



Προγραμματισμός Ι (ECE115)

#2

δομή προγράμματος C
τύποι δεδομένων
λειτουργίες εισόδου / εξόδου

Η γλώσσα προγραμματισμού C

- Από τις παλιές γλώσσες προγραμματισμού
- Γλώσσα «**προστακτικού**» (imperative) προγραμματισμού με σχετικά λίγες εντολές
- Ιδιαίτερα διαδεδομένη, σε επίπεδο υλοποίησης λειτουργικών συστημάτων αλλά και εφαρμογών
- Υπάρχει **συντακτικός έλεγχος** αλλά και **ευελιξία**
- Η ελάχιστη μονάδα υποπρογράμματος (δόμησης κώδικα) είναι η **συνάρτηση** (function)
- Ένα πρόγραμμα C **μεταφράζεται** και μετά εκτελείται
- Υποστηρίζεται ξεχωριστή μετάφραση και διασύνδεση διαφορετικών υποπρογραμμάτων
- Οι γλώσσες C++ και Java είναι «απόγονοι» της C

Μορφοποίηση

- Σχεδόν κάθε εντολή τερματίζεται με ';'
- Μπορούμε να γράφουμε πολλές διαδοχικές εντολές στην ίδια γραμμή του κειμένου
- «**σώμα**» ή «**μπλοκ**»: μια ή περισσότερες εντολές ανάμεσα σε '{' '}' και συμπεριφέρονται ως «ομάδα»
- Σχόλια (κείμενο που δεν λαμβάνει υπ' όψη του ο μεταφραστής) δίνονται ανάμεσα σε '/' '*' και '*' '/'
- Κενοί χαρακτήρες / γραμμές ανάμεσα σε εντολές δεν λαμβάνονται υπ' όψη από τον μεταφραστή
 - όμως χρησιμεύουν, όπως και τα σχόλια, για την αναγνωσιμότητα του κώδικα

Πως να **ΜΗΝ** γράφετε κώδικα

```
/*
#include
#include
#define c(C)/* - . */return ( C); /* 2004*/
#include <stdio.h> /* Moekan "' \b-' */
typedef/* */char p;p* u ,w [9
][128] ,*v;typedef int _;_ R,i,N,I,A ,m,o,e
[9], a[256],k [9], n[ 256];FILE*f ;_ x (_ K,_ r
,_ q){; for(; r< q ; K _=((
0xfffff) &(K>>8))^ n[255] & ( K
^u[0 + (p*r, p*q ){ c( f =
)} _ E (r ,q))} _ B(_ q){c( fseek (f, 0
,q))} _ D(){c( fclose(f ))} _ C( p *q){c( 0- puts(q ) )}_/* /
*/main(_ t,p**z){if(t<4)c( C("<in" "file" "\40<1" "a" "yout> "
/*b9213272*/"<outfile>" ) )u=0;i=I=(E(z[1],"rb")) ?B(2):0 : ((o =ftell
(f))>=8)?(u =(p*)malloc(o)) ?B(0):0:!fread(u,o,1,f):0:0):0 : D():0 ;if(
!u)c(C(" bad\40input ") );if(E(z[2],"rb" )){for(N=-1;256> i;n[i++] =-1 )a[
i]=0; for(i=I=0; i<o&&(R =fgetc( f ))>-1;i++)++a[R] ?(R==N)?( ++I>7)?(n[
N]+1 )?0:(n [N ]=i-7):0: (N=R) |(I=1):0;A =-1;N=o+1;for(i=33;i<127;i++
)( n[i ]+ 1&&N>a[i])? N= a [A=i] :0;B(i=I=0);if(A+1)for(N=n[A];
I< 8&&( R =fgetc(f ))> -1&& i <o ;i++) (i<N||i>N+7)?(R==A)?( (*w[I
] =u [i])?1: (*w[I]= 46))?(a [I+]=i):0:0:0:D() ;if(I<1)c(C(
" bad\40la" "yout ") )for(i =0;256>(R= i);n[i++]=R)for(A=8;
A >0;A --) R = ( (R&1)==0) ?(unsigned int)R>>(01):((unsigned
/*kero Q' ,KSS */)R>> 1)^ 0xedb88320;m=a[I-1];a[I
]=m <N)?(m= N+8): ++ m;for(i=00;i<I;e[i++]=0){
v=w [i]+1;for(R =33;127 >R;R++)if(R-47&&R-92
&& R-(_) * w[i])*( v++)= (p)R;*v=0;}for(sprintf
/*' _ G*/ (*w+1, "%0" "8x", x(R=time(i=0),m,o) ^~
0) ;i< 8; ++ i)u [N+ i]=* (*w+i+1);for(*k=x(~
0,i=0 ,*a);i>- 1; )for (A=i;A<I;A++){u[+a [ A]
]=w[A ] [e[A]] ; k [A+1]=x (k[A],a[A],a[A+1]
);}if (R==k[I]) c( (E(z[3 ],"wb"))?fwrite(
/* */ u,o,1,f)?D () |C(" \n OK.") :0 :C(
" \n WriteError" )) for (i +=I-
1 ;i >-1?!w[i][++ e[+ i]]:0;
) for( A=i--; A<I;e[A++
=0); (i <I-4 )?putchar
(( _ ) 46) | fflush
/*' ,*/ ( stdout
): 0& 0;}c(C
(" \n fail")
) /* dP' /
dP '
' zc
*/
}
```

