Διάλεξη 3 - Συναρτήσεις

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός

Ανακοινώσεις / Διευκρινίσεις

- Η εργασία 0 θα βγει την επόμενη εβδομάδα :(
- Χρήσιμες εντολές - ευχαριστώ τους TAs
    a. ls, cd, mkdir, rmdir, touch, rm, cp, mv, gcc, nano, cat, man, clear, alias, ssh
    b. vi (πως βγαινουμε), apropos, more, less, echo, head, tail, sort
- Pipe | operator & Redirect > operator

Την Προηγούμενη Φορά

- Μνήμη στους υπολογιστές
- Δηλώσεις μεταβλητών
- Αναπαράσταση μεταβλητών στην μνήμη
- Τύπωμα μεταβλητών

Σήμερα

- Μεταβλητές και Τύπωμα μεταβλητών (συνέχεια)
- Συνέχεια του Hello World που αφήσαμε την άλλη φορά
- Συναρτήσεις και Ακέραιοι (Μαθηματικά vs C)
- Pair Programming

Τύποι Μεταβλητών

Τύπος	Συνηθισμένο μέγεθος (bytes)	Εύρος τιμών (min-max)	Παράδειγμα Τιμής
char	1	-128 … 127	'B', 0x42
short int	2	-32.768 … 32.767	42
int	4	-2.147.483.648 … 2.147.483.647	42
long int	4	-2.147.483.648 … 2.147.483.647	42L
float	4	Μικρότερη θετική τιμή: $1.17*10^{-38}$ Μεγαλύτερη θετική τιμή: $3.4*10^{38}$	42.42F
double	8	Μικρότερη θετική τιμή: $2.2*10^{-308}$ Μεγαλύτερη θετική τιμή: $1.8*10^{308}$	42.42
long double	8, 10, 12, 16		42.42L
unsigned char	1	0 … 255	'B', 0x42
unsigned short int	2	0 … 65535	42
unsigned int	4	0 … 4.294.967.295	42
unsigned long int	4	0 … 4.294.967.295	42LU

Ανάθεση σε Μεταβλητή (Variable Assignment)

Ανάθεση σε μια μεταβλητή μπορεί να γίνει κατά τον ορισμό της:

```
int x = 42;
```

Ή μετά τον ορισμό της:

```
int x;

x = 42;
```

Ή με δεκαεξαδικό τρόπο:

```
int x = 0x2A;
```

Ανάθεση σε Μεταβλητή (Variable Assignment)

Ανάθεση σε μια μεταβλητή μπορώ να γίνει κατά τον ορισμό της:

```
int x = 42;
```

Ή μετά τον ορισμό της:

```
int x;

x = 42;
```

Ή με δεκαεξαδικό τρόπο:

```
int x = 0x2A;
```

Γίνεται και σε οκταδικό: `x = 052;`

Περιεχόμενο της x **μετά** την ανάθεση

Byte 0	0	0	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	1	0	1	0	1	0
…				…				
Byte N-1	0	0	0	0	0	0	0	0
Byte N	0	1	1	0	1	0	0	0

Υπερχείλιση Ακεραίων (Integer Overflow)

