Titolo unità didattica: Numeri pseudocasuali e simulazioni stocastiche [16]

Titolo modulo : Numeri pseudocasuali in C [02-T]

Le function C per la generazione di numeri pseudocasuali ed esempi di utilizzo in simulazioni stocastiche

Argomenti trattati:

- ✓ le function C **`rand`** e **`srand`**
- ✓ generazione di numeri pseudocasuali interi in intervalli qualunque in C
- ✓ esempi di programmi in C per simulazioni Monte Carlo
- ✓ generazione di numeri pseudocasuali reali in C

Prerequisiti richiesti: AP-05-*-C, AP-16-01-T

generazione di numeri **pseudocasuali** in C

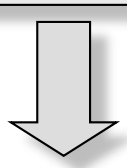
function `rand()`

```
int nc;
nc = rand();
```

genera solo **numeri interi** (tipo `int`) che appartengono all'intervallo [0,`RAND_MAX`] e che sono uniformemente distribuiti in tale intervallo

la costante `RAND_MAX` è definita in `stdlib.h`

generazione di numeri **pseudocasuali** in C



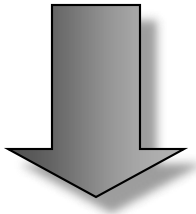
function `rand()`

```
int nc;
nc = rand();
```

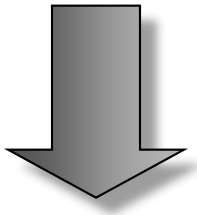
ogni chiamata alla function `rand` restituisce un numero pseudocasuale della successione pseudocasuale del generatore

`rand` è in `stdlib` → `#include <stdlib.h>`

in C il seed è fissato per default



a ogni esecuzione del programma C, viene generata la **stessa sequenza** di numeri pseudocasuali



riproducibilità delle simulazioni basate su numeri pseudocasuali (Monte Carlo)

in C il seed è fissato per default

è possibile **modificare** il valore del seed

a ogni esecuzione del programma C, viene generata una **diversa sequenza** di numeri pseudocasuali

```
srand(nuovo_seed);
```

l'attivazione della function `srand` modifica il seed del generatore (il nuovo valore del seed è l'argomento della chiamata)

```c
#include <stdio.h>
#include <stdlib.h>
void main()
{
   int i,nc,seed;
   printf(" RAND_MAX = %d\n",RAND_MAX);
   printf(" inserire un seed: ");
   scanf("%d",&seed);
   srand(seed);
   for (i=1;i<=30;i++){
/*genera 30 numeri pseudocasuali uniformemente
          distribuiti in [0,RAND_MAX] */
      nc = rand();
      printf(" %d-simo numero casuale =
                %d\n",i,nc);
      }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
   int i,nc;
   srand((unsigned int)time(0));
   for (i=1;i<=30;i++){
/*genera 30 numeri pseudocasuali uniformemente
          distribuiti in [0,RAND_MAX] */
      nc = rand();
      printf(" %d-simo numero casuale=
                %d\n",i,nc);
      }
}
```

rende **automatica** la scelta del valore del **seed**

```
srand((unsigned int)time(0));
```

`time` è una function
(prototipo in `time.h`)
che restituisce l'ora attuale, in unità
che è dipendente dal sistema

l'utilizzo di `time` ha solo lo scopo di passare a `srand` un valore (`unsigned int`) che cambia continuamente

in alternativa:

```
srand(time(NULL));
```

cambio di intervallo

M e B interi

$$n_k \in [0, M] \Rightarrow t_k \in [0, B]$$

$$t_k = \mathrm{mod}(n_k, B+1)$$

```
int nc;
nc = rand()%11;
```

generazione di un numero pseudocasuale intero in `0,..,10`

cambio di intervallo

M e B interi

$$n_k \in [0, M] \Rightarrow t_k \in [0, B]$$

$$t_k = \mathrm{mod}(n_k, B+1)$$

```
int nc,n;
n = ...;
nc = rand()%(n+1);
```

generazione di un numero pseudocasuale intero in `0,..,n`

cambio di intervallo

M e B interi

$$n_k \in [0, M] \Rightarrow t_k \in [A, B]$$

$$t_k = A + \mathrm{mod}(n_k, B + 1 - A)$$

```
int nc;
nc = 1 + rand()%10;
```

generazione di un numero pseudocasuale intero in $1,..,10$

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
  int i,nc;
  srand(time(NULL));
  for (i=1;i<=30;i++)
    {
/*genera 30 numeri casuali uniformemente
 distribuiti in 1-6 (30 lanci di un dado)*/
     nc = 1 + rand()%6;
     printf(" %d-simo numero casuale=
              %d\n",i,nc);
     }
}
```

simulazione lanci di una coppia di dadi

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
   int i,dado_1,dado_2;
   srand(time(NULL));
   for (i=1;i<=30;i++){
/* simulazione di 30 lanci di una coppia di
     dadi*/
     dado_1 = 1 + rand()%6;
     dado_2 = 1 + rand()%6;
     printf(" %d-simo lancio = %d\n",i,
                    dado_1+dado_2);
     }
}
```

generazione di 20 stringhe pseudocasuali di 5 caratteri dell'alfabeto a-m

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
   int i,j,nc,lungh_alf;
   char alfabeto[]={'a','b','c','d',
           'e','f','g','h','i','l','m'};
   char stringa5_cas[]="12345";
   lungh_alf = 11;
   srand(time(NULL));
   for (i=1;i<=20;i++){
/*genera 20 stringhe casuali di
   lunghezza 5 */
        for (j=0;j<5;j++){
```

continua….

genera a caso un indice dell'array **alfabeto**

(numero pseudocasuale in **0**,..,**lungh_alf-1**)

generazione di 20 stringhe pseudocasuali di 5 caratteri dell'alfabeto a-m