Από το Obfuscated C Contest
<http://www.ioccc.org/>

Η συνάρτηση `main`

- Η εκτέλεση του προγράμματος αρχίζει με την κλήση της **κυρίως συνάρτησης** `main`
 - η κλήση πραγματοποιείται από το περιβάλλον εκτέλεσης (λειτουργικό), χωρίς να υπάρχει εντολή μέσα στο πρόγραμμα
- Η εκτέλεση ακολουθεί την «συνηθισμένη» ροή
 - εκτελούνται οι εντολές της `main`
- Η `main` μπορεί να καλέσει άλλες συναρτήσεις
 - αρχικά (για απλότητα) θα εστιάσουμε σε προγράμματα που αποτελούνται μόνο από την συνάρτηση `main`
- Η εκτέλεση του προγράμματος ολοκληρώνεται (το πρόγραμμα τερματίζει) όταν επιστρέψει η `main`
- Η `main` επιστρέφει μια ακέραια τιμή
 - αν όλα πάνε καλά, **κατά σύμβαση** επιστρέφεται 0

Ένα απλό παράδειγμα

```
/* Αρχείο hello.c: εμφανίζει στην  
οθόνη το μήνυμα hello world  
*/
```

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

```
int main(int argc, char* argv[]) {
```

```
    printf("hello world\n");
```

```
    return(0);
```

```
}
```

σχόλια

συμπερίληψη δηλώσεων
εξωτερικών συναρτήσεων

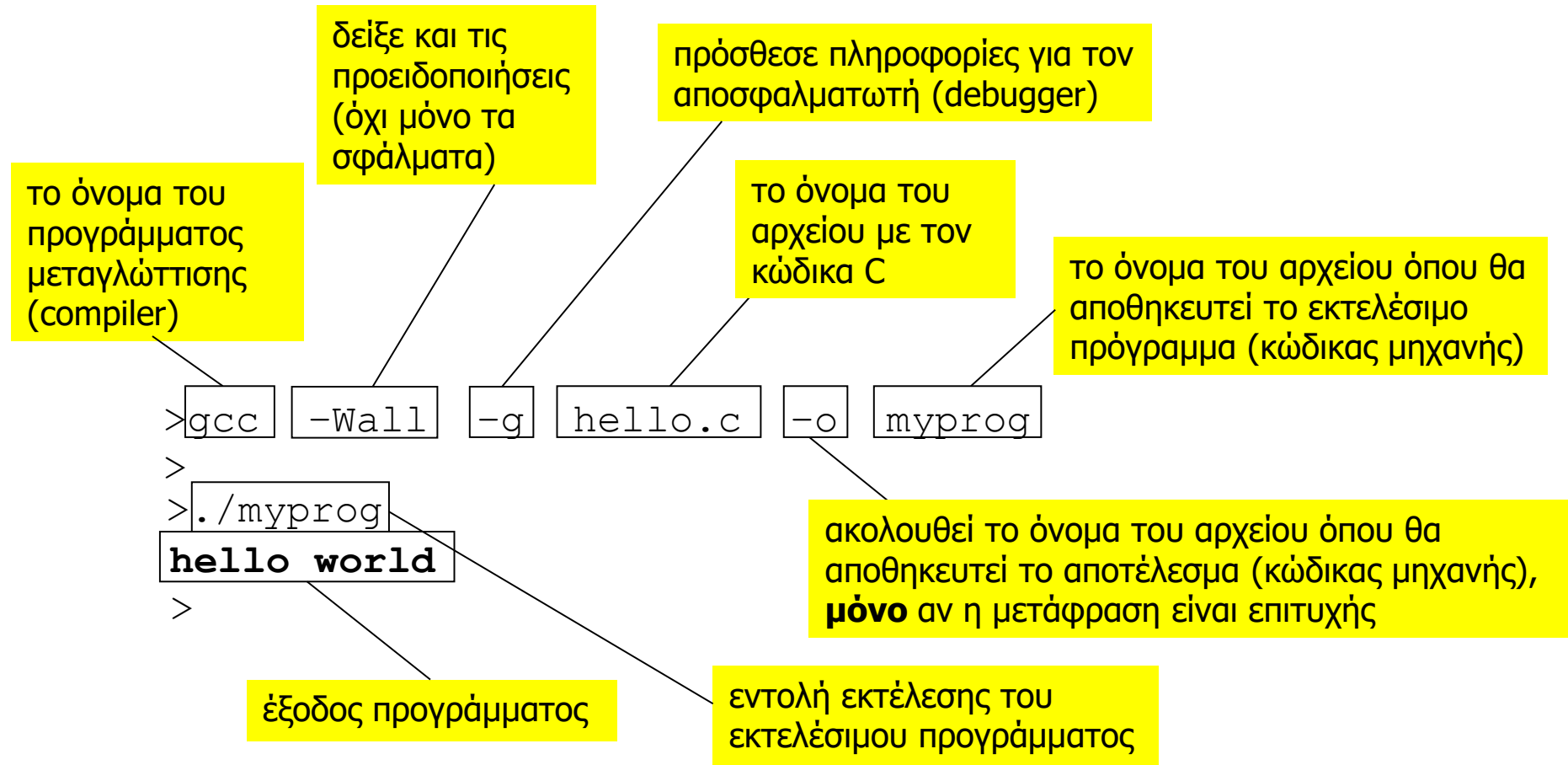
παράμετροι συνάρτησης
(ορίσματα προγράμματος)