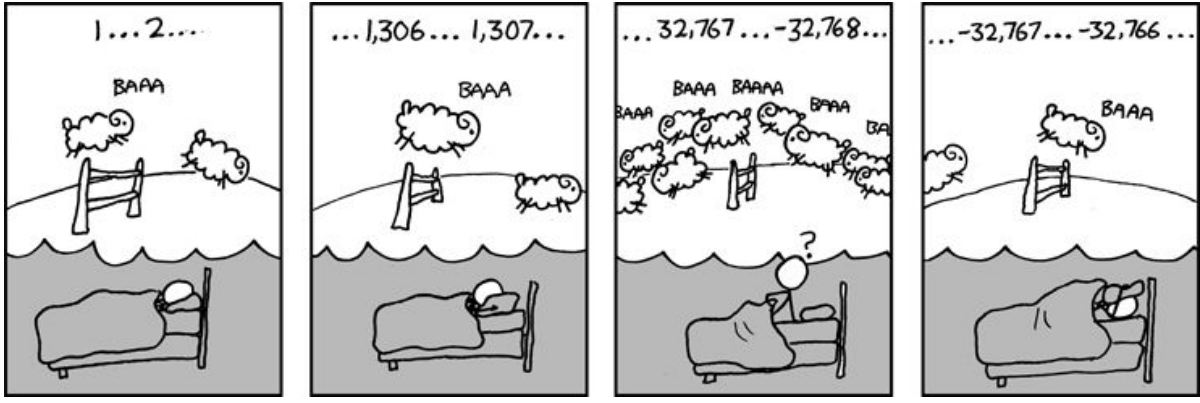
Τι θα τυπώσει το παρακάτω πρόγραμμα;

```c
#include <stdio.h>

int main() {
 printf("%d\n", 2000000000 + 2000000000);
 return 0;
}
```

```
$ ./overflow
-294967296
```

Τι συνέβη; Πως το διορθώνουμε;



Ακέραιοι με συγκεκριμένη ακρίβεια

Είπαμε ότι το μέγεθος ενός int είναι συνήθως 4 bytes - υπάρχει τρόπος να είμαστε βέβαιοι;

- Με την χρήση της `sizeof` μπορούμε να βρούμε το μέγεθος ενός τύπου - περισσότερα σε μελλοντικές διαλέξεις
- Με την χρήση της βιβλιοθήκης `<stdint.h>` μπορούμε να δηλώσουμε ακεραίους με την επιθυμητή ακρίβεια:

`int8_t, uint8_t, int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t`

Η Συνάρτηση printf

Η συνάρτηση printf() χρησιμοποιείται για το τύπωμα δεδομένων στο αρχείο εξόδου stdout (standard output)

- Έχει μία μεταβλητή λίστα παραμέτρων:
  a. Η πρώτη παράμετρος είναι ένα αλφαριθμητικό μορφοποίησης (format string), δηλαδή μία ακολουθία χαρακτήρων μέσα σε διπλά εισαγωγικά (" ") η οποία καθορίζει τον τρόπο με τον οποίο θα τυπωθούν τα δεδομένα.
  b. Οι επόμενες παράμετροι είναι προαιρετικές και, αν υπάρχουν, η printf() μπορεί να χρησιμοποιήσει τις τιμές τους
- Το αλφαριθμητικό μορφοποίησης (format string) μπορεί να περιέχει:
  a. Απλούς χαρακτήρες (οι οποίοι εμφανίζονται όπως είναι στην οθόνη)
  b. Ακολουθίες διαφυγής (Escape sequences)
  c. Προσδιοριστικά μορφοποίησης (Format specifiers)

Ακολουθίες Διαφυγής (Escape Sequences)

Η ακολουθία διαφυγής αποτελείται από μία ανάστροφη κεκλιμένη (****) (backslash) και έναν χαρακτήρα

Escape Sequence	Σημασία
\n	Αλλαγή γραμμής, σαν το πλήκτρο Enter
\r	Επαναφορά του δρομέα (cursor) στην αρχή της τρέχουσας γραμμής
\t	Τύπωμα ενός κενού ίσο με το tab, σαν το πλήκτρο Tab
****	Τύπωμα ανάστροφης κεκλιμένης (backslash)
\"	Τύπωμα διπλών εισαγωγικών (double quotes) (")
\xNN	Τύπωμα του χαρακτήρα που αντιστοιχεί σε NN σε δεκαεξαδικό
\a	Δημιουργία ηχητικού σήματος

Προσδιοριστικά Μορφοποίησης (Format Specifiers)

