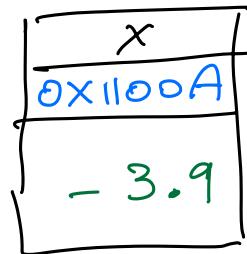


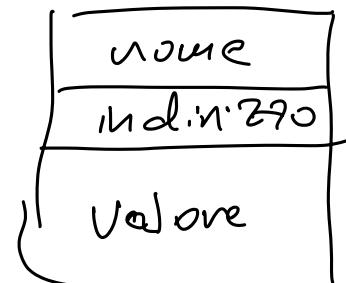
double $x = -3.9;$

printf("indirizzo di x: %p\n", &x);



double dati[3] = {-2, 23, 59};

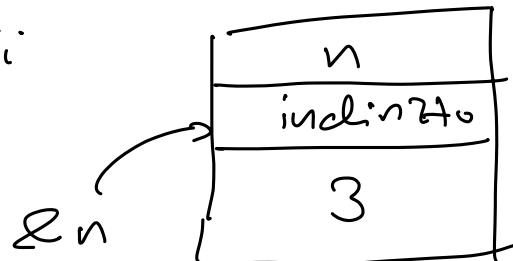
dati[0]	dati[1]	dati[2]
- - -	- - -	- - -
-2	23	59



printf("indirizzo di dati[1] = %p\n", &dati[1]);

3: lunghezza di array dati:

dati[i] : i = 0, 1, 2



int n = 3;

printf("indirizzo di n = %p\n", &n);

double * pd; pd è puntatore a variabile di tipo double

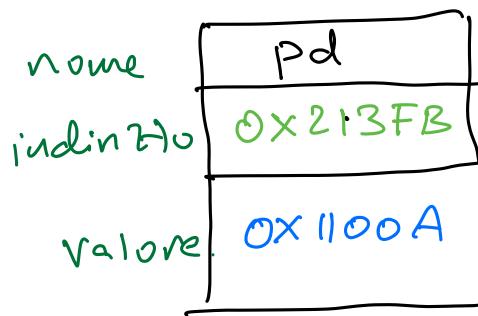
pd = &x;

printf("pd = %p\n", pd);

0X1100A

printf("x = %f\n", x);

-3.9



Stampa valore di variabili:

printf("&pd = %p\n", &pd);

0X213FB

printf("&x = %p\n", &x);

0X1100A

Dereferenziazione pd (Puntatore) : accesso a localazione di memoria alla quale punta il puntatore pd

*pd → &x.

*pd → valore di x.

printf ("*pd = %f\n", *pd);

= 3.9

printf ("*x = " met l'osto

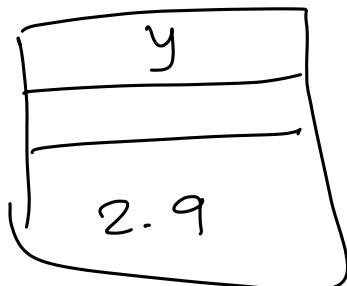
*: moltiplici usi

x * y : prodotto

operazione binario

* puntatore : dereferenzia operazione unario.

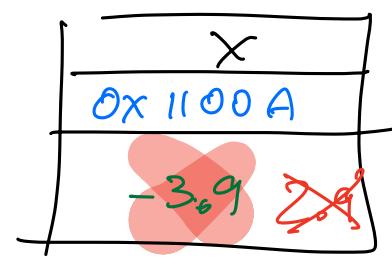
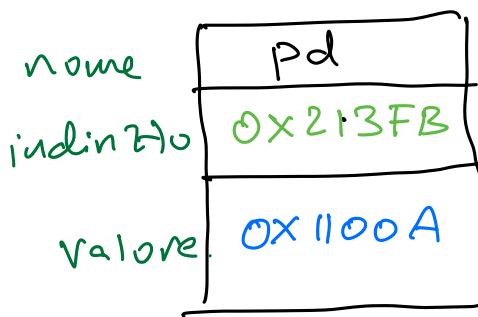
x - y binario
- 3 unario



double y = 2.9;

x = y;

y ~~= x;~~



*pd = y/2;
1.45

equivale a : x = y/2;

pd = &dati[2];

*pd = 1; equivale a dati[2] = 1

dati puntatore a dati[0]

printf("dati = %p\n", dati);

printf("&dati[0] = %p\n", &dati[0]);

