Η ΓΛΩΣΣΑ C

Μάθημα 19:

Βιβλιοθήκες συναρτήσεων: math.h και time.h

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Σύνοψη

- Στο σημερινό μάθημα θα ασχοληθούμε με δύο δημοφιλείς βιβλιοθήκες της C:
  - Την βιβλιοθήκη math.h που περιέχει συναρτήσεις για συνήθεια μαθηματικές πράξεις (τριγωνομετρικές, υψώσεις σε δύναμη κ.λπ.)
  - Την βιβλιοθήκη time.h που επιτρέπει να διαχειριστούμε χρόνους (όπως π.χ. την τρέχουσα ημερομηνία, τον χρόνο εκτέλεσης του προγράμματος μας κ.λπ.)

Α. Η Βιβλιοθήκη math.h

1. Τριγωνομετρικές Συναρτήσεις

- Τα πρωτότυπα των τριών (βασικών) τριγωνομετρικών συναρτήσεων:

```
double sin(double x)
```

```
double cos(double x)
```

```
double tan(double x)
```

  - οι οποίες έχουν οριστεί στο αρχείο κεφαλίδας `math.h`
- οι οποίες υπολογίζουν (αντίστοιχα) το ημίτονο, το συνημίτονο και την εφαπτομένη του x.

Α. Η Βιβλιοθήκη math.h

1. Τριγωνομετρικές Συναρτήσεις

- Επιπλέον συναρτήσεις:

```
double asin(double x)
```

  - Επιστρέφει το τόξο ημιτόνου του ορίσματος x

```
double acos(double x)
```

  - Επιστρέφει το τόξο συνημιτόνου του ορίσματος x

```
double atan(double x)
```

  - Επιστρέφει το τόξο εφαπτομένης του ορίσματος x

```
double atan2(double x, double y)
```

  - Επιστρέφει το τόξο εφαπτομένης του ορίσματος x/y

- Και έχουν οριστεί στο `math.h`

Α. Η Βιβλιοθήκη math.h

1. Τριγωνομετρικές Συναρτήσεις

- Επιπλέον συναρτήσεις:

```
double sinh(double x)
```

  - Επιστρέφει το υπερβολικό ημίτονο του ορίσματος x

```
double cosh(double x)
```

  - Επιστρέφει το υπερβολικό συνημίτονο του ορίσματος x

```
double tanh(double x)
```

  - Επιστρέφει την υπερβολική εφαπτομένη του ορίσματος x

- Και έχουν οριστεί στο `math.h`

Α. Η Βιβλιοθήκη math.h
1. Τριγωνομετρικές Συναρτήσεις

- Το παρακάτω παράδειγμα αναδεικνύει τις τριγωνομετρικές συναρτήσεις:

```
/* math.trig.c: Anadeiknyei tis trigonometrikes sinartiseis */
#include <stdio.h>
#include <math.h>

#define PI 3.1415926

main()
{
   int i;
   double v;

   for (i=0; i<4; i++)
   {
      v=i*(PI/2);
      printf("\nHmitono tou %d*PI/2=%.2f",i,sin(v));
      printf("\nSinimitono tou %d*PI/2=%.2f",i,cos(v));
      printf("\nEfaptomeni tou %d*PI/2=%.2f",i,tan(v));
      printf("\n====================");
   }
}
```

A. Η Βιβλιοθήκη math.h
2. Υψώσεις σε Δύναμη

- Οι ακόλουθες συναρτήσεις υπολογίζουν ποσότητες που σχετίζονται με δυνάμεις:

```
double pow(double x, double y)
```

  - Επιστρέφει την ποσότητα: x^y

```
double exp(double x)
```

  - Επιστρέφει την ποσότητα: e^x
    - Όπου e η σταθερά του Euler: $e \approx 2.71828$

- Και οι (πιο εξειδικευμένες) συναρτήσεις:

```
double frexp(double x, int *y)
```

  - Επιστρέφει εκείνο το $0.5 \leq r \leq 1$ και το y έτσι ώστε: $x = r \cdot 2^y$

```
double ldexp(double x, int y)
```

  - Επιστρέφει την ποσότητα $x \cdot 2^y$

- Όλες οι παραπάνω συναρτήσεις έχουν οριστεί στο `math.h`

A. Η Βιβλιοθήκη math.h
2. Υψώσεις σε Δύναμη

- Το παρακάτω παράδειγμα αναδεικνύει την συνάρτηση frexp

```
/* math.frexp.c: Anadeikniei tin frexp */
#include <stdio.h>
#include <math.h>

main()
{
   double x,r;
   int y;

   for (x=0; x<15; x+=0.5)
   {
      r=frexp(x,&y);
      printf("\n %.2f = %.2f * 2^%d",x,r,y);
   }
}
```

