Laboratorio 16/05/03

Uso di funzioni

strutture

vettori

Es. 1

Data la seguente definizione di tipo:

```
typedef struct cont
{int g; int h;}ELE;
```

scrivere una funzione C che, dato un parametro `rec` di tipo `ELE`, restituisce attraverso il valore di ritorno il seguente risultato: 1 se la somma dei divisori di rec.g (escluso rec.g) è minore di rec.h, altrimenti 0.

Es. 1

```
#include <stdio.h>
typedef struct cont {int g;int h;} ELE;
int minore (ELE rec);
ELE carica();
int main(void)
   { ELE e;
     e = carica();
     printf("%d\n",minore(e));
     return(0);}
```

Es. 1

```
ELE carica(void)
{
  ELE x;
  printf("dammi un intero: ");
  scanf("%d",&x.g);
  printf("dammi un intero: ");
  scanf("%d",&x.h);
   return(x);
}
```

Es. 1

```
int minore (ELE rec)
   { int sommadiv, i;
     sommadiv=0;
     for (i=1;  i <= rec.g / 2; i=i+1)
       if ((rec.g % i) == 0)
        sommadiv=sommadiv+i;
     if (sommadiv < rec.h)
      return 1;
     else
      return 0;
   }
```

Es. 2

Scrivere il programma che gestisce un vettore di N elementi di tipo:

```
typedef struct {int a, b; double c;} EL;
typedef EL VTEL[N];
```

Il programma deve:

- caricare i campi a e b di una variabile globale Vt di tipo VTEL
- caricare nel campo c il quoziente del campo a con il campo b
- stampare il vettore

Es. 2: prototipi

```
#include <stdio.h>
#define N 4
typedef struct {int a,b;double c;} EL;
typedef EL VTEL[N];
VTEL Vt;
void carica();
void riempic();
void stampa ();
```

Es. 2: `main()`

```
int main()
{
  carica();
  riempic();
  stampa();
  return(0);
}
```

Es. 2: `carica()`

```
void carica()
   {int i;
    for(i=0;i<N;i++)
    {printf("dammi due interi: ");
    scanf("%d%d",&Vt[i].a,&Vt[i].b);}
   }
```

Es. 2: `riempic()`

```
void riempic()
  {int j;
  for(j=0; j<N; j++)
    Vt[j].c = Vt[j].a/(double)Vt[j].b;
  }
```

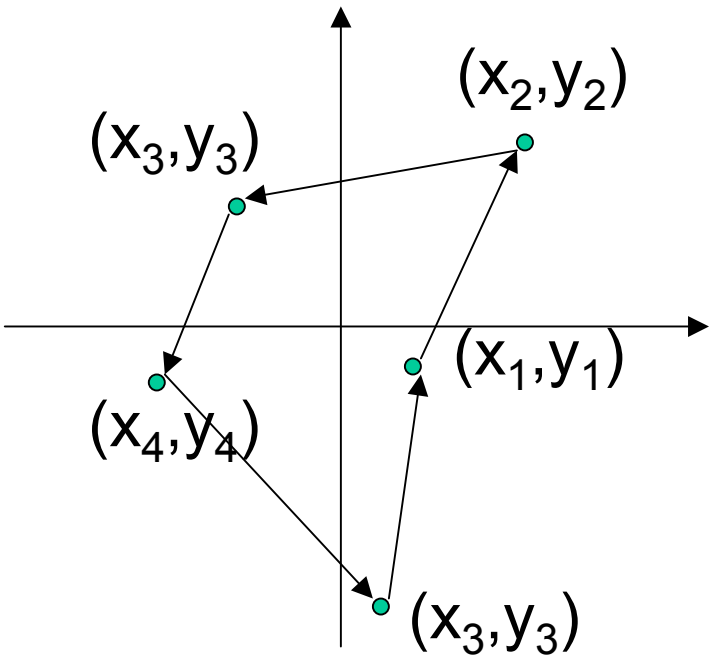
Es. 2: `stampa()`

```
void stampa ()
{ int j;
  for(j=0; j<N; j++)
    printf("%d  %d
  %f\n",Vt[j].a,Vt[j].b,Vt[j].c);
}
```

