Διαδικαστικός Προγραμματισμός

(ECE_Y215)

Επιπρόσθετες Βιβλιοθήκες στη C

Στόχοι

- Ο σκοπός της σημερινής διάλεξης είναι να παρουσιάσει τον **ρόλο και την λειτουργία** των επιπρόσθετων βιβλιοθηκών στη C
- Να αναφερθούμε σε **παραδείγματα** ώστε να εμπεδώσουμε καλύτερα τις θεωρητικές γνώσεις
- Να υποστηρίξει την περεταίρω ενασχόληση των φοιτητών στο αντικείμενο της διάλεξης μέσω της 1ης **Δραστηριότητας**

Η σημερινή διάλεξη συνοδεύεται **από μια δραστηριότητα**

- Ημερομηνία παράδοσης: **10 Μαΐου 2021**
- Επιβράβευση με +0.25 μονάδες στον τελικό βαθμό του μαθήματος

Πρόγραμμα Διαλέξεων & Δραστηριοτήτων

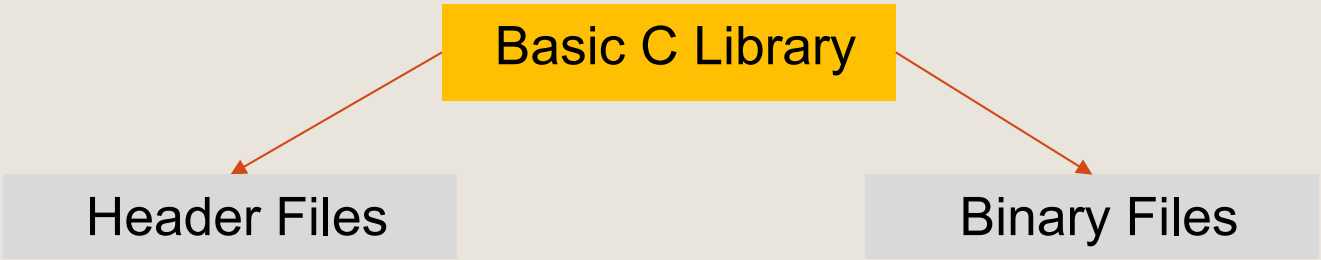
Ημερομηνία	Περιεχόμενο
23 Απριλίου 2021 09:00 – 10:00	Εισαγωγή & GSL - GNU Scientific Library (1/2) • *εξειδικευμένος επιστημονικός υπολογισμός*
14 Μαΐου 2021 09:00 – 10:00	GSL - GNU Scientific Library (2/2) • *εξειδικευμένος επιστημονικός υπολογισμός*
21 Μαΐου 2021 09:00 – 10:00	GMP - GNU MultiPrecision Library (1/2) • *αυθαίρετη αριθμητική ακρίβεια σε αριθμούς*
28 Μαΐου 2021 09:00 – 10:00	GMP - GNU MultiPrecision Library (2/2) • *αυθαίρετη αριθμητική ακρίβεια σε αριθμούς*

Δραστηριότητες	Παράδοση
1η Δραστηριότητα: 23 Απριλίου	10 Μαΐου 2021 (+ 0.25 μονάδες στον τελικό βαθμό)
2η Δραστηριότητα: 14 Μαΐου	28 Μαΐου 2021 (+ 0.25 μονάδες στον τελικό βαθμό)

Βιβλιοθήκες στη C

Βιβλιοθήκη είναι μια συλλογή **μεταγλωττισμένων μονάδων** που μπορούν να συνδεθούν (**linked**) στα προγράμματά μας μέσω των διεπαφών (**header files - interfaces**) που παρέχουν.

