Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Dati aggregati - Primi esercizi*

1 Per cominciare

Considerate il codice contenuto nel file `es.c`. Riassumete in una frase cosa fa il programma.

```c
#include <stdio.h>

#define LENGTH 100

int main( void) {

    int a[LENGTH], i, lun;

    for( i = 0; i < LENGTH; i++ ) {
        scanf( "%d", &a[i] );
        if ( a[i] == 0 )
            break;
    }

    lun = i;

    for( i = lun -1; i > 0; i-- )
        printf( "%d ", a[i] );

    printf( "%d\n", a[0] );
    return 0;
}
```

2 Palindrome

Una stringa si dice *palindroma* se è uguale quando viene letta da destra a sinistra e da sinistra a destra. Quindi "enne" è palindroma, ma "papa" non lo è. Scrivete un programma che legge una stringa terminata da un punto e stabilisce se è palindroma.

3 Cifre

Considerate la seguente porzione di codice (vedi file `cifre.c`) assumendo che sia contenuta in un programma che include `ctype.h`.

*Ultima modifica 14 ottobre 2019

```
1       int c, f[10] = {0};
2
3       c = getchar();
4       while ( c != '.' ) {
5           if ( isdigit( c ) )
6               f[ c - '0' ]++;
7           c = getchar();
8       }
```

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

1. Se `c` è una cifra, che cosa indica `c - '0'`?
2. A cosa serve l'istruzione nella riga 6?
3. Come descrivereste il valore `f[i]`?
4. Date un nome più significativo all'array `f`.
5. Inserite istruzioni per stampare su standard output dei messaggi esplicativi di cosa fa questa porzione di codice.
6. Riassumete in una frase cosa fa questa porzione di codice.

4 Ripetizioni

1. Usando la porzione di codice `cifre.c` dell'esercizio precedente, scrivete un programma che legga una sequenza di caratteri arbitrari terminata da `'.'` e stampi tutte e sole le cifre ivi contenute (non è importante l'ordine in cui appaiono).
2. Modificate il programma precedente in modo che stampi tutte e solo le cifre che appaiono ripetute nell'input.

5 Divisioni

Considerate il codice contenuto nel file `divisioni.c`.

```
1   #include <stdio.h>
2   #define N 10
3
4   int main( void) {
5       int b, n, i = 0;
6       int c[N] = {0};
7
8       scanf( "%d %d", &n, &b );
9
10      do c[i++] = n % b;
11          while ( ( n /= b ) > 0 );
12
13      while ( i > 0 )
14          printf( "%d", c[--i] );
15
16      printf( "\n" );
17
18      return 0;
19  }
```

Analizzate il codice sorgente e rispondete, possibilmente per iscritto, alle seguenti domande. Se avete dubbi, potete testarlo, eseguendolo su casi di input significativi e modificandolo.

- Date dei nomi più significativi alle variabili `n` e `b`

- Date un nome più significativo all'array `c`.
- Date un nome più significativo al programma.
- Inserite istruzioni per stampare su standard output dei messaggi esplicativi di cosa fa il programma.
- Riassumete in una frase cosa fa il programma.
- Se `b` vale 10, descrivete il valore di `c[2]` in relazione all'intero `n`.

6 Il cifrario di Cesare rivisto

Scrivete un programma che legga (usando `getchar`) un testo da cifrare, sotto forma di una sequenza di caratteri terminata da un punto, poi legga (usando `scanf`) la chiave di cifratura k e quindi stampi il testo cifrato usando il cifrario di Cesare con chiave k.

Suggerimento: osservate cosa cambia rispetto alla versione originaria dell'esercizio, svolta durante l'esercitazione precedente.

7 Cancella l'ultimo carattere

Scrivete un programma che legga una sequenza di caratteri (terminata da un a-capo) e la ristampi identica ma saltando tutte le occorrenze dell'ultimo carattere. Potete assumere che la sequenza contenga al più 100 caratteri. Ad esempio, se il programma riceve da standard input il testo

```
La vispa Teresa avea tra l'erbetta a volo sorpresa gentil farfalletta
```

l'output deve essere

```
L visp Teres ve tr l'erbett \ volo sorpres gentil frfllett
```

8 Figure geometriche

Considerate il codice contenuto nel file `figure.c`. Nel programma si usano **`typedef`** e **`struct`** per definire dei nuovi tipi (`Punto` e `Rettangolo`) e si calcola l'area e il perimetro di un rettangolo.

```
1  #include<stdio.h>
2  #include<math.h>
3
4  typedef struct {
5    float x, y;
6  } Punto;
7
8  typedef struct {
9    Punto p1;
10   Punto p2;
11 } Rettangolo;
12
13 int main( void ){
14
15   float b, h, area, duep;
16   Rettangolo r;
17
18   printf( "RETTANGOLO:\n" );
19   printf( "Inserisci le coordinate del punto in basso a sinistra\n" );
20   scanf( "%f%f" , &r.p1.x, &r.p1.y );
21   printf( "Inserisci le coordinate del punto in alto a destra\n" );
22   scanf( "%f%f", &r.p2.x, &r.p2.y );
```

```
23    b = r.p2.x - r.p1.x;
24    h = r.p2.y - r.p1.y;
25    area = b * h;
26    duep = 2 * ( b + h );
27    printf( "L'area del rettangolo vale %f, il perimetro vale %f\n", area, duep );
28
29    return 0;
30  }
```

Espandete il programma affinché calcoli anche perimetro e area di un cerchio qualunque, definito tramite le coordinate del suo centro e dal suo raggio.

9 Quadrato magico

Considerate il codice contenuto nel file `quadratoMagico.c`. Il codice serve a costruire un quadrato magico memorizzandolo in un array bidimensionale. Un quadrato magico è una disposizione dei numeri $1, 2, \cdots, n^2$ –con n dispari– tale che in ogni riga, in ogni colonna e nelle due diagonali la somma dei numeri sia la stessa.

```
1    int quadrato[n][n] = {{0}}, k = 1;
2    int i, j, inew, jnew;
3
4    i = 0; j = n/2;
5    nn = n * n;
6    for ( k = 1; k <= nn; k++ ) {
7      quadrato[i][j] = k;
8      inew = ( i == 0 ) ? n - 1 : i - 1;
9      jnew = ( j == n - 1 ) ? 0 : j + 1;
10     if ( quadrato[ inew ][ jnew ] == 0 ) {
11       i = inew;
12       j = jnew;
13     }
14     else {
15       i++;
16     }
17   }
```

Rispondete alle domande seguenti:

1. Come deve essere dichiarato `n`? Che assunzioni ha senso fare sul valore di `n`?
2. Spiegate come è usata la variabile `k` e cosa rappresenta.
3. Spiegate a parole quale valore viene assegnato a inew `inew` `inew` nella riga 8.
4. Riscrivete l'assegnamento nella riga 8 usando il costrutto **`if`** invece dell'operatore ternario.
5. Senza eseguire il programma al computer, tracciatene l'esecuzione quando `n` è pari a 5.
6. Verificate la correttezza della vostra risposta al punto precedente, completando il programma con la stampa del quadrato magico.
7. Descrivete a parole la strategia per costruire il quadrato magico che è implementata dalla suddetta porzione di codice. Iniziate con "Si parte mettendo il numero 1 al centro della prima riga . . . ..."
8. Completate il codice in modo da verificare che su ogni riga, su ogni colonna e sulle due diagonali la somma dei numeri sia la stessa. (Quanto deve valere tale somma in un qualunque quadrato magico di dimensione n?)