Α. Η Βιβλιοθήκη math.h

3. Λογαριθμικές συναρτήσεις

- Οι ακόλουθες συναρτήσεις υπολογίζουν ποσότητες που σχετίζονται με λογαρίθμους:

```
double log(double x)
```

  - Επιστρέφει τον φυσικό λογάριθμο του x: $\log_e x = \ln x$

```
double log10(double x)
```

  - Επιστρέφει τον δεκαδικό λογάριθμο του x: $\log_{10} x$

- Όλες οι παραπάνω συναρτήσεις έχουν οριστεί στο `math.h`

Α. Η Βιβλιοθήκη math.h

4. Υπολογισμός Ριζών

- Η ακόλουθη συνάρτηση υπολογίζει την τετραγωνική ρίζα του ορίσματός της:

```
double sqrt(double x)
```

  - Επιστρέφει την τετραγωνική ρίζα του x: $\sqrt{x}$
  - Το x πρέπει να είναι ≥ 0
- Η παραπάνω συνάρτηση έχει οριστεί στο `math.h`

Σημείωση:

- Δεν έχει οριστεί συνάρτηση στην C που να υπολογίζει μεγαλύτερης τάξης ρίζες (π.χ. κυβική)
- Ο υπολογισμός τέτοιων συναρτήσεων μπορεί να γίνει με χρήση της pow και ιδιοτητών δυνάμεων. Π.χ. για τον υπολογισμό της $\sqrt[3]{x}$ έχουμε $\sqrt[3]{x} = x^{\frac{1}{3}}$ συνεπώς μπορεί να υπολογιστεί με την κλήση: `pow(x, 1/3)`.

A. Η Βιβλιοθήκη math.h
4. Υπολογισμός Ριζών

- Το παρακάτω παράδειγμα αναδεικνύει τους δύο τρόπους για να υπολογίσουμε ριζικά:

```
/* math.sqrt.c: Anadeikniei ti xrisi rizwn sti C */
#include <stdio.h>
#include <math.h>

main()
{
   double x=2.0;
   double y,r;

   r=sqrt(x);
   printf("sqrt(%.2f)=%.2f",x,r);

   for (y=1; y<10; y+=1.0)
   {
      r=pow(x,1/y);
      printf("\n %.2f^(1/%.2f) = %.2f",x,y,r);
   }
}
```

A. Η Βιβλιοθήκη math.h

5. Άνω και κάτω ακέραιος

- Οι ακόλουθες συναρτήσεις υπολογίζουν τον άνω και κάτω ακέραιο του ορίσματός τους:

```
double floor(double x)
```

  - Επιστρέφει τον κάτω ακέραιο του x: $\lfloor x \rfloor$

```
double ceil(double x)
```

  - Επιστρέφει τον άνω ακέραιο του x: $\lceil x \rceil$

- Και οι δύο συναρτήσεις έχουν οριστεί στο `math.h`

Παράδειγμα:

- Για το 3.12: $\lfloor 3.12 \rfloor = 3$ και $\lceil 3.12 \rceil = 4$
- Για το -5.71: $\lfloor -5.71 \rfloor = -6$ και $\lceil -5.71 \rceil = -5$
- Για έναν ακέραιο: $\lfloor 2 \rfloor = 2$ και $\lceil 2 \rceil = 2$

Α. Η Βιβλιοθήκη math.h

5. Άνω και κάτω ακέραιος

- Το παρακάτω παράδειγμα αναδεικνύει τις συναρτήσεις floor και ceil

```
/* math.floor-ceil.c: Panw kai katw akeraios */
#include <stdio.h>
#include <math.h>

main()
{
   double x;

   x=1.2;
   printf("\n x=%.2f: floor=%.2f, ceil=%.2f",x,floor(x),ceil(x));
   x=1.5;
   printf("\n x=%.2f: floor=%.2f, ceil=%.2f",x,floor(x),ceil(x));
   x=2.0;
   printf("\n x=%.2f: floor=%.2f, ceil=%.2f",x,floor(x),ceil(x));
}
```

Α. Η Βιβλιοθήκη math.h

6. Απόλυτη Τιμή

- Η συνάρτηση abs επιστρέφει την απόλυτη τιμή ενός ακεραίου x: $|x|$

```
int abs(int x)
```

- Η συνάρτηση fabs επιστρέφει την απόλυτη τιμή ενός πραγματικού x: $|x|$

```
double fabs(double x)
```