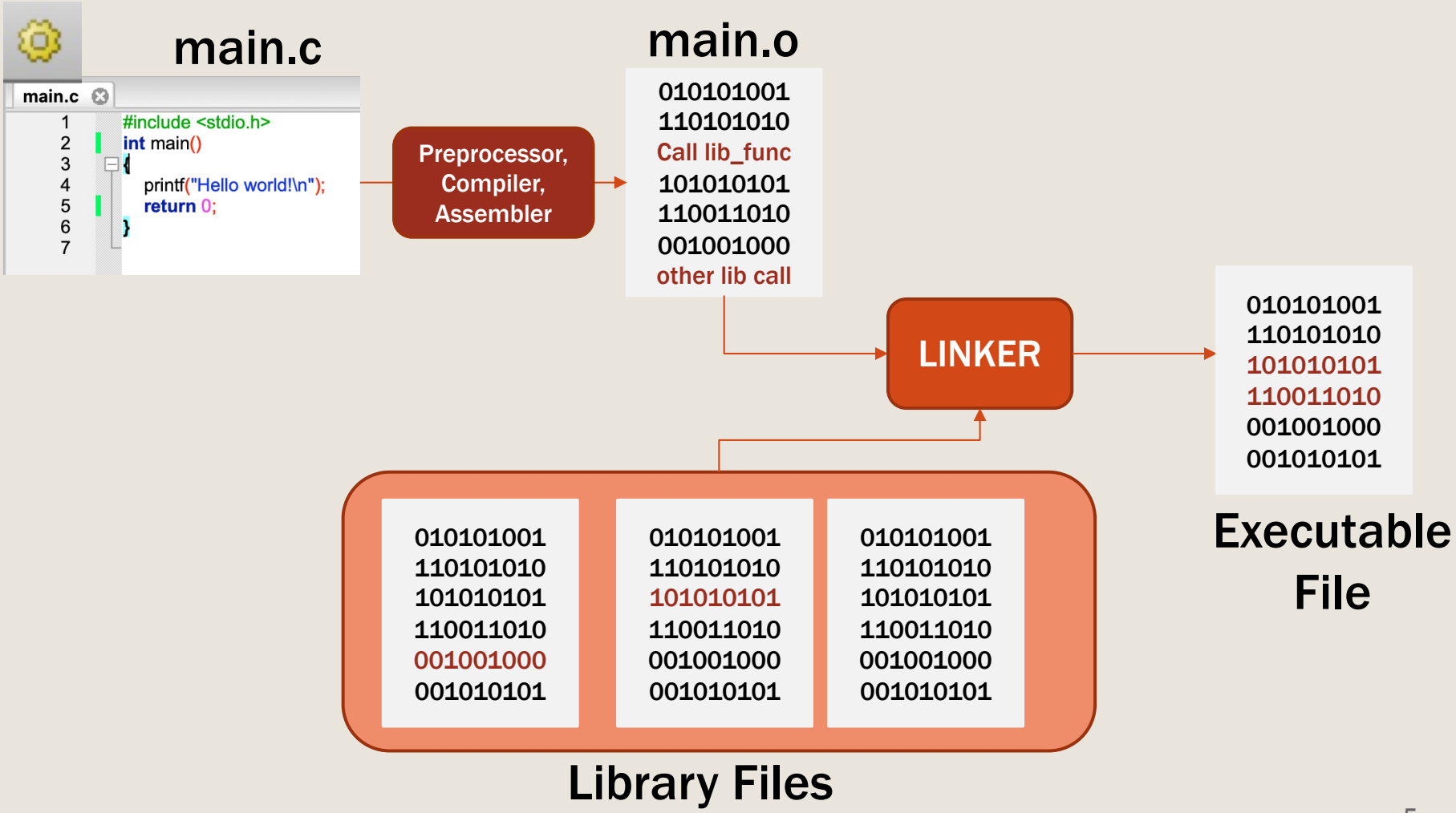
Με άλλα λόγια κάθε βιβλιοθήκη αποτελείτε από **δυαδικά** αρχεία σε object code (*.o) που περιέχουν **υλοποιήσεις** όλων των συναρτήσεων που έχουν **δηλωθεί** στα αρχεία επικεφαλίδων .h



- `<stdio.h>` είναι η διεπαφή (interface) που περιέχει συναρτήσεις I/O.