Τα προσδιοριστικά μορφοποίησης αποτελούνται από τον χαρακτήρα % ακολουθούμενο από έναν ή περισσότερους χαρακτήρες που προσδιορίζουν πως να γίνει η μορφοποίηση (πλήρης λίστα εδώ)

Format Specifier	Σημασία
%c	Τύπωμα ASCII χαρακτήρα
%d ή %i	Τύπωμα ακεραίου
%u	Τύπωμα unsigned (μη-προσημασμένου) ακεραίου
%f	Τύπωμα float
%llu	Τύπωμα long long unsigned int
%%	Τύπωμα του χαρακτήρα %

Note: Στον Προγραμματισμό *συνήθως* υπάρχει συμμετρία

Παρένθεση που ανοίγει πρέπει να κλείνει

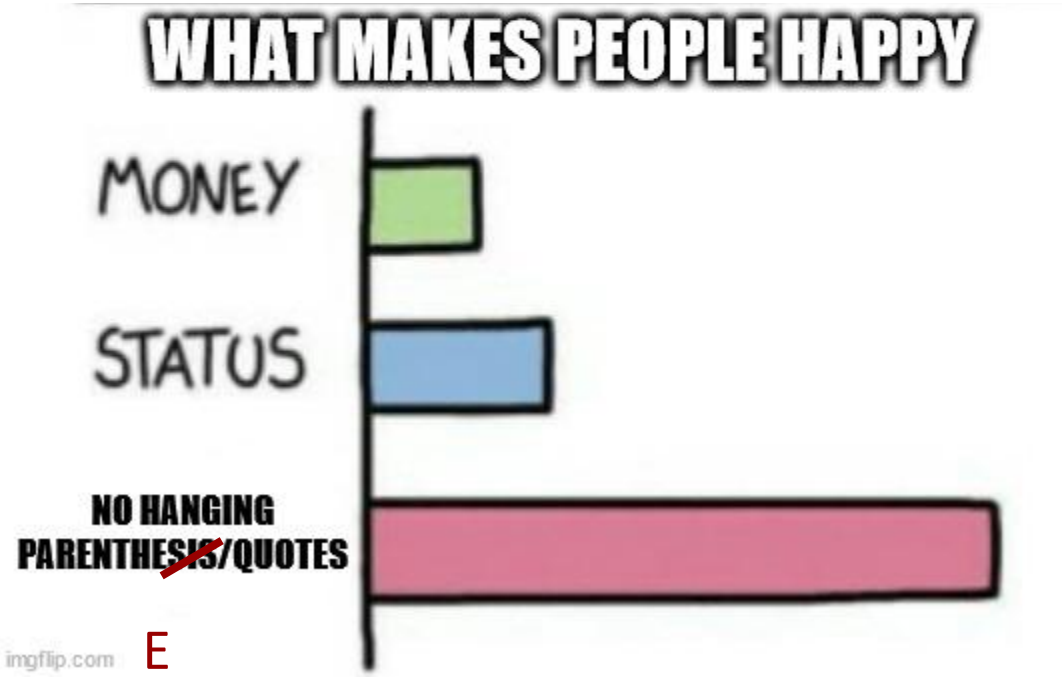
```
printf(   "      "   )
```

Εισαγωγικά (quotes) που ανοίγουν πρέπει να κλείνουν

```
main( ) {

}
```

Αγκύλες (curly braces) που ανοίγουν πρέπει να κλείνουν



Διαβάζουμε τα error messages

```
$ gcc -o hello hello.c
hello.c: In function ‘main’:
hello.c:4:26: error: expected ‘;’ before ‘return’
    4 |   printf("Hello world\n")
      |                          ^
      |                          ;
    5 |   return 0;
      |   ~~~~~~
```

Δηλώσεις Πολλών Μεταβλητών

Μεταβλητές του ίδιου τύπου μπορούν να δηλωθούν στην ίδια γραμμή, διαχωρισμένες με κόμμα (,):

```
int a;
int b;
int c;
```

