Esercizi Aggiuntivi su Array e Stringhe

Credits Prof. Campi

Esercizio 1

- Scrivere un programma in C che legge un vettore di interi di dimensione fissata e ne stampa la somma

```c
#include <stdio.h>
#define LUNG 5
int main() {
    int vett[LUNG],i,sum=0;
    printf("Inserire un vettore di interi di dimensione %d\n",
       LUNG);
    for (i = 0; i < LUNG; i++)
       scanf("%d", &vett[i]);
    /* Somma gli elementi del vettore */
    for (i = 0; i < LUNG; i++) {
         sum=sum+vett[i];
    }
    /* Stampa la somma */
    printf("Somma: %d\n", sum);

    return 0;
}
```

Esercizio 2

- Scrivere un programma in C che legge un vettore di interi di dimensione fissata, inverte il vettore e lo stampa.

```c
#include <stdio.h>
#define LUNG 5
int main() {
    int vett[LUNG],i,temp; /*usata per scambiare due elementi del vettore */
    printf("Inserire un vettore di interi di dimensione %d\n", LUNG);
    for (i = 0; i < LUNG; i++)
        scanf("%d", &vett[i]);
    /* Inverti il vettore senza l'utilizzo di un vettore ausiliario */
    for (i = 0; i < LUNG/2; i++) {
        temp = vett[i];
        vett[i] = vett[LUNG-1-i];
        vett[LUNG-1-i] = temp;
    }
    /* Stampa il vettore, che ora e' invertito */
    for (i = 0; i < LUNG; i++)
        printf("%d %d\n", i, vett[i]);
    return 0;
}
```

Esercizio 3

- Scrivere un programma C che legge un vettore di lunghezza arbitraria e stampa **1** se il vettore contiene solo valori diversi, **0** altrimenti.

```
int main() {
  int v[n], i, j, si=1;
 /* ometto codice lettura vettore */
  for ( i = 0; i < n-1; i++ )
    for ( j = i+1; j < n; j++ )
      if ( v[i] == v[j] )
        si=0;
  printf("%d\n",si);
  return 0;
}
```

Esercizio 4

- **Scrivere un programma C che:**
  - **Legge una sequenza di numeri interi e quei numeri compresi tra 0 e 1023 vengono memorizzati in un vettore di nome V. La lettura termina quando nel vettore sono stati inseriti 10 numeri**
  - **Per ogni numero in V il programma esegue la conversione in binario, memorizza i resti ottenuti in un vettore R opportunamente dimensionato e stampa il contenuto di R**

```
#define N 10
#define MAXVAL 1023

#include<stdio.h>

int main() {
  int i=0,j,V[N],R[N];
  do {  scanf("%d",&V[i]);
          if (V[i]>=0 && V[i]<=MAXVAL)
                i++;
  } while(i<N);
  for(i=0;i<N;i++){
    for(j=N-1;j>=0;j--){
      R[j]=V[i]%2;
      V[i]=V[i]/2;
    }
    for(j=0;j<N;j++)
       printf("%d ",R[j]);
    printf("\n");
  }
  return 0;
}
```

Esercizio 5

- Si scriva un programma C che all'interno del main consenta di inizializzare da tastiera un vettore di interi di lunghezza massima pari a 20.
- La lunghezza effettiva della sequenza acquisita è stabilita dall'utente (ad esempio, acquisendo da tastiera il valore di una variabile n).
- Il programma dovrà stampare il vettore, scorrere il vettore e stampare gli elementi del vettore che hanno un numero pari nella posizione immediatamente precedente alla propria.
- Per esempio, se l'utente sceglie di inserire 10 valori, avendo letto in input e memorizzato nell'array i seguenti interi:

  1 2 3 4 5 6 7 8 9 0

  Il programma produrrà il seguente output:

  1 2 3 4 5 6 7 8 9 0

  3 5 7 9

```
# include <stdio.h>
# define MAXDIM 20

int main() {
	int vet[MAXDIM], i=0, valore;

	do { /* leggo la lunghezza che deve essere accettabile */
		printf("\nInserisci un numero intero \n");
		scanf("%d",&valore);
	} while(valore<0||valore>MAXDIM);
	printf("\n Inserisci una sequenza di %d interi:\n",valore);
	for(i=0; i<valore; i++) {
		scanf("%d",&vet[i]);
	}
	printf("\n La sequenza di %d interi inserita e':\n",valore);
	for(i=0; i<valore; i++) {
		printf("%d ",vet[i]);
	}
	printf(“Stampo elementi con un pari nella posizione precedente");
	for(i=1; i<valore; i++) /* Il primo elemento non ha precedente*/
		if (vet[i-1]%2==0)
			printf("%d ",vet[i]);
	return 0;
}
```

Esercizio 6

- Scrivere un programma C che legge due stringhe da tastiera, le concatena in un’unica stringa e stampa la stringa così generata

```
#include <stdio.h>
#define LUNG 200
int main () {
        char str1[LUNG],str2[LUNG],strTot[2*LUNG];
        int i,j;
        printf("Inserisci la prima stringa:\n");
        scanf("%s",str1);
        printf("Inserisci la seconda stringa:\n");
        scanf("%s",str2);
        for (i=0;str1[i]!='\0';i++)
                strTot[i]=str1[i];
        /* i rappresenta il numero di caratteri copiati da str1 a strTot */
        for (j=0;str2[j]!='\0';j++) /* accodo str2 a str1 */
                strTot[i+j]=str2[j];
        strTot[i+j]='\0';
        printf("\n%s",strTot);
        return 0;
}
```

Esercizio 7

- Scrivere un programma che acquisisce una sequenza di caratteri terminata dal carattere 'invio' e stabilisce se la sequenza è palindroma oppure no (per esempio, "ada" è palindroma perché si legge allo stesso modo sia da destra sia da sinistra)

```
#include<stdio.h>
#include<string.h>
#define MAX_DIM 100
int main() {
        char stringa[MAX_DIM];
        int contatore=0, i, quanti=0 ;
        scanf("%s“,stringa);
        contatore=strlen(stringa)-1;
        for(i=0;i<contatore/2;i++)
                if(stringa[i]==stringa[contatore-i])
                        quanti++;
        if(quanti==contatore/2)
                printf(“Palindromo”);
        else
                printf(“Non Palindromo”);
        return 0;
}
```

```
#include<stdio.h>
#define MAX_DIM 100
int main() {
	char stringa[MAX_DIM];
	int contatore=0, i;
	scanf("%s",stringa);
	for (i=0;stringa[i]!='\0';i++)
		contatore++;
	contatore--;
	i=0;
	while(stringa[i]==stringa[contatore-i] && i<contatore-i)
		i++;
	if(i>=contatore-i)
		printf("Stringa palindroma");
	else
		printf("Stringa non palindroma");
	return 0;
}
```

Esercizi 8

- Scrivere un programma che chieda di inserire una sequenza di caratteri, fino allo spazio, riconosca se i caratteri inseriti sono cifre pari o dispari o se sono altro, visualizzi le tre sequenze divise per tipo.
- *Memorizzo le pari in un array, le dispari in un secondo array e tutto il resto in un terzo, controllando le dimensioni*

```
#include <stdio.h>
#define lung 10
int main() {
	int p=0,d=0,a=0,i=0;
	char cosa,pari[lung],dispari[lung],altro[lung];
	do {
		scanf("%c",&cosa);
		fflush(stdin);
		switch(cosa) {
		case '0': case '2': case '4': case '6': case '8': pari[p]=cosa; ++p; break;
		case '1': case '3': case '5': case '7': case '9': dispari[d]=cosa; ++d; break;
		default: if(cosa!=' ' ) {altro[a]=cosa;++a;} break;
		}
	} while (cosa !=' '&& p<lung && d<lung && a<lung);
	printf("\nle cifre pari sono: ");
	for (i=0;i<p;i++) printf(" %c ",pari[i]);
	printf("\nle cifre dispari sono: ");
	for (i=0;i<d;i++) printf(" %c ",dispari[i]);
	printf("\ni caratteri non cifre sono: ");
	for (i=0;i<a;i++) printf(" %c ",altro[i]);
	return 0;
}
```

Esercizio 9

- Scrivere un programma in linguaggio C che legge un polinomio di grado n a coefficienti reali e lo valuta per un dato valore x.