- Και οι δύο συναρτήσεις έχουν οριστεί στο

```
math.h
```

Παράδειγμα:

- Για το 3.12: $|3.12| = 3.12$
- Για το -5.71: $|-5.71| = 5.71$

Α. Η Βιβλιοθήκη math.h
7. Συναρτήσεις διάσπασης πραγματικών

- Η συνάρτηση modf χρησιμεύει για το διαχωρισμό ακέραιου και πραγματικού μέρους:

```
double modf(double x, double *y)
```

  - Επιστρέφει εκείνο το r και το y έτσι ώστε: $x = y + r$ όπου y ακέραιος και $0 \leq r < 1$
- Η συνάρτηση fmod:

```
double fmod(double x, double y)
```

  - Επιστρέφει το πραγματικό μέρος της διαίρεσης x/y

- Και οι δύο συναρτήσεις έχουν οριστεί στο `math.h`

Α. Η Βιβλιοθήκη math.h
7. Συναρτήσεις διάσπασης πραγματικών

- Το παρακάτω παράδειγμα αναδεικνύει την συνάρτηση fmod

```
/* math.modf.c: Anadeikniei ti xrisi tis modf */
#include <stdio.h>
#include <math.h>

main()
{
   double x;
   double r,y;

   for (x=0.0; x<5.0; x+=0.6)
   {
      r=modf(x,&y);
      printf("\n%.2f = %.2f + %.2f",x,y,r);
   }
}
```

Β. Η Βιβλιοθήκη time.h

1. Αναπαράσταση του χρόνου

- Ο χρόνος αναπαρίσταται με 3 τρόπους:
  - time_t (συνώνυμο του long)
    - Το χρησιμοποιούμε για να απεικονίσουμε πλήθος δευτερολέπτων που έχουν περάσει από την ημερομηνία γέννησης του UNIX (1.1.1970)
  - clock_t (συνώνυμο του long)
    - Το χρησιμοποιούμε για να απεικονίσουμε κύκλους ρολογιού που έχουν περάσει από κάποιο γεγονός (έναρξη του προγράμματος)
  - Με τα στοιχεία της δομής tm, που αποθηκεύει τα εξής στοιχεία:

```
struct tm {
  int tm_sec; // Δευτερόλεπτα
  int tm_min; // Λεπτά
  int tm_hour; // Ώρες
  int tm_mday; // Ημέρα (του μήνα)
  int tm_mon; // Μήνας
  int tm_year; // Έτος. Έναρξη το 1900. (π.χ. 100=2000)
  int tm_wday; // Ημέρες από την Κυριακή (π.χ.Δευτέρα = 1)
  int tm_yday; // Ημέρα από την 01.01 (π.χ. 33 = 02.02)
  int tm_isdst; // Σημαία αν η ώρα έχει φως
};
```

- Τα πρωτότυπα αυτά έχουν οριστεί στο `time.h`

Β. Η Βιβλιοθήκη time.h
2. Τρέχων χρόνος

- Με τον ακόλουθο κώδικα παίρνουμε την ώρα του συστήματος:

```
/* time.c: Deuterolepta apo tin 1.1.1970 */
#include <stdio.h>
#include <time.h>

main()
{
   time_t t;
   t = time(0);
   printf("Deyterolepta= %ld", t);
}
```

- Η συνάρτηση:

```
time_t time(time_t *timeptr)
```

  - Επιστρέφει το πλήθος των δευτερολέπτων που έχουν περάσει από την 1.1.1970.
  - Το όρισμα είναι προαιρετικό (αν περάσουμε με αναφορά μία μεταβλητή τύπου time_t, τότε αποθηκεύεται σε αυτήν επίσης το πλήθος των δευτερολέπτων από την 1.1.1970).

Β. Η Βιβλιοθήκη time.h

3. Μετατροπές (οι συναρτήσεις localtime και mktime)

- Η συνάρτηση:

```
struct tm *localtime(time_t *ptr);
```

  - Δέχεται ως όρισμα το πλήθος των δευτερολέπτων και επιστρέφει μέσω δείκτη, τις τιμές της δομής tm που αντιστοιχουν σε αυτήν την ημερομηνία
  - Η συνάρτηση δεσμεύει δυναμικά το χώρο που απαιτείται για το στιγμιότυπο της δομής.
- Αντίστοιχα η συνάρτηση:

```
time_t mktime(struct tm *ntime);
```

  - Κάνει το αντίστροφο: Δέχεται ως όρισμα μία δομή tm και επιστρέφει το πλήθος των δευτερολέπτων από την 1/1/1970