Πηγαίος κώδικας της βασικής βιβλιοθήκης: **http**://ftp.gnu.org/gnu/glibc/

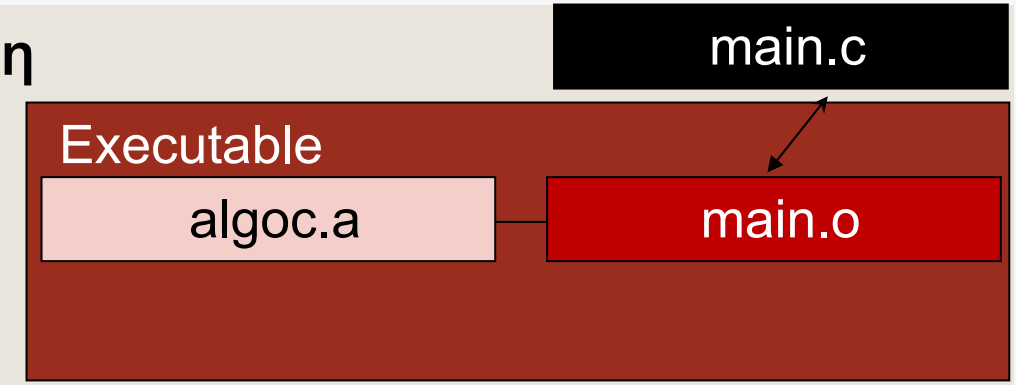
Βιβλιοθήκες στη C



Στατικά και Δυναμικά Διασυνδεδεμένες Βιβλιοθήκες στη C

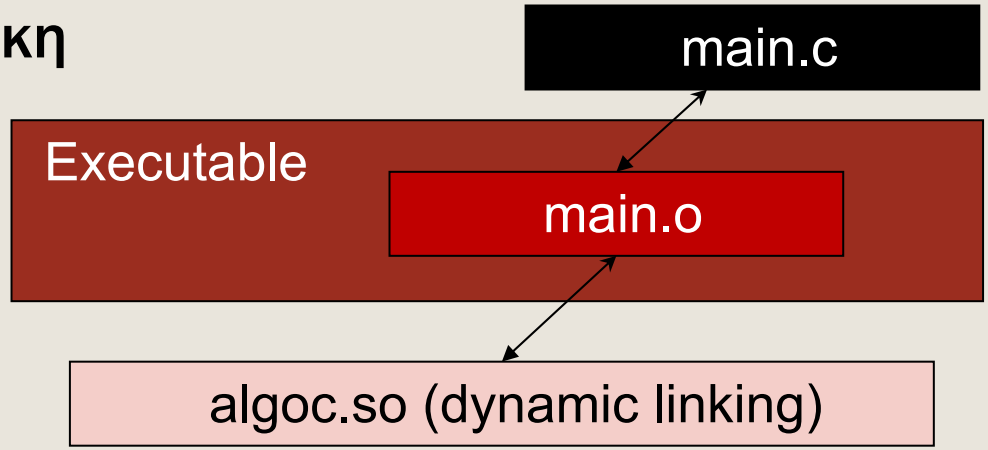
Σύνδεση με στατική βιβλιοθήκη

Η σύνδεση της βιβλιοθήκης, με τον πηγαίο κώδικά των προγραμμάτων μας, γίνεται ***κατά την μεταγλώττιση*** σε ένα εκτελέσιμο αρχείο.



Σύνδεση με δυναμική βιβλιοθήκη

Η σύνδεση της βιβλιοθήκης, με τον πηγαίο κώδικά των προγραμμάτων μας, γίνεται το ***λειτουργικό σύστημα*** σε χρόνο εκτέλεσης του προγράμματός μας.



Δημιουργία Βιβλιοθήκης (Bubble Sort)

Η βιβλιοθήκη θα περιέχει συνάρτηση που υλοποιεί τον αλγόριθμο ευθείας ανταλλαγής - Bubble sort: *https://en.wikipedia.org/wiki/Bubble_sort*

1ος Κύκλος

1	3	5	4	2
1	3	5	4	2
1	3	5	4	2
1	3	4	5	2
1	3	4	2	5

2ος Κύκλος

1	3	4	2	5
1	3	4	2	5
1	3	4	2	5
1	3	2	4	5

3ος Κύκλος

1	3	2	4	5
1	3	2	4	5
1	2	3	4	5

4ος Κύκλος

1	2	3	4	5
1	2	3	4	5

N-1 Κύκλοι

1	2	3	..	N

Δημιουργία Βιβλιοθήκης με Γραμμή Εντολών

Βήμα 1ο : Υλοποιούμε τις διεπαφές: *algorithms.h*

algorithms.h

```
extern void bubble_sort(int, int []);
```

Βήμα 2ο : Υλοποιούμε τις διεπαφές: *algorithms.c*

algorithms.c

```
#include "algorithms.h"
void bubble_sort(int elements,int tbl[])
{
    int x,y,buff;
     for (x=0; x < elements-1; x++)
        for (y=0; y < elements-x-1; y++)
            if (tbl[y] > tbl[y+1])
            {
                buff=tbl[y];
                tbl[y]=tbl[y+1];
                tbl[y+1]=buff;
            }
}
```

Βήμα 3ο : Δημιουργούμε το αρχείο αντικειμένου:*algorithms.o*

```
gcc -c -g algorithms.c
```