Il primo modo che sicuramente si può individuare è quello di sostituire ad x un certo valore, valutare le diverse potenze di x e moltiplicare per il corrispettivo coefficiente accumulando man mano il risultato in una variabile.
Si suppone di leggere i coefficienti del polinomio da quello di grado massimo a quello di grado nullo nelle celle di un vettore a partire dalla prima (cella 0).

es: n = 3, $f(x) = 3x^3 + 5x^2 + 2x + 1$ lo rappresento come V = < 3, 5, 2, 1>

Iniziando a leggere il polinomio dal termine noto, quindi dall'ultimo elemento del vettore (avrà n+1 elementi) cioè dalla posizione n-esima, si ha:

$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = (a_n x^n + (a_{n-1}x^{n-1} + \ldots + (a_1 x + (a_0))\ldots))$

es: $f(x) = 3x^3 + 5x^2 + 2x + 1 = ((((1) + 2x) + 5x^2) + 3x^3)$

Al variare di un indice tra 0 e il grado del polinomio, si utilizzeranno due variabili accumulatore: Una, per costruire man mano il valore della potenza di x opportuna, l'altra usata per sommare man mano i valori parziali della f

legge n, il vettore V e x dallo standard input

f = 0; pot = 1; i = n;

finché (i >= 0)

- f = V[i]*pot + f;
- pot = pot * x;
- i = i - 1;

```
#include <stdio.h>
#define MAX_LEN 100
int main( ) {
    float V[MAX_LEN],pot=1.0,f=0.0,x;
    int i=0,n;
    do {
       printf("\ninserire il grado del polinomio, n = "); scanf("%d",&n);
    } while ((n <= 0)||(n >= MAX_LEN));
    printf("\ninserire i coeff. del polinomio: \n");
    do { scanf("%f",&V[i]); i++; } while(i<=n);
    printf("\nInserire il valore di x:"); scanf("%f",&x);
    for (i = n; i >=0; i--) {
       f=f+V[i]*pot;
       pot=pot*x;
    }
    printf("\nf(%f) = %f\n ",x,f);
    return 0;
}
```

Esercizio 10

- Scrivere un programma C che legge da tastiera una sequenza di 100 numeri interi.
- Dopo avere letto tutti i numeri cercare le coppie di numeri tali che il primo sia il doppio dell'altro.

- Esempio

  1 5 6 10 3 4 7 0 2

  stampa

  6 3 / 10 5 / 4 2 / 2 1

```
#include <stdio.h>
#define MAX_NUM 100
int main() {
  int dati[MAX_NUM];
  int i=0, j=0;
  do { scanf("%d",&dati[i]); i++; } while(i<MAX_NUM);

  for (i=0; i<MAX_NUM; i++)
     for(j=0;j<MAX_NUM;j++) {
        if (dati[i]==2*dati[j] && i!=j) {
             printf("\ncoppia: %d e %d",dati[i],dati[j]);
        }
     }
  return 0;
}
```

Esercizio 11

- Scrivere un programma C che legge da tastiera una sequenza di numeri reali ***diversi da zero***
- La lettura termina quando la somma dei numeri immessi è maggiore di 50, e comunque non si possono immettere più di 100 numeri.
- Dopo avere letto tutti i numeri, se l'utente ha inserito almeno 3 valori, cercare se esiste una coppia di numeri tali che il loro rapporto (o il suo inverso) sia uguale al primo numero immesso e, se esiste, stamparla.