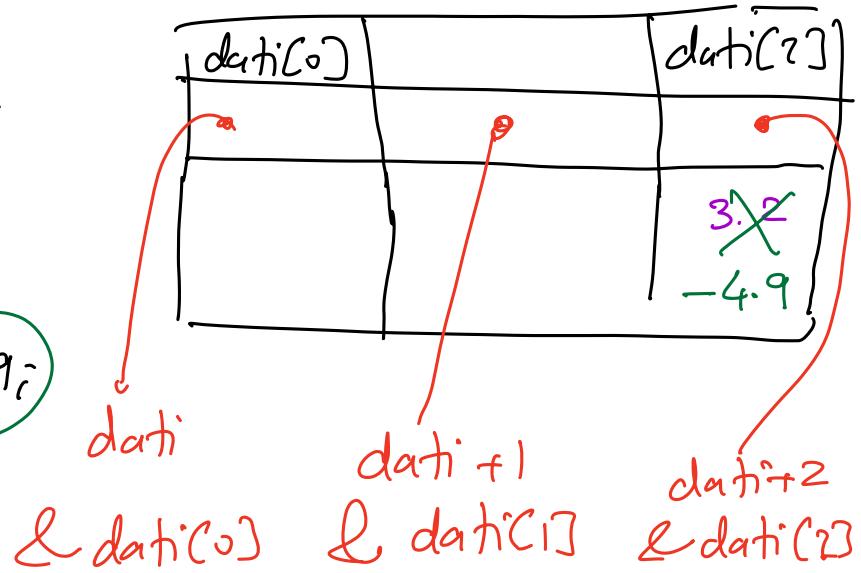
dati[0] = -1.1;

*dati = -2.2;

(dati[2] = 3.2) ←

(*dati + 2) = -4.9;

aritmetica
dei puntatori



(&pd)

double * p1 = &x;

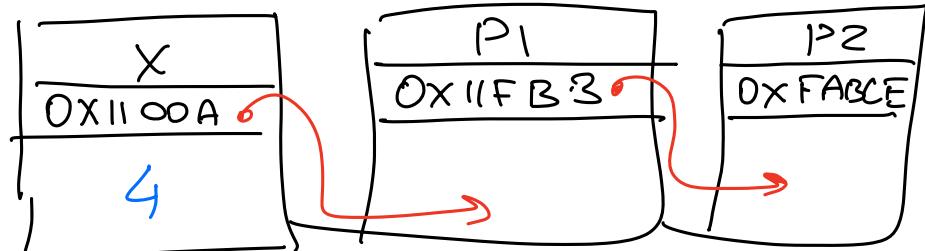
double ** p2 = &p1;

(*&pd)

(*p2)

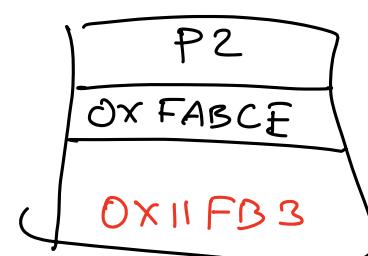
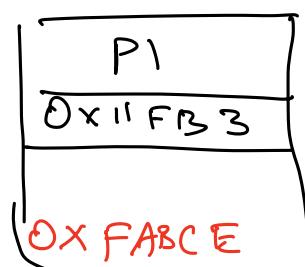
*x

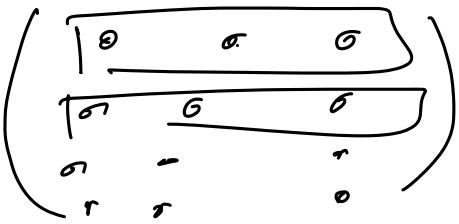
(*(&p2)) = 4



*p2 equivale a P1 OSS: &x

P1 = &P2.





double mat[4][3];

- 0 0 , 0 0 0 , 0 1 0 , - - -

for (int i=0; i<4; i++) {

 for (int j=0; j<3; j++) {

 mat[i][j] = 0;

*(&mat + i) + j) = 0;

}

}

int n;

n++;

double mat[4][3];

Diagonal mat come
array 2D

altera(mat, 4, 3);

~~altera(mat[6][3], 4, 3);~~

mat[2][1]

$$\begin{pmatrix} \cdot & 0 & 0 \\ 0 & \cdot & \vdots \\ 0 & 0 & \vdots \\ 0 & 0 & \vdots \end{pmatrix}$$

```

// dichiarazione
void affera( double S[NMAX][NMAX], int, int );
#define NMAX 10

int main() {
    double mat[4][3];
    affera( mat, 4, 3 ); // chiamare funzione
    double A[3][2];
    affera( A, 3, 2 );
}

```

// implementazione

```

void affera( double m[NMAX][NMAX], int nr, int nc ) {
    if( nr >= NMAX || nc >= NMAX ) {
        printf("errore dimensione array\n");
    }
    for( int i = 0; i < nr, i++ ) {
        for( int j = 0; j < nc; j++ ) {
            m[i][j] = 0;
        }
    }
}

