Funzioni

```
double x;
int N;
double dati[1000];
int matricola[162];
char nome[] = "Albert";
char c = 'A';
int * p = &N;

double * px = dati;
```

Funzione: insieme di istruzioni per implementare un algoritmo

```
printf("Ciao!\n");
```

"Ciao!\n" → input

```
scanf("%lf", &x);
```

due argomenti

"%lf" → stringa

&x → puntatore a double

```
scanf("%d", &N);
```

"%d" → stringa

&N → puntatore a int

```
int m = rand48();
```

nessun argomento

valore di ritorno di tipo int

```
x = sqrt(2.4);
```

E.g.: valore di input (argomento)

x: uscita a valore di ritorno della funzione;

```
x = sqrt();
```

```
double y = pow(x, 3);        x^3
```

due argomenti

```
y = pow(3, x);        3^x
```

ordine dell'argomenti è importante

```
y = pow(sqrt(x), x+3);
```

$(\sqrt{x})^{x+3}$

```
y = sin(x);
```

un solo argomento.

In C

Funzione restituisce/ritorna un solo valore

```
tipo nome_funzione(argomenti)
```

```
i = printf("ciao\n");   Stampa sullo schermo
sqrt(3.1);
```

```
int main() {
    printf("ciao\n");
    sqrt(3.1);
    printf("%.3f\n", sqrt(4));
}
```

Valore di ritorno di sqrt(3.1) non usato

double x = sqrt(3.1);

sqrt(4) = x;

2 = x;

2 = √3.1

Non potete assegnare un valore alla funzione.

Implementazione di funzione

argomenti separati da ,

double
int
float
char
void

tipo nome (..., ..., ..., ...) {

{ ≡ } corpo della funzione

return valore;

}

int randG8() {

int valore;

≡ algoritmo di generazione

return valore;

}

int dado() { → main()

}

```
int dado( int nfacce ) {
    int ris;
    ris = rand48() % nfacce + 1;
    return ris;
}

int main() {
    srand48( time(0) );
    int lancio;
    lancio = dado(48);
    return 0;
}
```

funzione principale

gioco.c

```
gcc -o app.exe gioco.c
./app.exe  ⟶  chiama main()
```

```
main
  ↳ time
  ↳ srand48
  ↳ dado
```

(X) = ./app.exe nella shell x = valore di r. torno di main()

genereare interi tra [m, n]

```
rand48() % (n - m) + 1
```

```
int generaInt(int a, int b) {
    int r;
    r = lrand48()%(b-a+1) + a;
    return r;
}
```

[0, b-a] → [a, b-a]

a, b, r: variabili locali nella funzione generaInt

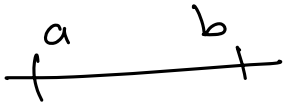
```
int main() {
    int i,j;
    i = dado(48);
    j = generaInt(-2, 48);
    printf("a = %d\n", a);
}
```

j = generaInt(-2, 48): argom. in input.

i: valore di output

variabile a non esiste all'interno di main()

```
double z = uniforme(0.3, 0.9);
```




```
double uniforme(double a, double b) {
    double r;
    r = ((double)lrand48()/RAND_MAX) * (b-a)
        + a;
    r = drand48() * (b-a) + a;
    return r;
}
```

xlow xhi

```
int main() {
    srand48( time(0) );
    double y = 1.2;
    for( int i = 0; i < 10; i++ ) {
        double x = uniforme( 1.1, 3.9 );
        y = sqrt( x );
        double z = 2.1;
    }
    printf(" i = %d ", i ); X non compila
    printf(" y = %lf \n", y ); ✓ √x9
    printf(" z = %lf \n", z ); X non compila
}
```

i variabile locale in questo {}

Visibilità delle variabili (scope)