Esempio di funzionamento

Inserisci numero: 6.25
Inserisci numero: -2.5
Inserisci numero: 20
Inserisci numero: 13.863
Inserisci numero: -15.625
Inserisci numero: 4
Inserisci numero: 38.192

Il rapporto (o il suo inverso) tra -2.5 e -15.625 vale 6.25.

```
#include <stdio.h>
#define MAX_NUM 100
#define MAX_SUM 50
#define TOL 0.00000001
int main() {
  float dati[MAX_NUM], sum = 0, rapp, inv_rapp;
  int i, j, n_dati=0;
  int trovata = 0;
  do {
     do {
          printf("Inserisci numero: "); scanf("%f", &dati[n_dati]);
     } while (dati[n_dati] < TOL && dati[n_dati] > -TOL)
     sum = sum + dati[n_dati];
     n_dati++;
  } while (sum <= MAX_SUM && n_dati < MAX_NUM);
  if (n_dati < 3) {
     printf("Sono stati inseriti solo %d elementi\n", n_dati);
   } else {
    …
   }
} // fine del main
```

Dentro l'else

```
i = 1;
while (!trovata && i<n_dati-1) {
  j = i+1;
  while (!trovata && j<n_dati) {
    rapp = dati[i]/dati[j];
    inv_rapp = dati[j]/dati[i];
    if (rapp - dati[0] < TOL && rapp - dati[0] > -TOL ||
                  inv_rapp - dati[0] < TOL && inv_rapp - dati[0] > -TOL) {
      trovata = 1;
      printf("Il rapporto (o il suo inverso) tra %f e %f e' %f\n",dati[i],dati[j],dati[0]);
    }
    j++;
  }
  i++;
}
```

Esercizio 12

- Scrivere un programma C che calcola il parallelo di n resistenze, con n specificato dall'utente
- Si faccia l'ipotesi l'utente non commetta errori nell'inserire i dati.

- Si ricorda che in caso di resistenze in parallelo

  $1/R_{tot}=1/R_1+1/R_2+\ldots+1/R_n$

```c
#include <stdio.h>
#define MAX 100
main() {
    int n,i;
    float resistenze[MAX],RP = 0;
    printf("\nQuante resistenze vuoi combinare (max %d)? ", MAX);
    scanf("%d", &n);
    for (i=0; i < n; i++) {
        printf("Inserire R%d (non 0)", i+1); scanf("%f", &resistenze[i]);
     }
    for (i=0; i < n; i++)
        resistenze[i] = 1/resistenze[i];
    for (i=0; i < n; i++)
        RP += resistenze[i];
    RP = 1/RP;
    printf("La resistenza equivalente vale %f Ohm\n", RP);
    return 0;
}
```

Esercizio 13 (tdeB 24-11-2005)

- Si codifichi un programma C che legge due stringhe che rappresentano due parole e stampa un valore intero, da interpretarsi come valore di verità, che indica se le parole sono **anagrammi**, cioè se è possibile ottenere l'una dall'altra tramite permutazione delle loro lettere.
- Ad esempio le parole POLENTA e PENTOLA sono anagrammi. Si presti attenzione al fatto che parole come TAPPO e PATTO **non** sono anagrammi, anche se ogni lettera dell'una è contenuta nell'altra.

```
int main() {
        char a[100], b[100]; int len, contA, contB, i, k, si=1;
        printf("……"); scanf("%s",a); scanf("%s",b);
        len=strlen(a);
        if ( len != strlen(b) )
                si=0;
        for ( i = 0 ; i < len && si==1; i++ ) {  //per ogni a[i] in a (escluso il \0)
                contA = 0; contB = 0;
                for ( k = 0 ; k < len ; k++ ) {  //scandisco le stringhe
                        if ( a[k] == a[i] )
                                ++contA; //conto le occorrenze di a[i] in a
                        if ( b[k] == a[i] )
                                ++contB; //e le occorrenze di a[i] in b
                }
                if ( contA != contB )
                        si = 0;
        }
        printf("%d\n",si);  // se corrispondono tutti -> 1
        return 0;
}
```