επικεφαλίδα συνάρτησης

Τύπος τιμής επιστροφής
συνάρτησης

σώμα (εντολές)

Μεταγλώττιση & εκτέλεση (στο Linux)



Στοιχεία ενός προγράμματος

- Literal / κυριολεκτικό
 - μια συγκεκριμένη («καρφωτή») τιμή στον κώδικα
 - π.χ. 3, "Hello", 5.19
- Variable / μεταβλητή
 - μια ονομασμένη θέση μνήμης όπου αποθηκεύεται μια τιμή
 - που μπορεί να διαβαστεί/αλλαχθεί κατά την εκτέλεση του
- Constant / σταθερά
 - μια ονομασμένη θέση μνήμης όπου αποθηκεύεται μια τιμή
 - η τιμή που αποθηκεύεται σε μια σταθερά **δεν** μπορεί να αλλαχθεί κατά την εκτέλεση του προγράμματος
- Τελεστές
 - συνδυάζουν τα παραπάνω σε πράξεις / εκφράσεις
 - αριθμητικές, λογικές, κλπ.

Στοιχεία ενός προγράμματος (2)

- Expression / έκφραση
 - οτιδήποτε μπορεί να αποτιμηθεί
 - συνδυασμός μεταβλητών, κυριολεκτικών, τελεστών, κ.α.
- Statement / εντολή
 - μια ολοκληρωμένη οδηγία προς εκτέλεση
 - κάθε έκφραση γίνεται εντολή αν βάλουμε στο τέλος ';'
- Function / συνάρτηση
 - ονομασμένη ομάδα εντολών, που κάνουν μια λειτουργία
 - μπορεί να δέχεται παραμέτρους και να επιστρέφει μια τιμή
- Library / βιβλιοθήκη
 - συλλογή συναρτήσεων που υλοποιούν παρεμφερείς λειτουργίες, που μπορεί να έχουν **γενικότερη χρησιμότητα** σε πολλά διαφορετικά προγράμματα



Τύποι δεδομένων

Βασικοί τύποι: ακέραιοι

Όνομα	Τύπος / Κωδικοποίηση	Μέγεθος (bytes)
char	χαρακτήρας	1
int	ακέραιος	2 ή 4 (*)
float	πραγματικός απλής ακρίβειας	4
double	πραγματικός διπλής ακρίβειας	8
void	‘τίποτα’	

(*) εξαρτάται από την αρχιτεκτονική του επεξεργαστή

Προσδιοριστές μεγέθους / πεδίου τιμών

Όνομα	Τύποι / Ερμηνεία	Μέγεθος (bytes)
short	«μικρός» <code>int</code>	1, 2 ή 4 (*)
long	«μεγάλος» <code>int</code>	2, 4 ή 8 (*) $\geq$ <code>int</code>
long	«μεγάλος» <code>double</code>	12 ή 16 (*)

(*) εξαρτάται από την αρχιτεκτονική του επεξεργαστή / πλατφόρμα

- `short` / `long` χωρίς τύπο: υπονοείται `int`

Όνομα	Τύποι	Πεδίο Τιμών
signed	<code>int</code> / <code>char</code> με πρόσημο	$[-2^{N-1} \dots 2^{N-1}-1]$
unsigned	<code>int</code> / <code>char</code> χωρίς πρόσημο	$[0 \dots 2^N-1]$

N: αριθμός bits

- τύπος χωρίς `signed` / `unsigned`: υπονοείται `signed`

Τύποι / μεγέθη / πεδία τιμών (μηχανήματα x86 64-bit του εργαστηρίου)

Τύπος	bytes	Πεδίο Τιμών
unsigned char	1	0 ... 255
char	1	-128 ... 127
unsigned int	4	0 ... 4,294,967,295
short int	2	-32,768 ... 32,767
int	4	-2,147,483,648 ... 2,147,483,647
unsigned long	8	0 ... 18,446,744,073,709,551,615
long	8	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807
float	4	$1.17549435 \cdot (10^{-38}) \dots$ $3.40282347 \cdot (10^{+38})$
double	8	$2.2250738585072014 \cdot (10^{-308}) \dots$ $1.7976931348623157 \cdot (10^{+308})$
long double	16	$3.4 \cdot (10^{-4932}) \dots 1.1 \cdot (10^{4932})$

Πως τα βρίσκω;

```
/* εκτύπωση μεγέθους βασικών τύπων */

#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("sizeof(char) = %lu\n", sizeof(char));
    printf("sizeof(short) = %lu\n", sizeof(short));
    printf("sizeof(int) = %lu\n", sizeof(int));
    printf("sizeof(long) = %lu\n", sizeof(long));
    printf("sizeof(float) = %lu\n", sizeof(float));
    printf("sizeof(double) = %lu\n", sizeof(double));
    printf("sizeof(long double) = %lu\n", sizeof(long double));

    return(0);
}
```

- `sizeof()` επιστρέφει τον αριθμό των bytes ενός τύπου ή αντικειμένου δεδομένων / μεταβλητής

Δηλώσεις μεταβλητών

- Οι δηλώσεις μεταβλητών δίνονται πριν από (σχεδόν) όλες τις υπόλοιπες εντολές ενός προγράμματος
- Μορφή των εκφράσεων δήλωσης είναι
 - `<τύπος> <όνομα> ;`
 - `<τύπος> <όνομα>, ..., <όνομα> ;`
 - `<τύπος> <όνομα> = <τιμή> ;`
- Το όνομα μιας μεταβλητής επιτρέπεται να περιέχει μόνο γράμματα, ψηφία ή τον χαρακτήρα `_` (κάτω παύλα) και δεν επιτρέπεται να ξεκινά από ψηφίο.