Es. 3

- Scrivere il programma che gestisce un insieme di N punti (N>2) in uno spazio 2-D, vertici di una figura chiusa:
  - carica l'insieme dei punti
  - verifica se sono tutti diversi
  - calcola il perimetro della figura (si supponga che i lati congiungano i vertici nell'ordine dato)

Es. 3



- distanza fra due punti nello spazio 2-D:
- $\sqrt{((x_i-x_j)^2+(y_i-y_j)^2)}$

Es. 3: struttura dati

```
typedef struct {double x,y;}
  PUNTO;
typedef PUNTO VTPt[N];
```

Variabile globale:

```
VTPt v;
```

Es. 3: `main()`

```
int main()
{
   carica();
  printf("%d\n",VerDupl(x));
  printf("peri:%lf\n",Perimetro(x));
  return(0);
}
```

Es. 3: carica()

```
void carica()
   {int i;

    for(i=0;i<N;i++)
    {printf("punto %d: ",i);
    scanf("%lf%lf",&v[i].x,&v[i].y);
    }
   }
```

Es. 3: VerDupl()

```
int VerDupl()
   {int i,j;
     for(j=0; j<N-1; j++)
     for(i=j+1;i<N;i++)
      if (v[j].x == v[i].x)
        if (v[j].y == v[i].y)
          return 0;
    return 1;
   }
```

Es. 3: Perimetro()

```
double Perimetro ()
{ int i;
  double per=0.0;
  for(i=0; i<N-1; i++)
   per=per+sqrt(pow(v[i].x-v[i+1].x,2)\
+pow(v[i].y-v[i+1].y,2));
  per = per + sqrt(pow(v[N-1].x,\
v[0].x,2)+pow(v[N-1].y-v[0].y,2));
  return (per);}
```

Es. 4

Una funzione ricorsiva è definita nel seguente modo:

$$f(x,y) \begin{cases} 1 & x \leq 0 \\ f(x-2,y-1)+x*y & \text{altrimenti} \end{cases}$$

Scrivere un programma che permetta di richiamarla finché l'utente non decide di smettere

Es. 4

```
#include <stdio.h>
#include <ctype.h>
int funz(int x, int y);
int prosegui();
void main()
{int a, b;
  do { printf("dammi 2 interi: ");
       scanf("%d%d",&a,&b);
       printf("ris.: %d\n",funz(a,b));
     }while(prosegui() == 's');}
```

Es. 4

```
int prosegui()
  {char c;
   do{ printf("vuoi continuare? ");
       scanf("%1s",&c);
       c = tolower(c);
     }while((c!='s') && (c!='n'));
   return(c);
 }
```

Es. 4

```
int funz(int x, int y)
{int ret;
  if (x <= 0)
     ret =1;
  else
     ret = funz(x-2,y-1) + x * y;
  return(ret);
}
```

Es. 5

- Scrivere la funzione per il calcolo ricorsivo del coefficiente binomiale e confrontare il valore ottenuto mediante il calcolo con il metodo iterativo.

$$\binom{n}{k} = \frac{(k+1)\times\cdots\times n}{(n-k)!} = \begin{cases} 1 & k = n, k = 0 \\ n & k = 1 \\ \binom{n-1}{k} + \binom{n-1}{k-1} & altrimenti \end{cases}$$

Es. 5

```
#include <stdio.h>
int prosegui();
long BinRic(int n, int k);
long BinIt(int n, int k);
void main()
{int a, b;
  do { printf("dammi 2 interi: ");
       scanf("%d%d",&a,&b);
      printf("%ld = “,BinRic(a,b));
      printf(“%ld\n",BinIt(a,b));
     }while(prosegui() == 's');}
```

Es. 5

```
int prosegui()
   {char c;
    do{ printf("vuoi continuare? ");
        scanf("%1s",&c);
      c = tolower(c);
      }while((c != 's')&&(c != 'n'));
    return(c);
  }
```

Es. 5

```
long BinRic(int n, int k)
{long ret;
  if ((k == 0) || (k == n))
     ret =1;
  else
    if (k == 1)
     ret = n;
    else
     ret = BinRic(n-1,k)+BinRic(n-1,k-1);

  return(ret);
}
```

Es. 5

```
long BinIt(int n, int k)
   {  int i;
    long num=1, den=1;
    for(i=k+1; i<=n; i++)
      num = num * i;
    for(i=2; i<=n-k; i++)
      den = den * i;
     return(num/den);
   }
```