```

}

// dichiarazione

```

double media( double *, int );

```

```

double media( double S[NMAX], int ); } in alternativa

```

```

int main() {

```

```

    double dati[123] = { 0 };

```

```

    double m = media( dati, 123 );

```

```

    m = media( &dati[0], 123 );

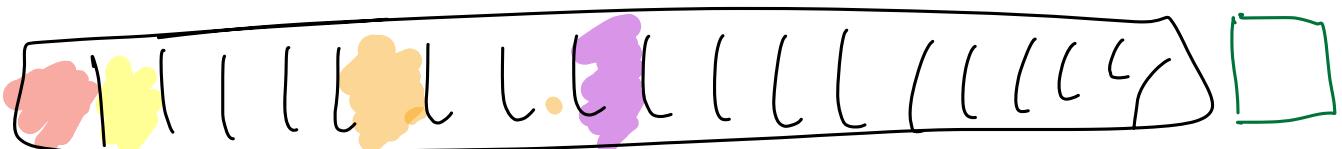
```

```

}
double media( double *d , int n ) {
    double s=0;
    for (int i=0; i<n; i++) {
        s += d[i]; // s += *(d+i);
    }
    return s/n;
}

```

double dati[123]



$m = \text{media}(\text{dati}, 123);$

$m = \text{media}(\&\text{dati}[0], 123);$

$m = \text{media}(\&\text{dati}[1], 123);$ comincia da $\text{dati}[1]$

$m = \text{media}(\&\text{dati}[29], 123);$ ignora $0 \rightarrow 28$ celle

$m = \text{media}(\&\text{dati}[37], 123);$

Dichiarazione

```
19 double uniforme(double,double);
20 double getDouble(char*,double,double);
21 int getInt(char*,int, int);
22
23 // genera un punto nell'ellisse
24 void ellisse(double,double,double*, double*);
25
26 // genera posizione atomi
27 void atomi(int, double[NMAX] [2], double, double);
28
29 // verifica interazione di un proiettile con gli atomi
30 int ...interazione(double*, double, double[NMAX] [2], int);
31
32 // calcola distanza tra due punti
33 double ...distanza(double*, double*);
```



```
36 int main() {
37
38     FILE* fp;
39
40     double bersaglio[NMAX][2]; /* posizione dei bersagli. solo array statici in C */
41     double xp[2]; // posizione di un punto generico
42
43     int N, MA;
44     double R, A, B;
45
46     int nchar;
47     char c, fname[100];
48     int i, nhit, nhitTot, nhitMax;
49
50     /* 1 pt: inizializza */
51     srand48(time(0));
52
53     /* 2 pt: condizioni corrette per i dati dall'utente */
54
55     MA = getInt("numero di atomi M", 1, NMAX);
56     N = getInt("numero particelle incidenti N", 1, MA);
57
58     A = getDouble("semiasse maggiore A: ", 0, LMAX);
59     B = getDouble("semiasse minore B: ", 0, A);
60     R = getDouble("raggio interazione: ", 0, RMAX);
```



```
62 /* genera atomi di bersaglio */
63 atomi(MA, bersaglio, A, B);
64
65 /* 2 pt: nome del file come variabile */
66 printf("inserire nome di file (max 100 caratteri) per scrivere posizione atomi: ");
67 scanf("%s", fname);
68
69 /* 1 pt: apertura file */
70 fp = fopen(fname,"w+");
71 if(!fp) {
72     printf("file %s non valido. abort.\n", fname);
73     exit(-1);
74 }
75
76 /* 2 pt: scrittura su file */
77 for(i=0; i<MA; i++) {
78     /* 1 pt: scrittura su file */
79     fprintf(fp, "%f %f\n", bersaglio[i][0], bersaglio[i][1]);
80 }
81
82 /* 1 pt: chiusura file */
83 fclose(fp);
```

MA <= NMAX



MA atom:
(x_i, y_i)

```

85  /* 1 pt: ciclo su bersagli */
86  nhitTot = 0;
87  nhitMax = 0;
88  for(i=0; i<N; i++) {
89      double xp[2];
90      double xi,yi;
91      /* 2 pt: genera proiettile e controlla interazione */
92      ellisse(A,B, xp, xp+1);
93      nhit = interazione(xp, R, bersaglio, MA);
94
95      nhitTot += nhit;
96      if(nhit>nhitMax) nhitMax = nhit;
97
98 }
99
100 /* 3 pt: calcolo numero medio e max di interazioni per proiettile con giusta inizializzazione */
101 printf("Numero medio di interazioni per proiettile: %.1f\n", (double)nhitTot/N);
102 printf("Numero max di interazione per un proiettile: %d\n", nhitMax);
103
104 /* fine del programma */
105 return 0;
106 }
```