Esercizio 14 (tdeB 21-7-2006)

- Definiamo il *grado minimo* **g** e il *grado massimo* **G** di una parola **P** rispettivamente come il minimo e il massimo numero di occorrenze delle lettere di P in P. Ad esempio:
  - ISTANBUL → **g**=1, **G**=1 (tutte le lettere della parola compaiono in essa una e una sola volta)
  - BOSFORO → **g**=1, **G**=3 (B, S, F, R compaiono una sola volta, O compare tre volte)
  - GALATASARAY → **g**=1, **G**=5 (G, L, T, S, R, Y compaiono una sola volta, A compare cinque volte)
  - MARMARA → **g**=2, **G**=3 (M e R compaiono due volte, A compare tre volte)
  - G e g valgono convenzionalmente 0 per la parola vuota (cioè per una parola priva di caratteri).
- Si scriva un programma C che legge una stringa di lunghezza generica che rappresenta P, calcola G e g, e li stampa

```
int main () {
        char p[100]; int g, G, i, j, count, lun;
        printf("…");
        scanf("%s",p);
        lun=strlen(p);
        g = lun;
        G = 0;
        for ( i=0; i<lun; i++ ) {
                count = 0;
                for ( j=0; j<lun; j++ ) {
                        if (p[i]==p[j]) {
                                count++;
                        }
                }
                if ( count < g )
                        g = count;
                if ( count > G )
                        G = count;
        }
        printf("g=%d,G=%d",g,G);
        return 0;
}
```

Esercizio 15 (tde 18-9-2006)

- Scrivere un programma C che legge un vettore di 100 elementi e stampa **1** se l'insieme rappresentato contiene tutti i numeri di un intervallo senza duplicati, senza omissioni, e **0** altrimenti.
- Ad esempio la proprietà sussiste per gli insiemi {4, 6, 5} e {-2, 0, 1, -3, -1}, ma non per {0, 3, -1, 1} (dove evidentemente manca il numero 2).

```
#define n 100
int main() {
        int v[n], i, k, min, trovato, si=1;
        /* ometto codice lettura vettore */
        min = v[0];
        for ( i = 1; i < n; i++ )
                if ( min > v[i] )
                        min = v[i];
        for ( k = min; k < min + n; k++ ) {
                trovato = 0;
                for ( i = 0; !trovato && i < n; i++ )
                        if ( v[i] == k )
                                trovato = 1;
                if ( trovato==0 )
                        si = 0;
        }
        printf("%d\n",si);
        return 0;
}
```

Esercizio 16

- Scrivere un programma C che legge un vettore di 100 elementi e stampa **1** se l'insieme rappresentato contiene tutti i numeri di un intervallo anche in presenza di duplicati, senza omissioni, e **0** altrimenti.
- Ad esempio la proprietà sussiste per gli insiemi {4, 5, 5} e {-2, 0, 1, -3, -1}, ma non per {0, 3, -1, 1} e {0, 3, 0, 1} (dove evidentemente manca il numero 2).

```
#define n 100
int main() {
        int v[n],i,k,min,max,trovato,si=1;/*ometto codice lettura vettore*/
        min = v[0];
        max = v[0];
        for ( i = 1; i < n; i++ ) {
                if ( min > v[i] )
                        min = v[i];
                if ( max < v[i] )
                        max = v[i];
        }
        for ( k = min; k <= max; k++ ) {
                trovato = 0;
                for ( i = 0; !trovato && i < n; i++ )
                        if ( v[i] == k )
                                trovato = 1;
                if ( !trovato )
                        si = 0;
        }
        printf("%d\n",si);
        return 0;
}
```

Esercizio 17 (tde 16-11-2007)

