Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

6

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #6

Παναγιώτης Παύλου

intro-hy-24@allos.gr

Αναπαράσταση χαρακτήρων

Αναπαράσταση μεμονωμένου χαρακτήρα

Κωδικοποίηση χαρακτήρων 1/2

Πέρα από όσα έχουμε δει στη θεωρία, η τυπική C ως γλώσσα χρησιμοποιεί την κωδικοποίηση ASCII, οπότε στα πλαίσια του μαθήματος θα περιοριστούμε σε κείμενα με Λατινικούς χαρακτήρες και στην κωδικοποίηση ASCII με 7bits.

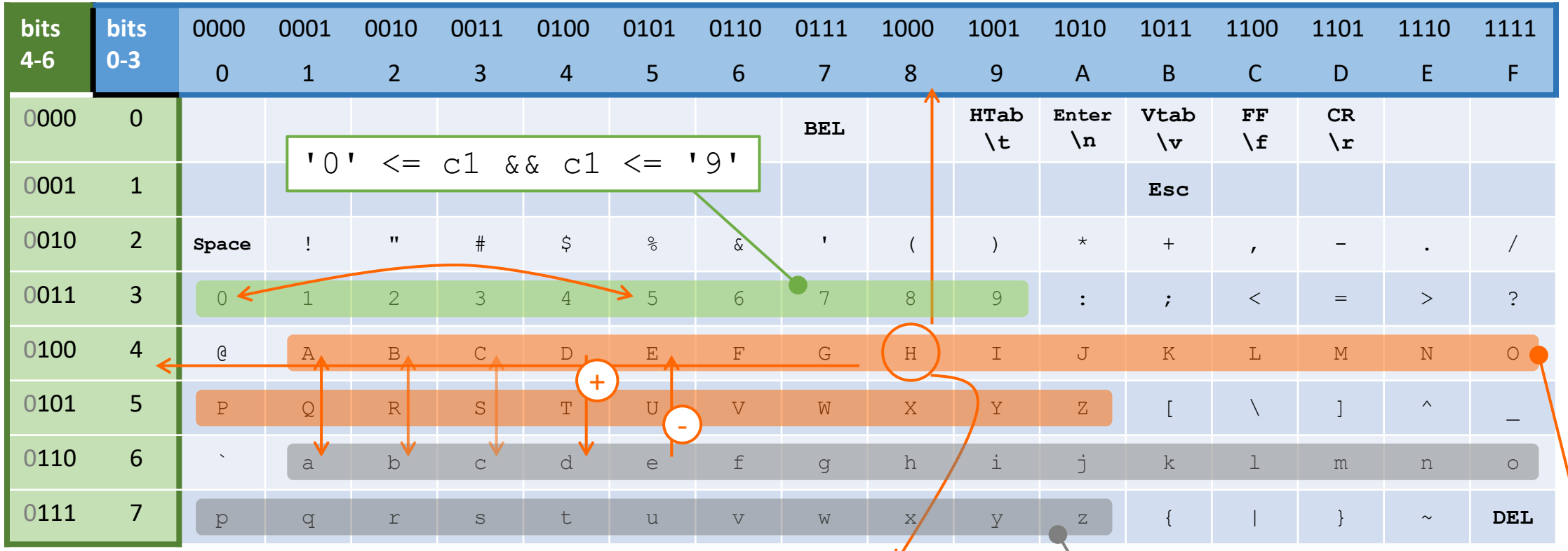
Η εμφάνιση στην οθόνη ενός μεμονωμένου χαρακτήρα γίνεται με τη χρήση της **%c** στην **printf**. Σε αυτή θα πρέπει να αντιστοιχίσουμε έναν 8bit (άρα τύπου δεδομένων char ή unsigned char) αριθμό που θα είναι ο κωδικός του χαρακτήρα. Π.χ.

```
printf("%c\n", 65);
```

