριο. Επιστρέφεται μία τιμή στο διάστημα [0, 2π)

Μερικές έτοιμες συναρτήσεις - stdlib

Η βιβλιοθήκη stdlib μεταξύ άλλων περιλαμβάνει και τις σταθερές και συναρτήσεις του πίνακα.

Περισσότερες πληροφορίες θα δείτε στη σχετική ενότητα των σημειώσεων αλλά και online εδώ.

Σταθερές	
`NULL`	Η σταθερά αυτή παριστάνει τη μηδενική τιμή αλλά σε τύπο δείκτη (void *)
`RAND_MAX`	Η σταθερά αυτή παριστάνει τη μέγιστη τιμή που μπορεί να επιστρέψει η συνάρτηση rand
Διάφορες συναρτήσεις	
`void exit(int status)`	Τερματίζει την εκτέλεση του προγράμματος με κωδικό εξόδου status.
`int abs(int x)`	Επιστρέφει την απόλυτη τιμή του ακεραίου int x
`long int labs(long int x)`	Επιστρέφει την απόλυτη τιμή του ακεραίου long x
`int rand(void)`	Παράγει τον επόμενο ψευδοτυχαίο ακέραιο μεταξύ 0 και RAND_MAX
`void srand(unsigned int seed)`	Αλλάζει την τιμή του seed για τη γεννήτρια ψευδοτυχαίων αριθμών. Συνήθως χρησιμοποιείται σε συνδυασμό με κάποια συνάρτηση που παράγει μια τιμή που δεν μπορεί εύκολα να προβλεφθεί (π.χ. την τρέχουσα χρονική στιγμή).

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-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

Ορισμός & Δήλωση Συναρτήσεων

Πως δημιουργείται μια συνάρτηση

Ορισμός συνάρτησης

Για να δημιουργηθεί μία νέα συνάρτηση σε ένα πρόγραμμα θα πρέπει να δοθεί σε αυτό ο **ορισμός** (**definition**) της. Για παράδειγμα ο ορισμός μιας μαθηματικής συνάρτησης π.χ.

$$f(x) = x^2$$

η οποία θα διαχειρίζεται πραγματικούς αριθμούς γράφεται:

```
double squareOf(double x)
{
  return x*x;
}
```

Ο **ορισμός** αυτός περιλαμβάνει με τη σειρά:

1. Τον **τύπο δεδομένων** που επιστρέφει η συνάρτηση, ή εφόσον δεν επιστρέφει κάτι, την λέξη κλειδί `void` που σημαίνει κενό
2. Το **όνομα** της συνάρτησης, το οποίο ακολουθεί τους κανόνες των identifiers, άρα πρέπει να είναι μοναδικό σε όλο το πρόγραμμα
3. Οι **παράμετροι** (parameters) της συνάρτησης, δηλαδή τις τιμές που δέχεται ως δεδομένα, χωρισμένα με κόμμα μεταξύ τους και μπροστά από το κάθε όρισμα τον τύπο δεδομένων του.
4. Το **σώμα των εντολών** της συνάρτησης, δηλωμένο ως block (δηλαδή οι εντολές της συνάρτησης μέσα σε άγκιστρα)

Σημεία προσοχής

Παράμετροι

Στον ορισμό μίας συνάρτησης οι παράμετροι λειτουργούν ως τοπικές μεταβλητές, ακριβώς όπως και αν ήταν δηλωμένες στην αρχή του σώματος της συνάρτησης.

Προσέξτε τη διαφορά από την έννοια των ορισμάτων που αναφέραμε νωρίτερα : Κάθε φορά που καλείται η συνάρτηση, οι τιμές που παίρνουν οι παράμετροι είναι αντίγραφα των αντίστοιχων ορισμάτων που δίνονται.