- Dato un array quotX di 365 valori interi, che rappresentano le quotazioni nell'anno solare corrente del titolo azionario X, si vuole realizzare un programma in linguaggio C in grado di:
  - Inizializzare il contenuto dell'array quotX con valori letti da standard input che devono essere compresi tra 1 e 100 (estremi inclusi), scartando eventuali valori fuori dall'intervallo.
  - Salvare in una opportuna/e variabile di tipo struttura (di cui si chiede anche la definizione del tipo) la quotazione massima e il primo giorno (espresso come posizione nell'array) in cui tale quotazione è stata memorizzata
  - Determinare e stampare a video il numero di periodi dell'anno in cui il titolo si è mantenuto costante (ossia, quante volte la medesima quotazione si è presentata in giorni successivi).
    - Ad esempio, se le quotazioni fossero **10 10** 15 14 **12 12 12** 8 **9 9 12 12**….. il titolo avrebbe avuto 4 periodi di quotazione costante.

```c
#include <stdio.h>
#define MAX 365
typedef struct {  int massimo;  int giorno;  } quotMax;

int main () {
  int quot[MAX], cont=0,i,inZona;
  quotMax qm;

  qm.massimo = -1;
  qm.giorno = -1;

  do {
            printf("Inserire l'elemento numero %d: ", cont);//o cont+1
            scanf("%d", &quot[cont]);
            if (quot[cont] >=0 && quot[cont] < 100) {
                        cont++;
            }
            else
                        printf("Inserimento non valido!\n");
  } while(cont<MAX);
```

```
for (cont=0;cont<MAX;cont++) {
        if (quot[cont] > qm.massimo) {
                qm.massimo = quot[cont];
                qm.giorno = cont;
                        //o cont+1, dipenda da come si interpreta il testo
        }
}
printf("Il valore massimo e': %d\n", qm.massimo);
printf("Il valore massimo si e' presentato in data: %d\n", qm.giorno);
```

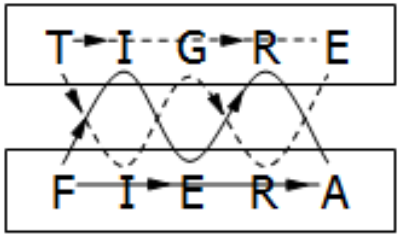
```
//conto periodi costanti
 cont = 0;
 inZona = 0;
 for (i=1; i< MAX; i++) {
        if (quot[i] == quot[i-1] && !inZona) {
                cont++;
                inZona=1;
        }
        else if (quot[i] != quot[i-1]) {
                inZona=0;
        }
 }

 printf("Numero di zone con valore costante: %d\n", cont);
 return 0;
}
```

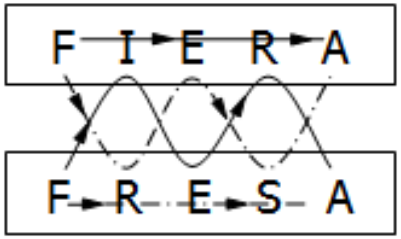
Esercizio 18

- Si vogliono calcolare tutti i numeri primi compresi tra 2 ed un numero assegnato. Un metodo antichissimo per determinare tutti i numeri primi è noto come Crivello di Eratostene: consiste nello scrivere di seguito tutti i numeri interi compresi tra 1 e n, e nel cancellare successivamente tutti i multipli di 2, eccetto 2; tutti i multipli di 3, eccetto 3; tutti i multipli di 5, eccetto 5 e così via. I numeri rimasti a seguito di un simile setacciamento saranno tutti e soli i numeri primi compresi tra 1 e n.
- Es: lim = 10
  1 2 3 4 5 6 7 8 9 10
  1 2 X X X X X
  1 2 3 X X X X X X
  .....
  1 2 3 X 5 X X X X X
  .....
  1 2 3 X 5 X 7 X X X

```
#include <stdio.h>
#define MAX_P 100
int main( ) {
   int multiples[MAX_P+1], i,noPrimo,lim;
   do { printf("Inserire il limite per il calcolo dei num. primi(<=%d): ",MAX_P);
        scanf("%d",&lim);  } while (( lim > MAX_P )||(lim <= 0));
   for (i=0; i < MAX_P+1; i++) {  multiples[i] = 1; }
   noPrimo = 2;
   while ( noPrimo <= lim ) {
            i = 2*noPrimo;
            while ( i <= lim ) {
                     multiples[i] = 0;
                     i = i+noPrimo;
            }
            do { noPrimo++; } while ((multiples[noPrimo]==0)&&(noPrimo<= lim));
   }
   printf("\nI numeri primi minori di %d sono: ",MAX_P);
   for (i = 1; i <= lim; i++)
            if (multiples[i] == 1)  printf("- %d -",i);
   return 0;
}
```