Που όπως θα δούμε θα εμφανίσει τον χαρακτήρα **A**. Βέβαια εμείς δεν θα θυμόμαστε τους κωδικούς των χαρακτήρων, ούτε και θα ήταν σωστό να κάνουμε κάτι τέτοιο, καθώς σε κάποιες περιπτώσεις μπορεί να είναι διαφορετικοί από το ένα σύστημα σε άλλο. Οπότε για να παραστήσουμε τον κωδικό του χαρακτήρα τον σημειώνουμε μέσα σε μονά εισαγωγικά:

```
printf("%c\n", 'A');
```

Πίνακας ASC-II




```
› char c1 = someDigitCharacter;
  int digitValue = c1 - '0';
```

```
› const int offsetCaps2Small = 'a' - 'A';
  char c2 = someCapitalLetter;
  char c3 = c2 + offsetCaps2Small;
```

Ομάδες χαρακτήρων – ctype

- **Ψηφία / Digits**
  Τα αριθμητικά ψηφία 0 1 2 3 4 5 6 7 8 9

`int isdigit(int c)`

- **Δεκαεξαδικά ψηφία / Hexadecimal digits**
  Τα αριθμητικά ψηφία και οι χαρακτήρες a-f
  0 1 2 3 4 5 6 7 8 9 A a B b C c D d E e F f

`int isxdigit(int c)`

- **Πεζά γράμματα / Lowercase letters**
  Οι πεζοί χαρακτήρες του Αγγλικού αλφαβήτου
  a b c d e f g h i j k l m n o p q r s t u v w x y z

`int islower(int c)`

- **Κεφαλαία γράμματα / Uppercase letters**
  Οι κεφαλαίοι χαρακτήρες του Αγγλικού αλφαβήτου
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

`int isupper(int c)`

- **Γράμματα ή Αλφαβητικοί χαρακτήρες / Letters ή Alphabetic characters**
  Τα κεφαλαία και τα πεζά γράμματα μαζί

`int isalpha(int c)`

- **Αλφαριθμητικοί χαρακτήρες / Alphanumeric characters**
  Τα γράμματα μαζί με τα (αριθμητικά) ψηφία

`int isalnum(int c)`

- **Σημεία στίξης / Punctuation characters**
  Οι χαρακτήρες
  ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

`int ispunct(int c)`

0 1 2 3 4 5 6 7 8 9

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

ΠΡΟΣΟΧΗ!
Χρησιμοποιήστε τις επιστρεφόμενες τιμές ως λογικές μεταβλητές.

- **Γραφικοί χαρακτήρες / Graphical characters**
  Οι αλφαριθμητικοί χαρακτήρες μαζί με τα σημεία στίξης. Δηλαδή όλοι οι χαρακτήρες οι οποίοι έχουν οπτική αναπαράσταση (εμφανίζουν κάτι στην οθόνη).

`int isgraph(int c)`

- **Κενοί χαρακτήρες / Space ή Whitespace characters**
  Χαρακτήρες που έχουν επίδραση στο αποτέλεσμα αλλά δεν έχουν ορατό περιεχόμενο. Πιο συγκεκριμένα είναι το κενό ( ), το οριζόντιο tab (\t), το κατακόρυφο tab (\v), αλλαγή γραμμής (\n), επιστροφή (\r) και form feed (\f).

`int isspace(int c)`

Αυτοί είναι και οι χαρακτήρες που χρησιμοποιούνται για την ελεύθερη σύνταξη ενός προγράμματος C

- **Εκτυπώσιμοι χαρακτήρες / Printable characters**
  Οι γραφικοί χαρακτήρες και οι κενοί μαζί.

`int isprint(int c)`

- **Χαρακτήρες ελέγχου / Control characters**
  Χαρακτήρες με κωδικό 0 (\000) ως και 31 (\037) και ο 127 (\177)

`int iscntrl(int c)`

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

  intro-hy-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!

Πρακτική 1 – Σύγκριση χαρακτήρων

Γράψτε τη συνάρτηση:

```
bool isSameLetter(char a, char b, bool caseInsensitive)
```

η οποία θα πρέπει να απαντά στο ερώτημα:

«είναι οι χαρακτήρες `a` και `b` , χαρακτήρες κειμένου και ίδιοι»;

Για τη σύγκριση θα πρέπει να λαμβάνει υπόψη του εάν η παράμετρος `caseInsensitive` είναι αληθής, οπότε η σύγκριση θα πρέπει να θεωρεί τα πεζά και τα κεφαλαία ίσα μεταξύ τους.

Αναπαράσταση κειμένων

Αναπαράσταση κειμένου ως ομάδα (πίνακας) χαρακτήρων

Κείμενα

Τα κείμενα στη C είναι απλά πίνακες χαρακτήρων. Η μόνη διαφορά είναι ότι το επόμενο στοιχείο από τον τελευταίο χαρακτήρα θα πρέπει να περιέχει τον κωδικό ASCII **0**, αντί να χρειάζεται να είναι γνωστό το μήκος τους!

Άρα το κείμενο Hello θα μπορούσε να γραφεί όπως ξέρουμε ως πίνακας:

```
char greeting[] = { 'H', 'e', 'l', 'l', 'o', 0 };
```

NULL terminated string

Ένωση σταθερών κειμένων κατά το compilation ως εξής:
`"one,""two""",three"`
που ταυτίζεται με το
`"one,two,three"`

Αλλά αυτό είναι και κάπως "κουραστικό". Γι'αυτό το ισοδύναμα γράφεται:

```
char greeting[] = "Hello";
```

το οποίο τοποθετεί από μόνο του το πρόσθετο στοιχείο με κωδικό 0. Δηλαδή ο πίνακας όπως και πριν είναι με 6 στοιχεία.

Η χρήση του μηδενικού στο τέλος του κειμένου δεν είναι σπατάλη μνήμης ενός χαρακτήρα, αλλά είναι οικονομία αφού για την αποθήκευση του μήκους ενός κειμένου εν γένει θα χρησιμοποιούνταν μία ακέραια μεταβλητή τουλάχιστον 2 ή 4 bytes.

Εάν βέβαια χρειαστεί να γνωρίζουμε το μήκος του κειμένου, τότε θα χρειαστεί να το υπολογίσουμε:

```
int calculateLength(char text[]) {
    int length;
    for (length = 0; text[length] != 0; length++) {
        ;
    }
    return length;
}
```

Πρακτική 2 – Αντιγραφή κειμένου

Γράψτε τη συνάρτηση:

```
bool copyText(char from[], char to[], int lengthOfTo)
```

η οποία θα πρέπει να αντιγράφει το κείμενο που έχει η `from` στον πίνακα/κείμενο `to`, το οποίο έχει πλήθος στοιχείων `lengthOfTo`.

Εάν ο πίνακας `to` έχει αρκετά στοιχεία για να χωρέσει το κείμενο, τότε γίνεται η αντιγραφή και επιστρέφεται αληθές. Εάν δεν έχει αρκετά στοιχεία, τότε αντιγράφονται όσοι χαρακτήρες χωρούν και επιστρέφεται ψευδές.

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

  intro-hy-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!

Διάλειμμα

μικρή ανάσα

Ανέκδοτο της ημέρας

- Καλά, μένεις κοντά σε θάλασσα και δεν ξέρεις να κολυμπάς;

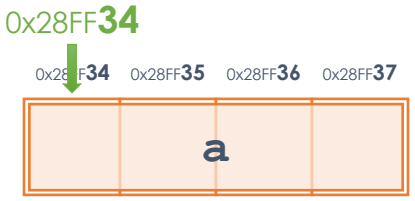
- Γιατί;! Εσύ που μένεις κοντά σε αεροδρόμιο ξέρεις να πετάς;

Δείκτες/Pointers μεταβλητών

Πως αποθηκεύονται οι μεταβλητές στη μνήμη
και βάσει αυτού ποιες δυνατότητες ανοίγονται

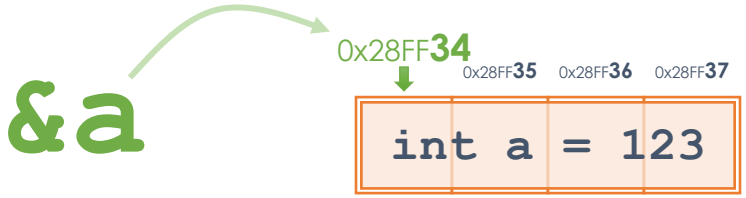
Pointer σε μεταβλητή 1/4

Ας θυμηθούμε από τη θεωρία ότι οι μεταβλητές που απαιτούν περισσότερες από μία θέσεις μνήμης αποθηκεύονται σε διαδοχικά bytes, λέμε όμως ότι αποθηκεύονται στην πρώτη από αυτές.



Δηλαδή η μεταβλητή `a` δίπλα, **λέμε** ότι είναι αποθηκευμένη στη θέση μνήμης `0x28FF34`.

Πως μπορούμε να προσδιορίσουμε αυτή την πρώτη θέση μνήμης που είναι αποθηκευμένη μια μεταβλητή;



Γίνεται με τη χρήση του τελεστή **&** ο οποίος διαβάζεται και "address of".

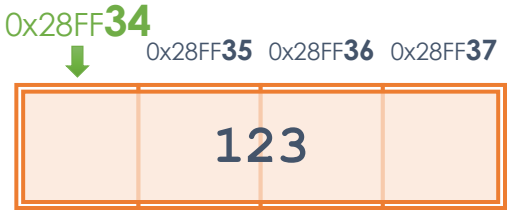
Η τιμή αυτή ονομάζεται **δείκτης** ή **pointer** στην μεταβλητή **a**.

Η πρώτη θέση μνήμης (από τον pointer) μαζί με το πλήθος των bytes (από τον τύπο δεδομένων) αρκούν για τον επακριβή εντοπισμό της περιοχής της μνήμης στην οποία είναι αποθηκευμένη η τιμή της μεταβλητής.

Pointer σε μεταβλητή 2/4

Την τιμή του pointer σε τι τύπο μεταβλητής την αποθηκεύουμε;

```
int a = 123;
int *Ap = &a;
printf("%p\n", Ap);
```

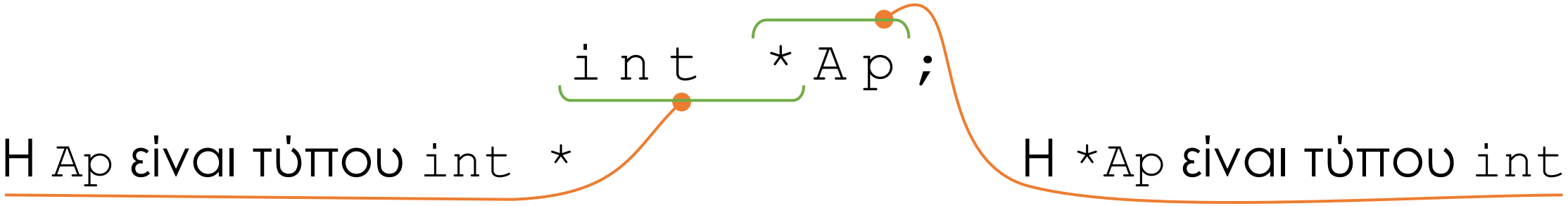


Η μεταβλητή `Ap` είναι τύπου pointer και για την ακρίβεια **δείκτη/pointer σε int** και συντάσσεται γράφοντας πρώτα τον τύπο δεδομένων της μεταβλητής στην οποία δείχνει ακολουθούμενο από τον μοναδιαίο τελεστή *****

Και εκτύπωσή της γίνεται με τη χρήση του format specifier **%p** για την εντολή printf, η οποία εκτυπώνει τη δεκαεξαδική τιμή της διεύθυνσης χωρίς το πρόθεμα **0x** και έχει νόημα μόνο στα πλαίσια του debugging.

Pointer σε μεταβλητή 3/4

Η δήλωση μιας τέτοιας μεταβλητής τύπου pointer (σε int) διαβάζεται με δύο τρόπους:



Και όντως η παράσταση `*Ap` αντιστοιχεί στην αντίστροφη λειτουργία από το `&a`, δηλαδή επιστρέφει την (ακέραια) τιμή εκεί που δείχνει ο pointer (άρα στο παράδειγμα αυτό `int`).

Άρα όταν **`Ap = &a`** τότε τα **`*Ap`** και **`a`** ταυτίζονται.

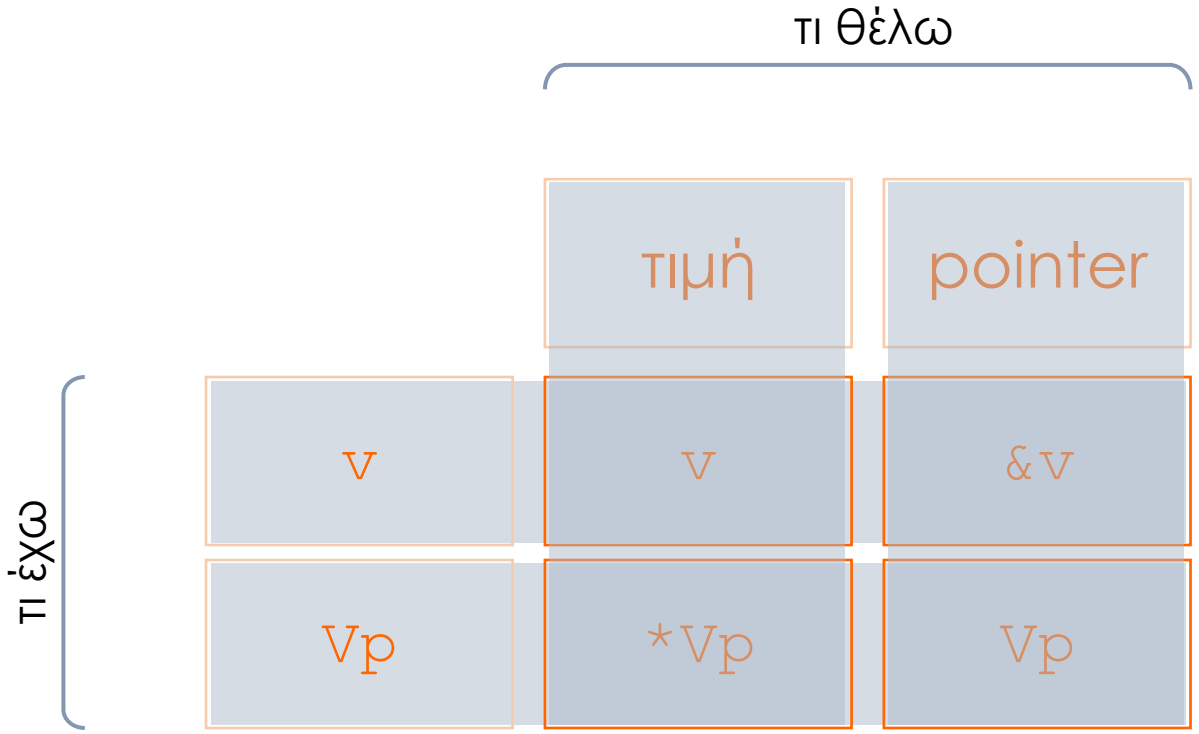
Παράδειγμα

```
int a = 123;
int *Ap = &a;
printf("a = %d , *Ap = %d\n", a, *Ap);
*Ap = 234;
printf("a = %d , *Ap = %d\n", a, *Ap);
```


```
a = 123 , *Ap = 123
a = 234 , *Ap = 234
```

Pointer σε μεταβλητή 4/4

Άρα για μια μεταβλητή v οποιουδήποτε τύπου δεδομένων και για έναν κατάλληλο pointer σε αυτή `vp = &v` ισχύει:



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

  intro-hy-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!

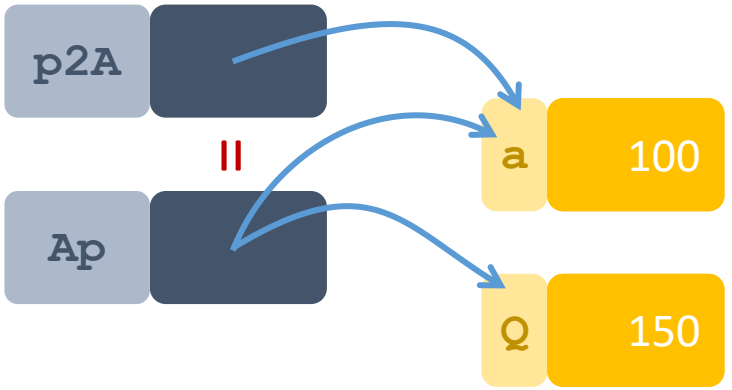
Call by reference

Η χρήση των pointers είναι πολύ εκτεταμένη και αποτελεί το στοιχείο κλειδί για την ουσιαστική χρήση της γλώσσας C. Η απλούστερη εφαρμογή της είναι η παράκαμψη της προστασίας των ορισμάτων κατά την κλήση μιας συνάρτησης. Η πρακτική αυτή ονομάζεται **call by reference** σε αντίθεση με την τυπική κλήση της C που ονομάζεται call by value.

Ουσιαστικά αυτό που συμβαίνει είναι ότι αντί να δοθεί ως όρισμα η ίδια η μεταβλητή και να αντιγραφεί η τιμή της στη συνάρτηση, δίνεται ως όρισμα ο pointer στη μεταβλητή που μας ενδιαφέρει, οπότε και το αντίγραφο του pointer δείχνει στην ίδια μεταβλητή.

ΠΡΟΣΟΧΗ! Είναι πολύ εύκολο εκ παραδρομής ο δείκτης σε μια τοπική μεταβλητή να διατηρηθεί (π.χ. επιστρεφόμενος από μία συνάρτηση) έξω από την εμβέλεια της μεταβλητής, πράγμα το οποίο είναι προφανώς λανθασμένο.

```
void changeTo100(int *Ap) {
    int Q = 123;
    *Ap = 100;
    Ap = &Q;
    *Ap = 150;
}
```

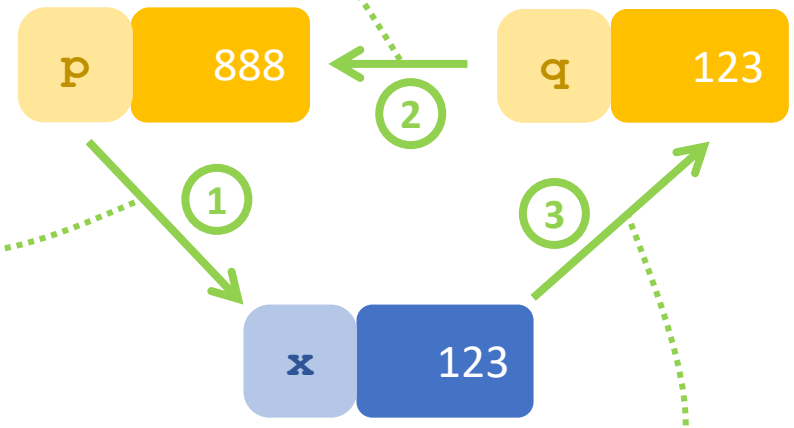



```
int main() {
    int a = 200;
    int *p2A = &a;
    changeTo100( p2A );
    return 0;
}
```

Εφαρμογή - Swap variables

```
void swap(double *a, double *b) {
     double x = *a;
     *a = *b;
     *b = x;
}

...
swap( &p, &q );
...
```



scanf - εισαγωγή από το πληκτρολόγιο

Όπως η εντολή `printf` εκτυπώνει στην οθόνη, έτσι και η εντολή `scanf` διαβάζει από το πληκτρολόγιο. Και για να μπορεί να αποδίδει σε μεταβλητές τις τιμές που διαβάζει, δέχεται τα ορίσματά της ως δείκτες στις μεταβλητές αυτές.