```
/*genera una 5-stringa pseudocasuale */
            nc = rand()%lungh_alf;
            stringa5_cas[j] = alfabeto[nc];
            }
    printf(" %d-sima 5stringa= %s\n"
                  ,i,stringa5_cas);
   }
}
```

generazione di 6 posizioni pseudocasuali su una scacchiera di 5x5 caselle

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
   int i,j,nx,ny;
   char scacchiera[][5]=
         {{' ',' ',' ',' ',' '},
          {' ',' ',' ',' ',' '},
          {' ',' ',' ',' ',' '},
          {' ',' ',' ',' ',' '},
          {' ',' ',' ',' ',' '}};
   srand(time(NULL));
```

continua….

```
   for (i=1;i<=6;i++)
    {
/* genera a caso una
   posizione nella
   scacchiera (5x5) */
       nx = rand()%5;
       ny = rand()%5;
       scacchiera[nx][ny] = 'x';
    }
   for (i=0;i<5;i++){
     for (j=0;j<5;j++)
        printf(" %c",scacchiera[i][j]);
     printf("\n");
    }
}
```

generazione di 6 posizioni pseudocasuali su una scacchiera di 5x5 caselle

simulazione di *dare 3 carte* a caso da un mazzetto di carte napoletane

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef enum {coppe,bastoni,spade,
     denari} Semi_Nap;
struct carta_nap{
        int valore;
        Semi_Nap seme;
        };
typedef struct carta_nap  Carta_Nap;
void main()
{
   int i,nc,lungh_mazzetto;
   srand(time(NULL));
```

continua….

```
Carta_Nap mazzetto[]={{1,coppe},
     {1,bastoni},{1,spade},{1,denari},
     {2,coppe},{2,bastoni},{2,spade},
                              {2,denari}};
   lungh_mazzetto = 8;
   for (i=1;i<=3;i++)
     {
/* dare 3 carte a caso */
      nc = rand()%lungh_mazzetto;
      printf("carta n.%d= %d
      di %d\n",i,mazzetto[nc].valore,
      mazzetto[nc].seme);
     }
}
```

simulazione di *dare 3 carte* a caso da un mazzetto di carte napoletane

attenzione: può essere necessario tenere conto del fatto che una carta sia **già stata estratta!**

```
struct carta_nap{
          int valore;
          Semi_Nap seme;
          int estratta;
          };
```

simulazione di
mischiare un mazzo
di carte francesi

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef enum {picche,fiori,quadri,cuori}
      Semi_Fr;
struct carta_Fr{
       int valore;
       Semi_Fr seme;
       };
typedef struct carta_Fr  Carta_Fr;
void mischia_mazzo(Carta_Fr *mazzo,int n);
void scambiare_carta(Carta_Fr *carta1,
       Carta_Fr *carta2);
```

continua….

simulazione di *mischiare un mazzo* di carte francesi

```
void main()
{
  int k,i,n = 52;
  Carta_Fr mazzo[52];
 char *Semi[]={"Picche","Fiori","Quadri", "Cuori"};
  srand(time(NULL));
  for (k=0;k<=3;k++)
   for (i=0;i<13;i++)
   {
     mazzo[k*13+i].valore = i+1;
     mazzo[k*13+i].seme = (Semi_Fr)k;
    }
  mischia_mazzo(mazzo,n);
  for (k=0;k<n;k++)
    printf("%d   %s\n",mazzo[k].valore,
    Semi[(int)mazzo[k].seme]);
}
```

```
void mischia_mazzo(Carta_Fr *mazzo,int n)
{ int k,p,q;
 for (k=1;k<=8*n/2;k++)
   {
    p = rand()%n;
    q = rand()%n;
    scambiare_carta(&mazzo[p],&mazzo[q]);
   }
}
```

simulazione di *mischiare un mazzo* di carte francesi

```
void scambiare_carta(Carta_Fr *carta1,
       Carta_Fr *carta2)
 { int k,p,q;
   Carta_Fr temp;
   temp = *carta1;
   *carta1 = *carta2;
   *carta2 = temp;
}
```

i numeri pseudocasuali generati da **`rand`** sono interi e appartengono all'intervallo [0,**`RAND_MAX`**]

la costante simbolica **`RAND_MAX`** dipende dal compilatore C ed è definita in **`stdlib.h`**

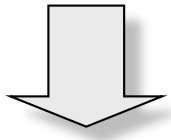
tecnica per generare numeri pseudocasuali di **tipo reale** uniformemente distribuiti nell'intervallo [0,1]

se un numero x è in $[0,b]$ allora

x / b è in [0,1]

tecnica per generare numeri pseudocasuali di
tipo reale
uniformemente distribuiti nell'intervallo [0,1]

```
nc_f =(float)rand()/(float)RAND_MAX;
```



tecnica per generare numeri pseudocasuali di
tipo reale
uniformemente distribuiti nell'intervallo [a,b]

```
nc_f = a+(b-a)*
       (float)rand()/(float)RAND_MAX;
```