Esercizio 19 (tdeB)





- Due parole p e q si definiscono Hertzianamente compatibili se entrambe sono leggibili anche "oscillando" e leggendo alternativamente i caratteri dell'una e dell'altra. La figura mostra tigre con fiera e fiera con fresa. In figura parole uguali sono tracciate da linee di ugual stile.

  Si noti anche che la relazione sussiste in due modi (diretto o inverso). In figura: fiera e tigre si leggono iniziando dalla stessa lettera (modo diretto), per fiera e fresa occorre iniziare dall'iniziale dell' "altra" parola nella coppia (modo inverso). Se la relazione fosse definita solo in modo diretto o solo in modo inverso varrebbe anche la proprietà transitiva (esempio: aria, prua, erba, orma sono tutte direttamente compatibili tra loro), ma consideriamo due parole compatibili indipendentemente dal modo in cui la proprietà si manifesta. Si scriva un programma che verifica la compatibilità Hertziana di due stringhe.

```
#include <stdio.h>
#define N 100
int main( ) {
   char str1[N], str2[N];
   int i=0, j=0, lun1, lun2, compatibili;
   printf(“Inserire due stringhe\n”);
   scanf(“%s”,str1);
   scanf(“%s”,str2);
   lun1=strlen(str1);
   lun2=strlen(str2);
   if(lun1!=lun2)
         printf(“Stringhe non compatibili”);
   else{
      // codice che verifica
      ……..
   }
   return 0;
}
```

```
else {
   // codice che verifica
   compatibili=1;
   for(i=1;i<lun1  && compatibili; i=i+2) {
         if(str1[i]!=str2[i])
               compatibili=0;
    }
    if(compatibili==1)
         printf("str1 e str2 direttamente compatibili");
    else
         printf("str1 e str2 non direttamente compatibili");

   compatibili=1;
   for(i=0;i<lun1 && compatibili; i=i+2) {
         if(str1[i]!=str2[i])
               compatibili=0;
    }
    if(compatibili==1)
         printf("str1 e str2 inversamente compatibili");
    else
         printf("str1 e str2 non inversamente compatibili");
}
```

Esercizio 20 (tde 10-9-2012)

- Due parole *p*, *q* si definiscono *allacciabili* se un suffisso proprio ***s*** di *p* è anche prefisso proprio di *q*, cioè hanno la forma $p=w_1\mathbf{s}$ $q=\mathbf{s}w_2$ (dove ***s*** è *proprio* se è lungo almeno 2 lettere).
- *Esempi di parole allacciabili:*

  (oca, carina) → o**ca**rina

  (coraggio, raggio) → co**raggio**

  (bugiardi, giardino) → bu**giardi**no *(dei farmaci)*

  (spora, radici) → spo**ra**dici

  (imposta, stazione) → impo**sta**zione
- *Esempi di coppie non allacciabili:*

  (violon**cell**o, **cell**ulare), (corp**o**, **o**razione)
- Si codifichi in C un frammento di codice che stampa la lunghezza del massimo suf-/pre-fisso proprio in comune tra le due stringhe p e q (cioè la lunghezza di s, se $p=w_1s$ $q=sw_2$) e che inserisce nella stringa t la parola ottenuta dall'allacciatura che fattorizza la massima sovrapposizione, oppure la stringa vuota se le parole non sono allacciabili