Δηλώσεις μεταβλητών

- Κάθε μεταβλητή (και συνάρτηση) πρέπει να έχει διαφορετικό όνομα (για να μην υπάρχει αμφισημία)
- Κατά την δήλωση της, μια μεταβλητή μπορεί (προαιρετικά) να λάβει και μια αρχική τιμή
 - το πρόθεμα `const` σε συνδυασμό με την ανάθεση αρχικής τιμής υποδηλώνει ότι η τιμή της μεταβλητής δε μπορεί να αλλάξει κατά την διάρκεια της εκτέλεσης

Υπολογισμός $1+2+3+4+5$ με ενδιάμεση αποθήκευση σε μεταβλητή

```
/* άθροισμα 1 έως 5 */
```

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

```
int main(int argc, char *argv[]) {
```

```
    int sum = 0;
```

```
    sum = sum + 1;
```

```
    sum = sum + 2;
```

```
    sum = sum + 3;
```

```
    sum = sum + 4;
```

```
    sum = sum + 5;
```

```
    printf("%d\n", sum);
```

```
    return(0);
```

```
}
```

μεταβλητή

έκφραση

κυριολεκτικό

τελεστής
(ανάθεσης)

Υπολογισμός $1+2+3+4+5$ με αποθήκευση σε μεταβλητή

```
/* άθροισμα 1 έως 5 */
```

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

```
int main(int argc, char *argv[]) {
```

```
    int sum;
```

```
    sum = 1 + 2 + 3 + 4 + 5;
```

εντολή

```
    printf("%d\n", sum);
```

```
    return(0);
```

```
}
```

σταθερή έκφραση που μπορεί να αποτιμηθεί κατά την μετάφραση

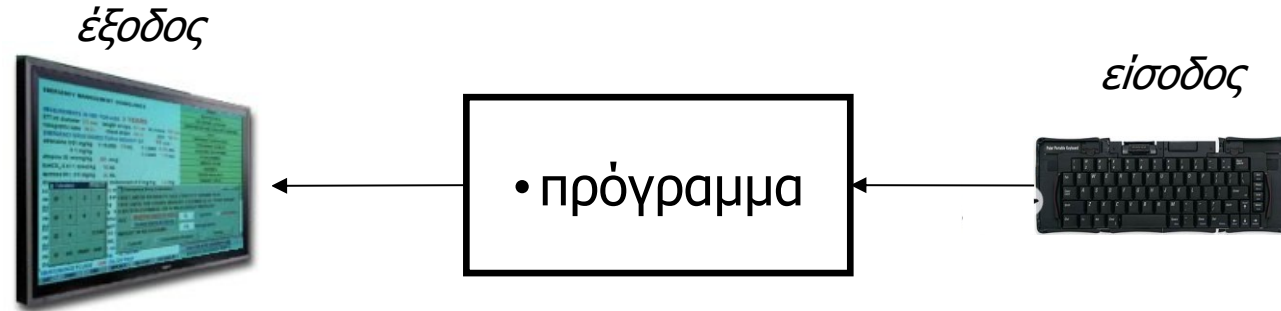
Υπολογισμός $1+2+3+4+5$ χωρίς αποθήκευση σε μεταβλητή

```
/* άθροισμα 1 έως 5 */  
  
#include <stdio.h>  
  
int main(int argc, char *argv[]) {  
  
    printf("%d\n", 1 + 2 + 3 + 4 + 5);  
  
    return(0);  
}
```

σταθερή έκφραση που μπορεί να
αποτιμηθεί κατά την μετάφραση

Είσοδος / Έξοδος προγράμματος

Είσοδος και έξοδος



- Το πρόγραμμα πρέπει να μπορεί να εισάγει/εξάγει δεδομένα από/προς το «περιβάλλον» του
- Συνήθως από το πληκτρολόγιο και προς την οθόνη
 - ή από και προς το δίκτυο, το δίσκο κλπ.
- Αυτό γίνεται χρησιμοποιώντας ειδικές **συναρτήσεις εισόδου/εξόδου** (input/output functions)

Η βιβλιοθήκη stdio

- Οι συναρτήσεις εισόδου/εξόδου υλοποιούνται στην **βιβλιοθήκη stdio**
- Τα «πρωτότυπα» των συναρτήσεων της stdio υπάρχουν στο **αρχείο κεφαλίδων** `stdio.h`
- Πρέπει να **συμπεριληφθεί** στο πρόγραμμα
 - για να γνωρίζει ο μεταφραστής την ύπαρξη των συναρτήσεων με τις παραμέτρους που δέχονται και τα αποτελέσματα που επιστρέφουν
- Μέσω της εντολής `#include <stdio.h>`
- Περισσότερα για το τις βιβλιοθήκες και τον ρόλο των αρχείων κεφαλίδων, πολύ-πολύ αργότερα ...
 - Για την ώρα αρκεί να ξέρετε: Άλλο η βιβλιοθήκη και άλλο το αρχείο κεφαλίδων για αυτή την βιβλιοθήκη, παρόλο που συχνά πάνε μαζί.