Βήμα 4ο : Δημιουργούμε την βιβλιοθήκη

Στατική

```
ar -cvq algoc.a algorithms.o
```

*Σε windows (algoc.lib)

Δυναμική

```
gcc –shared -o algoc.so algorithms.o –lm
gcc -dynamiclib algorithms.o -o algoc.dylib
```

*Σε windows (algoc.dll)

Βήμα 5ο : Χρήση της βιβλιοθήκης: *test.c*

- Στατική διασύνδεση του αρχείου **test.c** με την βιβλιοθήκη **algoc.a** στο εκτελέσιμο αρχείο **myapp**

```
gcc -o myapp test.c algoc.a
```

*Σε windows (myapp.exe)

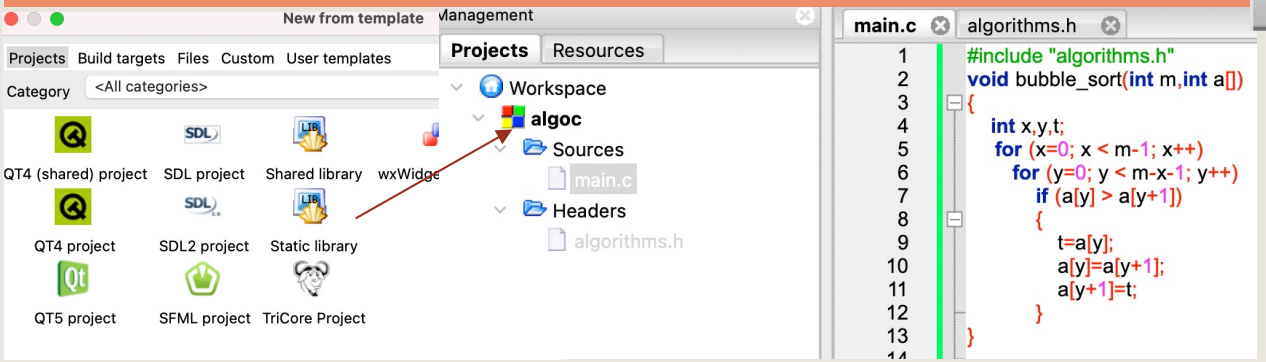
test.c

```
#include <stdio.h>
#include "algorithms.h"
int main()
{
    int x[] = {1,3,5,4,2};
    int i;
    bubble_sort(5,x);
    for (i=0; i < 5; i++)
        printf("%d\n",x[i]);
  return 0;
}
```

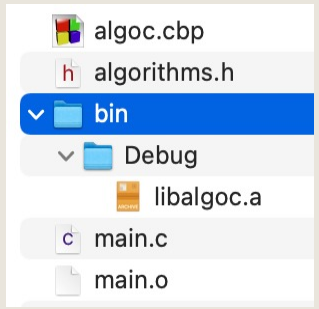
Δημιουργία Βιβλιοθήκης στο Code::Blocks

Το ίδιο παράδειγμα στο Code::Blocks

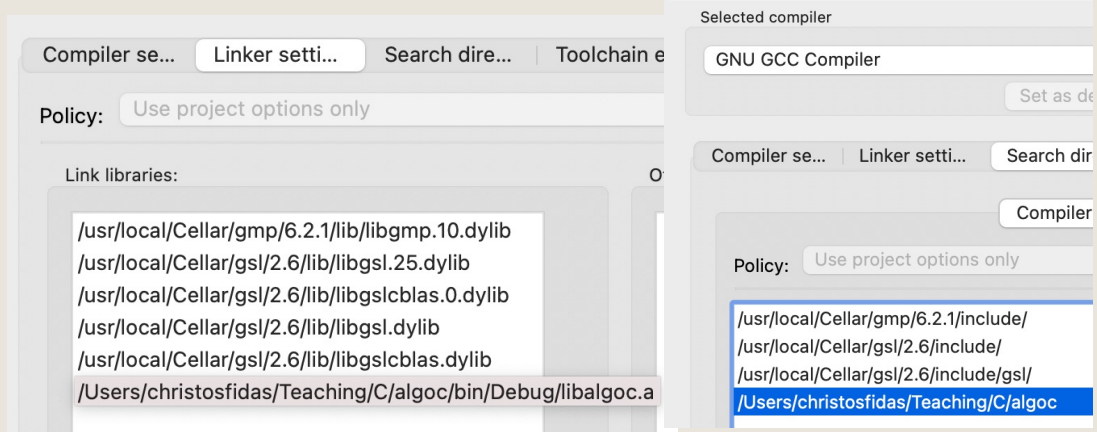
Υλοποίηση Βιβλιοθήκης (Βήμα 1ο και 2ο)