Il cifrario di Cesare

Uno dei modi più semplici (e ormai meno efficaci) di cifrare un messaggio consiste nell'applicare una trasformazione "rotatoria" alle lettere che lo compongono, sostituendo ogni lettera con la lettera che si trova, nell'ordinamento alfabetico, k posizioni più avanti. Le ultime k-1 lettere dell'alfabeto sono trattate "rientrando" dall'inizio dell'alfabeto. Esempio:

```
offset? 2
mi piace questo corso
ok rkceg swguvq eqtuq
```

Per k=0 il messaggio resta inalterato, per k=1 ogni lettera è sostituita dalla successiva (e la z dalla a), per k=-1 dalla precedente (e la a dalla z).

Si scriva un programma C che legga un intero k e una sequenza di caratteri e restituisca su stdout la sequenza cifrata. (per brevità, si considerino solo i caratteri minuscoli).

Si noti che, essendo 26 le lettere dell'alfabeto inglese, per k=13 l'algoritmo di cifratura è identico a quello di decifratura, e che per k = 26 il messaggio resta inalterato come per k=0. Si verifichi la "robustezza" della soluzione proposta utilizzando valori di k pari a 26 o superiori (per k=27 il comportamento dev'essere uguale a quello con k = 1)..

Le cifre dei numeri

Scrivere un programma che legge da stdin un numero intero positivo (non una sequenza di caratteri, un numero! – si usi `scanf("%d", …)`) e verifica se si tratta di un numero *vario* o *ripetitivo*. Un numero è definito come **ripetitivo** se nella sua codifica decimale qualche cifra compare più di una volta, **vario** se le sue cifre sono tutte diverse. Ad esempio:

I numeri 100 121 2002 e 124565 sono **ripetitivi**

I numeri 5 124 321 e 1789 sono **vari**

Scrivere poi un programma che legge dati due numeri interi positivi stabilisce se sono uno l'"anagramma" dell'altro, cioè sono ottenibili uno dall'altro con una permutazione delle cifre. Esempi:

```
1234  2143  -->  anagrammi
1131  3113  -->  no
1314  1143  -->  anagrammi
 112  1122  -->  no
```

Per entrambi i problemi, come esercizio di "elasticità espressiva" si progettino (almeno) **due soluzioni**: se la prima soluzione concepita non utilizza gli array, ci si domandi se gli array possono "semplificare" la formulazione. Se la prima soluzione concepita fa uso di array, se ne sviluppi anche una che non li utilizza.

Numeri Fortunati

Scrivere un programma che stampi tutti i numeri fortunati minori di 200.

Un **numero fortunato** è un numero naturale in un insieme generato come segue:

Si inizia con la successione di tutti i numeri interi:

1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25…

Si eliminano poi **tutti i secondi numeri** (ovvero si lasciano solo i numeri dispari):

1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43…

Il **secondo** termine rimasto in questa sequenza **è 3**. Si eliminano dunque **tutti i terzi numeri** che rimangono nella sequenza:

1, 3, 7, 9, 13, 15, 19, 21, 25, 27, 31, 33, 37, 39, 43…

Il **terzo** numero rimasto ora **è 7**. Si eliminano dunque **tutti i settimi numeri** che rimangono nella sequenza:

1, 3, 7, 9, 13, 15, 21, 25, 27, 31, 33, 37, 43… poi i **noni** numeri, poi i **tredicesimi**…

Ripetendo la procedura indefinitamente, i rimanenti sono numeri fortunati:

1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, 87, 93, 99...

Istogrammi

Scrivere un programma visualizzi i valori di un array di 10 interi (anche negativi) in forma di istogramma, e ne tracci anche il valor medio, nel modo seguente (nell'esempio, il valor medio è 6. In generale):