Επιστρεφόμενη τιμή

Κάθε συνάρτηση πρέπει να έχει τουλάχιστον μία εντολή **`return`**. Η εντολή αυτή όταν εκτελεστεί τερματίζει την εκτέλεση της συνάρτησης και αμέσως μετά τη λέξη `return` δίνεται η τιμή που επιστρέφει η συνάρτηση (δηλαδή το αποτέλεσμά της). Εάν η συνάρτηση έχει δηλωθεί ως `void`, τότε η **`return`** δεν θα έχει δίπλα της κάποια τιμή. Σε κάθε περίπτωση, όπως όλες οι εντολές, τελειώνει με τον τελεστή **`;`**

Όταν η τελευταία εντολή μιας συνάρτησης είναι η **`return`** και ο τύπος της συνάρτησης είναι **`void`**, μόνο τότε η **`return`** μπορεί να παραληφθεί. Κατά την περίοδο εκμάθησης όμως καλύτερα να γράφεται πάντα.

Τοποθέτηση του ορισμού

Ο ορισμός της συνάρτησης πρέπει να γίνεται πάντα πριν την χρήση της (όπως και οι δηλώσεις των μεταβλητών).

Το πρόβλημα είναι…

```c
#include <stdio.h>

void printOranges(int oranges) {
    printf("Exw %d portokalia\n",
           oranges);
    return;
}

int main() {
    printOranges(5);
    return 0;
}
```

```c
#include <stdio.h>
#include <math.h>

double ypoteinousa(double a, double b) {
    return sqrt( a*a + b*b );
}

void proveUnitCycle() {
    double f = M_PI / 4;
    printf("%lf\n", ypoteinousa(
                        sin(f),
                        cos(f)
                    )
          );
}

int main() {
    proveUnitCycle();
    return 0;
}
```

Δηλώσεις συναρτήσεων

Όπως φαίνεται και στα προηγούμενα παραδείγματα είναι «παράλογη» η τοποθέτηση των συναρτήσεων αφού πρώτα εμφανίζονται οι πιο «εξειδικευμένες» συναρτήσεις και μετά οι γενικότερες και στο τέλος η… αρχική (η main)!

Επίσης σε περίπτωση όπου δύο συναρτήσεις χρειάζονται η μία να καλέσει την άλλη για να ολοκληρωθούν, τότε δεν γίνεται να γραφτούν και οι δύο… πρώτες!

Γι'αυτό τον λόγο υπάρχει και η δυνατότητα απλής **δήλωσης** (**declaration**) των συναρτήσεων, όπου γίνεται περιγραφή της συνάρτησης χωρίς να δίνεται ο κώδικάς της. Η δήλωση γίνεται όπως ο ορισμός, αλλά χωρίς το σώμα των εντολών ενώ αντί για άγκιστρα απλά μπαίνει ο τερματισμός εντολής `;` π.χ.

```
double inverse(double x);
```

Επίσης το όνομα των παραμέτρων μπορεί να παραληφθεί.

Μετά από τη δήλωση της, μια συνάρτηση μπορεί να κληθεί οπουδήποτε. Θα πρέπει όμως σε οποιοδήποτε σημείο του κώδικα από εκεί και κάτω να δοθεί και ο ορισμός.

Παράδειγμα

```c
#include <stdio.h>
#include <math.h>

double ypoteinousa(
        double a, double b
) {
    return sqrt( a*a + b*b );
}

void proveUnitCycle() {
    double f = M_PI / 4;
    printf("%lf\n",
        ypoteinousa(
            sin(f),
            cos(f)
        )
    );
}

int main() {
    proveUnitCycle();
    return 0;
}
```

```c
#include <stdio.h>
#include <math.h>

void proveUnitCycle();
double ypoteinousa(double a, double b);

int main() {
    proveUnitCycle();
    return 0;
}

void proveUnitCycle() {
    double f = M_PI / 4;
    printf("%lf\n", ypoteinousa( sin(f), cos(f) ));
}

double ypoteinousa(double a, double b) {
    return sqrt( a*a + b*b );
}
```

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-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!

Ο προεπεξεργαστής και οι βιβλιοθήκες

Μια επεξήγηση της βοηθητικής αυτής λειτουργίας

Preprocessor (`#include`)

Στη C κατά τη διαδικασία του build, πριν ξεκινήσει το compilation, προηγείται η εκτέλεση του προεπεξεργαστή ή preprocessor. Για τη θέση του στη διαδικασία του build δείτε ένα απλό άρθρο εδώ. Ο σκοπός του είναι να παράγει από το αρχείο του κώδικα ένα νέο αρχείο κώδικα τροποποιημένο βάσει των ψευδοεντολών του.