Β. Η Βιβλιοθήκη time.h

3. Μετατροπές (οι συναρτήσεις localtime και mktime)

- Ο ακόλουθος κώδικας επιδεικνύει την αρχικοποίηση της δομής tm με το «τώρα»

```
/* time.convert.c: MEtatropes stis anaparastaseis */
#include <stdio.h>
#include <time.h>

main()
{
        time_t t;
        struct tm *now;
        t = time(0);
        now = localtime(&t);
        printf("%d:%d:%d",now->tm_hour, now->tm_min, now->tm_sec);
}
```

Β. Η Βιβλιοθήκη time.h

4. Εκτύπωση Χρόνων

- Η συνάρτηση:

```
char *asctime(struct tm *ptr);
```

  - Δέχεται ως όρισμα μία δομή tm και εκτυπώνει τον χρόνο που απεικονίζεται.
- Αντίστοιχα η συνάρτηση:

```
char *ctime(time_t *ptr);
```

  - Δέχεται ως όρισμα το πλήθος των δευτερολέπτων και επιστρέφει τον χρόνο που απεικονίζεται.
- Και οι δύο συναρτήσεις επιστρέφουν μία στατική συμβολοσειρά σταθερού μήκους
  - π.χ. Wed Jul 15 12:11:15 2021

```
/* time.print.c: Ektipwsi tou xronou */
#include <stdio.h>
#include <time.h>

main()
{
        time_t t;
        t = time(0);
        printf("%s", ctime(&t));
}
```

Β. Η Βιβλιοθήκη time.h

5. Μέτρηση Χρόνου Εκτέλεσης Προγράμματος

- Η συνάρτηση:

```
double difftime(time_t later, time_t earlier);
```

  - Επιστρέφει την διαφορά των χρονικών στιγμών (later-earlier) σε δευτερόλεπτα.
- Ενώ η συνάρτηση:

```
clock_t clock(void);
```

  - Επιστρέφει (σε κύκλους του ρολογιού) το χρόνο που έχει περάσει από την έναρξη του προγράμματος.
  - Η σταθερά CLOCKS_PER_SEC δίνει πόσοι κύκλοι του ρολογιού γίνονται σε 1 δευτερόλεπτο.

Β. Η Βιβλιοθήκη time.h

5. Μέτρηση Χρόνου Εκτέλεσης Προγράμματος

- Στο ακόλουθο πρόγραμμα βλέπουμε τη χρήση της clock()

```
/* clock.c: Ektipwsi toy xronoy apo tin arxi toy programmatos */
#include <stdio.h>
#include <time.h>

main()
{
        int i;
        clock_t t;
        t = clock();
        printf("Kikloi: %ld\n", t);

        printf("Kikloi ana deyterolepto: %ld\n", CLOCKS_PER_SEC);

        printf("Xronos apo tin enarksi: %f", (float)t/CLOCKS_PER_SEC);

}
```

Β. Η Βιβλιοθήκη time.h

5. Μέτρηση Χρόνου Εκτέλεσης Προγράμματος

- Το ακόλουθο πρόγραμμα επιδεικνύει τον τρόπο για να μετρήσουμε τον χρόνο που απαιτείται για να τρέξουμε ένα τμήμα κώδικα:

```
/* time.timer_program.c: Ypologismos Xronoy Ektelesis */
#include <stdio.h>
#include <time.h>
#define N 1000000000
main()
{
   clock_t c1,c2;
   long i;

   c1 = clock();

   for (i=1; i<=N; i++);

   c2 = clock();

   printf("Me time: Arxi=%ld, Telos=%ld, Xronos=%.3f", c1,c2,
                                                (double)(c2-c1)/CLOCKS_PER_SEC);
}
```

Γ. Ασκήσεις

1. Υπολογισμός παράστασης

Γράψτε ένα πρόγραμμα το οποίο:

1. Να διαβάζει από την είσοδο έναν πραγματικό αριθμό x.
2. Να τυπώνει το αποτέλεσμα της λογιστικής συνάρτησης (μέσω δημιουργίας και κλήσης μιας κατάλληλης συνάρτησης):

$$f(x) = \frac{1}{1 + e^{-x}}$$

Γ. Ασκήσεις

2. Αριθμοί Fibonacci και χρόνος εκτέλεσης

Στο μάθημα 6, εφαρμογή 4, είδαμε τον αναδρομικό ορισμό των αριθμών Fibonacci.

Κατασκευάστε ένα πρόγραμμα το οποίο να υπολογίζει το χρόνο που απαιτείται για τον υπολογισμό κάθε αριθμού Fibonacci από το 1 έως το 50.