Μπορεί να γραφτεί ισοδύναμα ως:

```
int a, b, c;
```

Δεσμευμένες Λέξεις (Reserved Keywords) στην C

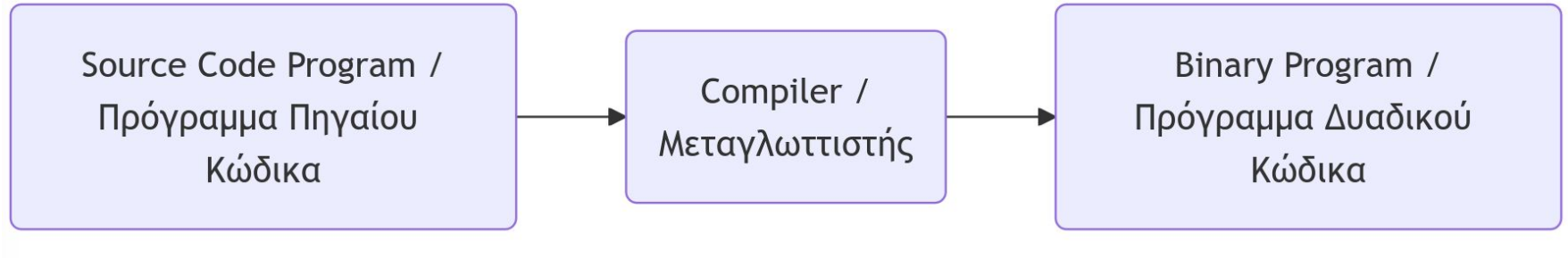
Οι δεσμευμένες λέξεις έχουν προκαθορισμένο νόημα για τον μεταγλωττιστή και **δεν** επιτρέπεται να χρησιμοποιηθούν για τον ορισμό μεταβλητών ή συναρτήσεων.

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	for	return	typedef	
default	float	short	union	

Hello World - άλλη μια φορά :)

Last Time: Compilers (Μεταγλωττιστές)

Compiler (μεταγλωττιστής) είναι ένα πρόγραμμα που μετατρέπει εντολές μιας γλώσσας προγραμματισμού σε κώδικα μηχανής ώστε να μπορεί να διαβαστεί και να τρέξει από τον υπολογιστή.



Η παραπάνω εικόνα είναι προσεγγιστική - λείπουν κάποιες λεπτομέρειες που θα προσθέσουμε σε επόμενες διαλέξεις. Στο μάθημα θα χρησιμοποιήσουμε τον GNU C Compiler ή αλλιώς `gcc`.

Ας κάνουμε compile το Hello World!

```c
/* File: helloworld.c */
#include <stdio.h>

int main() {
  printf("Hello world\n");
  return 0;
}
```

Ας κάνουμε compile το Hello World!

```
thanassis@linux14:~/examples$ gcc hello.c

thanassis@linux14:~/examples$ ls

a.out  hello.c

thanassis@linux14:~/examples$ ./a.out

Hello world

thanassis@linux14:~/examples$ file a.out

a.out: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=52fa5999c10d767a5ff30f662346478333de74bf, for GNU/Linux 3.2.0, not
stripped

thanassis@linux14:~/examples$ gcc -o hello hello.c

thanassis@linux14:~/examples$ ./hello

Hello world
```

Ανάλυση του Hello World 1/6

```c
/* File: helloworld.c */
#include <stdio.h>


int main() {
  printf("Hello world\n");
  return 0;
}
```

Σχόλια (comments): κείμενο του προγραμματιστή που συνοδεύει τον κώδικα για να τον κάνει περισσότερο σαφή για τρίτους ή και εμάς τους ίδιους, ειδικά αν έχει περάσει καιρός από τότε που γράψαμε τον κώδικα :)

Περιέχεται ανάμεσα στα /* και */

Ή μπορεί να είναι σε μία γραμμή με //:

// single line comment

Ανάλυση του Hello World 2/6

```
/* File: helloworld.c */
#include <stdio.h>

int main() {
  printf("Hello world\n");
  return 0;
}
```

#include Directive (Οδηγία): Με την οδηγία `#include <stdio.h>` ο μεταγλωττιστής συμπεριλαμβάνει (include) τα περιεχόμενα του αρχείου `stdio.h` (standard input output) στον κώδικα του προγράμματος. Το αρχείο `stdio.h` περιέχει τις βασικές (standard) δηλώσεις των συναρτήσεων με τις οποίες γίνεται εμφάνιση δεδομένων στην οθόνη (output) και εισαγωγή δεδομένων από το πληκτρολόγιο (input).

Ανάλυση του Hello World 3/6

```c
/* File: helloworld.c */

#include <stdio.h>


int main() {
  printf("Hello world\n");
  return 0;
}
```

`printf`: Η συνάρτηση βιβλιοθήκης `printf` δηλώνεται μέσα στο αρχείο `stdio.h` (για αυτό κάνουμε `#include`) και επιτρέπει την εκτύπωση του αλφαριθμητικού "Hello world\n". Ο χαρακτήρας '\n' δημιουργεί μια νέα γραμμή μετά την εμφάνιση του μηνύματος στην οθόνη.

Ανάλυση του Hello World 4/6 - Όνομα Συνάρτησης

- Ένα πρόγραμμα C ορίζεται από ένα σύνολο **συναρτήσεων** (next time).

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

Προκειμένου να μπορούμε να το τρέξουμε, πρέπει να έχει **ακριβώς μία** συνάρτηση `main`, η οποία καλείται πρώτη όταν αρχίσουμε να τρέχουμε το πρόγραμμα.

Ανάλυση του Hello World 5/6 - Επιστροφή Συνάρτησης

- Ένα πρόγραμμα C ορίζεται από ένα σύνολο **συναρτήσεων** (next time).

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

Η εντολή `return 0` επιστρέφει την τιμή της συνάρτησης όταν αποτιμηθεί. Η τιμή που επιστρέφει η `main` είναι επίσης και το **exit code** του προγράμματος, δηλαδή η τιμή που δείχνει αν το πρόγραμμα ολοκληρώθηκε με επιτυχία ή όχι. Τρέχοντας `echo $?` σε ένα Linux shell μπορούμε να δούμε την τιμή με την οποία επέστρεψε το πρόγραμμα. Τιμές διάφορες του 0 σημαίνουν ότι το πρόγραμμα **απέτυχε**.

Ανάλυση του Hello World 6/6 - Εντολές Συνάρτησης

- Ένα πρόγραμμα C ορίζεται από ένα σύνολο **συναρτήσεων** (next time).

```
int main() {
  printf(...);
  return 0;
}
```

Οι εντολές (statements) της συνάρτησης περιέχονται μέσα σε άγκιστρα `{ }` και η κάθε μία τελειώνει με `;` (semicolon / ελληνικό ερωτηματικό).

Τι κρατάμε:

1. Ένα πρόγραμμα C είναι μια σειρά από συναρτήσεις
2. Η εκτέλεση του προγράμματος ξεκινά από την συνάρτηση main

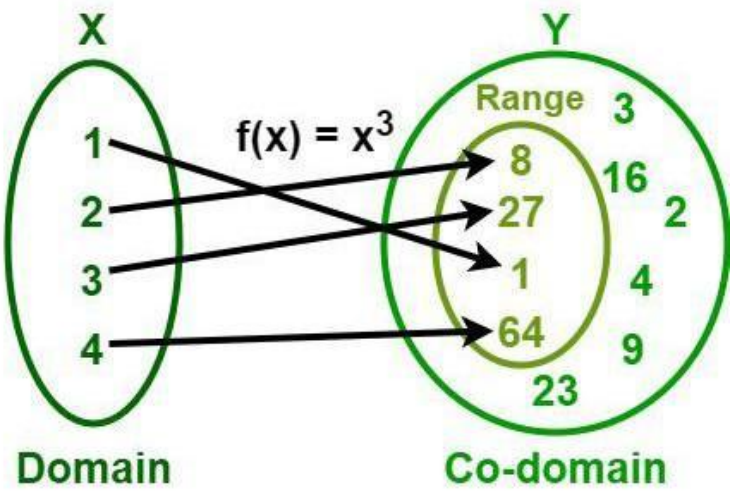
Ναι, αλλά τι είναι *συνάρτηση*;

Συναρτήσεις

Συνάρτηση: Μια αντιστοίχιση μεταξύ δύο συνόλων, που καλούνται σύνολο ορισμού και σύνολο τιμών, κατά την οποία κάθε ένα στοιχείο του πεδίου ορισμού αντιστοιχίζεται σε ένα και μόνο στοιχείο του πεδίου τιμών.

$f(x) = x^3$

$f: Z \rightarrow Z$



Πολυπαραμετρικές Συναρτήσεις

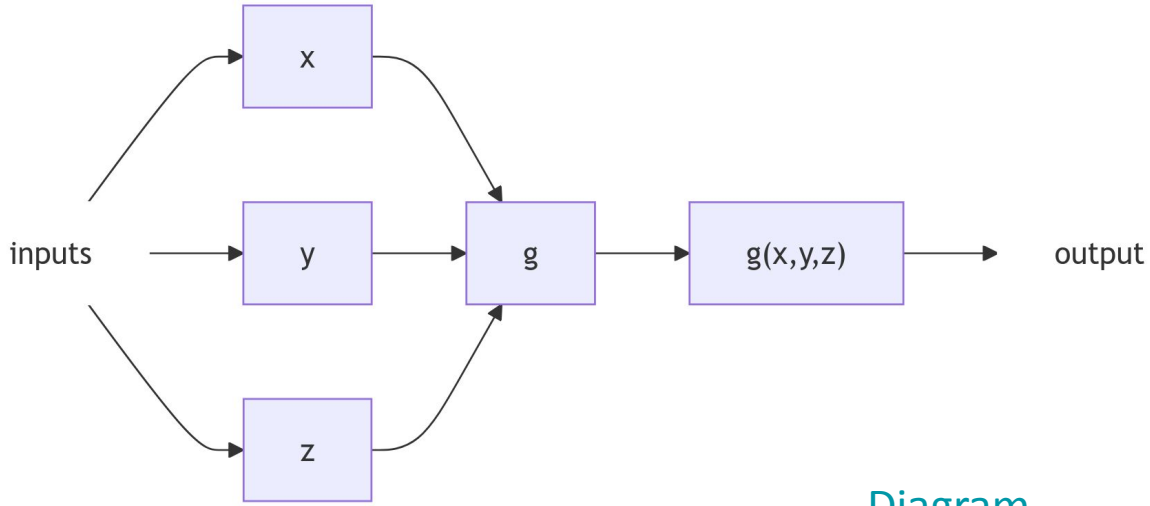
Συνάρτηση: Μια αντιστοίχιση μεταξύ δύο συνόλων, που καλούνται σύνολο ορισμού και σύνολο τιμών, κατά την οποία κάθε ένα στοιχείο του πεδίου ορισμού αντιστοιχίζεται σε ένα και μόνο στοιχείο του πεδίου τιμών.

$g(x, y, z) = x^2 + y^2 + z^2 + 42$

$g: R \times R \times R \longrightarrow R$

$h(x, y) = x + y + 1$

$h: R \times N \longrightarrow R$

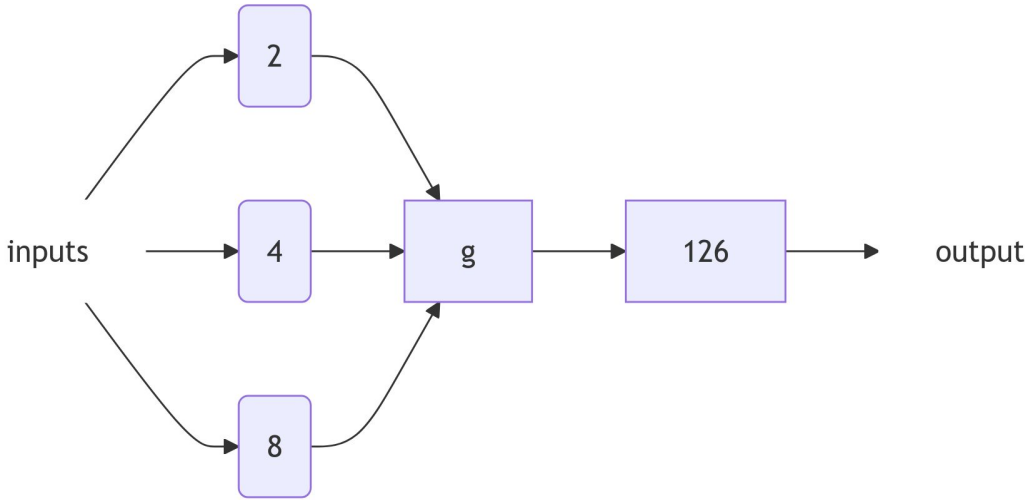


Diagram

Αποτίμηση Συναρτήσεων

$g(x, y, z) = x^2 + y^2 + z^2 + 42$

$g(2, 4, 8) = 2^2 + 4^2 + 8^2 + 42 = 126$



Συναρτήσεις στην C

Συνάρτηση είναι μια σειρά υπολογισμών που παίρνουν τις εισόδους της συνάρτησης και παράγουν την έξοδο.

Όπως και στα μαθηματικά, κάθε δεδομένο εισόδου και εξόδου της συνάρτησης έχει έναν **τύπο**, το σύνολο τιμών που μπορεί να πάρει. Για παράδειγμα, ο τύπος `int` στην C αντιπροσωπεύει τους ακεραίους (integers).

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}
```

Ορισμός Συνάρτησης (Function Definition)

Το όνομα της συνάρτησης και τον τύπο της τιμής που επιστρέφει στην μορφή `τύπος όνομα`

- `int g` στο παράδειγμα. Αν καλέσουμε `g(...)` περιμένουμε ακέραιο αποτέλεσμα.

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}
```

Ορισμός Συνάρτησης

Τα δεδομένα εισόδου ή ορίσματα της συνάρτησης εντός παρενθέσεων, επίσης της μορφής `τύπος όνομα`.

- `int x`, `int y`, `int z` είναι 3 ακέραια ορίσματα με ονόματα x, y και z.

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}
```

Ορισμός Συνάρτησης

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}
```

Το σώμα της συνάρτησης εντός των αγκυλών { } περιέχει τον υπολογισμό της τιμής της συνάρτησης.

- `return x * x + y * y + z * z + 42;` θα επιστρέψει ένα ακέραιο αποτέλεσμα.
- Οι εντολές στο σώμα της συνάρτησης διαχωρίζονται με semicolon (το ερωτηματικό `;`).

Ορισμός Συνάρτησης

Το όνομα της συνάρτησης και τον τύπο της τιμής που επιστρέφει στην μορφή `τύπος όνομα`

- `int g` στο παράδειγμα. Αν καλέσουμε `g(...)` περιμένουμε ακέραιο αποτέλεσμα.

Τα δεδομένα εισόδου ή ορίσματα της συνάρτησης εντός παρενθέσεων, επίσης της μορφής `τύπος όνομα`.

- `int x, int y, int z` είναι 3 ακέραια ορίσματα με ονόματα x, y και z.