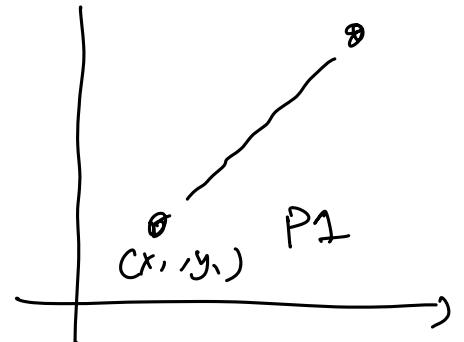
$\&xp[0], \&xp[1]$
 $\&x, \&y$

```

109 double getDouble(char* msg, double min, double max) {
110     double x;
111     do{
112         printf("inserisci %s compreso tra (%.2f,%.2f): ", msg, min, max);
113         scanf("%lf", &x);
114     } while(x<=min || x>=max);
115     return x;
116 }
117
118
119 /* 1 pt: prendi intero */
120 int getInt(char* msg, int min, int max) {
121     int x;
122     do{
123         printf("inserisci %s compreso tra [%d,%d]: ", msg, min, max);
124         scanf("%d", &x);
125     } while(x<min || x>max);
126     return x;
127 }
128
129 /* 2 pt: generazione numeri casuali */
130 double uniforme(double a, double b) {
131     return a + (b-a)*lrand48()/RAND_MAX;
132 }
133
134 double distanza(double p1[2], double p2[2]) {
135     return sqrt( (p1[0]-p2[0])*(p1[0]-p2[0]) + (p1[1]-p2[1])*(p1[1]-p2[1]) );
136 }
```

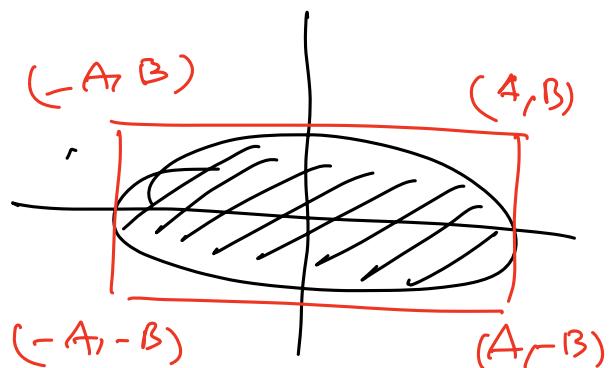
P2

(x2,y2)



```

139 /* 1 pt: interfaccia corretta */
140 // A, B: semiassi dell'ellisse
141 // x,y le coordinate da generare
142 void ellisse(double A,double B, double* x, double* y) {
143     /* 2 pt: generazione punto nell'ellisse */
144     double dist;
145     do{
146         *x = uniforme(-A, A);
147         *y = uniforme(-B, B);
148         dist = (*x/A)*(*x/A) + (*y/B)*(*y/B);
149     } while( dist > 1 );
150
151 }
```



```

154 /* 2 pt: interfaccia funzione */
155 // M: numero di atomi
156 // bers: array 2D per immagazzinare le posizioni di atomi
157 // A,B: semiassi dell'ellisse dove generare gli atomi
158 void atomi(int M, double bers[NMAX][2], double A, double B) {
159     int i;
160
161     /* 2 pt: generazione atomi e salvare nell'array */
162     for(i=0; i<M; i++) {
163
164         ellisse(A, B, *(bers+i), *(bers+i)+1);
165
166         /* modo alternativo per passare elementi dell'array */
167         /* ellisse(A, B, &bers[i][0], &bers[i][1]); */
168     }
169 }

```



```

171 // xp: posizione del proiettile come array di lunghezza 2
172 // R: raggio di interazione
173 // bers: array 2D con le posizioni di atomi
174 // M: numero di atomi
175 // la funzione restituisce il numero di interazioni per questa particella
176 int interazione(double xp[2], double R, double bers[NMAX][2], int M) {
177
178     int i, n=0;
179     double p;
180
181     /* 3 pt: condizione di interazione */
182     for(i=0; i<M; i++) {
183         if( distanza(xp, *(bers+i)) < R && uniforme(0,1) < XSEC) n++;
184     }
185     /* printf(" numero interazioni: %d\n", n); */
186
187     return n;
188 }

```

Ciclo su atomi:
 x, y ;
 $\text{ellisse}(A, B, \&x, \&y)$;
 $\text{bers}[i][0] = x$.
 $\text{bers}[i][1] = y$.