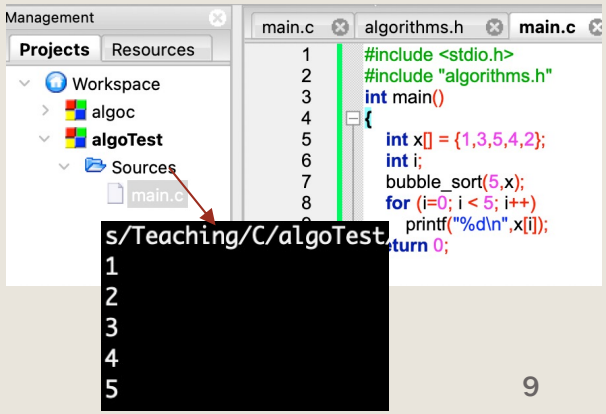
Δημιουργία Βιβλιοθήκης (Βήμα 3ο και 4ο)



Ρυθμίσεις Compiler (Linker Settings και Search Directory)



Χρήση Βιβλιοθήκης (Βήμα 5ο)



GSL – GNU Scientific Library
https://www.gnu.org/software/gsl/

- Η βιβλιοθήκη GSL **είναι μια αριθμητική βιβλιοθήκη** η οποία μπορεί να χρησιμοποιηθεί σε προγράμματα C
- **Είναι δωρεάν** βάσει της άδειας GNU General Public License.
- Η βιβλιοθήκη **παρέχει ένα ευρύ φάσμα με πληθώρα διαθέσιμων λειτουργιών σχετικά με:**
  - **Διανύσματα και Πίνακες**
  - **Γεννήτριες Ψευδοτυχαίων Αριθμών (PRNGs)**
  - Μετασχηματισμοί Fourier
  - Διατάξεις, Μεταθέσεις, Συνδυασμοί
  - Προσομοιώσεις Monte Carlo
  - Ρίζες Πολυωνύμων
  - κ.α.

Οδηγίες εγκατάστασης της GSL είναι διαθέσιμες στο eclass

Εγκατάσταση βιβλιοθηκών σε linux

Για **ubuntu** 18.04, 20.4 και τη βιβλιοθήκη GNU Scientific Library (gsl) πρώτα εγκαθιστούμε τη βιβλιοθήκη, με την ακόλουθη εντολή. Στο command prompt:

```
sudo apt-get install libgsl-dev
```

Η διαδικασία εγκατάστασης θα γίνει μία μόνο φορά.

Στη συνέχεια μπορούμε να κάνουμε compile και link ένα πρόγραμμα (έστω ότι το αρχείο πηγαίου κώδικα λέγεται gsltest.c) που χρησιμοποιεί τη βιβλιοθήκη, ως εξής

```
gcc -std=c90 -Wall -Wextra -pedantic -pedantic-errors gsltest.c -o gsltest.out -lgsl -lgslcblas
```

Δημιουργείται το gsltest.out το οποίο εκτελείται ως

```
./gsltest.out
```

```
paliuras@kissavos:~/gsltest$ gcc -std=c90 -Wall -Wextra -pedantic -pedantic-errors gsltest.c -o gsltest.out -lgsl -lgslcblas
paliuras@kissavos:~/gsltest$ ./gsltest.out
initial permutation: 0 1 2 3 4 5 6 7 8 9
 random permutation: 0 5 4 3 7 8 6 2 1 9
inverse permutation: 0 8 7 3 2 1 6 4 5 9
paliuras@kissavos:~/gsltest$
```

Ο πηγαίος κώδικας είναι το πρώτο παράδειγμα από το https://www.gnu.org/software/gsl/doc/html/permutation.html

Για **centos**, η εγκατάσταση της gsl γίνεται με `sudo yum install gsl-devel`

Εγκατάσταση βιβλιοθηκών με MSYS2 για windows

Χρήση Διανυσμάτων Vector της GSL

Η διεπαφή για την πρόσβαση στις συναρτήσεις διαχείρισης διανυσμάτων είναι η <gsl/gsl_vector.h>