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}
```

Το σώμα της συνάρτησης εντός των αγκυλών {} περιέχει τον υπολογισμό της τιμής της συνάρτησης.

- `return x * x + y * y + z * z + 42;` θα επιστρέψει ένα ακέραιο αποτέλεσμα.
- Οι εντολές στο σώμα της συνάρτησης διαχωρίζονται με semicolon (το ερωτηματικό `;`).

Κλήση Συνάρτησης (Function Call)

```
int g(int x, int y, int z) {
    return x * x + y * y + z * z + 42;
}


int main(int argc, char** argv) {
    ...
    int x = g(1, 2, 3);
    ...
}
```

Η κλήση της συνάρτησης γίνεται με το όνομά της και στην συνέχεια περνάμε τα ορίσματά της ανάμεσα σε παρενθέσεις χωρισμένα με ,

Ποια η τιμή του x μετά την εκτέλεση αυτής της ανάθεσης;

Pair Programming

Pair programming is a software development technique in which two programmers work together at one workstation. One, the *driver*, writes code while the other, the *observer* or *navigator*, reviews each line of code as it is typed in. The two programmers switch roles frequently.

You need to find someone that you're gonna pair-program with who's compatible with your way of thinking, so that the two of you together are a complementary force.

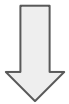
Jeff Dean

Challenge #1: Πυθαγόρειο Θεώρημα (pyth.c)

Χρειαζόμαστε ένα πρόγραμμα το οποίο **με δεδομένα τα μήκη των καθέτων πλευρών ενός ορθογωνίου τριγώνου** να τυπώνει τα ακόλουθα:

1. Το εμβαδόν του τριγώνου.
2. Την περίμετρο του τριγώνου.

```c
#include <stdio.h>
#include <math.h>
/* Returns the area of a right triangle */
double compute_area(double a, double b) {
    return (a * b) / 2.0;
}
/* Returns the perimeter of a right triangle */
double compute_perimeter(double a, double b) {
    double c = sqrt(a * a + b * b);  // hypotenuse
    double perimeter = a + b + c;
    return perimeter;
}
```



Συνεχίζεται

```c
int main(int argc, char ** argv) {

    double a = 3.0, b = 4.0;

    printf("Area of triangle: %f\n", compute_area(a, b));

    printf("Triangle perimeter: %f\n", compute_perimeter(a, b));

    return 0;

}
```

```
thanassis@linux14:~$ gcc -o pyth pyth.c
/usr/bin/X11/ld: /tmp/cc9jUwzF.o: in function `compute_perimeter':
pyth.c:(.text+0x5b): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
thanassis@linux14:~$ gcc -o pyth pyth.c -lm
thanassis@linux14:~$ ./pyth
Area of triangle: 6.000000
Triangle perimeter: 12.000000
```

Challenge #2: Προσέγγιση του π

Ο Leibniz έδειξε ότι μπορούμε να προσεγγίσουμε την τιμή του π με την ακόλουθη φόρμουλα:

$$4 \cdot \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

Θέλουμε να γράψουμε ένα πρόγραμμα που να προσεγγίζει το π. Μπορούμε;

```c
#include <stdio.h>

/* A simple way to approximate pi */

int main() {

  int i;

  double numerator = 1;

  double denominator = 1;

  double sum = 0;

  for(i = 0; i < 100000; i++) {

    sum += numerator / denominator;

    numerator = numerator * (-1);

    denominator = denominator + 2;

  }

  printf("Pi is approximately: %f\n", 4 * sum);

}
```

```
thanassis@linux14:~$ gcc -o pi pi.c
thanassis@linux14:~$ ./pi
Pi is approximately: 3.141583
```

Kahoot Time!

Για την Επόμενη Φορά - Επανάληψη

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 35 + σελίδες 58-71.
- Escape Sequences in C
- Printf tips and reference
- Ubuntu

Ευχαριστώ και καλό Σαββατοκύριακο εύχομαι!
Keep coding ;)