Προσοχή! Οι ψευδοεντολές του δεν αποτελούν εντολές της C και δεν παράγουν εκτελέσιμο κώδικα. Όλες ξεκινούν με τον χαρακτήρα **#**

Αν και οι δυνατότητές του preprocessor είναι πολλές θα περιοριστούμε σε δύο βασικές. Την `#define` και την `#include`.

Η **`#include`** φέρνει τα περιεχόμενα ενός αρχείου στο σημείο στο οποίο γράφεται, σαν να είχαν γραφεί αυτά εκεί. Γράφεται ως:

```
#include <some_file>   ή   #include "some_file"
```

Οι δύο παραλλαγές με τα < > ή τα εισαγωγικά " " αφορούν το σημείο στο οποίο θα αναζητηθεί το εν λόγω αρχείο. Εν γένει τα αρχεία των τυπικών βιβλιοθηκών συντάσσονται με τα < >, ενώ τυχόν «δικά μας» ή άλλων βιβλιοθηκών με τα εισαγωγικά.

Preprocessor (`#define`)

Η **`#define`** αντικαθιστά ένα λεκτικό με την παράσταση που το ακολουθεί σε όλη την έκταση του κώδικα από την ψευδοεντολή και κάτω, ακριβώς όπως ένας editor θα έκανε find/replace (εύρεση/αντικατάσταση). Π.χ.

```
#define THIS_YEAR 2021
```

Το όνομα (identifier) συνηθίζεται να γράφεται με κεφαλαίους χαρακτήρες.

Επίσης παρέχεται και μία άλλη παραλλαγή όπου το όνομα ακολουθείται από παρενθέσεις με παραμέτρους όπως οι συναρτήσεις. Αυτά τα «κατασκευάσματα» ονομάζονται macros ή μακροεντολές και δεν είναι συναρτήσεις. Π.χ.

```
#define PRODUCT(x,y) ((x)*(y))
```

Σε οποιοδήποτε σημείο του κώδικα πλέον γραφεί `PRODUCT(...,...)` τότε γίνεται η παραπάνω αντικατάσταση. Όμως με τα macros αυτά **δεν** θα ασχοληθούμε

Βιβλιοθήκες

Οι βιβλιοθήκες (libraries), **πολύ απλουστευτικά**, είναι συλλογές από συναρτήσεις, σταθερές και τύπους δεδομένων που συνεργάζονται μεταξύ τους για να πετύχουν κάποιον σκοπό.

Για παράδειγμα η μαθηματική βιβλιοθήκη έχει μαθηματικές σταθερές και μαθηματικές συναρτήσεις. Οι βιβλιοθήκες είναι κώδικας που έχει ήδη γίνει build.

Όταν όμως ένα πρόγραμμα θέλει να χρησιμοποιήσει μία βιβλιοθήκη πρέπει να έχει δύο πράγματα:

1. Τον κώδικα (=το αρχείο της βιβλιοθήκης), το οποίο πρέπει να γίνει link μαζί με τον κώδικα του προγράμματος (δεν θα χρειαστεί για τις ανάγκες του μαθήματος, αλλά για την πληρότητα των σημειώσεων, η προσθήκη μιας βιβλιοθήκης στο project του CLion περιγράφεται εδώ).
2. Τις δηλώσεις των συναρτήσεων, των σταθερών και των τύπων δεδομένων ώστε να μπορεί να γίνει compile ο κώδικας του προγράμματος που τις χρησιμοποιεί.

Όλα τα απαραίτητα περιέχονται σε αρχεία που ονομάζονται header files και έχουν κατάληξη .h , είναι δε, απλά αρχεία της C που περιέχουν τα #defines όλων των σταθερών (ίσως και κάποιες μακροεντολές), τους τύπους δεδομένων και τις δηλώσεις όλων των συναρτήσεων της βιβλιοθήκης.

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-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!