- **Δημιουργία:**

```
gsl_vector * gsl_vector_alloc(size_t n)
```

- **Απελευθέρωση μνήμης:**

```
void gsl_vector_free(gsl_vector* v)
```

- **Πολλές διαθέσιμες συναρτήσεις:**

```
int gsl_vector_reverse(gsl_vector * v)
double gsl_vector_max(const gsl_vector * v)
```

https://www.gnu.org/software/gsl/doc/html/vectors.html

Χρήση Διανυσμάτων Vector της GSL

Η διεπαφή για την πρόσβαση στις συναρτήσεις διαχείρισης διανυσμάτων είναι η <gsl/gsl_vector.h>

```
#include <stdio.h>
#include <gsl/gsl_vector.h>
int main (void) {
   int i;
   gsl_vector * pv = gsl_vector_alloc (10);

    for (i = 0; i < 10; i++)
       gsl_vector_set (pv, i, 5.25);

   for (i = 0; i < 10; i++)
      printf ("element %d = %g\n", i, gsl_vector_get (pv, i));

   gsl_vector_free (pv);
   return 0;
}
```

Source: https://www.gnu.org/software/gsl/doc/html/vectors.html#example-programs-for-matrices

Χρήση Πινάκων Matrices της GSL

Η διεπαφή για την πρόσβαση στις συναρτήσεις διαχείρισης πινάκων είναι η <gsl/gsl_matrix.h>

- `gsl_matrix * gsl_matrix_alloc(size_t n1, size_t n2)`
- `void gsl_matrix_set (gsl_matrix * m,const size_t i, const size_t j, double x)`
- `double gsl_matrix_get (const gsl_matrix * m, const size_t i, const size_t j)`

```c
#include <gsl/gsl_matrix.h>

int main (void)
{
      int i, j;
      gsl_matrix * m = gsl_matrix_alloc (10, 3);
      for (i = 0; i < 10; i++)
         for (j = 0; j < 3; j++)
         gsl_matrix_set (m, i, j, 0.23 + 100*i + j);

      for (i = 0; i < 10; i++)
         for (j = 0; j < 3; j++)
               printf ("m(%d,%d) = %g\n", i, j, gsl_matrix_get (m, i, j));

      gsl_matrix_free (m);
return 0;
}
```

https://www.gnu.org/software/gsl/doc/html/vectors.html#matrices

1η Δραστηριότητα: Γεννήτριες Ψευδοτυχαίων Αριθμών

- Οι γεννήτριες ψευδοτυχαίων αριθμών χρησιμοποιούνται ευρέως στον επιστημονικό υπολογισμό
- Παράδειγμα
  - προσομοίωση Monte Carlo για τον υπολογισμό του αριθμού π

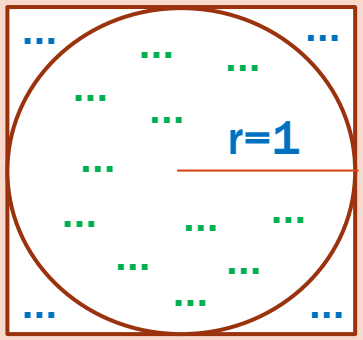
Στο Σχήμα 1, ο λόγος των εμβαδών κύκλου και τετραγώνου είναι π, που σημαίνει ότι:

π=Σ(σημείων μέσα στον κύκλο)/Σ(σημείων μέσα στο τετράγωνο)

Αλγόριθμος

1. Αρχικοποίηση μεταβλητών *in_circle* και *total* σε 0
2. Δημιουργία ψευδοτυχαίου αριθμού *x* στο διάστημα [0,1)
3. Δημιουργία ψευδοτυχαίου αριθμού *y* στο διάστημα [0,1)
4. Υπολογισμός *d = x*x + y*y*
5. Εάν *d* <= 1, *in_circle*++
6. *total*++

Για μεγάλο αριθμό επαναλήψεων των βημάτων 1-6 και με ποιοτικές **γεννήτριες ψευδοτυχαίων** αριθμών μπορούμε να υπολογίσουμε τον αριθμό *π = in_circle/total*



Σχήμα 1

Γραμμικές Γεννήτριες Ψευδοτυχαίων Αριθμών

- Linear Congruential Generators (Lehmer, 1951)

$X_{n+1}=(ax_n + c) \bmod(m)$, Οι τιμές: a, c, m αναλόγως με την γεννήτρια

Ποιοτικά χαρακτηριστικά των PRNGs:

- Ομοιόμορφη κατανομή
- Μεγάλη περιοδικότητα
- Μη επαναληψιμότητα
- Ταχύτητα εκτέλεσης