Βασικές συναρτήσεις εισόδου/εξόδου

- `getchar()` : διαβάζει ένα (τον επόμενο) χαρακτήρα από την είσοδο του προγράμματος (πληκτρολόγιο)
- `putchar()` : γράφει ένα χαρακτήρα στην έξοδο του προγράμματος (οθόνη)
- `scanf()` : διαβάζει από την είσοδο χαρακτήρες και αναθέτει τιμές σε συγκεκριμένες μεταβλητές
 - διαβάζονται όσοι χαρακτήρες είναι απαραίτητοι για να ανατεθεί τιμή στις «συγκεκριμένες» μεταβλητές
- `printf()` : εκτυπώνει στην έξοδο κείμενο καθώς και τιμές από συγκεκριμένες μεταβλητές


```

#include <stdio.h>

int main(int argc, char* argv[]) {
    char c1, c2, c3;

    putchar('3');
    putchar(' ');
    putchar('c');
    putchar('h');
    putchar('a');
    putchar('r');
    putchar('s');
    putchar(':');
    putchar('\n');

    c1=getchar();
    c2=getchar();
    c3=getchar();

    putchar(c1);
    putchar(c2);
    putchar(c3);
    putchar('\n');

    return(0);
}

```

> ./myprog<enter>

3 chars:

a2\$<enter>

a2\$

>

> ./myprog<enter>

3 chars:

a b c d e f g<enter>

a b

>

το πρόγραμμα **δεν** διαβάζει
τα «επιπλέον» δεδομένα ...

... απλά γιατί δεν υπάρχουν
αντίστοιχες εντολές στον
κώδικα του προγράμματος

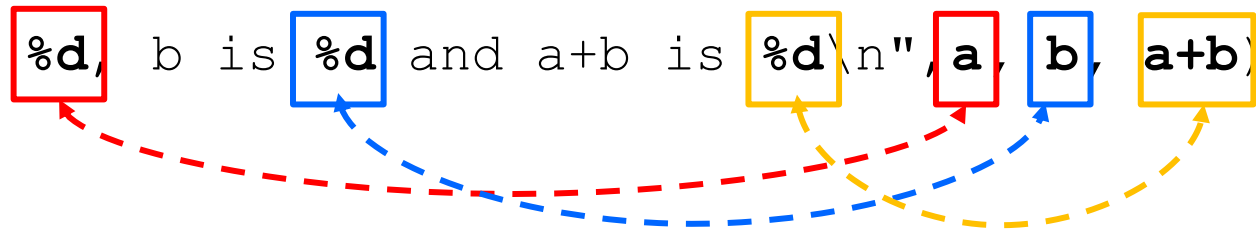
printf()

- Δέχεται ως παράμετρο μια συμβολοσειρά και προαιρετικά έναν (ανοιχτό) αριθμό **εκφράσεων αποτίμησης**
- Η πρώτη παράμετρος περιέχει το κυριολεκτικό κείμενο προς εκτύπωση και τους **προσδιορισμούς εκτύπωσης** για τις τιμές κάθε μιας έκφρασης αποτίμησης
- Ο ειδικός χαρακτήρας ' \n ' οδηγεί σε άμεση εκτύπωση
- Προσοχή:
 - δείτε **οπωσδήποτε** το [εγχειρίδιο](#) της γλώσσας για τις συμβάσεις των προσδιορισμών εκτύπωσης
 - ο μεταγλωττιστής **δεν** εμφανίζει λάθος αν οι εκφράσεις αποτίμησης είναι λιγότερες ή περισσότερες από τους προσδιορισμούς εκτύπωσης
 - ο μεταγλωττιστής δίνει **μόνο** μια προειδοποίηση αν οι προσδιορισμοί εκτύπωσης δεν είναι συμβατοί με τους αντίστοιχους τύπους των εκφράσεων αποτίμησης

```
#include <stdio.h>

int main(int argc, char* argv[]) {
    int a=1,b=2;

    printf("a is %d, b is %d and a+b is %d\n", a, b, a+b);
    return(0);
}
```



```
>./myprog<enter>
a is 1, b is 2 and a+b is 3
>
```

Προσδιοριστές της `printf()`

<http://en.cppreference.com/w/cpp/io/c/printf>

- `%c` : `char`
- `%d` : `int`
- `%u` : `unsigned int`
- `%x` : δεκαεξαδική αναπαράσταση
- `%f` : `float`
- `%ld` : `long int`
- `%lu` : `long unsigned int`
- `%lf` : `double`
- `%s` : συμβολοσειρά (`string` – για αργότερα)
- `%p` : δείκτης (`pointer` – για αργότερα)

Επιπλέον προσδιορισμοί εκτύπωσης

- Εκτύπωση ακέραιου αριθμού με συνολικό πλάτος 4 θέσεις:

```
int x = 12;  
printf("%4d", x);
```



Εκτυπώνει:

 12

2 κενά + 2 ακέραια ψηφία → 4 θέσεις

- Εκτύπωση ακέραιου αριθμού με συνολικό πλάτος 4 θέσεις και "γέμισμα" με μηδενικά:

```
int x = 12;  
printf("%04d", x);
```



Εκτυπώνει:

0012

Επιπλέον προσδιορισμοί εκτύπωσης

- Εκτύπωση αριθμού κινητής υποδιαστολής με 2 δεκαδικά ψηφία:

```
double x = 1.2345;  
printf("%.2lf", x);
```