Η σύνταξή της είναι ανάλογη της printf:

```
scanf("%lf %d", &d, &c);
```

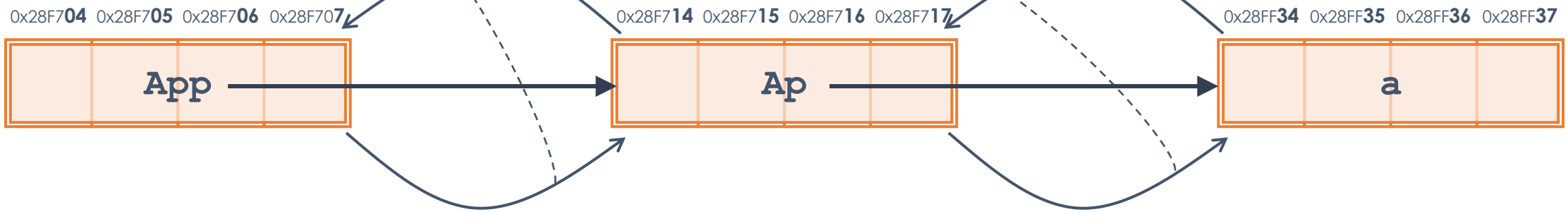
Δηλαδή υπάρχει το format string που δηλώνει τη μορφή των δεδομένων που αναμένεται να διαβαστούν και ακολουθούν ένα όρισμα για κάθε "όρο", του format string σε αντιστοιχία. Τα ορίσματα αυτά είναι (αναγκαστικά) δείκτες στις μεταβλητές οι οποίες θα υποδεχθούν τις τιμές.

Προσέξτε τις διαθέσιμες επιλογές για τους όρους του format string, δεν είναι το σύνολο των υποστηριζόμενων για την printf. Συμβουλευτείτε τα σχετικά παραρτήματα των σημειώσεων. Δείτε ιδιαιτέρως το `%i`

Pointer σε Pointer

Εφόσον μια μεταβλητή pointer είναι ... μια μεταβλητή, αποθηκεύεται και αυτή κάπου στη μνήμη. Άρα μπορώ να γράψω:

```
int a = 123;
int *Ap = &a;
int **App = &Ap;
**App = 888;          // αλλάζει η τιμή του a
// K.O.K.
```



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

  intro-hy-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!

ΠΡΟΣΟΧΗ!

Όταν έχουμε μία μεταβλητή τύπου pointer, την οποία δεν θέλουμε ακόμα να την βάλουμε να δείχνει σε κάποια μεταβλητή, τότε για να μην έχει κάποια τυχαία τιμή και δείχνει σε τυχαίο κομμάτι της μνήμης, της δίνουμε την ειδική τιμή NULL. Πχ

```
int *A = NULL;
```

Το NULL είναι ένα #define στο stdio.h το οποίο είναι ουσιαστικά η τιμή 0 σε τύπο δεδομένων pointer. Δηλαδή δείχνει στην θέση μνήμης 0, πράγμα το οποίο πάντα οδηγεί το πρόγραμμά μας σε ανώμαλο τερματισμό, αλλά αυτό σημαίνει πάντα σταθερή συμπεριφορά.

Σημαντικά σημεία

Μετά από τη σημερινή διάλεξη θα πρέπει να γνωρίζετε:

- Την αναπαράσταση χαρακτήρων (ASCII) στη C
- Την αναπαράσταση κειμένων (ως πίνακες χαρακτήρων) στη C
- Δείκτες σε μεταβλητές
- Την κλήση με αναφορά / call-by-reference