Διαφορετικές Υλοποιήσεις Γεννητριών Ψευδοτυχαίων Αριθμών

Rand της Βασικής Βιβλιοθήκης C

```
/* Αρχικοποίηση της γεννήτριας */
void srand(unsigned int seed);

/* Δημιουργία ψευδοτυχαίων αριθμών*/
int rand();
```

Περίοδος Rand ($2^{31} - 1$)

The Mersenne Twister της Βιβλιοθήκης GSL

```
/* Δημιουργία της γεννήτριας*/
gsl_rng
*r=gsl_rng_alloc(gsl_rng_mt19937);

/* Αρχικοποίηση της γεννήτριας*/
void gsl_rng_set(const gsl_rng
* r, unsigned long int s)

/* Παραγωγή ψευδοτυχαίων αριθμών */
double gsl_rng_uniform(const
gsl_rng * r)
```

Περίοδος MT ($2^{19937} - 1$)

https://www.gnu.org/software/gsl/doc/html/rng.html

1η Δραστηριότητα

- **Ζητούμενα της Δραστηριότητας**
  - Δημιουργήστε ένα πρόγραμμα στη C το οποίο να αξιολογεί, **αναφορικά με την ομοιόμορφη κατανομή και την προβλεψιμότητα**, τις ακόλουθες γεννήτριες ψευδοτυχαίων αριθμών:
    - **rand** της βασικής βιβλιοθήκης <**stdlib.h**>
    - **Mersenne Twister** της GSL <**gsl/gsl_rng.h**>

- **Παραδοτέα της Δραστηριότητας**
  - **Έκδοση 1:** Δημιουργία, αρχικοποίηση των γεννητριών και έλεγχος λειτουργίας
  - **Έκδοση 2:** Έλεγχος της ομοιόμορφης κατανομής των τυχαίων αριθμών για 10^9 κληρώσεις
  - **Έκδοση 3:** Έλεγχος της προβλεψιμότητας των αποτελεσμάτων

Ημερομηνία παράδοσης: **10 Μαΐου 2021**
Επιβράβευση με +0.25 μονάδες στον τελικό βαθμό του μαθήματος

Έκδοση 1: Δημιουργία, αρχικοποίηση των γεννητριών και έλεγχος λειτουργίας

- **Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης της 1ης έκδοσης**

  Βήματα: α) Αρχικοποιήσετε της γεννήτριες, β) Δημιουργήστε 10 τυχαίους αριθμούς στο διάστημα [1,45] και γ) δείτε το αποτέλεσμα

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include<unistd.h>
#include <gsl/gsl_rng.h>

void seed_gsl_mt_rng();
void seed_rand_rng();
void draws(unsigned long int draws, int range) ;
static gsl_rng *r;

int main()
{
   seed_gsl_mt_rng();
   seed_rand_rng();

   printf("PRNGs seeded succesfully. Initiate trials.\n");
   draws(10, 45);

    gsl_rng_free(r);
   return 0;
}
```

```
PRNGs seeded succesfully. Initiate trials.
        [Draw]          [Rand]          [Mersen Twister]
        [1]             25               14
        [2]             19               8
        [3]             40               30
        [4]             32               2
        [5]             35               24
        [6]             20               12
        [7]             42               27
        [8]             25               10
        [9]             33               30
        [10]            14               32
```

Έκδοση 2: Έλεγχος της ομοιόμορφης κατανομής των τυχαίων αριθμών για 10^9 κληρώσεις

- **Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης 2ης έκδοσης**

  Βήματα: Για 10^9 κληρώσεις αποθηκεύστε την ποσοστιαία συχνότητα εμφάνισης των αριθμών σε κάθε γεννήτρια και ερμηνεύστε το αποτέλεσμα. Χρησιμοποιήστε GSL - Vectors

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include<unistd.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_vector.h>
void seed_gsl_mt_rng();
void seed_rand_rng();
void draws() ;
void print_frequency_tables();

static gsl_rng *r;
static gsl_vector * v_rand_frequency;
static gsl_vector * v_mt_frequency;
static unsigned long int nof_draws=1000000000;
static int range=45;

int main()
{
   seed_gsl_mt_rng();
   seed_rand_rng();

   v_rand_frequency = gsl_vector_alloc (range);
   v_mt_frequency = gsl_vector_alloc (range);

   draws();
   print_frequency_tables();

   gsl_vector_free (v_rand_frequency);
   gsl_vector_free (v_mt_frequency);
   gsl_rng_free(r);
   return 0;
}
```

```
---Frequency tables ---
 [Number]        [Perc. Freq.Rand Rng]   [Perc. Freq.MT Rng]
[1]                2.222                   2.223
[2]                2.222                   2.223
[3]                2.222                   2.223
[4]                2.222                   2.223
[5]                2.223                   2.223
[6]                2.223                   2.223
[7]                2.222                   2.223
[8]                2.222                   2.223
[9]                2.222                   2.223
[10]               2.223                   2.223
[11]               2.223                   2.223
[12]               2.222                   2.223
[13]               2.223                   2.223
[14]               2.222                   2.223
[15]               2.222                   2.223
[16]               2.222                   2.223
[17]               2.222                   2.223
[18]               2.222                   2.223
[19]               2.222                   2.223
[20]               2.222                   2.223
[21]               2.222                   2.223
[22]               2.223                   2.223
[23]               2.223                   2.223
[24]               2.223                   2.223
[25]               2.222                   2.223
[26]               2.221                   2.223
[27]               2.222                   2.223
[28]               2.222                   2.223
[29]               2.222                   2.223
[30]               2.223                   2.223
[31]               2.223                   2.223
[32]               2.222                   2.223
[33]               2.222                   2.223
[34]               2.222                   2.223
[35]               2.223                   2.223
[36]               2.222                   2.223
[37]               2.223                   2.223
[38]               2.222                   2.223
[39]               2.223                   2.223
[40]               2.222                   2.223
[41]               2.222                   2.223
[42]               2.222                   2.223
[43]               2.222                   2.223
[44]               2.222                   2.223
[45]               2.222                   2.223
```