Εκτυπώνει:
1.23

- Εκτύπωση αριθμού κινητής υποδιαστολής με 2 δεκαδικά ψηφία και συνολικό πλάτος 6 θέσεις:

```
double x = 12.345;  
printf("%6.2lf", x);
```



Εκτυπώνει:
 12.34

1 κενό + 2 ακέραια ψηφία + τελεία + 2 δεκαδικά → 6 θέσεις

scanf ()

- Δέχεται ως παράμετρο μία συμβολοσειρά και έναν (ανοιχτό) αριθμό **διευθύνσεων μεταβλητών**
- Η πρώτη παράμετρος περιέχει το επιθυμητό **μοτίβο εισόδου** με **προσδιορισμούς ανάγνωσης** για τις τιμές κάθε μιας μεταβλητής η διεύθυνση της οποίας δίνεται ως παράμετρος
- Πάντα να **ελέγχετε** την τιμή επιστροφής!
- Προσοχή:
 - δείτε **οπωσδήποτε** το [εγχειρίδιο](#) της γλώσσας για τις συμβάσεις του μοτίβου εισόδου / προσδιορισμών ανάγνωσης
 - ο μεταγλωττιστής δίνει **μόνο** προειδοποίηση αν οι προσδιορισμοί ανάγνωσης δεν είναι συμβατοί με τους τύπους των μεταβλητών που δίνονται ως παράμετροι

```

#include <stdio.h>

int main(int argc, char* argv[]) {
    int a,b;

    printf("enter 2 int values: ");

    scanf("%d %d", &a, &b);

    printf("a is %d, b is %d and a+b is %d\n", a, b, a+b);

    return(0);
}

```

προσοχή: πάντα να υπάρχει το & πριν το όνομα της μεταβλητής (επεξήγηση προσεχώς)

```

> ./myprog<enter>
enter 2 int values: 5 10 20<enter>
a is 5, b is 10 and a+b is 15
>

```

το πρόγραμμα **δεν** διαβάζει τα «επιπλέον» δεδομένα αφού αυτό δεν ζητήθηκε από το πρόγραμμα

Προσδιοριστές της `scanf()`

<http://en.cppreference.com/w/cpp/io/c/scanf>

- `%c` : `char`
- `%d` : `int`
- `%u` : `unsigned int`
- `%x` : δεκαεξαδική αναπαράσταση
- `%f` : `float` (χωρίς εκθέτη)
- `%g` : `float` (χωρίς ή με εκθέτη)
- `%ld` : `long int`
- `%lu` : `long unsigned int`
- `%lf` : `double`
- `%s` : `string` (βλέπε συνέχεια)
- `%p` : `pointer` (βλέπε συνέχεια)

Παραδείγματα

- Ανάγνωση χαρακτήρα:

```
char letter;  
scanf("%c", &letter);
```

κενό ανάμεσα στο " και στο %c για να **αγνοηθούν** τυχόν προηγούμενοι λευκοί χαρακτήρες

- Ανάγνωση τιμών με μορφή εισόδου 1/10/2025:

```
int day, month, year;  
scanf("%d/%d/%d", &day, &month, &year);
```

- Ανάγνωση τιμών με μορφή εισόδου PRICE: 3.75

```
double price;  
scanf("PRICE: %lf", &price);
```

Είσοδος από πληκτρολόγιο

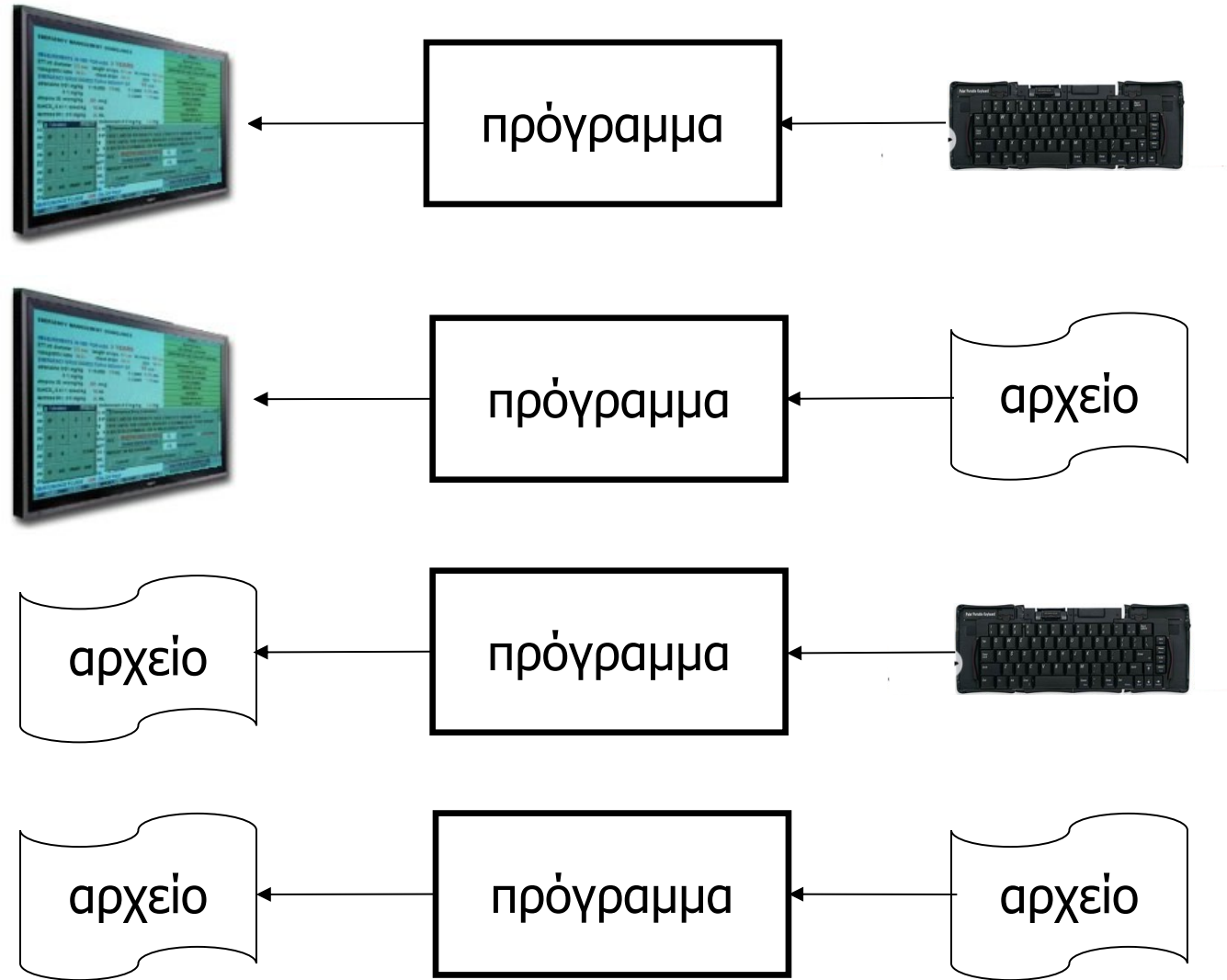
- Οι συναρτήσεις εισόδου μπλοκάρουν μέχρι να υπάρξουν τα δεδομένα προς ανάγνωση
- Οι χαρακτήρες στέλνονται για ανάγνωση στο πρόγραμμα **μόνο αφού** πατηθεί το `<enter>`
 - για να μπορεί να γίνουν διορθώσεις
 - το λάθος διαπιστώνεται «εκ των υστέρων» ...
- Το κενό ' ' είναι ένας κανονικός χαρακτήρας
- Το `<enter>` δημιουργεί τον «λευκό» χαρακτήρα `'\n'`
- Η είσοδος του προγράμματος μπορεί να τερματιστεί από το πληκτρολόγιο μέσω `Crtl-D`
 - αφού «καταναλωθούν» τα δεδομένα εισόδου που έχουν ήδη δοθεί, οι συναρτήσεις εισόδου επιστρέφουν `EOF` (end of file)
 - το πρόγραμμα πρέπει να κάνει κατάλληλο **έλεγχο**

Ανακατεύθυνση εισόδου/εξόδου (ε/ε)

- Οι συναρτήσεις εισόδου / εξόδου διαβάζουν / γράφουν από την είσοδο / έξοδο του προγράμματος
 - συνήθως η είσοδος αντιστοιχεί στο πληκτρολόγιο (με αυτόματη εκτύπωση των χαρακτήρων που εισάγονται και στην οθόνη) και η έξοδος αντιστοιχεί στην οθόνη
- Τόσο η είσοδος όσο και η έξοδος ενός προγράμματος μπορούν να **ανακατευθυνθούν**
- Π.χ. σε αρχεία έτσι ώστε οι πράξεις εισόδου να διαβάζουν δεδομένα από ένα αρχείο ή/και οι πράξεις εξόδου να γράφουν δεδομένα σε ένα αρχείο
 - τα αρχεία που δίνονται για είσοδο στο πρόγραμμα πρέπει να είναι αρχεία κειμένου (ASCII)
 - τα αρχεία που δημιουργούνται για την αποθήκευση των δεδομένων εξόδου του προγράμματος είναι ASCII

έξοδος

είσοδος



Σημειώσεις

Ανακατεύθυνση εισόδου σε αρχείο

- Αν το πρόγραμμα επιχειρήσει να διαβάσει έχοντας φτάσει το τέλος του αρχείου, επιστρέφεται EOF
 - παρόμοια συμπεριφορά με το τέλος εισόδου μέσω `Ctrl-D`

Ανακατεύθυνση εξόδου σε αρχείο

- Αν το αρχείο δεν υπάρχει, **δημιουργείται** αυτόματα
- Αν το αρχείο υπάρχει και έχει ήδη κάποια δεδομένα, **κόβεται** στην αρχή του (τα δεδομένα χάνονται)
- Εναλλακτικά, το αρχείο μπορεί να **επεκταθεί**
 - χρήση `>>` αντί για `>` κατά την ανακατεύθυνση
- Σε κάθε περίπτωση, το αρχείο μεγαλώνει **αυτόματα**, κάθε φορά που το πρόγραμμα γράφει στην έξοδο του

```

#include <stdio.h>

int main(int argc, char* argv[]) {
    char c1, c2, c3;

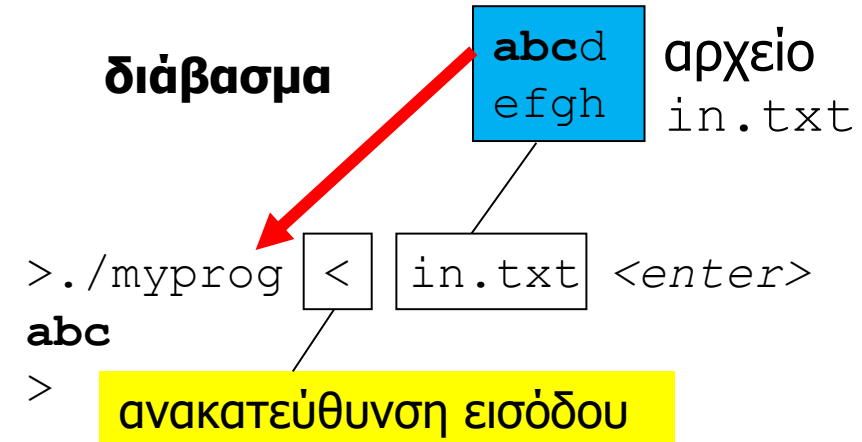
    c1=getchar();
    c2=getchar();
    c3=getchar();

    putchar(c1);
    putchar(c2);
    putchar(c3);

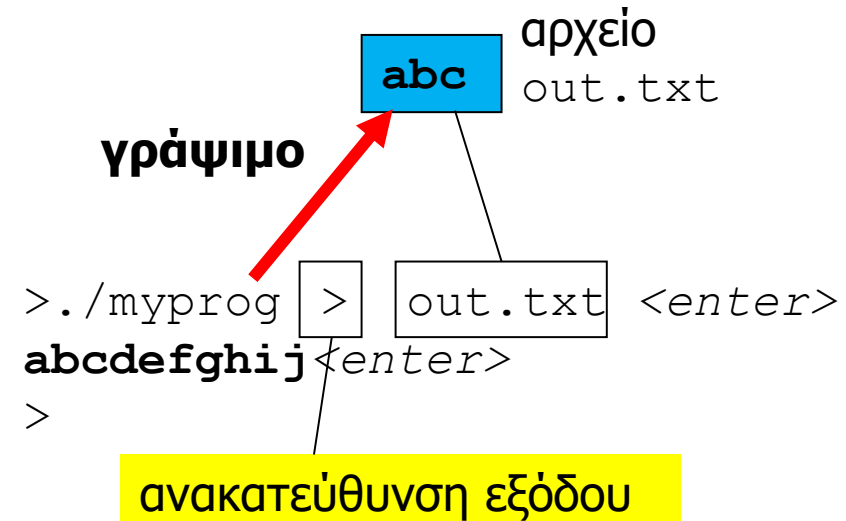
    return(0);
}

```

διάβασμα



γράψιμο



```

#include <stdio.h>

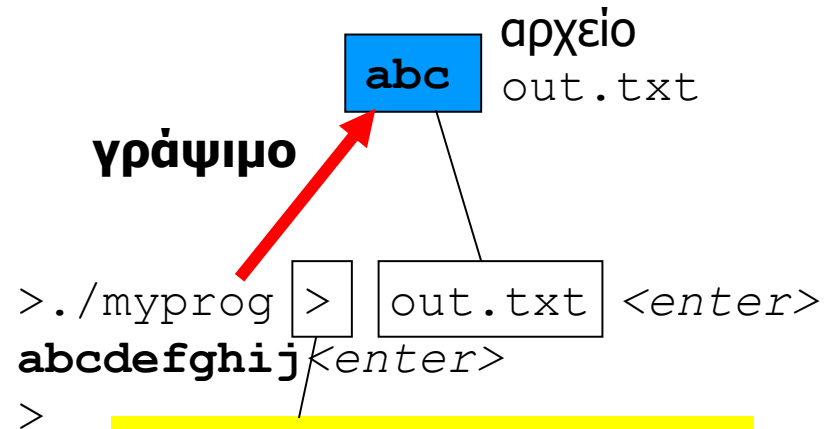
int main(int argc, char* argv[]) {
    char c1, c2, c3;

    c1=getchar();
    c2=getchar();
    c3=getchar();

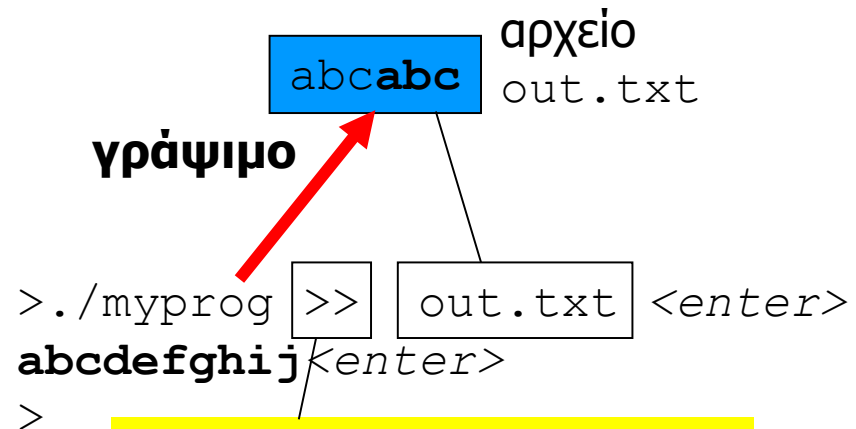
    putchar(c1);
    putchar(c2);
    putchar(c3);

    return(0);
}

```



ανακατεύθυνση εξόδου
(**χωρίς** διατήρηση των
περιεχομένων του αρχείου)



ανακατεύθυνση εξόδου
(με **διατήρηση** των
περιεχομένων του αρχείου)