Έκδοση 3: Έλεγχος της προβλεψιμότητας των αποτελεσμάτων

- **Ενδεικτικός πηγαίος κώδικας και παράδειγμα εκτέλεσης της 3ης έκδοσης**

  Βήματα: Αποθηκεύουμε για μια περίοδο της rand τους παραγόμενους τυχαίους αριθμούς των δυο γεννητριών. Στη συνέχεια κάνουμε επιπλέον κληρώσεις. Ερμηνεύστε το αποτέλεσμα.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include<unistd.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_vector.h>
void seed_gsl_mt_rng();
void seed_rand_rng();
void draws() ;
void forecast();

static gsl_rng *r;
static gsl_vector * v_rand_results;
static gsl_vector * v_mt_results;
static unsigned long int nof_draws=RAND_MAX-1;
static int range=45;
int main()
{
   seed_gsl_mt_rng();
   seed_rand_rng();

   v_rand_results = gsl_vector_alloc (nof_draws);
   v_mt_results = gsl_vector_alloc (nof_draws);

   draws();
   forecast();

   gsl_vector_free (v_rand_results);
   gsl_vector_free (v_mt_results);
   gsl_rng_free(r);
   return 0;
}
```

```
********************PREDCITING RAND*********************
 [Rand Predict vs. Draw ]          [GSL MT Predict vs. Draw]
       31  vs.  31                      8  vs.  29
       11  vs.  11                     32  vs.  41
       41  vs.  41                      9  vs.  24
        3  vs.   3                     26  vs.  22
       10  vs.  10                      6  vs.  41
        0  vs.   0                     38  vs.  13
       37  vs.  37                     25  vs.   7
       32  vs.  32                     12  vs.  35
        3  vs.   3                     24  vs.   9
       42  vs.  42                     33  vs.   2
       36  vs.  36                      1  vs.   7
       39  vs.  39                     13  vs.  17
       24  vs.  24                     12  vs.   9
       34  vs.  34                     16  vs.  11
        6  vs.   6                      4  vs.  16
       38  vs.  38                     19  vs.  28
       34  vs.  34                     39  vs.  11
       22  vs.  22                      1  vs.  15
        8  vs.   8                     23  vs.  31
       35  vs.  35                      3  vs.  24
        7  vs.   7                     36  vs.  42
        3  vs.   3                     36  vs.  10
       18  vs.  18                     24  vs.  12
       15  vs.  15                     41  vs.   4
       43  vs.  43                      9  vs.  25
       35  vs.  35                     33  vs.  13
       13  vs.  13                     11  vs.  23
       23  vs.  23                      8  vs.  38
       33  vs.  33                      2  vs.  17
       42  vs.  42                      3  vs.  29
```

Ευχαριστώ για την προσοχή σας

- **Επικοινωνία**
  - Skype: fidas.christos
  - Email: fidas@upatras.gr
  - Phone: 2610 – 996491
  - Web: http://cfidas.info

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass