RACCOLTA DI MATERIALE ED ESERCIZI PER LA PROGRAMMAZIONE IN LINGUAGGIO C

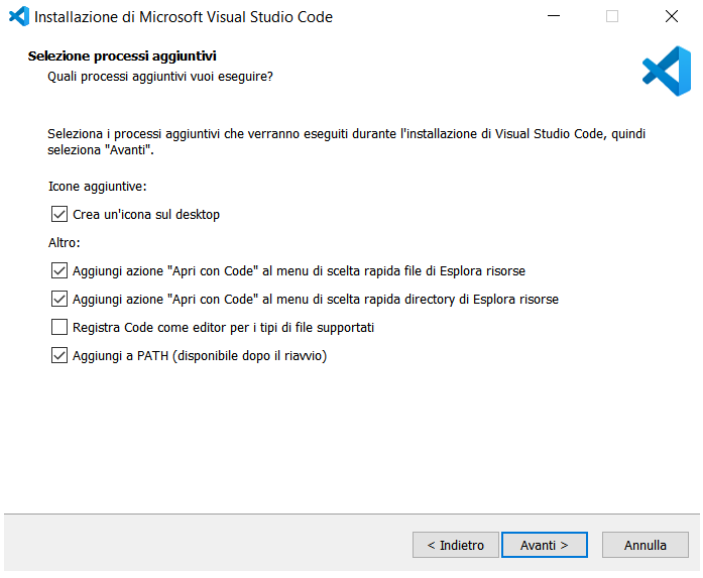
Ambiente di sviluppo

1. Installazione dell'ambiente di lavoro:

1.1 Visual Studio Code

Per installare visual studio code consiglierei di usare, a meno di esigenze particolari, il system installer da 64bit che puoi trovare sul sito https://code.visualstudio.com/download .

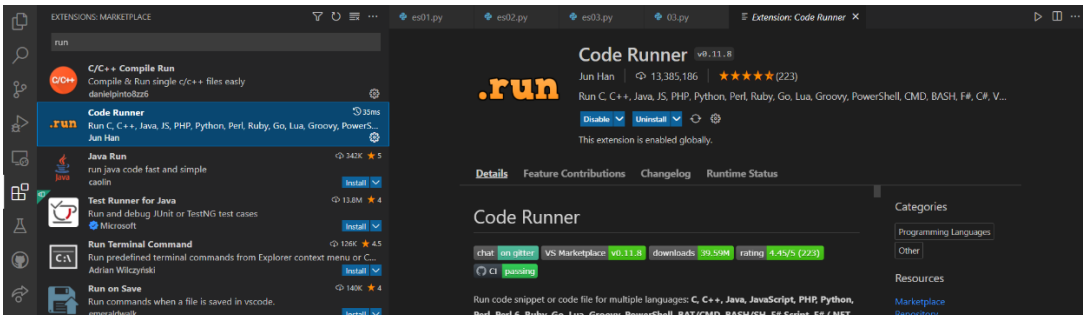
Ricordarsi durante l'installazione di flaggare le voci corrette durante tutti i passaggi come mostrato dalla seguente immagine:



Per programmare comodamente in python con Visual studio code consiglio di installare le seguenti estensioni:

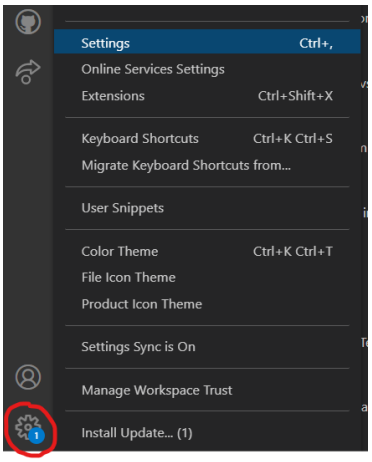
- C++.

- ~~Code Runner~~



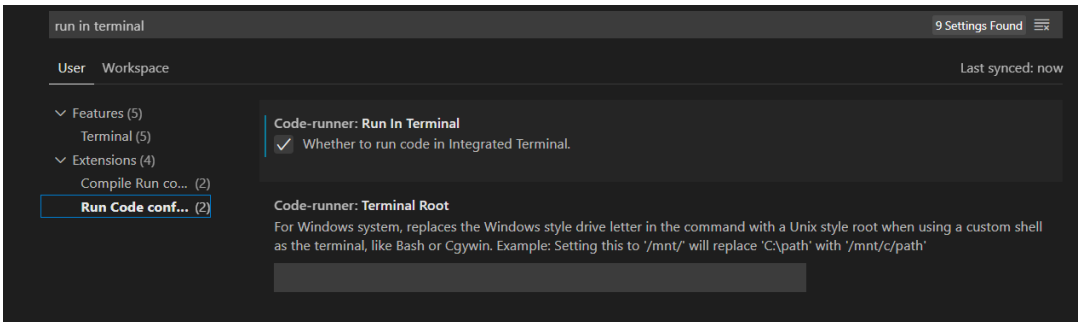
- Al posto di Code Runner ora consiglio C/C++ Compile Run

Una volta installate le estensioni vanno modificate alcune impostazioni. Il menù impostazioni è velocemente accessibile dal tasto in basso a sinistra:

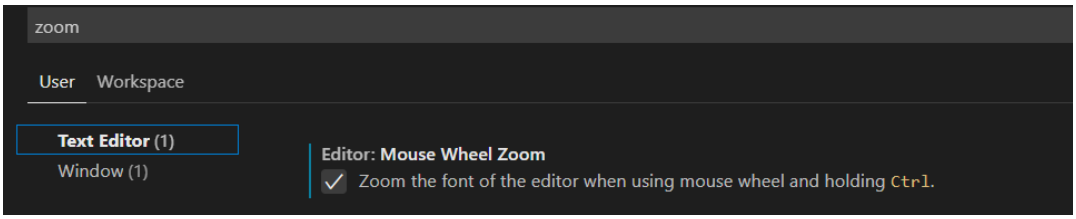


Le impostazioni da modificare sono:

- “run in terminal” da spuntare nelle impostazioni e trovabile velocemente inserendo la voce nella barra di ricerca e cliccando sull’estensione interessata come nella seguente immagine:



- “Mouse Wheel Zoom” da spuntare come fatto per l’impostazione precedente



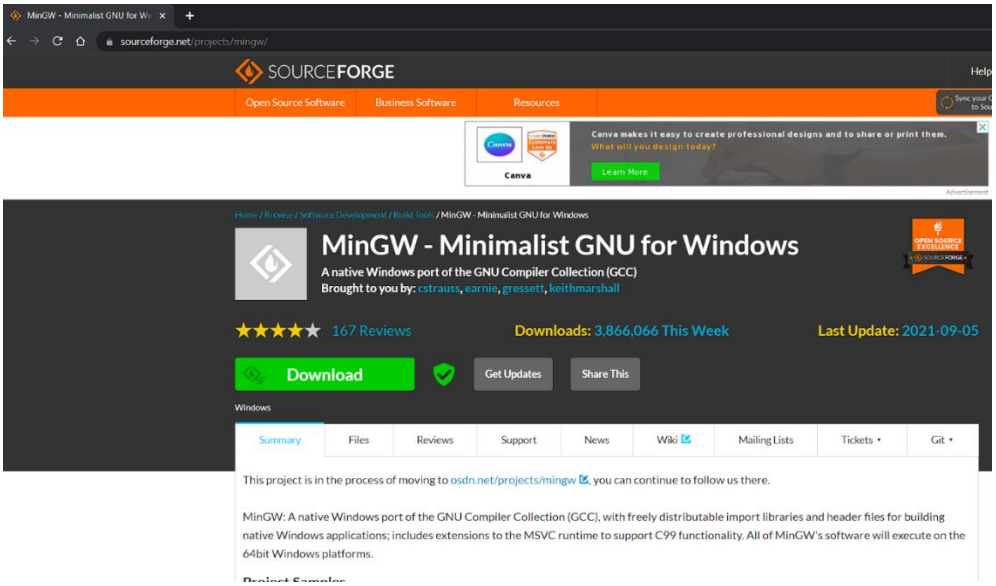
È possibile modificare a piacere altre impostazioni come:

- Save file before run
- Save all files before run
- Color Theme
- Clear Previous Output

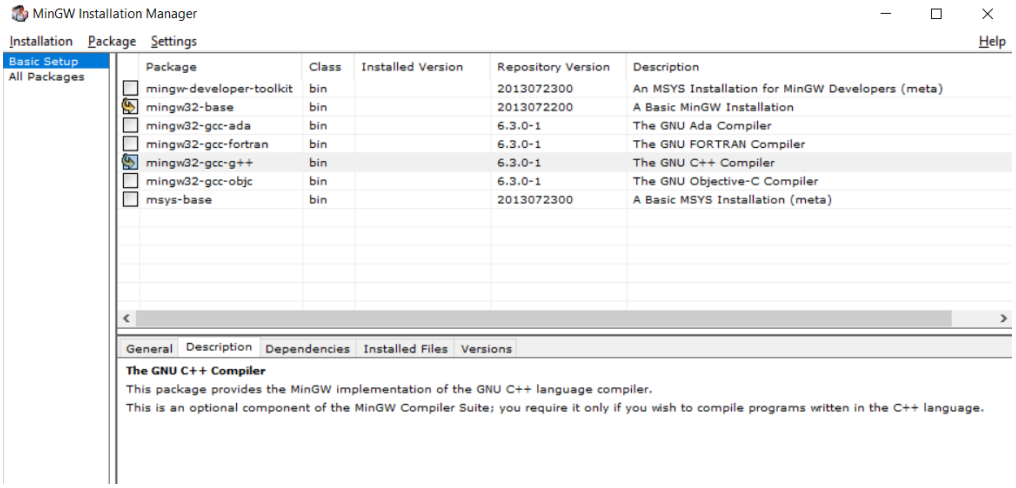
2. Installazione del Compilatore: MinGW

Per poter eseguire programmi scritti in C è necessario prima compilare il codice, per farlo serve un compilatore. Un buon compilatore per Windows è MinGW, per installarlo segui i seguenti passaggi:

1. Scarica l'installer da https://sourceforge.net/projects/mingw/ , cliccando sul pulsantone verde "download";



2. Apri l'installer che hai scaricato e alle prime due schermate premi i pulsanti "Install" e "Continue" senza cambiare nulla. L'installer ci metterà un po' di tempo a scaricare tutto, poi dopo aver premuto di nuovo "Contunue" comparirà una schermata da cui scegliere i componenti che si vogliono installare.



Cliccare a sinistra su “Basic Setup” e poi a destra sui quadratini a fianco di “mingw32-base” e mingw32-gcc-g++” e scegliere “Mark for Installation”. Infine scegliere dal menu in alto a sinistra “Installation” la voce “Apply changes”. Quando si apre la finestra premere su “Apply”.

Alla fine dell’installazione premere “Close”.

3. Cerca nel pannello di controllo di windows “modifica le variabili di ambiente relative al sistema” e aprilo (puoi cercarlo velocemente dal menu start con la parola chiave “path”). Sulla finestra che si apre clicca il bottone in basso a destra “Variabili d’ambiente…”.
   Nella finestra che si apre ci sono due elenchi, vai nel secondo dedicato alle variabili di sistema e fai doppio click sulla varibile “Path”.
   Nella nuova finestra clicca su nuovo e scrivi il percorso “C:\MinGW\bin” (posizione di default, se hai installato mingw in un'altra cartella dovrai inserire quel percorso).

A questo punto VSCode è in grado di trovare il compilatore ed usarlo. Se hai già aperto VSCode prima di installare il MinGW dovrai chiuderlo e riaprirlo perché VSCode carica le variabili di sistema all’avvio.

Indicazioni e materiale

Come manuale contenente esempi e spiegazioni di base su ogni aspetto del linguaggio consiglio di guardare il sito **https://www.w3schools.com/c/** .
Online sono presenti molti manuali e tutorial, c’è solo l’imbarazzo della scelta. Di seguito ne riporto alcuni che vi possono essere utili:

1. Manuale online in inglese con spiegazioni ed esempi su ogni aspetto del linguaggio: **https://www.w3schools.com/c/ ;**
2. Video tutorial in italiano, diviso in diversi episodi, molto bello e completo con molte spiegazioni**:** **https://www.youtube.com/playlist?list=PL83Ordjpzm5oUl7tFEjc4iirkPBiv7FxR ;**
3. Tutorial testuale in italiano: https://www.html.it/guide/guida-c/ .

Esercizi ed Esempi divisi per argomento

Per ora sono presenti solo i testi degli esercizi seguiti eventualmente dalla soluzione. Gli esempi base per ora sono solo raccolti in una serie di file contenenti codice d'esempio. In futuro saranno raccolti anche qui.

1. Variabili e operazioni elementari

1.1.1 Somma di due numeri letti in input (v1 - usando una terza variabile)

Soluzione

1.1.2 Data in input l'area del cerchio, stampare la circonferenza

Soluzione

```c
#include <math.h>
#include <stdio.h>

int main() {
    float cerchio, circonferenza, raggio;

    printf("Scrivi quanto misura l'area del cerchio: ");
    scanf("%f", &cerchio);

    raggio = sqrt(cerchio / 3.14);
    circonferenza = 2 * 3.14 * raggio;

    printf("Il cerchio con area %.2f ha un raggio che misura %.2f e una circonferenza che misura
%.2f \n", cerchio, raggio,
           circonferenza);
}
```

1.1.3 Data la base e l'altezza di un triangolo, scrivere l'area

Soluzione

```c
#include <stdio.h>

int main() {
    float base, altezza, area;

    printf("Scrivi la base: ");
    scanf("%f", &base);
    printf("Scrivi l'altezza: ");
    scanf("%f", &altezza);

    area = base * altezza / 2;

    printf("L'area misura %.2f", area);
}
```

1.1.4 Dati in input due numeri, scrivi il risultato di tutte le operazioni che è possibile fare con i due numeri (somma, sottrazione, moltiplicazione, divisione, resto della divisione)

1.1.5 Dato un numero decimale, ottieni e stampa:

1. Valore intero approssimato per difetto
2. Valore intero approssimato per eccesso
3. Valore intero approssimato in maniera intelligente
4. approssimazione del numero al secondo decimale
5. Valore assoluto del numero

Soluzione

2. Algoritmi e strutture di controllo

2.1.1 Data l'area del cerchio, stampare la circonferenza

Soluzione

```c
// 1.1  Data l'area del cerchio, stampare la circonferenza

#include <math.h>
#include <stdio.h>

int main() {
    float cerchio, circonferenza, raggio;

    printf("Scrivi quanto misura l'area del cerchio: ");
    scanf("%f", &cerchio);

    raggio = sqrt(cerchio / 3.14);
    circonferenza = 2 * 3.14 * raggio;

    printf("Il cerchio con area %.2f ha un raggio che misura %.2f e una circonferenza che misura
%.2f \n", cerchio, raggio,
           circonferenza);
}
```

2.1.2 Data la base e l'altezza di un triangolo, scrivere l'area

Soluzione

```c
// Data la base e l'altezza di un triangolo, scrivere l'area

#include <stdio.h>

int main() {
    float base, altezza, area;

    printf("Scrivi la base: ");
    scanf("%f", &base);
    printf("Scrivi l'altezza: ");
    scanf("%f", &altezza);

    area = base * altezza / 2;
```

```
    printf("L'area misura %.2f", area);
}
```

2.1.3 Somma di due numeri (v1 - usando una terza variabile)

Soluzione

```
#include <stdio.h>

int main() {
    float base, altezza, area;
    printf("Scrivi la misura della base: ");
    scanf("%f", &base);
    printf("Scrivi la misura dell'altezza: ");
    scanf("%f", &altezza);
    area = base * altezza;
    printf("%g", area);
}
```

2.1.4 Dati due numeri calcolare il loro quoziente se il divisore è != 0, ritornare “impossibile” se il divisore = 0

Soluzione

```
#include <stdio.h>

int main() {
    float a,b,c;
    printf("Scrivi il primo numero: ");
    scanf("%f", &a);
    printf("Scrivi il secondo numero: ");
    scanf("%f", &b);
    if (b != 0) {
        c = a / b;
        printf("%g", c);
    } else {
        printf("Non e' possibile dividere per 0\n");
    }
}
```

2.1.5 Dato in input un numero:

1. Stampa i numeri da 1 a quel numero
2. Stampa i numeri da quel numero a 0 (conto alla rovescia)

Soluzione

```
#include <stdio.h>

int main() {
    int n, i;

    // input - versione col while
    printf("Scrivi un numero positivo: ");
    scanf("%d", &n);
    while (n <= 0) {
        printf("Il numero non è positivo, riscrivilo: ");
```

```
        scanf("%d", &n);
    }

    // input - versione col do-while
    do {
        printf("Scrivi un numero positivo: ");
        scanf("%d", &n);
    } while (n <= 0);


    // 1.   Stampa i numeri da 1 a quel numero
    for (i = 1; i <= n; i++) {
        printf("%d ", i);
    }

    printf("\n");

    // 2.   Stampa i numeri da quel numero a 0 (conto alla rovescia)
    for (i = n; i >= 0; i--) {
        printf("%d ", i);
    }
}
```

2.1.6 Dato in input un numero positivo, stampa tutti i numeri pari minori di quel numero

Soluzione

```
// 2.1.6 - Dato in input un numero positivo, stampa tutti i numeri
// pari minori di quel numero

#include <stdio.h>

int main() {
    int n, i;

    printf("Scrivi un numero positivo: ");
    scanf("%d", &n);
    while (n <= 0) {
        printf("Il numero inserito e' negativo, scrivi un numero positivo: ");
        scanf("%d", &n);
    }

    // primo metodo
    for (i = 1; i < n; i++) {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }

    printf("\n");

    // secondo metodo
    for (i = 2; i < n; i+=2) {
        printf("%d ", i);
    }
}
```

2.1.7 Dato in input un numero n, stampa la prima potenza di 2 maggiore o uguale a n

Soluzione

```c
#include <stdio.h>

int main() {
    int n, ris;

    printf("Scrivi un numero: ");
    scanf("%d", &n);

    ris = 1;
    while (ris < n) {
        ris *= 2;
    }

    printf("%d", ris);
}
```

2.1.8 Moltiplicazione di due numeri senza usare l'operatore *

Soluzione

```c
#include <stdio.h>

int main() {
    int a, b, ris, i;

    a = 3;
    b = 5;

    ris = 0;

    for (i = 0; i < b; i++) {
        ris += a;
    }

    printf("%d * %d = %d", a, b, ris);
}
```

2.1.9 Divisione di due numeri interi senza usare l'operatore di divisione

Variante: stampa anche il resto

Soluzione

```c
#include <stdio.h>

int main() {
    int num, den, resto, risultato;

    do {
        printf("Scrivi il numeratore (positivo): ");
        scanf("%d", &num);
    } while (num < 0);

    do {
        printf("Scrivi il denominatore (positivo): ");
        scanf("%d", &den);
    } while (den <= 0);

    resto = num;
```

```
    for (risultato = 0; resto >= den; risultato += 1) {
        resto -= den;
    }

    printf("%d / %d = %d  con resto %d\n", num, den, risultato, resto);
}
```

2.1.10 dati dall'utente due numeri interi n1 e n2 il programma deve stampare:

1. tutti i numeri dal più piccolo dei due al più grande dei due
2. tutti i numeri dal più grande dei due al più piccolo dei due
3. tutti i numeri da n1 a n2 (comunque siano n1 e n2 quindi potresti dover andare in ordine crescente o decrescente)
4. tutti i numeri da n2 a n1 (comunque siano n1 e n2 quindi potresti dover andare in ordine crescente o decrescente)

Soluzione

```
#include <stdio.h>

int main() {
    int n1, n2, min, max, i;

    n1 = 12;
    n2 = 5;

    // a.   tutti i numeri dal più piccolo dei due al più grande dei due

    if (n1 < n2) {
        min = n1;
        max = n2;
    } else {
        max = n1;
        min = n2;
    }

    for (i = min; i <= max; i++) {
        printf("%d ", i);
    }
    printf("\n");

    // b.   tutti i numeri dal più grande dei due al più piccolo dei due
    for (i = max; i >= min; i--) {
        printf("%d ", i);
    }
    printf("\n");

    // c.   tutti i numeri da n1 a n2 (comunque siano n1 e n2 quindi potresti dover andare in
ordine crescente o decrescente)
    if (n1 < n2) {
        for (i = n1; i <= n2; i++) {
            printf("%d ", i);
        }
    printf("\n");
    } else {
        for (i = n1; i >= n2; i--) {
            printf("%d ", i);
        }
    }
```

```
    printf("\n");

    // d.   tutti i numeri da n2 a n1 (comunque siano n1 e n2 quindi potresti dover andare in
ordine crescente o decrescente)
    if (n2 < n1) {
        for (i = n2; i <= n1; i++) {
            printf("%d ", i);
        }
    printf("\n");
    } else {
        for (i = n2; i >= n1; i--) {
            printf("%d ", i);
        }
    }
    printf("\n");
}
```

2.1.11 Inserire n numeri != 0 (0 per finire), contare quanti sono i numeri inseriti

Soluzione

```
int main() {
    int n, numero;

    // for (n = 0; numero != 0;) {
    //     printf("Scrivi il %d° numero: ");
    //     scanf("%d", numero);
    //     if (numero != 0) {
    //         n++;
    //     }
    // }

    // for (n = 0; numero != 0; n++) {
    //     printf("Scrivi il %d° numero: ");
    //     scanf("%d", numero);
    //     if (numero == 0) {
    //         break;
    //     }
    // }

    for (n = 0; 1; n++) {
        printf("Scrivi il %d numero: ", n+1);
        scanf("%d", &numero);
        if (numero == 0) {
            break;
        }
    }

    printf("Hai inserito %d numeri.\n\n", n);

}
```

2.1.12 Variante avanzata: i numeri sono voti che devono essere validi e alla fine ne va calcolata la media.

2.1.13 Dati tre numeri reali dire che tipo di triangolo essi formano (classificazione dei triangoli in base ai lati).

2.1.14 Data una sequenza di n numeri interi (valore di n dato dall'utente), calcolare la somma dei pari ed il prodotto dei dispari

Soluzione

```c
// 2.1.14 - Data una
// sequenza di n numeri interi (valore di n dato dall'utente), calcolare la somma
// dei pari ed il prodotto dei dispari

#include <stdio.h>

int main() {
    int n, somma, prodotto, i, numero;

    printf("Quanti numeri vuoi inserire? ");
    scanf("%d", &n);
    while (n <= 0) {
        printf("Il valore non puo' essere negativo.");
        printf("Quanti numeri vuoi inserire? ");
        scanf("%d", &n);
    }

    somma = 0;
    prodotto = 1;
    for (i = 0; i < n; i++) {
        printf("Scrivi un numero positivo: ");
        scanf("%d", &numero);
        while (numero < 0) {
            printf("Scrivi un numero positivo: ");
            scanf("%d", &numero);
        }

        if (numero % 2 == 0) {
            somma += numero;
        } else {
            prodotto *= numero;
        }
    }

    printf("Somma dei pari: %d\n", somma);
    printf("Prodotto dei dispari: %d\n", prodotto);
}
```

2.1.15 Con un opportuno ciclo chiedere all'utente di inserire due numeri n1 e n2 compresi tra 1 e 100. Se i numeri non fossero corretti, rimanere nel ciclo e ripetere la richiesta. Stampare quanti sono i numeri pari compresi tra i due numeri

Soluzione

```c
// 2.1.15 -   Con un
// opportuno ciclo chiedere all'utente di inserire due numeri n1 e n2 compresi tra
// 1 e 100. Se i numeri non fossero corretti, rimanere nel ciclo e ripetere la
// richiesta. Stampare quanti sono i numeri pari compresi tra i due numeri

#include <stdio.h>

int main() {
    int n1, n2, ris, i, tmp;

    do {
        printf("Scrivi due numeri compresi tra 1 e 100: ");
        scanf("%d %d", &n1, &n2);
    } while (n1 < 1 || n1 > 100 || n2 < 1 || n2 > 100);

    if (n1 > n2) {
        tmp = n1;
```

```
        n1 = n2;
        n2 = tmp;
    }

    // metodo 1
    ris = 0;
    for (i = n1; i <= n2; i++) {
        if (i%2==0) {
            ris++;
        }
    }
    printf("I numeri pari compresi tra %d e %d sono %d\n", n1, n2, ris);

    // metodo 2
    ris = 0;
    i = n1;
    if (i % 2 == 1) {
        i++;
    }
    for (;i <= n2; i += 2) {
        ris++;
    }
    printf("I numeri pari compresi tra %d e %d sono %d\n", n1, n2, ris);

    // metodo 3
    ris = (n2-n1) / 2;
    if (n1 % 2 == 0 || n2 % 2 == 0) {
        ris++;
    }

    printf("I numeri pari compresi tra %d e %d sono %d", n1, n2, ris);
}
```

2.1.16 Data una sequenza di numeri terminata dal numero 0 (leggo numeri finchè non mi viene dato il numero 0), stampo il maggiore e il minore

Soluzione

```
// 2.1.16 - Data una sequenza di numeri terminata dal numero 0
// (leggo numeri finchè non mi viene dato il numero 0), stampo il maggiore e il
// minore

#include <stdio.h>

int main() {
    int num, max, min;

    printf("Scrivi un numero (0 per terminare): ");
    scanf("%d", &num);
    min = num;
    max = num;

    while (num != 0) {
        printf("Scrivi un numero (0 per terminare): ");
        scanf("%d", &num);
        if (num > max) {
            max = num;
        }
        if (num < min) {
            min = num;
        }
    }

    printf("I valori minimi e massimi inseriti sono: %d %d\n", min, max);
}
```

2.1.17 Letto un numero intero positivo n stampare il fattoriale: n! = 1 · 2 · 3 · . . . ·n

Soluzione

```c
#include <stdio.h>

int main() {
    int n, i;
    long long f;

    do {
        printf("Scrivi un numero positivo: ");
        scanf("%d", &n);
    } while (n < 0);

    f = 1;
    // for (i = 2; i <= n; i++) {
    for (i = n; i > 0; i--) {
        f *= i;
    }

    printf("%d! = %lld\n\n", n, f);
}
```

Usando i double si riesce a rappresentare numeri molto grandi perché il double (come il float ma usa più byte) usa una notazione scietifica

```c
#include <stdio.h>
#include <float.h>

int main() {
    int i;
    double n;
    double f;

    do {
        printf("Scrivi un numero positivo: ");
        scanf("%lf", &n);
    } while (n < 0);

    f = 1;
    // for (i = 2; i <= n; i++) {
    for (i = n; i > 0; i--) {
        f *= i;
    }

    printf("%.0lf! = %e\n\n", n, f);
}
```

2.1.18 Dati in input 3 numeri, stamparli in ordine crescente

2.1.19 Realizzare un programma che, presi in input 2 operandi reali e un operatore (+, -, *, /), esegue l'operazione stampandone il risultato

Soluzione

```c
#include <stdio.h>

int main() {
```

```
    int a, b;
    char op;

    printf("Scrivi l'operazione da fare: ");
    scanf("%d %c %d", &a, &op, &b);

    if (op == '+') {
        printf("%d %c %d = %d", a, op, b, a+b);
    } else if (op == '-') {
        printf("%d %c %d = %d", a, op, b, a-b);
    } else if (op == '*') {
        printf("%d %c %d = %d", a, op, b, a*b);
    } else if (op == '/') {
        printf("%d %c %d = %d", a, op, b, a/b);
    } else {
        printf("Non riconosco l'operazione da fare.");
    }
}
```

2.1.20 Progettare un algoritmo che legga da terminale una sequenza di interi positivi e negativi terminati dal valore 0 (uno su ogni linea) e stampi il prodotto degli interi positivi e la somma dei negativi.

2.1.21 Progettare un algoritmo che legga da terminale una sequenza di interi positivi e negativi fintanto che l'utente dice di volerne inserire ancora, e stampi il prodotto degli interi positivi e la somma dei negativi.

Soluzione

```
#include <stdio.h>

int main() {
    int numero, risposta, somma, prodotto;

    somma = 0;
    prodotto = 1;
    while (1) {
        printf("Vuoi inserire un numero? (0 per no, altro numero per sì): ");
        scanf("%d", &risposta);
        if (risposta == 0) {
            break;
        }
        printf("Scrivi il numero: ");
        scanf("%d", &numero);
        if (numero >= 0) {
            prodotto *= numero;
        } else {
            somma += numero;
        }
    }

    printf("Prodotto dei positivi: %d\n", prodotto);
    printf("Somma dei negativi: %d\n", somma);
}
```

Oppure

```
#include <stdio.h>

int main() {
    int numero, risposta, somma, prodotto;

    somma = 0;
    prodotto = 1;
```

```
    printf("Vuoi inserire un numero? (0 per no, altro numero per sì): ");
    scanf("%d", &risposta);

    // while (risposta != 0) {
    while (risposta) {
        printf("Scrivi il numero: ");
        scanf("%d", &numero);
        if (numero >= 0) {
            prodotto *= numero;
        } else {
            somma += numero;
        }

        printf("Vuoi inserire un numero? (0 per no, altro numero per sì): ");
        scanf("%d", &risposta);
    }

    printf("Prodotto dei positivi: %d\n", prodotto);
    printf("Somma dei negativi: %d\n", somma);
}
```

2.1.22 Dati due numeri, determinare il maggiore (verificare anche se sono uguali)

2.1.23 Data una sequenza di numeri terminata da 0, dire quanti sono i numeri inseriti (diversi da 0)

2.1.24 Dati due numeri in input b ed e, calcola e scrivi b^e senza usare la funzione pow.

Soluzione

```
// 2.1.24 - Dati due numeri in input b ed e, calcola e scrivi b^e senza usare la
// funzione pow.

#include <stdio.h>

int main() {
    int b, e, i;
    double ris;

    b = 2;
    e = 100;
    ris = 1;
    for (i = 0; i < e; i++) {
        ris *= b;
    }
    printf("%d^%d = %g", b, e, ris);
}
```

2.1.25 Dati in ingresso 4 numeri, che rappresentano gli orari in cui avvengono due diversi eventi della giornata, in modo che i primi 2 numeri siano ore e minuti del primo orario e gli altri 2 numeri siano ore e minuti del secondo orario, stabilire quale evento è avvenuto prima. Inserisci dei controlli durante l'input in modo che le ore possano andare da 0 a 23 e i minuti da 0 a 59.

Soluzione

```
// 2.1.25 - Dati in
// ingresso 4 numeri, che rappresentano gli orari in cui avvengono due diversi
// eventi della giornata, in modo che i primi 2 numeri siano ore e minuti del primo
// orario e gli altri 2 numeri siano ore e minuti del secondo orario, stabilire
```

```c
// quale evento è avvenuto prima. Inserisci dei controlli durante l'input in modo
// che le ore possano andare da 0 a 23 e i minuti da 0 a 59.

#include <stdio.h>

int main() {
    int h1, m1, h2, m2;
    printf("Scrivi h1: ");
    scanf("%d", &h1);
    while (h1 < 0 || h1 > 23) {
        printf("Valore non valido, reinserisci: ");
        scanf("%d", &h1);
    }
    printf("Scrivi m1: ");
    scanf("%d", &m1);
    while (m1 < 0 || m1 > 59) {
        printf("Valore non valido, reinserisci: ");
        scanf("%d", &m1);
    }
    printf("Scrivi h2: ");
    scanf("%d", &h2);
    while (h2 < 0 || h2 > 23) {
        printf("Valore non valido, reinserisci: ");
        scanf("%d", &h2);
    }
    printf("Scrivi m1: ");
    scanf("%d", &m2);
    while (m2 < 0 || m2 > 59) {
        printf("Valore non valido, reinserisci: ");
        scanf("%d", &m2);
    }

    if (h1 > h2 || (h1 == h2 && m1 > m2)) {
        printf("%d:%d > %d:%d", h1,m1,h2,m2);
    } else if (h1 == h2 && m1 == m2) {
        printf("%d:%d = %d:%d", h1,m1,h2,m2);
    } else {
        printf("%d:%d < %d:%d", h1,m1,h2,m2);
    }
}
```

2.1.26 Dato un numero di minuti e un numero di secondi rappresentare il conto alla rovescia.

Soluzione

```c
// 2.1.26 - Dato un numero di minuti e un numero di secondi rappresentare il
// conto alla rovescia.

#include <stdio.h>
#include <unistd.h>

int main() {
    int m, s;
    m = 1;
    s = 5;

    while (m > 0 || s > 0) {
        printf("%02d:%02d\n", m, s);
        s--;
        if (s < 0) {
            m--;
            s = 59;
        }
        sleep(1);
    }
```

```
}
```

2.1.27 Dato in input un numero intero n, stampa l'ennesimo numero della successione di fibonacci. La successione di fibonacci è quella successione di numeri in cui ogni numero è la somma dei due numeri precedenti. I primi due numeri sono 1 1. La successione inizia così: 1 1 2 3 5 8 13 21 …

Soluzione

```c
#include <stdio.h>

int main() {
    int prec1, prec2, ris, n, i;

    do {
        printf("Scrivi un numero >= 0: ");
        scanf("%d", &n);
    } while (n < 0);

    if (n == 0) {
        ris = 0; // Fib(0)
    } else {
        prec2 = 0;
        prec1 = 1;
        ris = 1; // che è sia Fib(1) che Fib(2)

        for (i = 2; i < n; i++) {
            prec2 = prec1;
            prec1 = ris;
            ris = prec1 + prec2;
        }
    }

    printf("Fib(%d) = %d", n, ris);
}
```

2.1.28 Dato in input un numero intero, conta da quante cifre è composto.

Soluzione

```c
// 2.1.28   Dato in input un numero intero, conta da quante cifre è composto.

#include <stdio.h>
#include <math.h>

int main() {
    int n, cifre, tmp;
    n = -235;
    cifre = 1;
    tmp = abs(n);
    while (tmp >= 10) {
        tmp /= 10;
        cifre++;
    }
    printf("%d e' composto da %d cifre.\n", n, cifre);
}
```

2.1.29 Dato in input un numero intero n di 3 cifre (in questo caso sarebbe utile mettere un controllo sull'input, cioè continuare a richiedere il numero fintanto che il numero dato non è di 3 cifre), stampa separatamente le sue cifre. Consiglio: usa divisioni per 10 e resti della divisione per 10. Puoi provare anche a generalizzare il programma in modo che funzioni anche con numeri di dimensione diversa da 3 cifre.

2.1.30 Dato in input un numero intero qualsiasi, stampa separatamente le sue cifre.

Soluzione

```c
// 2.1.30 - Dato in input un numero intero qualsiasi, stampa separatamente le sue cifre.

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
    int cifra, ncifre, tmp, potenza;
    int n = -234, i;
    ncifre = 1;
    tmp = abs(n);
    while (tmp >= 10) {
        tmp /= 10;
        ncifre++;
    }
    // printf("%d", ncifre);
    tmp = abs(n);
    while (ncifre > 0) {
        potenza = 1;
        for (i = 0; i < ncifre-1; i++) {
            potenza *= 10;
        }
        // printf("potenza %d\n", potenza);
        cifra = tmp / potenza;
        printf("%d ", cifra);
        tmp = tmp % potenza;
        ncifre--;
    }
}
```

2.1.31 Dato in input un numero intero n, stabilisci se è primo

Soluzione

```c
#include <stdio.h>

int main() {
    int n = 23, i, primo;

    primo = 1;
    for (i = 2; i <= n/2; i++) {
        if (n % i == 0) {
            primo = 0;
        }
    }

    if (primo) {
        printf("%d e' primo.", n);
    } else {
        printf("%d non e' primo.", n);
    }
}
```

2.1.32 Dato in input un numero intero n, stampa la sua scomposizione in fattori primi

Soluzione

```c
#include <stdio.h>

int main() {
    int n, val, div;

    do {
        printf("Inserisci un numero positivo: ");
        scanf("%d", &n);
    } while (n <= 0);

    val = n;
    div = 2;
    while (val >= div) {
        while (val % div == 0) {
            printf("%d ", div);
            val /= div;
        }
        div++;
    }
}
```

2.1.33 Data una sequenza di n numeri (n dato dall'utente) stabilire qual è il numero più grande, qual è il più piccolo e calcolare la media.

Soluzione

```c
// 2.1.33   Data una sequenza di n numeri (n dato dall'utente) stabilire qual è il
// numero più grande, qual è il più piccolo e calcolare la media.

#include <stdio.h>

int main() {
    int n, i;
    float media, numero, min, max;

    printf("Quanti numeri vuoi inserire? ");
    scanf("%d", &n);
    while (n < 1) {
        printf("Devi inserirne almeno 1, reinserisci:  ");
        scanf("%d", &n);
    }

    printf("Inserisci un numero: ");
    scanf("%f", &numero);
    min = numero;
    max = numero;
    media = numero;

    for (i = 1; i < n; i++) {
        printf("Inserisci un numero: ");
        scanf("%f", &numero);
        if (numero < min) min = numero;
        if (numero > max) max = numero;
        media += numero;
    }
    media /= n;
```

```
    printf("Minimo: %.2f\n", min);
    printf("Massimo: %.2f\n", max);
    printf("Media: %.2f\n", media);
}
```

2.1.34 Data una sequenza di n numeri (n dato dall'utente) stabilire qual è il secondo numero più grande, qual è il secondo più piccolo.

2.1.35 Data una sequenza di numeri positivi (fai il controllo sull'input) terminati da 0, scrivi qual è la maggior differenza tra due numeri dati consecutivamente

2.1.36 Data una sequenza di numeri positivi (fai il controllo sull'input) terminati da 0, scrivi la somma dei numeri divisibili per 3

2.1.37 Dato un numero n scrivi somma e prodotto di tutti i numeri minori o uguali a n.

2.1.38 Dato un numero n stabilire quante volte è possibile dividerlo per 2. (esempio 20 è divisibile per 2, 2 volte)

2.1.39 Dato un numero n, decidere se è primo

2.1.40 Data una sequenza di prezzi di prodotti, calcolare la spesa totale sapendo che se un prodotto costa meno di 100€ lo devo scontare del 10% altrimenti del 5%. Decidi tu il metodo per capire quando terminare la lettura della sequenza di prezzi.

2.1.41 Data una sequenza di 5 numeri che rappresentano i voti presi nelle diverse materie, stabilire se lo studente sarà promosso, bocciato o rimandato a settembre. Lo studente è promosso se non ha insufficienze, è bocciato se ha almeno 3 insufficienze, altrimenti è rimandato. Ricordati di controllare i valori dei voti in input che devono essere voti validi.

2.1.42 Leggi 3 parole e stampale tutte insieme in un'unica volta (solo un parallelogramma di scrittura).

2.1.43 Dato un numero n positivo stampa tutti i numeri da 1 a n, i primi n multipli di 2 (2 compreso che consideri il primo multiplo) e i primi n multipli di 3

2.1.44 Dati tre numeri positivi verificare che questi tre numeri siano una terna pitagorica (una terna pitagorica è un insieme di 3 numeri per cui la somma del quadrato di due numeri sia uguale al quadrato del terzo numero, in altre parole sono le lunghezze dei lati di un triangolo rettangolo)

2.1.45 Scrivere l'algoritmo per il pagamento della spesa che consiste nel chiedere inizialmente se si è in possesso della carta fedeltà, poi chiedere tutti i prezzi dei prodotti acquistati terminando la sequenza con un prezzo uguale a zero. Il programma deve sommare i prezzi, e se si è in possesso della carta, scontare del 10% i prezzi minori di 50 e del 5% i prezzi maggiori di 50

2.1.46 Un libro deve essere restituito in biblioteca dopo 15 giorni di prestito altrimenti si è multati di 0,80€ al giorno di ritardo. Ricevuto in ingresso il numero di giorni di un prestito, visualizza se il socio deve essere multato per il ritardo e a quanto ammonta la multa da pagare.

2.1.47 Calcolare il costo della bolletta telefonica sapendo che i primi 30 scatti costano 20 centesimi l'uno, gli scatti dal 31 al 100 costano 15 centesimi l'uno, mentre gli ulteriori scatti costano 10 centesimi l'uno. Aggiungere infine una tassa fissa di 2,50€ per le spese telefoniche. In input al programma è dato il numero di scatti effettuati.

2.1.48 Costruire uno schema di flusso che rappresenti l'algoritmo per il seguente gioco: il computer genera un numero segreto (usa la funzione random(100) che genera un numero casuale tra 0 e 99) e il giocatore deve individuarlo seguendo le indicazioni fornite dal computer (ti dice se il numero da trovare è più grande o più piccolo di quello che hai provato); una volta trovato il numero segreto, il numero di tentativi effettuati è visualizzato a video.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
    int segreto, tentativo;

    srand(time(NULL));

    segreto = rand() % 100;

    while (1) {
        printf("Indovina il numero (da 0 a 99): ");
        scanf("%d", &tentativo);

        if (tentativo < segreto) {
            printf("Il numero da trovare e' piu' grande.\n");
        } else if (tentativo > segreto){
            printf("Il numero da trovare e' piu' piccolo.\n");
        } else {
            printf("Hai indovinato!\n");
            break;
        }
    }
}
```

2.1.49 Calcolare il quoziente e il resto della divisione intera di due numeri interi positivi forniti in ingresso chiamati dividendo e divisore, applicando il metodo delle sottrazioni successive. Per esempio, se dividendo=13 e divisore=5, il programma dovrà restituire Quoziente=2 e Resto=3, calcolati sottraendo successivamente il valore di divisore dal valore di dividendo

Soluzione

```c
#include <stdio.h>

int main() {
    int num, den, quoz, resto;

    num = 13;
    den = 3;

    for (resto = num, quoz = 0; resto >= den; quoz++) {
        resto -= den;
    }

    resto = num;
    quoz = 0;
    while (resto >= den) {
        resto -= den;
        quoz += 1;
    }
```

```
    printf("%d : %d = %d resto %d\n", num, den, quoz, resto);
}
```

2.1.50 Visualizza i termini della successione di Fibonacci compresi nell'intervallo tra due interi positivi N1 e N2, entrambi forniti in ingresso, con N2>N1 (controlla bene gli input)

Soluzione

```
#include <stdio.h>

int main() {
    int n1, n2, i, prec1, prec2, fib;

    do {
        printf("Inserisci un numero positivo: ");
        scanf("%d", &n1);
    } while (n1 < 1);
    do {
        printf("Inserisci un numero positivo maggiore di %d: ", n1);
        scanf("%d", &n2);
    } while (n2 <= n1);

    prec2 = 1;
    prec1 = 1;
    fib = 2;

    if (n1 == 1) {
        printf("1 1 ");
    }

    for (i = n1; fib <= n2; i++) {
        if (fib >= n1 && fib <= n2) {
            printf("%d ", fib);
        }

        fib = prec1 + prec2;
        prec2 = prec1;
        prec1 = fib;
    }
}
```

2.1.51 Simula il lancio di un dado per N volte (con N intero e positivo in ingresso e usando la funzione random(7) per il lancio) e verifica che effettivamente la probabilità che esca 6 è 1/6. Per farlo fai tanti tiri e vedi se esce 6 un sesto delle volte (più tiri fai più dovresti avere un risultato positivo).

Soluzione

```
int main() {
    int n, dado, i;
    float prob;

    srand(time(NULL));

    n = 100000;
    prob = 0;
    for (i = 0; i < n; i++) {
        dado = rand() % 6 + 1;
        if (dado == 6) {
            prob += 1;
        }
```

```
    }

    prob /= n;

    printf("La probabilita' che esca 6 e' %.3f%%.\n", prob*100);
    printf("1/6=%.5f", 1.0/6.0);
}
```

2.1.52 Simula il lancio di due dadi e verifica le probabilità che escano: una coppia di 6, un valore totale uguale a 7 (due diverse probabilità)

Soluzione

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int nlanci, dado1, dado2, i;
    float prob1, prob2;

    srand(time(NULL));

    prob1 = 0;
    prob2 = 0;
    nlanci = 100000;

    for (i = 0; i < nlanci; i++) {
        dado1 = rand() % 6 + 1;
        dado2 = rand() % 6 + 1;

        if (dado1 == 6 && dado2 == 6) {
            prob1 += 1;
        }
        if (dado1 + dado2 == 7) {
            prob2 += 1;
        }
    }

    prob1 /= nlanci;
    prob2 /= nlanci;

    printf("Probabilita che esca coppia di 6: %.2f%%\n", prob1*100);
    printf("Probabilita che esca somma 7: %.2f%%\n", prob2*100);
}
```

2.1.53 Dati in ingresso due numeri positivi x e y, visualizza in ordine decrescente la sequenza di numeri interi compresi tra x e y che sono divisibili per il minore tra x e y. Ad esempio, se x = 7 e y = 35, la sequenza è 35 28 21 14 7.

Soluzione

```
#include <stdio.h>

int main() {
    int x, y, i, min, max;

    do {
        printf("Inserisci il primo numero, deve essere positivo: ");
        scanf("%d", &x);
    } while (x <= 0);
    do {
```

```
        printf("Inserisci il secondo numero, deve essere positivo: ");
        scanf("%d", &y);
    } while (y <= 0);

    if (x < y) {
        min = x;
        max = y;
    } else {
        min = y;
        max = x;
    }

    for (i = max; i >= min; i--) {
        if (i % min == 0) {
            printf("%d ", i);
        }
    }
}
```

2.1.54 Simula una serie di lanci di un dado a 6 facce (d6) e un dado da 20 facce (d20). Calcola il valore medio che si ottiene lanciando il d6 e quello ottenuto lanciando il d20.

2.1.55 Valuta se è più probabile ottenere 7 oppure ottenere 8 lanciando 2 dadi da 6 e sommando i 2 valori ottenuti.

2.1.56 Voglio valutare la probabilità che tirando due dadi da 6 esca il valore 2. Il valore calcolato deve essere abbastanza preciso e decido di calcolarlo misurando la media di tanti tiri di dado e fermandomi solo quando vedo che facendo uscire un nuovo 2 valore calcolato non varia di più di 0,01%

Soluzione

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int tiro, ntiri = 0, ndue = 0;
    float prob = 0, prec = 0, diff;
    float soglia = 0.0000001;

    srand(time(NULL));


    while (1) {
        tiro = rand() % 6 + 1 + rand() % 6 + 1; // rand() % (max-min+1) + min
        ntiri++;

        prec = prob;
        if (tiro == 2) {
            ndue++;
            prob = (float)ndue/(float)ntiri;

            diff = prob-prec;
            // if (diff < 0) diff * -1;
            // printf("%f - %f = %f\n", prob, prec, diff);
            if (diff < soglia && diff > -soglia) {
                break;
            }
        }
        printf("%d - %2d: %f\n", ntiri, tiro, prob);
    }
```

```
    printf("%f", prob);
}
```

2.1.57 Vengono dati in input i valori delle altezze di n persone (n chiesto all'utente). Per ogni persona oltre all'altezza viene indicato anche il sesso. Deve essere stampato il valore medio di altezza separatamente per i maschi e per le femmine. L'utente può decidere di non indicare né maschio né femmina, in quel caso non contare quella persona.

Soluzione

```
#include <stdio.h>

int main() {
    int n, i, altezza, numm, numf;
    float mediam, mediaf;
    char sesso;

    printf("Quanti valori vuoi inserire? ");
    scanf("%d", &n);

    mediam = 0;
    mediaf = 0;
    numm = 0;
    numf = 0;
    for (i = 0; i < n; i++) {
        printf("Scrivi l'altezza della persona n %d: ", i+1);
        scanf("%d", &altezza);
        printf("Scrivi se e' maschio o femmina (m maschio, f femmina altrimenti non considero il
sesso)");
        scanf(" %c", &sesso);

        if (sesso == 'm') {
            mediam += altezza;
            numm += 1;
        } else {
            mediaf += altezza;
            numf += 1;
        }
    }

    mediam /= numm;
    mediaf /= numf;

    printf("L'altezza media dei maschi e' %.1f.\n", mediam);
    printf("L'altezza media delle femmine e' %.1f.\n", mediaf);
}
```

2.1.58 Scrivi il codice di un gioco per 2 persone basato sul lancio di dadi. All'inizio del gioco si decide che tipo di dado usare, si possono scegliere i dadi da 4, 6, 8 o 20 facce. Il gioco consiste nel tirare uno per volta il dado e sommando ogni volta i numeri usciti finchè uno dei due giocatori non supera un valore limite che è 4 volte il valore massimo rappresentato sul dado scelto (ad esempio se si sceglie il dado da 6 il limite è 24). Per rendere il gioco equo si può fare in modo che i due giocatori debbano tirare per forza lo stesso numero di tiri e che entrambi i giocatori possano superare il limite facendo finire il gioco in parità

Variante: Scrivi una variante del gioco in cui l'obiettivo è avvicinarsi il più possibile al valore limite senza superarlo. Chi si avvicina di più vince. In ogni momento un giocatore può decidere di accontentarsi del valore ottenuto e smettere di tirare.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
    int valdado, limite, g1, g2, lancio;

    srand(time(NULL));

    do {
        printf("Che dado vuoi usare? (4, 6, 8, 20):  ");
        scanf("%d", &valdado);
    } while (valdado != 4 && valdado != 6 && valdado != 8 && valdado != 20);

    limite = 4*valdado;

    g1 = 0;
    g2 = 0;
    while (g1 <= limite && g2 <= limite) {
        lancio = rand() % valdado + 1;
        g1 += lancio;
        printf("Il giocatore 1 lancia %d e arriva a %d.\n", lancio, g1);
        system("pause");

        lancio = rand() % valdado + 1;
        g2 += lancio;
        printf("Il giocatore 2 lancia %d e arriva a %d.\n", lancio, g2);
        system("pause");
    }

    if (g1 > limite && g2 > limite) {
        printf("Pareggio.\n");
    } else if (g1 > limite) {
        printf("Vince il giocatore 2.\n");
    } else {
        printf("Vince il giocatore 1.\n");
    }

}
```

2.1.59 Leggi un numero positivo e controlla se esso è divisibile per 2 o divide 3 e non è compreso tra 10 e 20

Soluzione

```c
#include <stdio.h>

int main() {
    int n;

    do {
        printf("Scrivi un numero positivo: ");
        scanf("%d", &n);
    } while (n <= 0);

    if ((n % 2 == 0 || 30 % n == 0) && !(n >= 10 && n <= 20)) {
        printf("Si");
    } else {
        printf("No");
    }
}
```

2.1.60 Leggi un numero positivo e controlla se esso è divisibile per 2 o è un multiplo di 3 ed è compreso tra 10 e 20

Soluzione

```c
#include <stdio.h>

int main() {
    int n;
    do {
        printf("Scrivi un numero positivo: ");
        scanf("%d", &n);
    } while (n <= 0);

    if ((n % 2 == 0 || n % 3 == 0) && (n >= 10 && n <= 20)) {
        printf("Si");
    } else {
        printf("No");
    }
}
```

2.1.61 dato un numero n stampa tutti i numeri da 1 a n che sono divisibili da 2 o 3 o 5 ma non sono divisibili per 4

2.1.62 Dato un numero positivo n disegna le figure geometriche riportate di seguito:

per n = 5

```
xxxxx        xxxxx        x
xxxx         xxxxx        xx
xxx          xxxxx        xxx
xx           xxxxx        xxxx
x            xxxxx        xxxxx
```

Soluzione

```c
#include <stdio.h>

int main() {
    int n = 5, i, j;

    printf("Triangolo 1\n");
    // xxxxx
    // xxxx
    // xxx
    // xx
    // x
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i; j++) {
            printf("x");
        }
        printf("\n");
    }


    printf("\nQuadrato\n");
    // xxxxx
    // xxxxx
    // xxxxx
    // xxxxx
    // xxxxx
    for (i = 0; i < n; i++) {
```

```
        for (j = 0; j < n; j++) {
            printf("x");
        }
        printf("\n");
    }


    printf("\nTriangolo 2\n");
    // x
    // xx
    // xxx
    // xxxx
    // xxxxx
    for (i = 0; i < n; i++) {
        for (j = 0; j < i + 1; j++) {
            printf("x");
        }
        printf("\n");
    }
}
```

2.1.63 Esegui un programma che abbia il seguente output (chiaramente usando dei cicli):

```
1 2 3 4 5 6
2 3 4 5 6 +
3 4 5 6 + +
4 5 6 + + +
5 6 + + + +
6 + + + + +
```

Soluzione

```
#include <stdio.h>

int main() {
    int i, j, n = 6, val;

    printf("Modo 1\n");
    for (i = 0; i < n; i++) {
        for (j = 1 + i; j < 1 + i + n; j++) {
            if (j <= n) {
                printf("%d ", j);
            } else {
                printf("+ ");
            }
        }
        printf("\n");
    }

    printf("\nModo 2\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            val = 1 + i + j;
            if (val <= n) {
                printf("%d ", val);
            } else {
                printf("+ ");
            }
        }
        printf("\n");
    }
}
```

2.1.64 Simula l'esecuzione di un gioco di dadi in cui due persone si scontrano lanciando ognuno un dado. Il gioco è diviso in round. Al primo round i due giocatori lanciano ognuno un dado da venti facce e vince chi fa il numero più alto, in caso di pareggio ritirano i dadi finchè uno dei due non vince. Chi perde il round deve affrontare i round successivi con un dado con una faccia in meno. Perde la partita chi arriva ad avere un dado con 0 facce. Mostra ad ogni round quante facce hanno i dadi dei due giocatori e mostra il risultato di ogni lancio dei dadi. Dichiara infine il vincitore.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
    int dado1, dado2, tiro1, tiro2;

    srand(time(NULL));

    dado1 = 20;
    dado2 = 20;
    while (dado1 > 0 && dado2 > 0) {
        tiro1 = rand() % dado1 + 1;
        tiro2 = rand() % dado2 + 1;

        printf("giocatore 1: %2d,  giocatore 2: %2d - ", tiro1, tiro2);
        if (tiro1 > tiro2) {
            dado2--;
            printf("vince giocatore 1\n");
        } else if (tiro1 < tiro2) {
            dado1--;
            printf("vice giocatore 2\n");
        } else {
            printf("pareggio\n");
        }
    }
    if (dado1 == 0) {
        printf("\nVince il giocatore 2!\n");
    } else {
        printf("\nVince il giocatore 1!\n");
    }

}
```

2.1.65 Scrivi un programma che dato un numero n stampi una piramide come nel seguente esempio:

```
Con n = 3
  1
 121
12321
```

Soluzione

```c
#include <stdio.h>

int main() {
    int n, i, j, val;

    n = 10;
    for (i = 0; i < n; i++) {
```

```
        for (j = 0; j < n; j++) {
            val = -n+i+j+2;
            if (val > 0) {
                printf("%2d ", val);
            } else {
                printf("   ");
            }
        }
        for (j = n-2; j >= 0; j--) {
            val = -n+i+j+2;
            if (val > 0) {
                printf("%2d ", val);
            } else {
                printf("   ");
            }
        }
        printf("\n");
    }
}
```

2.1.66 Scrivere un programma che legga due interi n e m con valori compresi tra 1 e 9, i cui prodotto sia inferiore a 35, e stampi m piramidi di altezza n. L'esempio si riferisce al caso n=3 e m=4.

```
  1    1    1    1
 121  121  121  121
12321123211232112321
```

Soluzione

```
#include <stdio.h>

int main() {
    int n, i, j, k, m, val;

    n = 3;
    m = 4;
    for (i = 0; i < n; i++) {
        for (k = 0; k < m; k++) {
            for (j = 0; j < n; j++) {
                val = -n + i + j + 2;
                if (val > 0) {
                    printf("%2d ", val);
                } else {
                    printf("   ");
                }
            }
            for (j = n - 2; j >= 0; j--) {
                val = -n + i + j + 2;
                if (val > 0) {
                    printf("%2d ", val);
                } else {
                    printf("   ");
                }
            }
        }
        printf("\n");
    }
}
```

2.1.67 Scrivi un programma che permetta di controllare i dati di input immessi dall'utente:

se l'utente inserisce un intero N compreso tra 1 e 10, il programma deve stampare a video il valore N^N,

se l'intero N e' compreso tra 11 e 20, il programma deve stampare a video la somma 1 + 2 + 3 +… + N

altrimenti deve dare un segnale di errore.

Soluzione

```c
#include <stdio.h>

int main() {
    long long int n, i, ris;

    printf("Scrivi un numero: ");
    scanf("%lld", &n);

    if (n >= 1 && n <= 10) {
        for (i = 0, ris = 1; i < n; i++) {
            ris *= n;
        }
        printf("%lld^%lld=%lld", n, n, ris);
    }
    else if (n >= 11 && n <= 20) {
        for (i = 1, ris = 0; i <= n; i++) {
            ris += i;
        }
        printf("sommatoria da 1 a %lld = %lld", n, ris);
    } else {
        printf("Valore non valido.");
    }
}
```

2.1.68 Si realizzi un programma che legga un intero N da tastiera, e stampi a video il risultato della seguente sommatoria:

$$\sum_{i=0}^{N} \left[(-1)^i \frac{4}{2 * i + 1} \right]$$

Una volta calcolato e stampato il valore a video, il programma deve chiedere un nuovo numero all'utente e ripetere il calcolo. Il programma deve terminare solo qualora l'utente inserisca un valore negativo.

Soluzione

```c
#include <stdio.h>

int main() {
    int i, n;
    float ris, x;

    do {
        printf("Scrivi n (negativo per terminare): ");
        scanf("%d", &n);

        ris = 0;
        for (i = 0; i <= n; i++) {
            x = 4.0 / (2 * i + 1);
            if (i % 2 != 0) {
                x *= -1;
```

```
            }
            ris += x;
        }

        printf("%f\n", ris);
    } while (n >= 0);
}
```

2.1.69 Si progetti in C un programma che legge un float, rappresentante un ammontare di euro; di seguito il programma deve leggere un tasso d'interesse (in percentuale), ed un numero di anni. Il programma deve stampare, in uscita, per ogni anno, come l'ammontare cresce con gli interessi. Si ricordi che l'interesse si calcola con la seguente formula:

$$C_{fin} = C_{in} (1 + (r/100))^N$$

dove C_{fin} è il capitale finale, C_{in} è quello iniziale, r è l'interesse, e N rappresenta il numero di anni in cui si applicano gli interessi.

Ad esempio supponiamo che il capitale iniziale sia di 1000.0 €, con un tasso del 3%, per un periodo di 3 anni. L'output stampato deve avere all'incirca questo aspetto:

```
Capitale iniziale: 1000.00€
Dopo 1 anno: 1030.00 €
Dopo 2 anni: 1060.90 €
Dopo 3 anni: 1092.73 €
```

Soluzione

```
#include <stdio.h>
#include <math.h>

int main() {
    float c, r, cf;
    int n, i;
    c = 1000.0;
    r = 3;
    n = 10;

    printf("Capitale iniziale : %.2f e\n", c);
    cf = c;
    for (i = 1; i <= n; i++) {
        cf =  cf * ( 1 + (r/100));
        printf("Dopo %d anni: %.2f e\n", i, cf);
    }


    printf("\n\nCapitale iniziale : %.2f e\n", c);
    for (i = 1; i <= n; i++) {
        cf =  c * pow(1 + (r/100), i);
        printf("Dopo %d anni: %.2f e\n", i, cf);
    }

}
```

2.1.70 Realizzare un programma che legga da input un carattere dell'alfabeto e stampi a video il carattere stesso ed il suo valore ASCII. Il programma deve controllare che il carattere inserito sia compreso tra 'a' e 'z' o tra 'A' e 'Z' (in caso contrario si stampi un messaggio di errore). Dopo la stampa, il programma deve continuare a chiedere nuovi caratteri, finché l'utente non inserisce il carattere corrispondente al numero zero ('0'): in tal caso il programma termina.

Soluzione

```c
#include <stdio.h>

int main() {
    char c;


    do {
        printf("Scrivi un carattere: ");
        scanf(" %c", &c);

        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {
            printf("%d\n\n", c);
        } else {
            printf("'%c' non e' una lettera.\n\n", c);
        }
    } while (c != '0');

}
```

2.1.71 Realizzare un programma che prende in input una sequenza di caratteri ‘0’ e ‘1’ e conta la lunghezza della più lunga sotto-sequenza di ‘0’ di fila L’inserimento della sequenza termina quando si inserisce un carattere diverso da ‘0’ e ‘1’ A quel punto, si stampa a video il valore trovato.

Soluzione

```c
#include <stdio.h>

int main() {
    char car;
    int lung, max;

    lung = 0;
    max = 0;
    do {
        scanf(" %c", &car);
        if (car == '0'){
            lung++;
        } else {
            if (lung > max) {
                max = lung;
            }
            lung = 0;
        }
    } while (car == '0' || car == '1');

    printf("\n\n%d", max);
}
```

2.1.72 Realizzare un programma che prenda in input una sequenza di cifre (tra 1 e 9) e calcoli la somma massima fra le sottosequenze di cifre non decrescenti Il programma termina quando viene inserito lo 0

Esempio:

2	2	4	5	3	9	3	1	5	0
13				12		3	6		

Soluzione

```c
#include <stdio.h>

int main() {
    int n, prec, somma, sommamax;

    prec = 10;
    sommamax = 0;
    somma = 0;
    do {
        printf("Scrivi un numero tra 1 e 9: ");
        scanf("%d", &n);
        while (n < 0 || n > 9) {
            printf("Errore! Scrivi un numero tra 1 e 9: ");
            scanf("%d", &n);
        }

        if (prec <= n) {
            somma += n;
        } else {
            somma = n;
        }
        if (somma > sommamax) {
            sommamax = somma;
        }
        prec = n;
    } while (n != 0);

    printf("%d", sommamax);
}
```

2.1.73 Data in input una sequenza di n numeri (n dato dall'utente), stampare per ognuno il numero di quadrati perfetti minori di quel numero.

Soluzione

```c
#include <stdio.h>

int main() {
    int n, numero, i, j, nquadrati;

    do {
        printf("Quanti numeri vuoi inserire: ");
        scanf("%d", &n);
    } while (n < 1);

    for (i = 0; i < n; i++) {
        printf("inserisci il %d numero: ", i+1);
        scanf("%d", &numero);
        nquadrati = 0;
        for (j = 1; j*j <= numero; j++) {
            nquadrati++;
        }
        printf("esistono %d quadrati perfetti minori o ugali a %d.\n", nquadrati, numero);
    }
}
```

2.1.74 Conta quanti tiri di moneta devi fare prima di vedere uscire testa per 10 volte di fila.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
    int ntiri, tiro, cont;

    srand(time(NULL));

    for(ntiri = 0, cont = 0; cont < 10; ntiri++) {
        tiro = rand() % 2;
        if (tiro == 1) {
            cont += 1;
        } else {
            cont = 0;
        }
    }

    printf("%d", ntiri);
}
```

2.1.75 Conta quanti tiri di un dado da 6 devi fare prima di vedere uscire lo stesso numero 5 volte di fila.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
    int ntiri, tiro, cont, prec;

    srand(time(NULL));

    prec = 0;
    for(ntiri = 0, cont = 0; cont < 5; ntiri++) {
        tiro = rand() % 6 + 1;
        if (tiro == prec) {
            cont += 1;
        } else {
            cont = 0;
        }
        prec = tiro;
    }

    printf("%d", ntiri);
}
```

2.1.76 Scrivi un programma che deve indovinare un numero scelto dall'utente (che se lo tiene per sé e non lo dice al computer). Il computer continua a proporre numeri e l'utente deve rispondere se il numero segreto è maggiore o minore di quello proposto.

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```c
int main() {
    int n, risp;
    int min = 0, max = 1000;
    int ntentativi;

    printf("Pensa ad un numero tra %d e %d, ora cercherò di indovinarlo.\n", min, max);


    for (ntentativi = 0; 1; ntentativi++) {
        n = (max+min) / 2;
        printf("Il numero e' %d? (0 no, 1 si): ", n);
        scanf("%d", &risp);
        if (risp) {
            printf("Ho indovinato in %d mosse.\n", ntentativi);
            break;
        } else {
            printf("Il numero che hai pensato e' maggiore o minore di %d? (0 minore, 1 maggiore):
", n);
            scanf("%d", &risp);
            if (risp) {
                min = n+1;
            } else {
                max = n-1;
            }
        }
    }
}
```

2.1.77 Genera una sequenza di 10 numeri interi casuali e calcolane la media.

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int n = 10, numero, i, min, max;
    float media;

    srand(time(NULL));

    min = 8;
    max = 12;
    media = 0;
    for (i = 0; i < n; i++) {
        numero = rand() % (max-min+1) + min;
        printf("%d ", numero);
        media += numero;
    }
    media /= n;
    printf("\n\n%.2f", media);
}
```

2.1.78 Tira due dadi da 12 fino a che non escono due numeri uguali. Calcola la media dei tiri fatti (sommi sempre i due dadi)

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <time.h>

int main() {
    int d1, d2, c, min, max;
    float media;

    srand(time(NULL));

    media = 0;
    c = 0;
    while (1) {
        d1 = rand() % 12 + 1;
        d2 = rand() % 12 + 1;
        printf("%d %d = %d\n", d1, d2, d1+d2);
        if (d1 == d2) {
            break;
        }
        media += (d1+d2);
        c++;
    }
    media /= c;
    printf("\n\n%.2f", media);
}
```

2.1.79 Leggi due numeri e poi comunica se questi numeri hanno lo stesso segno o se la loro somma è negativa e contemporaneamente sono entrambi minori di 10.

Soluzione

```c
#include <stdio.h>

int main() {
    int a, b;

    printf("Scrivi due numeri interi qualsiasi: ");
    scanf("%d %d", &a, &b);

    if (a*b > 0 || (a+b < 0 && a < 10 && b < 10)){
        printf("Si");
    } else {
        printf("No");
    }
}
```

3. Array, stringhe e matrici

3.1 Array numerici

3.1.1 Riempi un array di 6 elementi nel modo che preferisci, poi:

1. Stampa l'array
2. Stampa l'array al contrario

3.1.2 Leggi n voti, caricali in un array, calcolane la media, poi stampa tutti i numeri e la media.

1. Variante: calcola e stampa la media dei soli voti sufficienti.

Soluzione

```c
#include <stdio.h>

#define N 100

int main() {
    int voti[N];
    int nvoti, i;
    float media;

    // lettura del numero di voti
    printf("Inserisci il numero di voti: ");
    scanf("%d", &nvoti);
    while (nvoti < 1) {
        printf("Ci deve essere almeno un voto.");
        printf("Inserisci il numero di voti: ");
        scanf("%d", &nvoti);
    }

    // lettura dei voti
    for (i = 0; i < nvoti; i++) {
        printf("Inserisci il %d voto: ", i+1);
        scanf("%d", &voti[i]);
        while (voti[i] < 1 || voti[i] > 10) {
            printf("Il voto non e' valido, deve essere compreso tra 1 e 10.");
            printf("Inserisci il %d voto: ", i+1);
            scanf("%d", &voti[i]);
        }
    }

    // calcolo la media (potevo farlo insieme alla lettura, così è più leggibile e modulare)
    media = 0;
    for (i = 0; i < nvoti; i++) {
        media += voti[i];
    }
    media /= nvoti;

    // stampe
    printf("\nElenco dei voti: ");
    for (i = 0; i < nvoti; i++) {
        printf("%d ", voti[i]);
    }
    printf("\nMedia: %.2f", media);
}
```

3.1.3 Carica un array di numeri casuali, chiedi poi all'utente un numero e digli se questo numero è presente nell'array.

Variante: stampa anche in che posizione si trova il numero trovato. Se il programma non trova il numero stampa una volta sola che il numero non è stato trovato.

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main() {
    int a[N];
    int num, i, trovato;
```

```
    srand(time(NULL));

    // riempimento
    for (i = 0; i < N; i++) {
        a[i] = rand() % 100;
    }

    // stampa
    for (i = 0; i < N; i++) {
        printf("%d ", a[i]);
    }

    printf("\n\nInserisci un numero da cercare: ");
    scanf("%d", &num);

    trovato = 0;
    for (i = 0; i < N; i++) {
        if (a[i] == num) {
            printf("Numero trovato in posizione %d\n", i);
            trovato = 1;
        }
    }
    if (!trovato) {
        printf("Numero non trovato \n");
    }
}
```

3.1.4 Carica un array, di dimensione a piacere, di numeri casuali compresi in un range di numeri deciso dall'utente. Esegui poi le seguenti operazioni:

1. inverti l'ordine dei numeri contenuti nell'array,
2. Date due posizioni a e b dell'array (chiaramente controlla che a sia minore di b e che siano compresi tra 0 e N-1) stampa prima i numeri dell'array dalla posizione a alla posizione b poi i numeri dalla posizione b alla posizione a
3. Dato un numero positivo x qualsiasi stampa tutti i numeri dell'array dalla posizione 0 alla posizione x. Se x > N-1, dopo aver stampato tutto l'array ricomincia a stampare dalla posizione 0 e continua fino ad aver stampato x+1 numeri.

Soluzione

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define N 10

int main() {
    int arr[N];
    int i, tmp;
    int min = 5;
    int max = 22;


    // Carica un array, di dimensione a piacere, di numeri casuali compresi
    // in un range di numeri deciso dall'utente

    srand(time(NULL));

    // riempimento
    for (i = 0; i < N; i++) {
        arr[i] = rand() % (max-min+1) + min;
```

```
    }

    // stampa
    for (i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");


    // 1. inverti l'ordine dei numeri contenuti nell'array,
    for (i = 0; i < N/2; i++) {
        tmp = arr[i];
        arr[i] = arr[N-1-i];
        arr[N-1-i] = tmp;
    }

    // stampa
    for (i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");


    // 2. Date due posizioni a e b dell'array (chiaramente controlla che a sia
    // minore di b e che siano compresi tra 0 e N-1) stampa prima i numeri
    // dell'array dalla posizione a alla posizione b poi i numeri dalla posizione b
    // alla posizione a

    int a = 2;
    int b = 7;

    for (i = a; i <= b; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    for (i = b; i >= a; i--) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // 3.   Dato un numero positivo x qualsiasi stampa tutti i
    // numeri dell'array dalla posizione 0 alla posizione x. Se x > N-1, dopo aver
    // stampato tutto l'array ricomincia a stampare dalla posizione 0 e continua
    // fino ad aver stampato x+1 numeri.
    int x = 22;
    for (i = 0; i <= x; i++) {
        printf("%d ", arr[i%N]);
    }
    printf("\n");
}
```

3.1.5 Dato un array di numeri interi casuali, calcola e stampa il valore massimo e il valore minimo contenuti nell'array

Soluzione

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 10

int main() {
    int arr[N];
    int max, min, i;
```

```
    srand(time(NULL));

    for (i = 0; i < N; i++) {
        arr[i] = rand() % 100;
    }
    min = arr[0];
    max = arr[0];
    for (i = 1; i < N; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    printf("Array: ");
    for (i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\nMin: %d\n", min);
    printf("Max: %d\n", max);
}
```

3.1.6 Dato un array di numeri interi casuali, stampa tutti i numeri superiori alla media partendo dal fondo

Soluzione

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 10

int main() {
    int arr[N];
    int max, min, i;
    float media;

    srand(time(NULL));

    for (i = 0; i < N; i++) {
        arr[i] = rand() % 100;
        printf("%d ", arr[i]);
    }
    printf("\n");

    media = 0;
    for (i = 0; i < N; i++) {
        media += arr[i];
    }
    media /= N;
    printf("%.2f\n", media);

    for (i = N-1; i >= 0; i--) {
        if (arr[i] > media) {
            printf("%d ", arr[i]);
        }
    }

}
```

3.1.7 Scrivi un programma che esegua le seguenti operazioni:

1. Chiedere con un ciclo all'utente di inserire i voti presi in matematica ed inserirli in un array, uscire dal ciclo quando l'utente inserisce un valore negativo;
2. Calcola la media di tutti i voti escludendo il più basso e il più alto.

Soluzione

```c
#include <stdio.h>

#define N 100

int main() {
    float voti[N], media;
    int nvoti, imax, imin, i;

    nvoti = 0;
    while (1) {
        printf("Scrivi il %d voto: ", nvoti+1);
        scanf("%f", &voti[nvoti]);
        while (voti[nvoti] < 0 || voti[nvoti] > 10) {
            printf("Voto non valido, deve essere compreso tra 1 e 10.\n");
            printf("Scrivi il %d voto: ", nvoti+1);
            scanf("%f", &voti[nvoti]);
        }
        if (voti[nvoti] == 0) {
            break;
        } else {
            nvoti++;
        }
    }
    if (nvoti < 3) {
        printf("Non ha senso calcolare la media di 0 voti.");
    } else {
        imax = 0;
        imin = 0;
        for (i = 1; i < nvoti; i++) {
            if (voti[i] > voti[imax]) {
                imax = i;
            }
            if (voti[i] < voti[imin]) {
                imin = i;
            }
        }

        media = 0;
        for (i = 0; i < nvoti; i++) {
            if (i != imin && i != imax) {
                media += voti[i];
            }
        }
        media /= (nvoti-2);

        printf("La media vale %.2f\n", media);
    }

}
```

3.1.8 Riempi un array di numeri casuali poi rappresenta l'array come un istogramma.

```
Ad es.
1: x
0:
4: xxxx
```

```
6: xxxxxx
3: xxx
9: xxxxxxxxx
```

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 6

int main() {
    int arr[N];
    int i, j;

    srand(time(NULL));

    for (i = 0; i < N; i++) {
        arr[i] = rand() % 10;
    }
    for (i = 0; i < N; i++) {
        printf("%2d: ", arr[i]);
        for (j = 0; j < arr[i]; j++) {
            printf("x");
        }
        printf("\n");
    }
}
```

3.1.9 Scrivi un programma che ordini un array con l'algoritmo selection sort, prima in ordine crescente poi in ordine decrescente.

Soluzione

```c
#include <stdio.h>

#define N 6

void stampaArray(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
}

void ordinaArrayCrescente(int a[], int n) {
    int i, j, jmin, tmp;
    for (i = 0; i < N-1; i++) {
        jmin = i;
        for (j = i+1; j < N; j++) {
            if (a[j] < a[jmin]) {
                jmin = j;
            }
        }
        tmp = a[i];
        a[i] = a[jmin];
        a[jmin] = tmp;
    }
}

void ordinaArrayDecrescente(int a[], int n) {
    int i, j, jmax, tmp;
    for (i = 0; i < N-1; i++) {
```

```
        jmax = i;
        for (j = i+1; j < N; j++) {
            if (a[j] > a[jmax]) {
                jmax = j;
            }
        }
        tmp = a[i];
        a[i] = a[jmax];
        a[jmax] = tmp;
    }
}

int main() {
    int arr[N] = {4,7,1,9,4,6};

    stampaArray(arr, N);
    printf("\n");

    // ordinamento (selection sort)
    ordinaArrayCrescente(arr, N);
    ordinaArrayDecrescente(arr, N);

    stampaArray(arr, N);
    printf("\n");

}
```

3.1.10 Scrivi un programma che ordini un array con l'algoritmo bubble sort, prima in ordine crescente poi in ordine decrescente.

Soluzione

```
// 3.1.10 -  Scrivi un programma che ordini un array con l'algoritmo bubble sort,
// prima in ordine crescente poi in ordine decrescente.
// (prova a cercare online l'algoritmo).

#include <stdio.h>

void stampaArray(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

void bubbleSortCresente(int a[], int n) {
    int i, j, tmp;

    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-1; j++) {
            if (a[j] > a[j+1]) {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }

}

void bubbleSortDecresente(int a[], int n) {
    int i, j, tmp;

    for (i = 0; i < n-1; i++) {
```

```
        for (j = 0; j < n-1; j++) {
            if (a[j] < a[j+1]) {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }
}

int main() {
    int arr[] = {1,7,3,9,5,6};
    int n = 6;

    bubbleSortCresente(arr,n);
    stampaArray(arr,n);
    bubbleSortDecresente(arr,n);
    stampaArray(arr,n);
}
```

3.1.11 Scrivi un programma che dato un array di dimensione n e due valori x y tali che x<=y<=n, ordini l'array dalla posizione x alla posizione y comprese.

Soluzione

```
// 3.1.11 - Scrivi un programma che dato un
// array di dimensione n e due valori x y tali che x<=y<=n, ordini l'array
// dalla posizione x alla posizione y comprese. Usa l'algoritmo che preferisci.

#include <stdio.h>

void stampaArray(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

void ordinaxy(int a[], int n, int x, int y) {
    int i, j, tmp;

    for (i = x; i < y; i++) {
        for (j = x; j < y; j++) {
            if (a[j] > a[j+1]) {
                tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }
}

int main() {
    int arr[] = {1,7,3,9,5,6,4,4,8};
    int n = 9;

    stampaArray(arr,n);
    ordinaxy(arr, n, 2, 6);
    stampaArray(arr,n);
    // bubbleSortDecresente(arr,n);
    // stampaArray(arr,n);
}
```

3.1.12 Riempi un array di numeri casuali tra 0 e 9 e calcola la somma massima fra le sottosequenze di cifre non decrescenti. Il valore deve essere calcolato da un'apposita funzione.

Esempio:

2	2	4	5	3	9	3	1	5	0
13				12		3	6		

la prima riga rappresenta l'array e la seconda i vari valori calcolati (viene restituito 13 che è il più grande)

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 10

int main() {
    // int A[N] = {2,2,4,5,3,9,3,1,5,0};
    int A[N];
    int i, somma, sommamax;

    srand(time(NULL));

    for (i = 0; i < N; i++) {
        A[i] = rand() % 10;
        printf("%d ", A[i]);
    }
    printf("\n");

    somma = A[0];
    sommamax = A[0];
    for (i = 1; i < N; i++) {
        if (A[i] >= A[i-1]) {
            somma += A[i];
            if (somma > sommamax) {
                sommamax = somma;
            }
        } else {
            somma = A[i];
            if (somma > sommamax) {
                sommamax = somma;
            }
        }
    }

    printf("%d", sommamax);
}
```

3.1.13 Scrivi un programma che riempia un array di numeri casuali di una cifra in modo tale che ogni numero sia sempre maggiore o minore di entrambi i numeri adiacenti (ad esempio 1 8 3 5 2 7 4 5 3). L'array generato deve essere stampato poi ordinato e infine ristampato. Per implementare le funzionalità descritte realizza due funzioni: una che riceve un array e lo riempie di numeri casuali come descritto, una che riceve un array e un indicazione che dica se ordinare l'array in ordine crescente o decrescente (ad esempio una variabile booleana).

3.1.14 Dato un array di numeri interi casuali, calcola e stampa il valore massimo e il valore minimo contenuti nell'array

3.1.15 Dato un array di numeri interi casuali, calcola e stampa media e mediana dell'array

3.1.16 Dato un array di numeri interi casuali, stampa tutti i numeri superiori alla media partendo dal fondo

3.1.17 Letta in input una sequenza di numeri interi positivi memorizzarla in una lista. Costruire una seconda lista contenente soltanto gli elementi della prima lista che non siano numeri primi. Stampare la seconda lista.

3.1.18 Date due liste di numeri ordinate (ottienile come vuoi), costruire una terza lista di numeri interi ordinata, ottenuta mediante la "fusione" delle prime due. Stampare la lista.

3.1.19 Leggere in input una sequenza di numeri interi ordinati in ordine crescente. Dopo aver memorizzato la sequenza in una lista, inserire nella posizione corretta all'interno della lista, tutti i numeri mancanti. Stampare in output la lista. Non devono essere usate altre iste o array di appoggio.

Esempio: supponiamo che sia fornita in input la sequenza 4, 7, 8, 9,15,17,21. Dopo aver memorizzato gli elementi nella lista 4, 7, 8, . . . 21, vengono inseriti i numeri mancanti, ottenendo la lista composta dagli elementi 4, 5, 6, 7, 8, . . . 19, 20, 21.

3.2 Stringhe

3.2.1 Leggi una stringa da input e poi comunica all'utente di quante lettere è composta la stringa. Il conteggio deve essere fatto in due modi diversi, usando la funzione strlen e contando i caratteri uno alla volta.

Soluzione

```c
// 3.2.1 - Leggi una stringa da input e poi comunica all'utente di quante lettere è composta la
// stringa. Il conteggio deve essere fatto in due modi diversi, usando la funzione
// strlen e contando i caratteri uno alla volta.

#include <stdio.h>
#include <string.h>

#define N 100

int corrado(char stringa[]) {
    int i;
    for (i = 0; i < N && stringa[i] != 0 && stringa[i] != '\n'; i++) {
        // niente
    }
    return i;
}

int main() {
    char stringa[N];

    printf("Scrivi qualcosa: ");
    // fgets(stringa, N, stdin);
    scanf(" %29[^\n]", stringa);

    printf("La stringa e' formata da %d caratteri.\n", strlen(stringa));
```

```
    printf("La stringa e' formata da %d caratteri.\n", corrado(stringa));
    printf("%s", stringa);
    printf("ciao");
}
```

3.2.2 Leggi una stringa da input e poi stampala in due modi diversi, la prima volta tutta insieme poi stampando una lettera per volta separando le lettere con uno spazio.

Soluzione

```
// 3.2.2 -   Leggi una stringa da input e poi
// stampala in due modi diversi, la prima volta tutta insieme poi stampando una
// lettera per volta separando le lettere con uno spazio.

#include <stdio.h>
#include <string.h>

#define N 100

int main() {
    char stringa[N];
    int i;

    printf("Scrivi qualcosa: ");
    // fgets(stringa, N, stdin);
    scanf(" %29[^\n]", stringa);

    printf("%s\n", stringa);
    for (i = 0; stringa[i] != 0; i++) {
        printf("%c ", stringa[i]);
    }
    printf("\n");
}
```

3.2.3 Leggi una stringa da input e poi stampala al contrario, cioè partendo dal fondo ma non modificandone il contenuto.

Soluzione

```
// 3.2.3 - Leggi una stringa da input e poi stampala al contrario, cioè partendo dal fondo ma non
// modificandone il contenuto.

#include <stdio.h>
#include <string.h>

#define N 100

int main() {
    char stringa[N];
    int i;

    printf("Scrivi qualcosa: ");
    // fgets(stringa, N, stdin);
    scanf(" %29[^\n]", stringa);

    for (i = strlen(stringa)-1; i >= 0; i--) {
        printf("%c", stringa[i]);
    }
}
```

3.2.4 Leggi una stringa da input e poi modificala in modo da ottenere la stringa inversa (ad esempio “ciao” diventa “oaic”). Stampa infine la stringa.

Soluzione

```c
// 3.2.4 -   Leggi una stringa da input e poi modificala in modo da ottenere la stringa inversa (ad esempio
“ciao” diventa
// “oaic”). Stampa infine la stringa.

#include <stdio.h>
#include <string.h>

#define N 100

int main() {
    char stringa[N];
    char tmp;
    int i, len;

    printf("Scrivi qualcosa: ");
    // fgets(stringa, N, stdin);
    scanf(" %29[^\n]", stringa);

    len = strlen(stringa);

    for (i = 0; i < len/2; i++) {
        tmp = stringa[i];
        stringa[i] = stringa[len-1-i];
        stringa[len-1-i] = tmp;
    }

    printf("%s", stringa);
}
```

3.2.5 Dichiara e inizializza due stringhe con valori a piacere, stampale. Scambia il contenuto delle due stringhe in modo che la prima contenga il contenuto della seconda e viceversa, poi ristampale.

Soluzione

```c
// 3.2.5 - Dichiara e inizializza due stringhe con valori a piacere, stampale. Scambia il
contenuto
// delle due stringhe in modo che la prima contenga il contenuto della seconda e
// viceversa, poi ristampale.

#include <stdio.h>
#include <string.h>

#define N 100

int main() {
    char stringa1[N];
    char stringa2[N];
    char tmp[N];
    int i, len;

    printf("Scrivi la prima stringa: ");
    fgets(stringa1, N, stdin);
    printf("Scrivi la seconda stringa: ");
    fgets(stringa2, N, stdin);

    printf("Inizio:\n");
    printf("%s", stringa1);
    printf("%s", stringa2);

    strcpy(tmp, stringa1);
```

```
    strcpy(stringa1, stringa2);
    strcpy(stringa2, tmp);

    printf("\n\nFine:\n");
    printf("%s", stringa1);
    printf("%s", stringa2);
}
```

3.2.6 Leggi da input due stringhe, se le due stringhe sono uguali comunica che sono uguali altrimenti scrivi quella che viene prima in ordine alfabetico

3.2.7 Leggi usando un'unica funzione scanf tre parole. Indica poi qual è la parola più lunga, la parola che viene prima in ordine alfabetico e la parola che contiene più vocali.

3.2.8 Scrivi e utilizza una funzione che riceve una stringa e un carattere e che stampi tutto il contenuto della stringa andando a capo ogni volta che trova il carattere dato (il carattere rappresenta un separatore della stringa in sottostringhe).

3.2.9 Scrivi un programma che contenga in un array la seguente stringa "Ciao a tutti, mi chiamo Francesco\nOggi vi voglio parlare delle stringhe, siete d'accordo?" e che la stampi in modo tale che quando trova una virgola va a capo(oltre a stampare la virgola) e quando trova un a capo, va a capo due volte.

3.2.10 Scrivi due funzioni che implementino il cifrario di Cesare cioè un cifrario che per cifrare un testo trasforma ogni lettera nella lettera che si trova tre posizioni più avanti nell'alfabeto. Le due funzioni sono le funzioni "cifra" e "decifra". Scrivi infine u programma che permetta di inserire un testo da cifrare o decifrare a scelta.

Variante: Fai in modo che il numero di posizioni sia un parametro da passare alle funzioni (consiste nella chiave di cifratura)

Soluzione

```
#include <stdio.h>

void cifra(char frase[], int chiave) {
    int i;
    int nlettere = 'z' - 'a' + 1;

    for (i = 0; frase[i] != 0; i++) {
        if (frase[i] >= 'a' && frase[i] <= 'z'){
            frase[i] = (((frase[i] - 'a') + chiave) % nlettere) + 'a';
        } else if (frase[i] >= 'A' && frase[i] <= 'Z') {
            frase[i] = (((frase[i] - 'A') + chiave) % nlettere) + 'A';
        }
    }
}

void decifra(char frase[], int chiave) {
    int i;
    int nlettere = 'z' - 'a' + 1;

    for (i = 0; frase[i] != 0; i++) {
        if (frase[i] >= 'a' && frase[i] <= 'z'){
            frase[i] = (((frase[i] - 'a') - chiave) % nlettere) + 'a';
        } else if (frase[i] >= 'A' && frase[i] <= 'Z') {
            frase[i] = (((frase[i] - 'A') - chiave) % nlettere) + 'A';
        }
    }
}
```

```c
int main() {
    char frase[] = "Ciao, come va?";
    printf("%s\n", frase);
    cifra(frase, 3);
    printf("%s\n", frase);
    decifra(frase, 3);
    printf("%s\n", frase);
}
```

3.2.11 Scrivi e utilizza una funzione che riceve due stringhe e sia in grado di dire se la stringa più piccola è una sottostringa di quella più grande. Nel tuo programma il main deve creare due stringhe di lunghezza a piacere contenenti lettere casuali (solo lettere maiuscole e minuscole senza altri simboli o cifre; la funzione non cambia)

Variante: invece di dire se la stringa è contenuta nell'altra stringa restituire la posizione di partenza della sottostringa trovata.

Ad esempio "mi c" è sottostringa di "Ciao a tutti mi chiamo Francesco"

Soluzione

```c
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>

int contiene(char piccola[], char grande[]) {
    int len_piccola = strlen(piccola);
    int len_grande = strlen(grande);
    int i, j;
    int ris;

    for (i = 0; i <= len_grande - len_piccola; i++) {
        ris = 1;
        for (j = 0; j < len_piccola; j++) {
            if (piccola[j] != grande[i + j]) {
                ris = 0;
                break;
            }
        }
        if (ris) {
            // return 1; // rispondo true se l'ho trovata
            return i; // per la posizione
        }
    }
    // return 0; // non l'ho trovata
    return -1; // quando non la trovo do posizione -1 che non esiste (metodo
    // standard)
}

void stringaCasuale(char stringa[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        stringa[i] = rand() % ('z' - 'a' + 1) + 'a';
    }
    stringa[i] = 0;
}

int main() {
    printf("%d\n", contiene("ciao", "ciao"));
    printf("%d\n", contiene("ciao", "aciao"));
    printf("%d\n", contiene("ciao", "ciaoa"));
    printf("%d\n", contiene("ciao", "aciaoa"));
    printf("%d\n", contiene("ciao", "ocia"));
```

```
    // fin qui non ho creato stringhe casuali, ho solo testato la funzione con
    // stringhe create da me

    srand(time(NULL));
    char piccola[10];
    char grande[100];
    stringaCasuale(piccola, 2);
    stringaCasuale(grande, 99);
    printf("Grande:  %s\n", grande);
    printf("Piccola: %s\n", piccola);
    printf("%d\n", contiene(piccola, grande));

}
```

3.2.12 Scrivi un programma che chieda all'utente nome e cognome e poi:

1. Verifichi che il nome e il cognome siano stati scritti con le iniziali maiuscole (devi controllare se la prima lettera ha i valori corretti secondo la tabella ASCII, i caratteri sono come numeri)
2. crei una stringa che contenga le generalità dell'utente (nome e cognome insieme)
3. stampi le generalità al contrario.
4. chieda nuovamente nome e cognome e verifichi se sono state riscritte le stesse cose di prima

3.2.13 Scrivi un programma che riempia una matrice di numeri casuali e che stampi i numeri contenuti in ogni diagonale, prima delle diagonali che vanno da in basso a sinistra a in alto a destra, poi da in alto a sinistra a in basso a destra;

Variante: invece che stampare solamente i numeri, crea una matrice che contenga per ogni riga i valori contenuti in una diagonale. Devono essere create due matrici, una per ogni diversa direzione delle diagonali. Attento che avrai bisogno di memorizzare il numero di elementi di ogni diagonale (e quindi riga della matrice)

Soluzione

```
// 3.2.13 - Scrivi un
// programma che riempia una matrice di numeri casuali e che stampi i numeri
// contenuti in ogni diagonale, prima delle diagonali che vanno da in basso a
// sinistra a in alto a destra, poi da in alto a sinistra a in basso a destra;

// Variante:
// invece che stampare solamente i numeri, crea una matrice che contenga per
// ogni riga i valori contenuti in una diagonale. Devono essere create due
// matrici, una per ogni diversa direzione delle diagonali. Attento che avrai
// bisogno di memorizzare il numero di elementi di ogni diagonale (e quindi riga
// della matrice)

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 3
#define M 5

int main() {
    int mat[N][M];
    int i, j;
```

```
    srand(time(NULL));

    printf("Matrice intera:\n");
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            mat[i][j] = rand() % 100;
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    // sola stampa diagonali su-sx a giu-dx \.
    printf("Diagonali \\:\n");
    for (j = -N + 1; j < M; j++) {
        for (i = 0; i < N; i++) {
            if (j + i >= 0 && j + i < M) {
                printf("%2d ", mat[i][j + i]);
            }
        }
        printf("\n");
    }
    printf("\n");

    // sola stampa diagonali /.
    printf("Diagonali /:\n");
    for (j = -N + 1; j < M; j++) {
        for (i = N - 1; i >= 0; i--) {
            if (j + (N - 1 - i) >= 0 && j + (N - 1 - i) < M) {
                printf("%2d ", mat[i][j + (N - 1 - i)]);
            }
        }
        printf("\n");
    }
    printf("\n");

    // sola stampa diagonali /. usando le i in maniera diversa (forse più
    // comprensibile)
    printf("Diagonali / alternativo:\n");
    for (j = -N + 1; j < M; j++) {
        for (i = 0; i < N; i++) {
            if (j + i >= 0 && j + i < M) {
                printf("%2d ", mat[N - 1 - i][j + i]);
            }
        }
        printf("\n");
    }
    printf("\n");
}
```

3.3 Matrici di numeri

3.3.1 Creare una matrice che contiene le tabelline dei numeri dall' 1 al 10. Alla fine stamparla.

Soluzione

```
#include <stdio.h>

#define N 10
#define M 10

int main() {
    int mat[N][M];
```

```
    int i, j;
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            mat[i][j] = (i+1)*(j+1);
        }
    }

    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            printf("%3d ", mat[i][j]);
        }
        printf("\n");
    }
}
```

3.3.2 Scrivi un programma che istanzia una matrice n x m di interi casuali (n e m scelti da te). Oltre a stampare l'intera matrice, chiedi all'utente quale riga o quale colonna stampare e stampala sullo schermo.

Soluzione

```
// 3.3.2 - Scrivi un programma che istanzia una matrice n x m di
// interi casuali (n e m scelti da te). Oltre a stampare l'intera matrice, chiedi
// all'utente quale riga o quale colonna stampare e stampala sullo schermo.

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 4
#define M 6

void stampaMatrice(int m[N][M]) {
    int i, j;

    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int m[N][M];
    int i, j;
    int rig, col;

    srand(time(NULL));

    // riempio la matrice
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            m[i][j] = rand() % 10;
        }
    }

    stampaMatrice(m);

    rig = 2;
    printf("\nRiga %d: ", rig);
    for (j = 0; j < M; j++) {
        printf("%d ", m[rig][j]);
```

```
    }

    col = 3;
    printf("\nColonna %d:\n", col);
    for (i = 0; i < N; i++) {
        printf("%d\n", m[i][col]);
    }
}
```

3.3.3 Scrivere un programma che riempia due matrici A e B di dimensione n x m e calcoli la matrice C = A + B e la matrice D = A x B (in realtà è un finto prodotto di matrici). Stampa le matrici ottenute. Per il calcolo della somma calcola normalmente C[i][j] = A[i][j] + B[i][j] mentre per la moltiplicazione hai due possibilità:

1. Versione semplice se non riesci proprio a fare la versione corretta: C[i][j] = A[i][j] x B[i][j].
2. Versione complicata: vai a guardare su internet come si calcola e scrivi l'algoritmo per farlo.

3.3.4 Scrivi un programma che data una matrice n x m di interi, scambia le righe pari con quelle dispari
variante: le dimensioni della matrice vengono scelte dall'utente tra i valori massimi 10 x 20.

Soluzione

```
// 3.3.4 - Scrivi un programma che data una matrice n x m di
// interi, scambia le righe pari con quelle dispari
// variante: le dimensioni della matrice vengono scelte dall'utente tra i valori
// massimi 10 x 20.

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 100
#define M 100

void riempiMatrice(int mat[N][M], int n, int m, int min, int max) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            mat[i][j] = rand() % (max-min+1) + min;
        }
    }
}

void stampaMatrice(int mat[N][M], int n, int m) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }
}

void scambiaRighe(int mat[N][M], int n, int m) {
    int i, j, tmp;
```

```
    for (i = 0; i < n-1; i+=2) {
        for (j = 0; j < m; j++) {
            tmp = mat[i][j];
            mat[i][j] = mat[i+1][j];
            mat[i+1][j] = tmp;
        }
    }
}

int main() {
    int mat[N][M];
    int n, m;

    srand(time(NULL));

    do {
        printf("Scrivi il numero di righe, min 1, max 10: ");
        scanf("%d", &n);
    } while (n < 1 || n > 10);
    do {
        printf("Scrivi il numero di colonne, min 1, max 20: ");
        scanf("%d", &m);
    } while (m < 1 || m > 10);

    riempiMatrice(mat, n, m, 0, 99);
    printf("\nMatrice iniziale:\n");
    stampaMatrice(mat, n, m);

    scambiaRighe(mat, n, m);
    printf("\nMatrice dopo gli scambi:\n");
    stampaMatrice(mat, n, m);
}
```

3.3.5 Scrivi un programma che data una matrice n x n di interi, con n scelto dall'utente, scambia le righe con le colonne

Soluzione

```
// 3.3.5 - Scrivi un programma che data una matrice n x n di
// interi, con n scelto dall'utente, scambia le righe con le colonne

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 100
#define M 100

void riempiMatrice(int mat[N][M], int n, int m, int min, int max) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            mat[i][j] = rand() % (max-min+1) + min;
        }
    }
}

void stampaMatrice(int mat[N][M], int n, int m) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            printf("%2d ", mat[i][j]);
        }
```

```
        printf("\n");
    }
}

void scambiaRigheColonne1(int mat[N][N], int n) {
    int i, j;
    // uso una seconda matrice
    int m2[n][n];
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            m2[j][i] = mat[i][j];
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            mat[i][j] = m2[i][j];
        }
    }
}

void scambiaRigheColonne2(int mat[N][N], int n) {
    int i, j, tmp;
    // lavoro sulla stessa matrice
    for (i = 0; i < n; i++) {
        for (j = i; j < n; j++) {
            tmp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = tmp;
        }
    }
}

int main() {
    int mat[N][N];
    int n;

    srand(time(NULL));

    do {
        printf("Scrivi il numero di righe, min 1, max 10: ");
        scanf("%d", &n);
    } while (n < 1 || n > 10);

    riempiMatrice(mat, n, n, 0, 99);
    printf("\nMatrice iniziale:\n");
    stampaMatrice(mat, n, n);

    scambiaRigheColonne1(mat, n);
    printf("\nMatrice dopo gli scambi 1:\n");
    stampaMatrice(mat, n, n);

    scambiaRigheColonne2(mat, n);
    printf("\nMatrice dopo gli scambi 2:\n");
    stampaMatrice(mat, n, n);
}
```

3.3.6 Scrivi un programma che riempia una matrice n x m di numeri interi a caso, n e m sono scelti dall'utente e il range dei numeri è da 0 a x scelto anch'esso dall'utente. Determina poi il numero più frequente generato nella matrice. Il riempimento delle matrici e la ricerca del numero più frequente devono essere svolti da due apposite funzioni, mentre l'interazione con l'utente dev'essere lasciata al main.

Soluzione

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define N 100
#define M 100
#define MAX 100

void riempiMatrice(int mat[N][M], int n, int m, int min, int max) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            mat[i][j] = rand() % (max-min+1) + min;
        }
    }
}

void stampaMatrice(int mat[N][M], int n, int m) {
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }
}

int piuFrequente(int mat[N][M], int n, int m) {
    int i, j;
    int occorrenze[MAX+1];
    int max;

    // azzero le occorrenze per ogni numero, la posizione indica il numero
    for (i = 0; i <= MAX; i++) {
        occorrenze[i] = 0;
    }

    // conto le occorrenze
    max = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            occorrenze[mat[i][j]] += 1;
            if (occorrenze[mat[i][j]] > occorrenze[max]) {
                max = mat[i][j];
            }
        }
    }
    return max;
}

int main() {
    int mat[N][M];
    int n, m, max, ris;

    srand(time(NULL));

    do {
        printf("Scrivi il numero di righe, min 1, max 10: ");
        scanf("%d", &n);
    } while (n < 1 || n > 10);
    do {
        printf("Scrivi il numero di colonne, min 1, max 20: ");
```

```
        scanf("%d", &m);
    } while (m < 1 || m > 20);

    do {
        printf("Scrivi il valore massimo da generare, min 0, max %d: ", MAX);
        scanf("%d", &max);
    } while (max < 1 || max > MAX);

    riempiMatrice(mat, n, m, 0, max);
    printf("\nMatrice iniziale:\n");
    stampaMatrice(mat, n, m);

    ris = piuFrequente(mat, n, m);
    printf("\nIl numero più frequente: %d\n", ris);
}
```

3.3.7 Scrivi un programma che riempia una matrice n x m di numeri interi a caso, n e m sono scelti dall'utente e il range dei numeri è da 0 a x scelto anch'esso dall'utente. Determina poi il la riga o la colonna con la somma dei numeri più grande. Il riempimento delle matrici e la ricerca della riga o della colonna devono essere svolti da apposite funzioni, mentre l'interazione con l'utente dev'essere lasciata al main.

3.3.8 Scrivi un programma che:

1. crei e riempia di valori casuali una matrice di dimensioni a tua scelta,
2. ne stampi il contenuto sullo schermo,
3. chieda all'utente un numero e cerchi se il numero è presente nella matrice
4. se il numero è presente si stampino le coordinate della posizione nella matrice

3.3.9 Scrivi un programma che istanzia una matrice n x m di interi casuali (n e m scelti da te). Oltre a stampare l'intera matrice, chiedi all'utente quale riga o quale colonna stampare e stampala sullo schermo

3.3.10 data una matrice n x m di numeri casuali (n e m scelti da te), stampa l'indice della riga con la massima somma dei numeri contenuti. Stampa anche l'indice della colonna con la massima somma dei numeri.

Soluzione

```
// 3.3.10   data una matrice n x m di numeri casuali (n e m scelti da te), stampa
// l'indice della riga con la massima somma dei numeri contenuti. Stampa anche
// l'indice della colonna con la massima somma dei numeri.

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 3
#define M 4

int main() {
    int mat[N][M];
    int i, j;
    int somma, sommamax, imax, jmax;

    srand(time(NULL));

    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
```

```
            mat[i][j] = rand() % 10;
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }

    // trovo la riga massima
    imax = 0;
    sommamax = 0;
    for (i = 0; i < N; i++) {
        somma = 0;
        for (j = 0; j < M; j++) {
            somma += mat[i][j];
        }
        if (somma > sommamax) {
            sommamax = somma;
            imax = i;
        }
    }
    printf("Riga con somma massima: %d\n", imax);

    // trovo la colonna massima
    jmax = 0;
    sommamax = 0;
    for (j = 0; j < M; j++) {
        somma = 0;
        for (i = 0; i < N; i++) {
            somma += mat[i][j];
        }
        if (somma > sommamax) {
            sommamax = somma;
            jmax = j;
        }
    }
    printf("Colonna con somma massima: %d\n", jmax);
}
```

3.3.11 Scrivi un programma che per mezzo di un'apposita funzione riempia una matrice di numeri casuali. Stampa poi il numero di numeri pari contenuti in ogni riga della matrice (riga per riga separatamente) e il numero di numeri dispari contenuti in ogni colonna della matrice. Anche in questo caso definisci una o due funzioni per farlo (a scelta).

3.3.12 Riempi una matrice di numeri casuali (dimensione a scelta), poi scegli di stampare solo una parte della matrice: a partire da una riga ad un'altra e a partire da una colonna ad un'altra.

Soluzione

```
// 3.3.12   Riempi una matrice di numeri casuali (dimensione a scelta), poi scegli
// di stampare solo una parte della matrice: a partire da una riga ad un'altra e
// a partire da una colonna ad un'altra.
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10
#define M 10

int main() {
    int mat[N][M];
    int i, j;
    int ri, rf, ci, cf;
```

```
    srand(time(NULL));

    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            mat[i][j] = rand() % 100;
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }

    ri = 2;
    rf = 5;
    ci = 1;
    cf = 3;

    printf("\n");
    for (i = ri; i <= rf; i++) {
        for (j = ci; j <= cf; j++) {
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }

}
```

3.4 Array di stringhe o matrici di caratteri

3.4.1 Scrivi un programma che chieda all'utente una serie di nomi che devono essere tutti memorizzati in un array di stringhe (e sarà quindi una matrice, ad esempio char nomi[n][m] con n numero dei nomi e m numero massimo di caratteri per stringa). Il numero di nomi da inserire deve essere chiesto all'utente. Quando viene inserito un nome bisogna controllare che inizi con una lettera e se questa è minuscola deve essere fatta diventare maiuscola. Alla fine stampa tutte le stringhe.

Soluzione

```
// 3.4.1 - Scrivi un
// programma che chieda all'utente una serie di nomi che devono essere tutti
// memorizzati in un array di stringhe (e sarà quindi una matrice, ad esempio char
// nomi[n][m] con n numero dei nomi e m numero massimo di caratteri per stringa).
// Il numero di nomi da inserire deve essere chiesto all'utente. Quando viene
// inserito un nome bisogna controllare che inizi con una lettera e se questa è
// minuscola deve essere fatta diventare maiuscola. Alla fine stampa tutte le
// stringhe.

#include <stdio.h>
#include <ctype.h>

#define N 100
#define M 100

int main() {
    int n, i;
    char nomi[N][M];

    n = 3;
    for (i = 0; i < n; i++) {
        printf("Scrivi il %d nome: ", i+1);
        fgets(nomi[i], M, stdin);
        // if (nomi[i][0] >= 'a' && nomi[i][0] <= 'z') {
        //     nomi[i][0] += 'A' - 'a';
        // }
```

```
        nomi[i][0] = toupper(nomi[i][0]);
    }

    for (i = 0; i < n; i++) {
        printf("%s", nomi[i]);
    }
}
```

3.4.2 Continua l'esercizio precedente aggiungendo le seguenti funzionalità:

1. Cerca e stampa il nome più lungo
2. Cerca e stampa il nome che in ordine alfabetico è il primo
3. Chiedi all'utente un nome e digli se il nome è presente nell'elenco dei nomi (usa strcmp).

Soluzione

```
// 3.4.2 - Continua l'esercizio precedente
// aggiungendo le seguenti funzionalità:

// 1.  Cerca e stampa il nome più lungo
// 2.  Cerca e stampa il nome che in ordine alfabetico è il primo
// 3.  Chiedi all'utente un nome e digli se il nome è presente
// nell'elenco dei nomi (usa strcmp).

#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define N 100
#define M 100

int main() {
    int n, i;
    char nomi[N][M];
    char lungo[M];
    char primo[M];
    char cercare[M];

    n = 3;
    for (i = 0; i < n; i++) {
        printf("Scrivi il %d nome: ", i+1);
        scanf(" %99[^\n]", nomi[i]);
        // if (nomi[i][0] >= 'a' && nomi[i][0] <= 'z') {
        //     nomi[i][0] += 'A' - 'a';
        // }
        nomi[i][0] = toupper(nomi[i][0]);
    }

    for (i = 0; i < n; i++) {
        printf("%s\n", nomi[i]);
    }

    // printf("-----");

    // cerco il più lungo
    strcpy(lungo, nomi[0]);
    for (i = 1; i < n; i++) {
        if (strlen(nomi[i]) > strlen(lungo)) {
            strcpy(lungo, nomi[0]);
        }
    }
    //alternativa
```

```
    int ilungo = 0;
    for (i = 1; i < n; i++) {
        if (strlen(nomi[i]) > strlen(nomi[ilungo])) {
            ilungo = i;
        }
    }
    printf("Il nome piu' lungo e': %s\n", nomi[ilungo]);


    // 2.  Cerca e stampa il nome che in ordine alfabetico è il primo
    int iprimo = 0;
    for (i = 1; i < n; i++) {
        if (strcmp(nomi[i], nomi[iprimo]) < 0) {
            iprimo = i;
        }
    }
    printf("Il primo nome in ordine alfabetico e': %s\n", nomi[iprimo]);

    // 3.  Chiedi all'utente un nome e digli se il nome è presente
    // nell'elenco dei nomi (usa strcmp).
    printf("Scrivi un nome da cercare: ");
    // fgets(cercare, M, stdin);
    scanf(" %99[^\n]", cercare);
    for (i = 0; i < n; i++) {
        if (strcmp(nomi[i], cercare) == 0) {
            printf("Il nome e' presente nell'elenco.\n");
            break;
        }
    }
    if (i == n) {
        printf("Il nome non e' presente nell'elenco.\n");
    }

}
```

3.4.3 Dato un array di stringhe (quindi una matrice), riempito come preferisci, stampa la stringa più lunga, la più corta e la lunghezza media delle stringhe. I tre risultati devono essere ottenuti per mezzo di apposite funzioni.

Soluzione

```
// 3.4.3 - Dato un
// array di stringhe (quindi una matrice), riempito come preferisci, stampa la
// stringa più lunga, la più corta e la lunghezza media delle stringhe. I tre
// risultati devono essere ottenuti per mezzo di apposite funzioni.

#include <stdio.h>
#include <string.h>

#define N 100
#define M 100

int main() {
    char mat[N][M];
    int n;
    int i;

    strcpy(mat[0], "Ciao");
    strcpy(mat[1], "belli");
    strcpy(mat[2], "evviva!");
    n = 3;

    // la più lunga
    int ilunga = 0;
```

```
    for (i = 1; i < n; i++) {
        if (strlen(mat[i]) > strlen(mat[ilunga])) {
            ilunga = i;
        }
    }
    printf("La parola piu' lunga e': %s\n", mat[ilunga]);

    // la più corta
    int icorta = 0;
    for (i = 1; i < n; i++) {
        if (strlen(mat[i]) < strlen(mat[icorta])) {
            icorta = i;
        }
    }
    printf("La parola piu' corta e': %s\n", mat[icorta]);

    // lunghezza media delle stringhe
    float media = 0;
    for (i = 0; i < n; i++) {
        media += strlen(mat[i]);
    }
    media /= n;
    printf("La lunghezza media delle parole e': %.2f\n", media);

}
```

3.4.4 Scrivi e usa una funzione che dato un array di stringhe (quindi una matrice), restituisca il numero di parole palindrome contenute nell'array. Riempi l'array come preferisci.

Soluzione

```
// 3.4.4 - Scrivi e usa una funzione che dato un array di stringhe (quindi una matrice),
// restituisca il numero di parole palindrome contenute nell'array. Riempi l'array
// come preferisci.

#include <stdio.h>
#include <string.h>

#define N 100
#define M 100

int palindrome(char elenco[][M], int n) {
    int i, j, ris = 0, palindroma, m;
    for (i = 0; i < n; i++) {
        palindroma = 1;
        m = strlen(elenco[i]);
        for (j = 0; j < m / 2; j++) {
            if (elenco[i][j] != elenco[i][m-1-j]) {
                palindroma = 0;
                break;
            }
        }
        if (palindroma) {
            ris++;
        }
    }
    return ris;
}

int main() {
```

```
    char elenco[N][M];
    int n;

    strcpy(elenco[0], "casa");
    strcpy(elenco[1], "anna");
    strcpy(elenco[2], "osso");
    strcpy(elenco[3], "cammello");
    strcpy(elenco[4], "onorarono");
    strcpy(elenco[5], "fragnelli");
    strcpy(elenco[6], "corrado");
    strcpy(elenco[7], "chiara");
    n = 8;

    // int x = palindrome(elenco, n);

    printf("Le parole palindrome sono %d\n", palindrome(elenco, n));
}
```

3.4.5 Scrivi un programma che permetta di rappresentare il campo di gioco di “Campo minato”. Il campo deve essere formato da una superficie suddivisa in caselle quadrate, quindi avrai una grande tabella (ad esempio di dimensioni 8x12 ma puoi scegliere diversamente). Sul campo devono essere disposte a caso una serie di mine (suggerisco 16 ma puoi cambiare). Fai poi in modo che ogni casella che non sia una mina abbia un numero che rappresenti il numero di mine adiacenti (anche in diagonale quindi un numero da 1 a 8) lasciando vuote le celle che non hanno mine adiacenti (non scrivere 0)

Soluzione

```
// 3.4.4 - Scrivi e usa una funzione che dato un array di stringhe (quindi una matrice),
// restituisca il numero di parole palindrome contenute nell’array. Riempi l’array
// come preferisci.

#include <stdio.h>
#include <string.h>

#define N 100
#define M 100

int palindrome(char elenco[][M], int n) {
    int i, j, ris = 0, palindroma, m;
    for (i = 0; i < n; i++) {
        palindroma = 1;
        m = strlen(elenco[i]);
        for (j = 0; j < m / 2; j++) {
            if (elenco[i][j] != elenco[i][m-1-j]) {
                palindroma = 0;
                break;
            }
        }
        if (palindroma) {
            ris++;
        }
    }
    return ris;
}

int main() {
    char elenco[N][M];
    int n;

    strcpy(elenco[0], "casa");
    strcpy(elenco[1], "anna");
    strcpy(elenco[2], "osso");
```

```
    strcpy(elenco[3], "cammello");
    strcpy(elenco[4], "onorarono");
    strcpy(elenco[5], "fragnelli");
    strcpy(elenco[6], "corrado");
    strcpy(elenco[7], "chiara");
    n = 8;

    // int x = palindrome(elenco, n);

    printf("Le parole palindrome sono %d\n", palindrome(elenco, n));
}
```

3.4.6 Crea un programma di gestione di una agenda. L'agenda deve permettere di visualizzare e inserire i dati riguardanti i propri contatti. Per ogni persona devono essere memorizzati nome cognome e numero di telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per nome o cognome (scegli tu). Aggiungi poi una funzione per cancellare un contatto in una specifica posizione dell'agenda. (Non avendo ancora fatto la gestione dei files ogni volta l'agenda partirà da uno stato predefinito che puoi decidere tu). Aggiungi anche un'interfaccia utilizzabile.

3.4.7 Scrivi un programma che chieda all'utente di scrivere una frase e le legga con la funzione gets o fgets. Le parole della frase devono poi essere stampate una per volta andando sempre a capo tra una parola e l'altra.

3.4.8 Scrivi un programma che chieda all'utente di scrivere una serie di parole lette una ad una. Queste parole devono essere unite a formare un'unica stringa che deve essere infine stampata.

Soluzione

```
// 3.4.8 - Scrivi un programma che chieda all'utente di scrivere
// una serie di parole lette una ad una. Queste parole devono essere unite a
// formare un'unica stringa che deve essere infine stampata.

#include <stdio.h>
#include <string.h>

#define N 5
#define M 100

int main() {
    char parole[N][M];
    int i, j, k;
    char frase[N*M];

    printf("Scrivere 5 parole.\n");
    for (i = 0; i < N; i++) {
        printf("La numero %d: ", i+1);
        scanf("%s", parole[i]);
    }

    // // opzione 1 veloce
    // strcpy(frase, parole[0]);
    // for (i = 1; i < N; i++) {
    //     strcat(frase, " ");
    //     strcat(frase, parole[i]);
    // }

    // opzione 2 lenta
    k = 0;
    for (i = 0; i < N; i++) {
```

```
        for (j = 0; parole[i][j] != 0; j++) { // 0 == '\0' come ' ' == 32
            frase[k] = parole[i][j];
            k++;
        }
        frase[k] = ' ';
        k++;
    }
    frase[k] = 0;

    printf("\n%s\n", frase);
}
```

3.4.9 Scrivi un programma che chieda all'utente di scrivere una frase e le legga con la funzione gets o fgets (otterrai quindi un'unica stringa). Le parole della frase devono poi essere ordinate e stampate in ordine alfabetico. (consiglio: prendi le parole dalla stringa, mettile in un array e ordina l'array)

3.5 Algoritmi di ordinamento

3.5.1 Carica un array di numeri casuali e utilizza il selection sort per ordinarlo. Deve essere fatta una stampa dell'array prima e dopo l'ordinamento

3.5.2 Carica un array di numeri casuali e utilizza il bubble sort per ordinarlo. Deve essere fatta una stampa dell'array prima e dopo l'ordinamento

3.5.3 Carica un array di numeri casuali e utilizza la funzione sort della libreria standard per ordinarlo. Deve essere fatta una stampa dell'array prima e dopo l'ordinamento

3.5.4 Scrivi un programma che dato un array di numeri interi, sia in grado di:

1. controllare se l'array è palindromo,
2. invertire l'ordine dei numeri dell'array,
3. ordinare in ordine decrescente l'array.

3.5.5 Dato un array di stringhe (quindi una matrice di caratteri), che puoi riempire come preferisci, ordina l'array in ordine alfabetico prima crescente e poi decrescente. Devono essere inserite le stampe dell'intero array prima e dopo ogni ordinamento.

3.5.6 Crea un programma di gestione di una agenda. L'agenda deve permettere di visualizzare e inserire i dati riguardanti i propri contatti. Per ogni persona devono essere memorizzati nome cognome e numero di telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per nome o cognome (scegli tu). (Non avendo ancora fatto la gestione dei files ogni volta l'agenda partirà da uno stato predefinito che puoi decidere tu).

Soluzione

```
// 3.5.6 - Crea un programma di gestione di una agenda. L'agenda
// deve permettere di visualizzare e inserire i dati riguardanti i propri
// contatti. Per ogni persona devono essere memorizzati nome cognome e numero di
// telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per
// nome o cognome (scegli tu). (Non avendo ancora fatto la gestione dei files ogni
// volta l'agenda partirà da uno stato predefinito che puoi decidere tu).

#include <stdio.h>
#include <string.h>

#define N 100
```

```c
#define M 100

char nomi[N][M];
char cognomi[N][M];
char numeri[N][M];
int n;

void ordinaAgenda() {
    int i, j;
    char tmp[M];
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-1-i; j++) {
            if (strcmp(nomi[j], nomi[j+1]) > 0) {
                strcpy(tmp, nomi[j]);
                strcpy(nomi[j], nomi[j+1]);
                strcpy(nomi[j+1], tmp);

                strcpy(tmp, cognomi[j]);
                strcpy(cognomi[j], cognomi[j+1]);
                strcpy(cognomi[j+1], tmp);

                strcpy(tmp, numeri[j]);
                strcpy(numeri[j], numeri[j+1]);
                strcpy(numeri[j+1], tmp);
            }
        }
    }
}

int inserisciContatto(char nome[M], char cognome[M], char numero[M]) {
    if (n < N-1) {
        strcpy(nomi[n], nome);
        strcpy(cognomi[n], cognome);
        strcpy(numeri[n], numero);
        n++;
        ordinaAgenda();
        return 0;
    }
    return 1;
}

void stampaContatti() {
    int i;
    for (i = 0; i < n; i++) {
        printf("%s %s - %s\n", nomi[i], cognomi[i], numeri[i]);
    }
}

int main() {
    int scelta = -1;
    char nome[M];
    char cognome[M];
    char numero[M];

    do {
        printf("   AGENDA   \n\n");
        printf("1. Visualizza contatti\n");
        printf("2. Inserisci contatto\n");
        printf("3. esci\n");

        printf("\nChe cosa vuoi fare? ");
        scanf("%d", &scelta);
        while (scelta < 1 || scelta > 3) {
            printf("Scelta non prevista, riprova: ");
            scanf("%d", &scelta);
        }
```

```
        if (scelta == 1) {
            printf("\n\nElenco dei contatti:\n\n");
            stampaContatti();
        } else if (scelta == 2) {
            printf("Scrivi il nome: ");
            scanf("%s", nome);
            printf("Scrivi il cognome: ");
            scanf("%s", cognome);
            printf("Scrivi il numero: ");
            scanf("%s", numero);

            if (inserisciContatto(nome, cognome, numero)) {
                printf("L'agenda e' piena.\n");
            } else {
                printf("Contatto inserito.\n");
            }
        }
    } while (scelta != 3);

}
```

4. Funzioni

4.1 Funzioni semplici

4.1.1 Scrivi e utilizza una funzione per calcolare e restituire la somma tra due numeri

4.1.2 Scrivi e utilizza 4 funzioni che effettuano le 4 operazioni e restituiscono il risultato.

4.1.3 Scrivi e utilizza una funzione che dati due valori e un simbolo di una delle 4 operazioni effettua quella operazione e restituisce il risultato

4.1.4 Scrivi e utilizza le seguenti funzioni:

1. raddoppia un numero passando parametri per valore,
2. raddoppia un numero passando parametri per indirizzo,
3. raddoppia due numeri contemporaneamente.

4.1.5 Scrivi e utilizza una funzione che restituisca il prodotto di due numeri e alla fine dell'esecuzione lasci modificati i due numeri in modo che valgano il doppio di prima.

4.1.6 Funzione che date le coordinate di due punti nel piano, calcolino e restituiscano le coordinate del punto medio tra i due (senza approssimazioni).

4.1.7 Funzione che dato il tempo di caduta di un oggetto, calcola l'altezza da cui è caduto l'oggetto e la velocità di impatto. Le formule che ti servono sono: h = 1/2 * g * t^2, con g=9,81 e v=g*t. Almeno uno dei due valori deve essere restituito normalmente

4.1.8 Funzione che prevede se un oggetto si rompe cadendo da una certa altezza. Quello che ti serve sapere è quindi l'altezza da cui cade l'oggetto e la velocità massima di impatto che l'oggetto può sopportare. Le formule che ti servono sono: h = 1/2 * g * t^2, con g=9,81 e v=g*t.

Variante: Modifica l'esercizio facendo in modo che la funzione restituisca anche la velocità di impatto.

4.1.9 Scrivi ed utilizza una funzione che dati due numeri li aumenta entrambi di un certo valore passato come parametro.

4.1.10 Scrivi e utilizza una funzione che dato un array di numeri, trova e restituisce il massimo e il minimo numero presenti.

4.1.11 Scrivi e utilizza una funzione che, data una stringa, restituisce true se la stringa inizia con una lettera maiuscola.

4.1.12 Funzione che data una stringa, la stampa separando ogni carattere della stringa con uno spazio.

4.1.13 Scrivi e utilizza una funzione che riceve come parametri due stringhe e restituisce una stringa che è la concatenazione delle altre due.

4.1.14 Scrivi un programma che, dati n voti in un array, li passi ad una funzione che restituisca la media dei voti escludendo il voto più alto e il voto più basso.

4.1.15 Scrivi e utilizza una funzione che stampa una stringa ricevuta come parametro.

4.1.16 Scrivi un programma che utilizzi una funzione che, ricevuto un array di numeri interi, restituisce il numero che compare con più frequenza (in caso di parità restituisce il primo numero trovato con la frequenza maggiore).

4.1.17 Scrivi una funzione che scambia il contenuto di due variabili intere (i valori devono rimanere scambiati per chi chiama la funzione).

4.1.18 Scrivi un programma che sappia riordinare un array. Per farlo dividi le seguenti funzionalità in diverse funzioni:

1. funzione che scambia il valore di due variabili
2. funzione che riceve un array e lo riempie di numeri casuali
3. funzione che stampa un array
4. funzione che riceve un array e lo ordina. Lo scambio del valore delle variabili deve essere fatto usando la funzione descritta sopra
5. main che dichiara un array e poi usando le funzioni descritte sopra lo riempie di numeri casuali, lo stampa, lo ordina e poi lo ristampa.

4.1.19 Scrivi e utilizza una funzione che calcola il quoziente tra due numeri interi e restituisce, oltre al quoziente anche il resto della divisione.

Soluzione

```c
// 4.1.19   Scrivi e utilizza una funzione che calcola il quoziente tra due numeri
// interi e restituisce, oltre al quoziente anche il resto della divisione.

#include <stdio.h>

void quozienteI(int num, int den, int *q, int *r) {
    *r = num;
    for (*q = 0; *r >= den; (*q)++) {
        *r -= den;
    }
}

int quozienteR(int num, int den, int *r) {
    if (num < den) {
        *r = num;
```

```
        return 0;
    } else {
        return 1 + quozienteR(num-den, den, r);
    }
}

int main() {
    int num = 14;
    int den = 3;
    int q, r;
    q = quozienteR(num, den, &r);
    printf("%d/%d = %d resto %d\n", num, den, q, r);
}
```

4.1.20 Scrivi e utilizza una funzione che data una stringa, fa in modo che la prima lettera sia maiuscola e tutte le altre minuscole. i caratteri che non sono lettere non devono essere modificati.

4.1.21 Scrivi una funzione che calcola il massimo comune divisore tra due numeri.
Scrivi ed utilizza poi una funzione che dati due valori numeratore e denominatore, semplifica la frazione, usando la funzione mcd appena definita.

Soluzione

```
// 4.1.21 - Scrivi una funzione che calcola il massimo comune
// divisore tra due numeri.
// Scrivi ed utilizza poi una funzione che dati due valori numeratore e
// denominatore, semplifica la frazione, usando la funzione mcd appena definita.

#include <stdio.h>

typedef struct {
    int num, den;
} Frazione;

int mcd(int a, int b) {
    int max, min, i;
    if (a > b) {
        max = a;
        min = b;
    } else {
        max = b;
        min = a;
    }
    for (i = min; i > 0; i--) {
        if (max % i == 0 && min % i == 0) {
            return i;
        }
    }
}

Frazione semplificaV(Frazione a) {
    int div = mcd(a.num, a.den);
    a.num /= div;
    a.den /= div;
    return a;
}

void semplificaI(Frazione *a) {
    int div = mcd(a->num, a->den);
    a->num /= div;
    a->den /= div;
```

```
}

int main() {
    Frazione a, b;
    a.num = 36;
    a.den = 60;
    b = semplificaV(a);
    semplificaI(&a);
    printf("a: %d/%d\n", a.num, a.den);
    printf("b: %d/%d\n", b.num, b.den);
}
```

4.1.22 Scrivi ed utilizza una funzione che riceva come parametri tre numeri e che restituisca true se questi numeri possono essere le misure di un triangolo. In questo caso la funzione deve anche restituire una stringa in cui c'è scritto il tipo di triangolo (equilatero, isoscele, scaleno).

Soluzione

```
// 4.1.22 - Scrivi ed utilizza una funzione che riceva come
// parametri tre numeri e che restituisca true se questi numeri possono essere le
// misure di un triangolo. In questo caso la funzione deve anche restituire una
// stringa in cui c'è scritto il tipo di triangolo (equilatero, isoscele,
// scaleno).

#include <stdio.h>
#include <string.h>

int triangolo(float a, float b, float c, char tipo[]) {
    if (a >= b + c || b >= a + c || c >= a + b || a <= 0 || b <= 0 || c <= 0) {
        return 0;
    } else {
        if (a == b && b == c) {
            strcpy(tipo, "equilatero");
        }
        else if (a == b || a == c || b == c) {
            strcpy(tipo, "isoscele");
        } else {
            strcpy(tipo, "scaleno");
        }
        return 1;
    }
}

int main() {
    char tipo[20];
    float a, b, c;

    a = 10;
    b = 2;
    c = 3;
    if (triangolo(a,b,c,tipo)) {
        printf("%g, %g, %g formano un triangolo %s\n", a, b, c, tipo);
    } else {
        printf("%g, %g, %g  non formano un triangolo \n", a, b, c);
    }

    a = 10;
    b = 20;
    c = 15;
    if (triangolo(a,b,c,tipo)) {
        printf("%g, %g, %g formano un triangolo %s\n", a, b, c, tipo);
    } else {
        printf("%g, %g, %g  non formano un triangolo \n", a, b, c);
```

```
    }

    a = 10;
    b = 10;
    c = 10;
    if (triangolo(a,b,c,tipo)) {
        printf("%g, %g, %g formano un triangolo %s\n", a, b, c, tipo);
    } else {
        printf("%g, %g, %g  non formano un triangolo \n", a, b, c);
    }

    a = 10;
    b = 10;
    c = 12;
    if (triangolo(a,b,c,tipo)) {
        printf("%g, %g, %g formano un triangolo %s\n", a, b, c, tipo);
    } else {
        printf("%g, %g, %g  non formano un triangolo \n", a, b, c);
    }
}
```

4.1.23 Scrivi e utilizza una funzione che dato un array restituisce i valori massimi e minimi dell'array

Soluzione

```
// 4.1.23   Scrivi e utilizza una funzione che dato un array restituisce i valori
// massimi e minimi dell'array

#include <stdio.h>

typedef struct {
    int min;
    int max;
} minmax;

minmax maxminI1(int a[], int n) {
    minmax ris;
    int i;
    if (n < 1) {
        printf("Array vuoto!\n");
        return ris;
    }
    ris.min = a[0];
    ris.max = a[0];
    for (i = 1; i < n; i)

}

int main() {

}
```

4.1.24 Scrivi e utilizza una funzione che restituisce media e mediana di una serie di numeri contenuti in un array

4.2 Funzioni ricorsive

4.2.1 Scrivi ed utilizza una funzione iterativa che calcola il fattoriale di un numero

Soluzione

```c
#include <stdio.h>

void scambia (int* a, int* b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int main() {
    int a = 1, b = 2;
    printf("%d %d\n", a, b);
    scambia(&a, &b);
    printf("%d %d\n", a, b);


}
```

4.2.2 Scrivi ed utilizza una funzione ricorsiva che calcola il fattoriale di un numero

Soluzione

```c
#include <stdio.h>

void scambia (int* a, int* b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

void bubbleSort(int v[], int n) {
    int i, j;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n - 1 -i; j++) {
            if (v[j] > v[j+1]) {
                scambia(&v[j], &v[j+1]);
                // scambia(v+j, v+j+1);
            }
        }
    }
}

void stampaVettore(int v[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", v[i]);
    }
    printf("\n");
}

int main() {
    int v[] = {3,8,1,2,7,6,8};
    int n = 7;

    stampaVettore(v, n);
    bubbleSort(v,n);
    stampaVettore(v,n);
}
```

4.2.3 Scrivi una funzione iterativa che calcola l'ennesimo numero della successione di Fibonacci.

Soluzione

```c
// 4.2.3 - Scrivi una funzione iterativa che calcola l'ennesimo
// numero della successione di Fibonacci.

#include <stdio.h>

int fibonacci(int n) {
    int i, fib = 1, fib1 = 1, fib2 = 0;
    if (n == 0) {
        return 0;
    }

    for (i = 2; i <= n; i++) {
        fib = fib2+fib1;
        fib2 = fib1;
        fib1= fib;
    }
    return fib;
}

int main() {
    int i;
    for (i = 0; i < 30; i++) {
        printf("f(%d) = %d\n", i, fibonacci(i));
    }

}
```

4.2.4 Scrivi una funzione ricorsiva che calcola l'ennesimo numero della successione di Fibonacci.

Soluzione

```c
// 4.2.4 - Scrivi una funzione ricorsiva che calcola l'ennesimo
// numero della successione di Fibonacci.

#include <stdio.h>

int fibonacci(int n) {
    // quando uscire
    if (n < 0) return -1;
    if (n == 0) return 0;
    if (n == 1) return 1;

    // ricorsione
    return fibonacci(n-1) + fibonacci(n-2);
}

int main() {
    int i;
    for (i = 0; i < 30; i++) {
        printf("f(%d) = %d\n", i, fibonacci(i));
    }
}
```

4.2.5 Scrivi una funzione ricorsiva che effettui la moltiplicazione tra due numeri

Soluzione

```c
// 4.2.5    Scrivi una funzione ricorsiva che effettui la moltiplicazione tra due numeri

#include <stdio.h>
```

```c
int moltiplicazione(int a, int b){
    if (a == 0 || b == 0){
        return 0;
    }

    return a + moltiplicazione (a, b - 1);
}


int main() {
   int a = 7, b = 3;
   printf ("%d", moltiplicazione(a, b));

}
```

4.2.6 Scrivi una funzione che calcola ricorsivamente il quoziente tra due numeri utilizzando le sottrazioni.

Soluzione

```c
// 4.2.6    Scrivi una funzione che calcola ricorsivamente il quoziente tra due
// numeri utilizzando le sottrazioni.

#include <stdio.h>
int divisione(int a, int b) {
    if (b == 0) {
        printf("impossibile");
        return -1;
    }

    if (a < b) {
        return 0;
    }

    return 1 + divisione(a - b, b);
}

int resto(int a, int b) {
    if (b == 0) {
        printf("impossibile");
        return -1;
    }

    if (a < b) {
        return a;
    }

    return divisione(a - b, b);
}

int main() {
    int a = 11, b = 3;
    printf("%d / %d = %d resto %d", a, b, divisione(a, b), resto(a, b));
}
```

4.2.7 Scrivi e utilizza tre diverse versioni di una funzione che stampa un array elemento per elemento. Le tre versioni sono:

1. Versione iterativa, cioè che usa un ciclo for

2. Versione ricorsiva che riceve come parametri solo l'array e il numero di elementi contenuti nell'array
3. Versione ricorsiva che riceve come parametri l'array e le posizioni di inizio e fine dell'array che vanno considerate.

Variante: Stampa l'array al contrario

Soluzione

```c
#include <stdio.h>

void stampa1(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
}
void stampa2(int a[], int n) {
    if (n < 1) return;
    stampa2(a, n-1);
    printf("%d ", a[n-1]);
}
void stampa3(int a[], int inizio, int fine) {
    if (inizio > fine) return;
    printf("%d ", a[inizio]);
    stampa3(a, inizio+1, fine);
}

void stampaContrario1(int a[], int n) {
    int i;
    for (i = n-1; i >= 0; i--) {
        printf("%d ", a[i]);
    }
}
void stampaContrario2(int a[], int n) {
    if (n < 1) return;
    printf("%d ", a[n-1]);
    stampaContrario2(a, n-1);
}
void stampaContrario3(int a[], int inizio, int fine) {
    if (inizio > fine) return;
    printf("%d ", a[fine]);
    stampaContrario3(a, inizio, fine-1);
}

int main() {
    int a[] = {1,2,3,4,5,6};
    int n  = 6;

    stampa1(a, n);
    printf("\n");
    stampa2(a, n);
    printf("\n");
    stampa3(a, 0, n-1);
    printf("\n");
    stampaContrario1(a, n);
    printf("\n");
    stampaContrario2(a, n);
    printf("\n");
    stampaContrario3(a, 0, n-1);
    printf("\n");
}
```

4.2.8 Scrivi e utilizza le versioni iterativa e ricorsiva di una funzione che riceve un array e inverte l'ordine dei numeri in esso contenuti.

Soluzione

```c
#include <stdio.h>

void invertiI(int a[], int n) {
    int i, tmp;
    for (i = 0; i < n/2; i++) {
        tmp = a[i];
        a[i] = a[n-1-i];
        a[n-1-i] = tmp;
    }
}

void invertiR(int a[], int inizio, int fine) {
    int tmp;
    if (inizio >= fine) {
        return;
    }

    tmp = a[inizio];
    a[inizio] = a[fine];
    a[fine] = tmp;

    invertiR(a, inizio+1, fine-1);
}

void stampa(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

int main() {
    int a[] = {4,7,4,8,6,7};
    int n = sizeof(a)/sizeof(a[0]);
    stampa(a, n);
    invertiI(a, n);
    stampa(a, n);
    invertiR(a, 0, n-1);
    stampa(a, n);
}
```

4.2.9 Funzione ricorsiva per trovare i valori massimi e minimi di un vettore.

Soluzione

```c
// 4.2.9    Funzione ricorsiva per trovare i valori massimi e minimi di un
// vettore.

#include <stdio.h>

typedef struct {
    int min;
    int max;
} coppia;
```

```c
int maxr(int a[], int n) {
    int resto;
    if (n <= 0) {
        printf("Hai passato un array vuoto!\n");
        return -1;
    }
    if (n == 1) {
        return a[0];
    }
    resto = maxr(a, n-1);
    if (resto > a[n-1]) {
        return resto;
    } else {
        return a[n-1];
    }
}

int minr(int a[], int n) {
    int resto;
    if (n <= 0) {
        printf("Hai passato un array vuoto!\n");
        return -1;
    }
    if (n == 1) {
        return a[0];
    }
    resto = minr(a, n-1);
    if (resto < a[n-1]) {
        return resto;
    } else {
        return a[n-1];
    }
}

coppia minmaxr(int a[], int n) {
    coppia ris;
    if (n <= 0) {
        printf("Hai passato un array vuoto!\n");
        ris.min = -1;
        ris.max = -1;
        return ris;
    }
    if (n == 1) {
        ris.min = a[0];
        ris.max = a[0];
        return ris;
    }
    ris = minmaxr(a, n-1);
    if (a[n-1] < ris.min) {
        ris.min = a[n-1];
    }
    if (a[n-1] > ris.max) {
        ris.max = a[n-1];
    }
    return ris;
}

void minmaxr2(int a[], int n, int *min, int *max) {
    coppia ris;
    if (n <= 0) {
        printf("Hai passato un array vuoto!\n");
        *min = -1;
        *max = -1;
        return;
    }
    if (n == 1) {
```

```
        *min = a[0];
        *max = a[0];
        return;
    }
    minmaxr2(a, n-1, min, max);
    if (a[n-1] < *min) {
        *min = a[n-1];
    }
    if (a[n-1] > *max) {
        *max = a[n-1];
    }
}

int main() {
    int a[] = {5,7,2,6,8,4};
    int n = sizeof(a)/sizeof(int);
    coppia minmax = minmaxr(a, n);
    int min, max;
    printf("max: %d\n", minmax.max);
    printf("min: %d\n", minmax.min);

    minmaxr2(a, n, &min, &max);
    printf("max: %d\n", max);
    printf("min: %d\n", min);
}
```

4.2.10 Funzione ricorsiva per cercare un valore all'interno di un vettore non ordinato (ricerca lineare). Risultato: –1 se non lo trova, altrimenti l'indice della posizione dove è stato trovato.

Soluzione

```
#include <stdio.h>

int trova(int a[], int n, int val) {
    if (n <= 0) {
        return -1;
    }
    if (a[n-1] == val) {
        return n-1;
    } else {
        trova(a,n-1,val);
    }
}

int main() {
    int a[] = {5, 7, 2, 6, 8, 4};
    int n = sizeof(a) / sizeof(int);
    int val = 4;
    printf("La posizione di %d: %d\n", val, trova(a,n,val));
}
```

4.2.11 Scrivere una funzione ricorsiva per cercare un valore all'interno di un vettore ordinato (ricerca dicotomica). Risultato: – 1 o l'indice.
Indicazioni: se il vettore è ordinato, iniziate a controllare il numero a metà del vettore, se il numero controllato è più piccolo di quello da trovare, cercherò nella parte del vettore a sinistra (richiamo la funzione indicando come intervallo in cui cercare solo la parte a sinistra), se è più piccolo dovrò cercare a destra, se è uguale l'ho trovato.

Soluzione

```c
#include <stdio.h>

int cerca(int a[], int num, int i, int f){
    int medio;
    if(i>f){
        return -1;
    }
    medio = (i+f)/2;
    if (a[medio] == num) {
        return medio;
    }
    if (num < a[medio]) {
        return cerca(a, num, i, medio-1);
    } else {
        return cerca(a, num, medio+1, f);
    }
}

int main() {
    int a[] = {2,5,7,11,19,25,27,37,50,68,99};
    int n = sizeof(a)/sizeof(int);

    printf("%d", cerca(a, 50, 0, n-1));
}
```

4.2.12 Scrivere una funzione ricorsiva che riceve un array di interi e restituisce true se gli elementi sono tutti dispari e false se non lo sono.

Soluzione

```c
#include <stdio.h>

int tuttoDispari(int a[], int n) {
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return (a[n-1] % 2 == 1);
    }
    else {
       return (a[n-1] % 2 == 1) && tuttoDispari(a, n-1);
    }
}

int main() {
    int a[] = {4,3,7,8,4,2,6};
    int b[] = {1,7,3,5,9};
    int c[] = {};
    printf("%d\n", tuttoDispari(a, 7));
    printf("%d\n", tuttoDispari(b, 5));
    printf("%d", tuttoDispari(c, 0));
}
```

4.2.13 Scrivere una funzione ricorsiva che riceve un array di interi e restituisce il prodotto degli elementi negativi.

Soluzione

```c
#include <stdio.h>
```

```c
int prodottoNegativi(int a[], int n) {
    if (n <= 0) {
        return 1;
    }
    if (a[n-1] < 0) {
        return a[n-1] * prodottoNegativi(a, n-1);
    } else {
        return prodottoNegativi(a, n-1);
    }
}

int main() {
    int a[] = {-3,-4};
    int b[] = {1,7,3,5,9};
    int c[] = {};
    printf("%d\n", prodottoNegativi(a, 2));
    printf("%d\n", prodottoNegativi(b, 5));
    printf("%d\n", prodottoNegativi(c, 0));
}
```

4.2.14 Scrivi una funzione di ordinamento ricorsiva.

Soluzione

```c
// 4.2.14 - Scrivi una funzione di ordinamento ricorsiva.

#include <stdio.h>

void ordinaArray(int a[], int n) {
    int i, tmp;
    if (n <= 1) {
        return;
    }
    for (i = 0; i < n-1; i++) {
        if (a[i] > a[i+1]){
            tmp = a[i];
            a[i] = a[i+1];
            a[i+1] = tmp;
        }
    }
    ordinaArray(a, n-1);
}

void stampaArray(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

int main() {
    int a[] = {1,6,3,8,5,7,7,3,2};
    int n = sizeof(a) / sizeof(int);

    stampaArray(a, n);
    ordinaArray(a, n);
    stampaArray(a, n);
}
```

4.2.15 Scrivi e utilizza le versioni iterativa e ricorsiva di una funzione che dato un array di interi, calcola e restituisce la massima differenza tra due numeri consecutivi.

Soluzione

```c
// 4.2.15 - Scrivi e utilizza le versioni iterativa e ricorsiva di
// una funzione che dato un array di interi, calcola e restituisce la massima
// differenza tra due numeri consecutivi.


#include <stdio.h>
#include <stdlib.h>

int maxDiffR(int a[], int n) {
    int diff1, diff2;
    if (n <= 1) {
        return 0;
    }
    else if (n == 2) {
        return abs(a[1] - a[0]);
    }
    else {
        diff1 = abs(a[n-1] - a[n-2]);
        diff2 = maxDiffR(a, n-1);
        if (diff1 > diff2) {
            return diff1;
        } else {
            return diff2;
        }
    }
}

void stampaArray(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

int main() {
    int a[] = {1,6,3,8,5,7,7,3,2};
    int n = sizeof(a) / sizeof(int);

    stampaArray(a, n);
    printf("Massima differenza tra due numeri consecutivi: %d", maxDiffR(a, n));
}
```

4.2.16 Scrivi una funzione ricorsiva che riceva un array di punti (definisci la struct punto) e che restituisca il punto più vicino all'origine (definisci e usa la funzione distanza tra punti)

Soluzione

```c
// 4.2.16 - Scrivi una funzione ricorsiva che riceva un array di
// punti (definisci la struct punto) e che restituisca il punto più vicino
// all'origine (definisci e usa la funzione distanza tra punti)

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
```

```c
#define N 10

typedef struct {
    float x, y;
} Punto;

float distanza(Punto p) {
    return sqrt(p.x*p.x + p.y*p.y);
}

Punto piuVicino(Punto elenco[], int n) {
    Punto vicino;
    float dist;
    if (n <= 0) {
        return vicino;
    } else if (n == 1) {
        return elenco[0];
    } else {
        dist = distanza(elenco[n-1]);
        vicino = piuVicino(elenco, n-1);
        if (distanza(vicino) < dist) {
            return vicino;
        } else {
            return elenco[n-1];
        }
    }
}

int main() {
    Punto elenco[N], vicino;
    float x, y;
    int i;
    srand(time(NULL));
    for (i = 0; i < N; i++) {
        x = ((float)rand() / ((float)RAND_MAX/20)) - 10;
        y = ((float)rand() / ((float)RAND_MAX/20)) - 10;
        elenco[i].x = x;
        elenco[i].y = y;
        printf("(%.2f, %.2f)\n", elenco[i].x, elenco[i].y);
    }
    vicino = piuVicino(elenco, N);
    printf("\n\n(%.2f, %.2f)\n", vicino.x, vicino.y);
}
```

5. Struct

5.1.1 Scrivi un programma che definisca un punto e poi dati due punti scelti da te sia in grado, per mezzo di due apposite funzioni, di calcolare la distanza fra due punti e trovare il punto medio.

Soluzione

```c
// 5.1.1 - Scrivi un programma che definisca un punto e poi dati
// due punti scelti da te sia in grado, per mezzo di due apposite funzioni, di
// calcolare la distanza fra due punti e trovare il punto medio.

#include <math.h>
#include <stdio.h>

typedef struct {
    float x;
```

```
    float y;
} punto;

float distanza(punto a, punto b) {
    return sqrt((b.x - a.x) * (b.x - a.x) + (b.y - a.y) * (b.y - a.y));
}

punto medio(punto a, punto b) {
    punto c;
    c.x = (a.x + b.x) / 2;
    c.y = (a.y + b.y) / 2;
    return c;
}

int main() {
    punto a, b, c;
    a.x = 1.1;
    a.y = 3.5;
    b.x = 0.5;
    b.y = 3;

    printf("La distanza tra (%g, %g) e (%g, %g) e' %g\n", a.x, a.y, b.x, b.y,
           distanza(a, b));
    c = medio(a, b);
    printf("il punto medio tra (%g, %g) e (%g, %g) e' (%g, %g)", a.x, a.y, b.x,
           b.y, c.x, c.y);
}
```

5.1.2 Definisci una struct punto. Scrivi ed utilizza poi le funzioni:

1. Funzione per stampare una rappresentazione del punto (es. A(1,2))
2. Funzione per calcolare il punto medio
3. Funzione per calcolare la retta passante tra due punti. Dovrai anche definire la struct retta che contiene le informazioni necessarie ad identificare una retta e definire una funzione per stamparla in modo standard (y=mx+c)
4. Funzione che verifica se tre punti sono allineati.

Soluzione

5.1.3 Scrivi un programma per gestire un corso. Al corso sono iscritte delle persone, massimo 30. Per ogni persona dovete sapere nome e cognome e ad ognuno alla fine del corso viene assegnato un voto da 1 a 10. Alla fine stampa tutti i dati di ogni studente, compresi i voti.

5.1.4 Completa il seguente programma in cui è stato dichiarato un array di stringhe in cui sono scritti nomi, cognomi e date di nascita e morte di esploratori famosi. Devi definire le strutture dati adeguate a contenere queste informazioni: una struct esploratore e un array di esploratori. Usa la funzione sscanf per leggere i dati dalla stringa dati e riempi le strutture dati da te definite. scrivi poi una serie di funzioni che lavorino sulle tue strutture dati:

1. una funzione che stampa l'array di esploratori.
2. una funzione che ordina l'array in ordine alfabetico secondo il nome degli esploratori
3. una funzione che ordina l'array in ordine crescente secondo le date di nascita degli esploratori

Consigli e indicazioni extra:

1. La funzione sscanf l'abbiamo usata pochissimo in passato ma trovate facilmente online indicazioni su come si usa.
2. Alcuni potrebbero trovare comodo definire una struttura dati contenente l'array degli esploratori insieme al numero di esploratori contenuti nell'array.

Di seguito i dati da utilizzare:

```
int main(){
   char dati[15][50] = {
        "Marco Polo 1254 1324",
        "Cristoforo Colombo 1451 1506",
        "Amerigo Vespucci 1454 1512",
        "Francisco Pizarro 1475 1541",
        "Ferdinando Magellano 1480 1521",
        "Hernan Cortez 1485 1547",
        "Walter Raleigh 1552 1618",
        "Henry Hudson 1570 1611",
        "James Cook 1728 1779",
        "Charles Darwin 1809 1882",
        "Kit Carson 1809 1866",
        "David Livingstone 1813 1873",
        "Charles Foucauld 1858 1916",
        "Ronald Amundsen 1872 1928",
        "Ernest Shakleton 1874 1922",
   };
}
```

Soluzione

```c
#include <stdio.h>
#include <string.h>

#define N 100

typedef struct {
    char nome[N];
    char cognome[N];
    int nascita, morte;
} esploratore;

void stampa(esploratore lista[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("- %s %s %d %d\n", lista[i].nome, lista[i].cognome,
               lista[i].nascita, lista[i].morte);
    }
}

void ordinaPerNome(esploratore lista[], int n) {
    int i, j;
    esploratore tmp;

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - 1 - i; j++) {
            if (strcmp(lista[j].nome, lista[j + 1].nome) > 0 ||
                (strcmp(lista[j].nome, lista[j + 1].nome) == 0 &&
                 strcmp(lista[j].cognome, lista[j + 1].cognome) > 0)) {
                tmp = lista[j];
                lista[j] = lista[j + 1];
                lista[j + 1] = tmp;
            }
        }
    }
}
```

```c
void ordinaPerNascita(esploratore lista[], int n) {
    int i, j;
    esploratore tmp;

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - 1 - i; j++) {
            if (lista[j].nascita > lista[j+1].nascita) {
                tmp = lista[j];
                lista[j] = lista[j + 1];
                lista[j + 1] = tmp;
            }
        }
    }
}

int main() {
    char dati[15][50] = {
        "Marco Polo 1254 1324",           "Cristoforo Colombo 1451 1506",
        "Amerigo Vespucci 1454 1512",     "Francisco Pizarro 1475 1541",
        "Ferdinando Magellano 1480 1521", "Hernan Cortez 1485 1547",
        "Walter Raleigh 1552 1618",       "Henry Hudson 1570 1611",
        "James Cook 1728 1779",           "Charles Darwin 1809 1882",
        "Kit Carson 1809 1866",           "David Livingstone 1813 1873",
        "Charles Foucauld 1858 1916",     "Ronald Amundsen 1872 1928",
        "Ernest Shakleton 1874 1922",
    };

    esploratore lista[N];
    int nesploratori = 0;

    for (nesploratori = 0; nesploratori < 15; nesploratori++) {
        sscanf(dati[nesploratori], "%s %s %d %d", lista[nesploratori].nome,
               lista[nesploratori].cognome, &lista[nesploratori].nascita,
               &lista[nesploratori].morte);
    }

    printf("Prima stampa:\n");
    stampa(lista, nesploratori);

    ordinaPerNome(lista, nesploratori);
    printf("\n\nOrdinati per nome:\n");
    stampa(lista, nesploratori);

    ordinaPerNascita(lista, nesploratori);
    printf("\n\nOrdinati per nascita:\n");
    stampa(lista, nesploratori);
}
```

Soluzione alternativa (con struttura esploratori)

```c
#include <stdio.h>
#include <string.h>

#define N 100

typedef struct {
    char nome[N];
    char cognome[N];
    int nascita, morte;
} esploratore;

typedef struct {
    esploratore elenco[N];
    int n;
} esploratori;

void aggiungiEsploratore(esploratori lista, esploratore e) {
```

```
    lista.elenco[lista.n] = e;
    lista.n++;
}

void stampa(esploratori lista) {
    int i;
    for (i = 0; i < lista.n; i++) {
        printf("- %s %s %d %d\n", lista.elenco[i].nome, lista.elenco[i].cognome,
               lista.elenco[i].nascita, lista.elenco[i].morte);
    }
}

void ordinaPerNome(esploratori lista) {
    int i, j;
    esploratore tmp;

    for (i = 0; i < lista.n - 1; i++) {
        for (j = 0; j < lista.n - 1 - i; j++) {
            if (strcmp(lista.elenco[j].nome, lista.elenco[j + 1].nome) > 0 ||
                (strcmp(lista.elenco[j].nome, lista.elenco[j + 1].nome) == 0 &&
                 strcmp(lista.elenco[j].cognome, lista.elenco[j + 1].cognome) > 0)) {
                tmp = lista.elenco[j];
                lista.elenco[j] = lista.elenco[j + 1];
                lista.elenco[j + 1] = tmp;
            }
        }
    }
}

void ordinaPerNascita(esploratori lista) {
    int i, j;
    esploratore tmp;

    for (i = 0; i < lista.n - 1; i++) {
        for (j = 0; j < lista.n - 1 - i; j++) {
            if (lista.elenco[j].nascita > lista.elenco[j+1].nascita) {
                tmp = lista.elenco[j];
                lista.elenco[j] = lista.elenco[j + 1];
                lista.elenco[j + 1] = tmp;
            }
        }
    }
}

int main() {
    char dati[15][50] = {
        "Marco Polo 1254 1324",           "Cristoforo Colombo 1451 1506",
        "Amerigo Vespucci 1454 1512",     "Francisco Pizarro 1475 1541",
        "Ferdinando Magellano 1480 1521", "Hernan Cortez 1485 1547",
        "Walter Raleigh 1552 1618",       "Henry Hudson 1570 1611",
        "James Cook 1728 1779",           "Charles Darwin 1809 1882",
        "Kit Carson 1809 1866",           "David Livingstone 1813 1873",
        "Charles Foucauld 1858 1916",     "Ronald Amundsen 1872 1928",
        "Ernest Shakleton 1874 1922",
    };

    esploratori lista;

    for (lista.n = 0; lista.n < 15; lista.n++) {
        sscanf(dati[lista.n], "%s %s %d %d", lista.elenco[lista.n].nome,
               lista.elenco[lista.n].cognome, &lista.elenco[lista.n].nascita,
               &lista.elenco[lista.n].morte);
    }

    printf("Prima stampa:\n");
    stampa(lista);
```

```
    ordinaPerNome(lista);
    printf("\n\nOrdinati per nome:\n");
    stampa(lista);

    ordinaPerNascita(lista);
    printf("\n\nOrdinati per nascita:\n");
    stampa(lista);
}
```

5.1.5 Scrivi e utilizza un insieme di funzioni da utilizzare per fare calcoli con le frazioni. In pratica devono essere definite le funzioni per fare almeno le 4 operazioni (magari anche altre operazioni). Deve anche essere fatta una funzione per effettuare la semplificazione delle frazioni (puoi decidere se creare e usare anche un'altra funzione mcd o utilizzare un altro metodo). Scrivi tutte queste funzioni in un file separato e poi utilizza queste funzioni per fare dei calcoli con le frazioni in un programma.
In una prima versione del programma considera numeratore e denominatore come due variabili intere separate. In una seconda versione definisci invece una struct frazione che contiene numeratore e denominatore. Le funzioni dovranno essere modificate di conseguenza.

Soluzione

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int num, den;
} frazione;

void semplifica(frazione* f) {
    int n;
    int segno;
    if (f->num * f->den >= 0 ){
        segno = 1;
    } else {
        segno = -1;
    }
    f->num = abs(f->num);
    f->den = abs(f->den);
    for (n = 2; n <= f->num && n <= f->den; n++) {
        while (f->num % n == 0 && f->den % n == 0) {
            f->num /= n;
            f->den /= n;
        }
    }
    f->num *= segno;
}

frazione somma(frazione a, frazione b) {
    frazione c;
    c.num = a.num * b.den + b.num * a.den;
    c.den = a.den * b.den;
    semplifica(&c);
    return c;
}

frazione sottrai(frazione a, frazione b) {
    frazione c;
    c.num = a.num * b.den - b.num * a.den;
    c.den = a.den * b.den;
    semplifica(&c);
    return c;
```

```
}

frazione moltiplica(frazione a, frazione b) {
    frazione c;
    c.num = a.num * b.num;
    c.den = a.den * b.den;
    semplifica(&c);
    return c;
}

frazione dividi(frazione a, frazione b) {
    frazione c;
    c.num = a.num * b.den;
    c.den = a.den * b.num;
    semplifica(&c);
    return c;
}

int main() {
    frazione a, b, c;
    a.num = 1;
    a.den = 2;
    b.num = 3;
    b.den = -2;
    c = somma(a, b);
    printf("%d/%d + %d/%d = %d/%d\n", a.num, a.den, b.num, b.den, c.num , c.den);
    c = sottrai(a, b);
    printf("%d/%d - %d/%d = %d/%d\n", a.num, a.den, b.num, b.den, c.num , c.den);
    c = moltiplica(a, b);
    printf("%d/%d x %d/%d = %d/%d\n", a.num, a.den, b.num, b.den, c.num , c.den);
    c = dividi(a, b);
    printf("%d/%d : %d/%d = %d/%d\n", a.num, a.den, b.num, b.den, c.num , c.den);
}
```

5.1.6 Scrivi un programma in cui è definita una struttura studente. Dello studente ci interessano il nome, i vori presi e la media dei voti. Il programma deve istanziare uno o più studenti e aggiungere una serie di voti casuali (anche mezzi voti ma non tutti i decimali possibili). Voglio creare uno studente e poi usare delle funzioni per aggiungere voti e calcolare la media. Infine stampare lo studente con tutti i suoi dati con una funzione apposita.

Soluzione

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

#define N 50

typedef struct {
    char nome[N];
    char cognome[N];
    float voti[N];
    int nvoti;
    float media;
} studente;

void calcolaMedia(studente* stud) {
    int i;
    stud->media = 0;
    for (i = 0; i < stud->nvoti; i++) {
        stud->media += stud->voti[i];
    }
```

```
    stud->media /= stud->nvoti;
}

void stampaStudenti(studente elenco[], int nstudenti) {
    int i, j;
    for (i = 0; i < nstudenti; i++) {
        printf("- %s %s: ", elenco[i].cognome, elenco[i].nome);
        for (j = 0; j < elenco[i].nvoti; j++) {
            printf("%g ", elenco[i].voti[j]);
        }
        printf(" - media: %g\n", elenco[i].media);
    }
}

int main() {
    int i;
    studente elenco[N];
    int nstudenti;

    srand(time(NULL));

    strcpy(elenco[0].nome, "Filippo");
    strcpy(elenco[0].cognome, "Brambilla");
    elenco[0].nvoti = 0;
    for (i = 0; i<3; i++) {
        elenco[0].voti[i] = (rand() % 19 + 2) / 2.0;
        elenco[0].nvoti ++;
    }
    calcolaMedia(&elenco[0]);

    strcpy(elenco[1].nome, "Andrea");
    strcpy(elenco[1].cognome, "Garofoli");
    elenco[1].nvoti = 0;
    for (i = 0; i<3; i++) {
        elenco[1].voti[i] = (rand() % 19 + 2) / 2.0;
        elenco[1].nvoti ++;
    }
    calcolaMedia(&elenco[1]);

    nstudenti = 2;

    stampaStudenti(elenco, nstudenti);
}
```

5.1.7 Crea una struttura, che chiamerai Lista, che gestisca un array di interi e contenga come dati oltre all'array anche due interi che indicano la capienza massima dell'array e il numero di valori effettivamente contenuti nell'array. Scrivi e utilizza poi le seguenti funzioni:

1. Funzione per aggiungere un numero in fondo all'array
2. Funzione per aggiungere un numero all'inizio dell'array (gli altri scalano)
3. Funzione per aggiungere un numero in una posizione a piacere dell'array (gli altri numeri scalano)
4. Funzione per eliminare il numero presente in una determinata posizione dell'array (non deve rimanere il buco)
5. Funzione per concatenare due liste

Soluzione

```c
#include <stdio.h>

#define N 1000

// 5.1.6    Crea una struttura, che chiamerai Lista, che gestisca un array di
// interi e contenga come dati oltre all'array anche due interi che indicano la
// capienza massima dell'array e il numero di valori effettivamente contenuti
// nell'array. Scrivi e utilizza poi le seguenti funzioni:
typedef struct {
    int a[N];
    int capienza;
    int n;
} Lista;

// 1.   Funzione per aggiungere un numero in fondo all'array
int push_back(Lista *l, int num) {
    if (l->n == l->capienza) {
        return 0;
    }
    l->a[l->n] = num;
    l->n++;
    return 1;
}

// 2.   Funzione per aggiungere un numero all'inizio dell'array (gli altri scalano)
int push_front(Lista* l, int num) {
    int i;
    if (l->n == l->capienza) {
        return 0;
    }
    l->n++;
    for (i = l->n-1; i > 0; i--) {
        l->a[i] = l->a[i-1];
    }
    l->a[0] = num;
    return 1;
}

// 3.   Funzione per aggiungere un numero in una posizione a piacere
// dell'array (gli altri numeri scalano)
int push(Lista* l, int num, int pos) {
    int i;
    if (l->n == l->capienza || pos < 0 || pos >= l->n) {
        return 0;
    }
    l->n++;
    for (i = l->n-1; i > pos; i--) {
        l->a[i] = l->a[i-1];
    }
    l->a[pos] = num;
    return 1;
}

// 4.   Funzione per eliminare il numero
// presente in una determinata posizione dell'array (non deve rimanere il buco)
int pop(Lista* l, int pos) {
    int i;
    if (pos < 0 || pos >= l->n) {
        return 0;
    }
    for (i = pos; i < l->n-1; i++) {
        l->a[i] = l->a[i+1];
    }
    l->n--;
```

```
    return 1;
}

// 5.   Funzione per concatenare due liste
Lista concatena(Lista l1, Lista l2) {
    Lista ris = l1;
    int i;

    for (i = 0; i < l2.n && ris.n < ris.capienza; i++, ris.n++) {
        ris.a[ris.n] = l2.a[i];
    }

    return ris;
}

void stampa(Lista l) {
    int i;
    for (i = 0; i < l.n; i++) {
        printf("%d ", l.a[i]);
    }
    printf("\n");
}


int main() {
    Lista lista;
    Lista l2;
    lista.capienza = N;
    lista.n = 0;
    lista.capienza = N;
    lista.n = 0;

    push_back(&lista, 2);
    push_back(&lista, 3);
    push_back(&lista, 5);
    push_back(&lista, 6);
    push_front(&lista, 1);
    push_front(&lista, 0);
    push(&lista, 4, 4);
    pop(&lista, 3);
    push(&lista, 3, 3);

    stampa(lista);

}
```

5.1.8 Completa un programma che legga i dati contenuti in un array (riportato di seguito) e li inserisca in opportune strutture dati. Di questi dati il programma deve fare le seguenti cose:

1. stamparli sullo schermo (con una funzione)
2. ordinarli in ordine decrescente di voto (con una funzione)
3. stamparli nuovamente in ordine (usando la funzione del punto 1)

Il programma da completare:

```
#include <stdio.h>

int main() {
    char dati[][50] = {
        "Amici 7.00",
        "Biella 9.00",
        "Brescia 6.00",
        "Carolla 8.00",
        "Cunegatti 7.00",
```

```
        "DeBella 4.00",
        "DeVecchi 9.00",
        "Fumagalli 7.00",
        "Galimberti 9.00",
        "Germano 9.00",
        "Gubellini 9.00",
        "Lepore 5.00",
        "Maconi 6.00",
        "Mariani 8.00",
        "Mattavelli 9.00",
        "Oggiano 4.00",
        "Passoni 5.00",
        "Pastori 4.00",
        "Pirovano 7.00",
        "Rudi 10.00",
        "Russell 6.00",
        "Sogos 4.00",
        "Tezza 6.00",
        "Varisco 8.00",
    };
    int n = 24, i;
    for (i = 0; i < n; i++) {
        printf("%s\n", dati[i]);
    }
}
```

5.1.9 Chiedi all'utente i dati riguardanti i giocatori di una squadra di basket. I dati sono: nome, cognome, ruolo, numero di maglia e altezza. Dopo aver letto i dati stampa il quintetto più alto possibile che contenga tutti i ruoli he sono: playmaker, guardia, ala piccola, ala grande, centro.

Variante: Scrivi e usa una funzione che dato l'elenco e un ruolo, restituisce il giocatore più alto che ha quel ruolo.

Soluzione

```
// 5.1.9    Chiedi all'utente i dati riguardanti i giocatori di una squadra di
// basket. I dati sono: nome, cognome, ruolo, numero di maglia e altezza. Dopo
// aver letto i dati stampa il quintetto più alto possibile che contenga tutti i
// ruoli he sono: playmaker, guardia, ala piccola, ala grande, centro.
// Variante:
// Scrivi e usa una funzione che dato l'elenco e un ruolo, restituisce il
// giocatore più alto che ha quel ruolo.

#include <stdio.h>
#include <string.h>

#define N 100
#define M 20

typedef struct {
    char nome[N];
    char ruolo[M];
    int numero;
    int altezza;
} Giocatore;

void stampaGiocatore(Giocatore g) {
    printf("%30s %20s %4d %5d\n", g.nome, g.ruolo,
               g.numero, g.altezza);
}

void stampaLista(Giocatore elenco[], int n) {
```

```
    int i;
    for (i = 0; i < n; i++) {
        printf("%30s %20s %4d %5d\n", elenco[i].nome, elenco[i].ruolo,
               elenco[i].numero, elenco[i].altezza);
    }
}

Giocatore scegli(Giocatore elenco[], int n, char ruolo[]) {
    int imax = -1;
    int i;
    for (i = 0; i < n; i++) {
        if (strcmp(elenco[i].ruolo, ruolo) == 0) {
            if (imax == -1 || elenco[i].altezza > elenco[imax].altezza) {
                imax = i;
            }
        }
    }
    if (imax == -1) {
        printf("Non ci sono giocatori con quel ruolo!!\n");
    }
    return elenco[imax];
}

void creaQuintetto(Giocatore elenco[], int n, Giocatore quintetto[]) {
    char ruoli[5][M] = {"playmaker", "guardia", "ala piccola", "ala grande", "centro"};
    int i;
    for (i = 0; i < 5; i++) {
        quintetto[i] = scegli(elenco, n, ruoli[i]);
    }
}

int main() {
    Giocatore elenco[N], quintetto[5];
    int n = 0, i, risp;

    while (n < N) {
        printf("Vuoi inserire un giocatore? (1 si, 0 no)");
        scanf("%d", &risp);
        while (risp != 1 && risp != 0) {
            printf("Risposta non prevista\n");
            printf("Vuoi inserire un giocatore? (1 si, 0 no)");
            scanf("%d", &risp);
        }
        if (risp == 0) {
            break;
        } else {
            printf("Scrivi il nome (e cognome): ");
            scanf(" %99[^\n]", elenco[n].nome);
            printf("Scrivi il ruolo: ");
            scanf(" %19[^\n]", elenco[n].ruolo);
            printf("Scrivi il numero di maglia: ");
            scanf(" %d", &elenco[n].numero);
            printf("Scrivi l'altezza (in cm): ");
            scanf(" %d", &elenco[n].altezza);
            n++;
        }
    }
    i = 0;
    // printf("%30s, %20s, %2d, %3d\n", elenco[i].nome, elenco[i].ruolo,
    //            elenco[i].numero, elenco[i].altezza);
    printf("\nElenco dei giocatori inseriti:\n");
    stampaLista(elenco, n);

    printf("\n\n");
    stampaGiocatore(scegli(elenco, n, "ala piccola"));

    printf("\nStampa del quintetto piu' alto:\n");
```

```
    creaQuintetto(elenco, n, quintetto);
    stampaLista(quintetto, 5);
}
```

6. Files in C

6.1.1 Scrivi un programma che scriva su un file un elenco di nomi, chiuda il file, lo riapra in lettura lo legga tutto e stampi ciò che ha letto.

6.1.2 Scrivi e usa una funzione che conta il numero di caratteri contenuti in un file.

Variante: Conta le lettere, non i caratteri.

Soluzione

```
// 6.1.2    Scrivi e usa una funzione che conta il numero di caratteri contenuti in un file
// variante: conta le lettere

#include <stdio.h>

int main() {
    FILE *fp;
    int nchar;
    char lettera;

    fp = fopen("6-1-2.txt", "r");
    if (fp == NULL) {
        printf("File non trovato\n");
        return 1;
    }

    nchar = 0;
    while(!feof(fp)) {
        fscanf(fp ,"%c", &lettera);
        // variante
        if ((lettera >= 'a' && lettera <= 'z') || (lettera >= 'A' && lettera <= 'Z'))
            nchar++;
    }

    printf("%d", nchar);

    return 0;
}
```

6.1.3 Scrivi una funzione che conta e restituisce il numero di righe di un file.
Scrivi poi un'altra funzione che calcola per ogni riga il numero di caratteri contenuti nella riga. I valori vanno inseriti in un array o un vettore da restituire al main.

1. Variante complicata che prevede l'uso di strtok: invece dei caratteri conta il numero di parole per riga

Soluzione

```
// 6.1.3    Scrivi una funzione che conta e restituisce il numero di righe di un
// file. Scrivi poi un'altra funzione che calcola per ogni riga il numero di
// caratteri contenuti nella riga. I valori vanno inseriti in un array o un
// vettore da restituire al main.
// 1.   Variante complicata che prevede l'uso di
```

```c
// strtok: invece dei caratteri conta il numero di parole per riga


// per ora non facciamo la variante

#include <stdio.h>
#include <string.h>

#define N 200

int contaCaratteri(char riga[]) {
    return strlen(riga);
}

int contaLettere(char riga[]) {
    int ris = 0, i;
    for (i = 0; riga[i] != 0; i++) {
        if ((riga[i] >= 'a' && riga[i] <= 'z') || (riga[i] >= 'A' && riga[i] <= 'Z')) {
            ris++;
        }
    }
    return ris;
}

int main() {
    FILE *fp;
    char riga[N];
    int i;

    fp = fopen("6-1-3.txt", "r");
    if (fp == NULL) {
        printf("Errore nell'apertura del file.\n");
        return 1;
    }

    for (i = 0; !feof(fp); i++) {
        fgets(riga, N-1, fp);
        printf("La riga %d contiene %d lettere\n", i+1, contaLettere(riga));
    }

    fclose(fp);
}
```

6.1.4 Scrivi e usa una funzione che conta il numero di parole contenute in un file.

Soluzione

```c
// 6.1.4    Scrivi e usa una funzione che conta il numero di parole contenute in un file.

#include <stdio.h>
#include <string.h>

#define N 200

int contaParole(char nomefile[]) {
    FILE *fp;
    char parola[N];
    int ris;

    fp = fopen(nomefile, "r");
    if (fp == NULL) {
        printf("Errore nell'apertura del file.\n");
        return 1;
    }
```

```
    for (ris = 0; !feof(fp); ris++) {
        fscanf(fp, "%s", parola);
    }

    fclose(fp);

    return ris;
}


int main() {
    char nomefile[] = "6-1-3.txt";
    int nparole = contaParole(nomefile);
    printf("Nel file %s sono contenute %d parole", nomefile, nparole);
}
```

6.1.5 Scrivi e usa una funzione che legga un file e scriva in un altro file tutte le parole del primo file in ordine alfabetico

Soluzione

```
// 6.1.5 -   Scrivi e usa una funzione che legga un file e scriva in
// un altro file tutte le parole del primo file in ordine alfabetico

#include <stdio.h>
#include <string.h>

#define N 1000
#define M 50

void bsort(char parole[N][M], int nparole) {
    int i, j;
    char tmp[M];
    for (i = 0; i < nparole-1; i++) {
        for (j = 0; j < nparole-1-i; j++) {
            if (strcmp(parole[j], parole[j+1]) > 0) {
                strcpy(tmp, parole[j]);
                strcpy(parole[j], parole[j+1]);
                strcpy(parole[j+1], tmp);
            }
        }
    }
}

int main() {
    FILE *fp;
    char parole[N][M];
    int nparole, i;

    fp = fopen("6-1-5-in.txt", "r");
    if (fp == NULL) {
        printf("Errore nell'apertura nel file.\n");
        return 1;
    }

    for (nparole = 0; !feof(fp); nparole++) {
        fscanf(fp, "%s", parole[nparole]);
    }
    fclose(fp);

    bsort(parole, nparole);
```

```
    fp = fopen("6-1-5-out.txt", "w");
    if (fp == NULL) {
        printf("Errore nell'apertura nel file in scrittura.\n");
        return 1;
    }

    for (i = 0; i < nparole; i++) {
        fprintf(fp, "%s\n", parole[i]);
    }

}
```

6.1.6 Scrivi un programma che legga un file di testo e che scriva in un secondo file, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi.

Soluzione

```
// 6.1.6 - Scrivi un
// programma che legga un file di testo e che scriva in un secondo file, lo stesso
// testo modificato in modo che tutte le lettere siano minuscole. Il file di testo
// iniziale generalo tu come vuoi.

#include <stdio.h>

int main() {
    FILE *fpin, *fpout;
    char carattere;

    fpin = fopen("6-1-6-in.txt", "r");
    if (fpin == NULL) {
        printf("Errore nell'apertura nel file.\n");
        return 1;
    }

    fpout = fopen("6-1-6-out.txt", "w");
    if (fpout == NULL) {
        printf("Errore nell'apertura nel file in scrittura.\n");
        return 1;
    }

    while (!feof(fpin)) {
        fscanf(fpin, "%c", &carattere);
        if (carattere >= 'A' && carattere <= 'Z') {
            carattere += ('a'-'A');
        }
        printf("%c", carattere);
        fprintf(fpout, "%c", carattere);
    }

    fclose(fpin);
    fclose(fpout);
}
```

6.1.7 Scrivi un programma che legga un file di testo e che aggiunga alla fine di tale file, dopo un paio di righe vuote, lo stesso testo modificato in modo che tutte le lettere siano minuscole. Il file di testo iniziale generalo tu come vuoi. Attento a fare in modo che se il programma viene eseguito più volte, le righe da considerare sono solo quelle iniziali.

6.1.8 Scrivi una funzione parametrica in grado di modificare un file di testo di nome "miofile.txt" letto da disco in modo tale che, se l'ultimo elemento della linea è una virgola, la linea successiva venga eliminata

6.1.9 Scrivi un programma che scriva su un file un elenco di nomi (o numeri se preferisci), chieda poi all'utente un nome da cercare nel file e stampi sullo schermo l'esito della ricerca (trovato o no). Le operazioni di scrittura su file devono essere eseguite da una funzione che riceve come parametro la stringa contenente il nome del file. Le operazioni di ricerca devono essere svolte da una funzione che riceve come parametro la stringa contenente il nome del file e la stringa da cercare.

1. Continua l'esercizio aggiungendo una funzione che dato il nome del file e il numero di riga, cerca, se esiste, la riga e la restituisce.
2. Aggiungi anche la possibilità di chiedere all'utente se vuole aggiungere dei nomi a quelli già presenti (aprendo poi il file in modalità append e aggiungendo i nomi dati dall'utente)

Soluzione

```c
// 6.1.9 - Scrivi un programma che scriva su un file un elenco di
// nomi (o numeri se preferisci), chieda poi all'utente un nome da cercare nel
// file e stampi sullo schermo l'esito della ricerca (trovato o no). Le operazioni
// di scrittura su file devono essere eseguite da una funzione che riceve come
// parametro la stringa contenente il nome del file. Le operazioni di ricerca
// devono essere svolte da una funzione che riceve come parametro la stringa
// contenente il nome del file e la stringa da cercare.

// 1.
// Continua l'esercizio aggiungendo una funzione che dato
// il nome del file e il numero di riga, cerca, se esiste, la riga e la
// restituisce.

// 2.
// Aggiungi anche la possibilità di chiedere all'utente se
// vuole aggiungere dei nomi a quelli già presenti (aprendo poi il file in
// modalità append e aggiungendo i nomi dati dall'utente)

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

# define N 100

int scriviNomi(char nomefile[]) {
    FILE *fp;
    int i;
    fp = fopen("6-1-9-1.txt", "w");
    if (fp == NULL) {
        return 1;
    }
    for (i = 0; i < 20; i++) {
        fprintf(fp, "%d\n", rand() % 100);
    }
    fclose(fp);
    return 0;
}

int cercaNumero(char nomefile[], int numero) {
    FILE *fp;
    int numfile;
```

```
    fp = fopen("6-1-9-1.txt", "r");
    if (fp == NULL) {
        return -1;
    }
    while (!feof(fp)) {
        fscanf(fp, "%d", &numfile);
        if (numfile == numero) {
            return 1;
        }
    }
    fclose(fp);
    return 0;
}

int getRiga(char nomefile[], int numriga, char ris[]) {
    FILE *fp;
    int i;
    char riga[N];
    fp = fopen("6-1-9-1.txt", "r");
    if (fp == NULL) {
        return -1;
    }

    fgets(riga, N, fp);
    for (i = 1; !feof(fp); i++) {
        if (i == numriga) {
            strcpy(ris, riga);
            return 1;
        }
        fgets(riga, N, fp);
    }

    return 0;
}

int aggiungiRoba(char nomefile[], char roba[]) {
    FILE *fp;
    fp = fopen("6-1-9-1.txt", "a");
    if (fp == NULL) {
        return -1;
    }
    fprintf(fp, "\n%s", roba);
    fclose(fp);
}

int main() {
    int i, j, numero, ris, numriga;
    char nomefile[] = "6-1-9-1.txt";
    char riga[N];
    srand(time(NULL));
    if(scriviNomi(nomefile)) {
        printf("Errore nella scrittura del file. ");
        return 1;
    }

    printf("Che numero vuoi cercare? ");
    scanf("%d", &numero);

    ris = cercaNumero(nomefile, numero);
    if (ris == -1) {
        printf("Errore nell'apertura del file.\n");
    }
    if (ris == 1) {
        printf("Il numero e' stato trovato\n");
    } else {
        printf("Il numero non e' stato trovato\n");
    }
```

```
    // punto 1
    printf("\nChe riga vuoi? ");
    scanf("%d", &numriga);
    ris = getRiga(nomefile, numriga, riga);
    if (ris == -1) {
        printf("Errore nell'apertura del file.\n");
    }
    if (ris == 0) {
        printf("La riga non esiste\n");
    } else {
        printf("Riga %d: %s\n", numriga, riga);
    }

    // punto 2
    printf("\nVuoi aggiungere qualcosa al file? (0 no, altro sì)");
    scanf("%d", &ris);
    if (ris != 0) {
        printf("Scrivi che cosa vuoi aggiungere: ");
        scanf(" %99[^\n]", riga);
        aggiungiRoba(nomefile, riga);
    }

}
```

6.2 Che richiedono anche le strutture dati

6.2.1 Leggi i dati contenuti nel seguente file di testo. Dopo averli salvati in un'opportuna struttura dati, scrivi due funzioni:

1. Funzione che riceve i dati e restituisce l'esploratore che ha vissuto più a lungo
2. Funzione che ordina gli esploratori in ordine decrescente di età di morte.

```
Marco Polo 1254 1324
Cristoforo Colombo 1451 1506
Amerigo Vespucci 1454 1512
Francisco Pizarro 1475 1541
Ferdinando Magellano 1480 1521
Hernan Cortez 1485 1547
Walter Raleigh 1552 1618
Henry Hudson 1570 1611
James Cook 1728 1779
Charles Darwin 1809 1882
Kit Carson 1809 1866
David Livingstone 1813 1873
Charles Foucauld 1858 1916
Ronald Amundsen 1872 1928
Ernest Shakleton 1874 1922
```

6.2.2 Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati (richiesta la conoscenza delle struct o delle classi). Di questi dati il programma deve fare le seguenti cose:

1. stamparli sullo schermo
2. ordinarli in ordine decrescente di voto
3. stamparli nuovamente in ordine
4. scriverli in ordine in un secondo file che puoi nominare come vuoi

Dati da mettere nel file di testo:

```
Amici  7.00
Biella 9.00
Brescia    6.00
Carolla    8.00
Cunegatti  7.00
DeBella    4.00
DeVecchi   9.00
Fumagalli  7.00
Galimberti 9.00
Germanò    9.00
Gubellini  9.00
Lepore 5.00
Maconi 6.00
Mariani    8.00
Mattavelli 9.00
Oggiano    4.00
Passoni    5.00
Pastori    4.00
Pirovano   7.00
Rudi   10.00
Russell    6.00
Sogos  4.00
Tezza  6.00
Varisco    8.00
```

6.2.3 Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo.

```
Amici 7.00 8.00 7.00
Biella 8.00 9.50 7.00
Brescia 5.00 7.50 9.00
Carolla 7.00 8.50 8.50
Cunegatti 7.00 7.00 5.50
DeBella 4.00 4.00 4.50
DeVecchi 8.00 10.00 6.00
Fumagalli 8.50 6.00 4.50
Galimberti 9.00 9.00 9.00
Germanò 8.50 9.00 7.50
Gubellini 8.00 10.00 7.00
Lepore 5.50 4.00 3.00
Maconi 7.50 5.00 7.50
Mariani 8.00 8.00 7.00
Mattavelli 9.00 9.50 8.50
Oggiano 5.00 3.00 3.00
Passoni 5.00 6.00 6.00
Pastori 3.50 5.00 7.00
Pirovano 6.50 7.50 7.50
Rudi 9.50 10.00 10.00
Russell 7.50 4.50 5.50
Sogos 4.00 3.50 3.50
Tezza 7.00 5.50 5.50
Varisco 8.00 9.00 8.50
```

Soluzione

```c
#include <stdio.h>

#define N 50

typedef struct {
    char cognome[N];
```

```c
    float voti[N];
    int nvoti;
} Studente;

int main() {
    FILE *fp;
    Studente elenco[N];
    int nstudenti = 0;

    fp = fopen("6-2-3.txt", "r");
    if (fp == NULL) {
        printf("errore");
        return 1;
    }

    for (nstudenti = 0; !feof(fp); nstudenti++) {
        fscanf(fp, "%s %f %f %f", elenco[nstudenti].cognome,
               &elenco[nstudenti].voti[0], &elenco[nstudenti].voti[1],
               &elenco[nstudenti].voti[2]);
    }
    fclose(fp);

    for (int i = 0; i < nstudenti; i++) {
        printf("%12s %4.1f %4.1f %4.1f\n", elenco[i].cognome,
               elenco[i].voti[0], elenco[i].voti[1],
               elenco[i].voti[2]);
    }
}
```

6.2.4 Scrivi un programma che legga i dati contenuti in un file di testo (testo riportato di seguito) e li inserisca in opportune strutture dati che rappresentino il registro dei voti per una materia. Il programma deve poi calcolare la media dei voti per ogni singolo studente e aggiungerla alla struttura dati (che quindi conterrà anche una variabile media). Il programma deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in un secondo file di testo. Questa versione del programma è più difficile della precedente perché si può notare che i cognomi possono essere formati da più parole e sono separati dai voti da un ":", inoltre il numero di voti varia per ogni studente.

```
Amici: 7.00 8.00 7.00
Biella: 8.00 9.50 7.00
Brescia: 5.00 7.50 9.00
Carolla: 7.00 8.50 8.50
Cunegatti: 7.00 7.00
De Bella: 4.00 4.00 4.50 5.00
De Vecchi: 8.00 10.00 6.00
Fumagalli: 8.50 6.00 4.50
Galimberti: 9.00 9.00 9.00
Germanò: 8.50 9.00 7.50
Gubellini: 8.00 10.00 7.00
Lepore: 5.50 4.00 3.00 5.00
Maconi: 7.50 5.00
Mariani: 8.00 8.00 7.00
Mattavelli: 9.00 9.50 8.50
Oggiano: 5.00 3.00 3.00 2.00
Passoni: 5.00 6.00 6.00 7.50
Pastori: 3.50 5.00 7.00 2.00
Pirovano: 6.50 7.50 7.50
Rudi: 9.50 10.00 10.00
Russell: 7.50 4.50
Sogos: 4.00 3.50 3.50 2.00
Tezza: 7.00 5.50 5.50
Varisco: 8.00 9.00 8.50
Pirovano: 6.50 7.50 7.50
Rudi: 9.50 10.00 10.00
Russell: 7.50 4.50 5.50
Sogos: 4.00 3.50 3.50
```

```
Tezza: 7.00 5.50 5.50
Varisco: 8.00 9.00 8.50
```

Soluzione con sscanf

```c
// 6.2.4    Scrivi un programma che legga i dati contenuti in un file di testo
// (testo riportato di seguito) e li inserisca in opportune strutture dati che
// rappresentino il registro dei voti per una materia. Il programma deve poi
// calcolare la media dei voti per ogni singolo studente e aggiungerla alla
// struttura dati (che quindi conterrà anche una variabile media). Il programma
// deve mostrare tutti i dati raccolti e calcolati sullo schermo e salvarli in
// un secondo file di testo. Questa versione del programma è più difficile della
// precedente perché si può notare che i cognomi possono essere formati da più
// parole e sono separati dai voti da un ":", inoltre il numero di voti varia
// per ogni studente.

#include <stdio.h>
#include <string.h>

#define N 50
#define M 200

typedef struct {
    char cognome[N];
    float voti[N];
    float media;
    int nvoti;
} Studente;

int main() {
    FILE *fp;
    Studente elenco[N];
    int nstudenti = 0;
    char riga[M], *token;
    int ntoken;
    int i, j;

    fp = fopen("6-2-4.txt", "r");
    if (fp == NULL) {
        printf("errore");
        return 1;
    }

    for (nstudenti = 0; !feof(fp); nstudenti++) {
        fgets(riga, M, fp);

        ntoken = sscanf(riga, "%[^:]: %f %f %f %f", elenco[nstudenti].cognome,
                        &elenco[nstudenti].voti[0], &elenco[nstudenti].voti[1],
                        &elenco[nstudenti].voti[2], &elenco[nstudenti].voti[3]);
        elenco[nstudenti].nvoti = ntoken-1;
    }
    fclose(fp);

    // calcolo della media
    for (i = 0; i < nstudenti; i++) {
        elenco[i].media = 0;
        for (j = 0; j < elenco[i].nvoti; j++) {
            elenco[i].media += elenco[i].voti[j];
        }
        elenco[i].media /= elenco[i].nvoti;
    }

    // stampo tutto con una funzione
    for (i = 0; i < nstudenti; i++) {
        printf("Studente: %12s - Media: %5.2f - Voti: ", elenco[i].cognome,
               elenco[i].media);
```

```
        for (j = 0; j < elenco[i].nvoti; j++) {
            printf("%3.1f ", elenco[i].voti[j]);
        }
        printf("\n");
    }

    // scrivo in un secondo file
    fp = fopen("6-2-4-out.txt", "w");
    if (fp == NULL) {
        printf("Apertura in scrittura del file fallita.\n");
        return 2;
    }
    for (i = 0; i < nstudenti; i++) {
        fprintf(fp, "%s: ", elenco[i].cognome);
        for (j = 0; j < elenco[i].nvoti; j++) {
            fprintf(fp, "%.2f ", elenco[i].voti[j]);
        }
        fprintf(fp, "- %.2f ", elenco[i].media);
        fprintf(fp, "\n");
    }
    fclose(fp);
}
```

Soluzione con strtok

```
#include <stdio.h>
#include <string.h>

#define N 50
#define M 200


typedef struct {
    char cognome[N];
    float voti[N];
    float media;
    int nvoti;
} Studente;


int main() {
    FILE *fp;
    Studente elenco[N];
    int nstudenti = 0;
    char riga[M], *token;
    int nvoti;
    int i, j;

    fp = fopen("6-2-4.txt", "r");
    if (fp == NULL) {
        printf("errore");
        return 1;
    }

    for (nstudenti = 0; !feof(fp); nstudenti++) {
        fgets(riga, M, fp);
        token = strtok(riga, ":");
        strcpy(elenco[nstudenti].cognome, token);
        // printf("%s ", elenco[nstudenti].cognome);

        token = strtok(NULL, " ");

        for (elenco[nstudenti].nvoti = 0; token != NULL; elenco[nstudenti].nvoti += 1) {
            nvoti = elenco[nstudenti].nvoti;
```

```
            sscanf(token, "%f", &elenco[nstudenti].voti[nvoti]);
            // printf("%g ", elenco[nstudenti].voti[nvoti]);
            token = strtok(NULL, " ");
        }
        printf("\n");
    }
    fclose(fp);

    // calcolo della media
    for (i = 0; i < nstudenti; i++) {
        elenco[i].media = 0;
        for (j = 0; j < elenco[i].nvoti; j++) {
            elenco[i].media += elenco[i].voti[j];
        }
        elenco[i].media /= elenco[i].nvoti;
    }

    // stampo tutto con una funzione
    for (i = 0; i < nstudenti; i++) {
        printf("Studente: %12s - Media: %5.2f - Voti: ", elenco[i].cognome, elenco[i].media);
        for (j = 0; j < elenco[i].nvoti; j++) {
            printf("%3.1f ", elenco[i].voti[j]);
        }
        printf("\n");
    }

    // scrivo in un secondo file
    fp = fopen("6-2-4-out.txt", "w");
    if (fp == NULL) {
        printf("Apertura in scrittura del file fallita.\n");
        return 2;
    }
    for (i = 0; i < nstudenti; i++) {
        fprintf(fp, "%s: ", elenco[i].cognome);
        for (j = 0; j < elenco[i].nvoti; j++) {
            fprintf(fp, "%.2f ", elenco[i].voti[j]);
        }
        fprintf(fp, "- %.2f ", elenco[i].media);
        fprintf(fp, "\n");
    }
    fclose(fp);
}
```

6.2.5 Leggi le informazioni riguardanti una serie di libri da un file (testo di seguito) e:

1. carica i dati in un array di libri (definisci un'apposita struttura libro)
2. ordina i libri per data di scrittura e riscrivili in un altro file nello stesso formato del file originale
3. scrivi e usa una funzione che riceve i dati già caricati e che restituisce il libro più recente
4. scrivi e usa una funzione che restituisca il secolo in cui sono stati scritti più libri

```
Divina Commedia;Dante Alighieri;1321
Promessi Sposi;Alessandro Manzoni;1840
Il Decamerone;Alessandro Boccaccio;1350
L'Orlando Furioso;Ludovico Ariosto;1516
Il fu Mattia Pascal;Luigi Pirandello;1904
Ultime lettere di Jacopo Ortis,Ugo Foscolo;1802
Se questo è un uomo;Primo Levi;1947
Il barone rampante;Italo Calvino;1957
I Malavoglia;Giovanni Verga;1881
```

```
Il Principe; Niccolò Macchiavelli;1532
La Gerusalemme Liberata;Torquato Tasso;1581
Cuore;Edmondo De Amicis;1886
Il deserto dei Tartari;Dino Buzzati;1940
La coscienza di Zeno;Italo Svevo;1923
Pinocchio;Carlo Collodi;1883
Il piacere;Gabriele D'Annunzio;1889
Myricae;Giovanni Pascoli;1891
Canti;Giacomo Leopardi;1831
Uno, nessuno e centomila;Luigi Pirandello;1926
Il nome della rosa;Umberto Eco;1980
La casa in collina;Cesare Pavese;1948
Il gattopardo;Giuseppe Tomasi di Lampedusa;1958
I Vicerè;Federico De Roberto;1894
Mastro don Gesualdo;Giovanni Verga;1889
```

Soluzione

```c
// 6.2.5 -
// Leggi le informazioni riguardanti una serie di libri da
// un file (testo di seguito) e:
// 1.    carica i dati in un array di libri (definisci un'apposita
// struttura libro)
// 2.    ordina i libri per data di scrittura e riscrivili in un
// altro file nello stesso formato del file originale
// 3.    scrivi e usa una funzione che riceve i dati già
// caricati e che restituisce il libro più recente
// 4.    scrivi e usa una funzione che restituisca il secolo in
// cui sono stati scritti più libri

#include <stdio.h>

# define N 100

typedef struct {
    char titolo[N];
    char autore[N];
    int anno;
} libro;

void stampaElenco(libro elenco[], int nlibri) {
    int i;
    for (i = 0; i < nlibri; i++) {
        printf("%s;%s;%d\n", elenco[i].titolo, elenco[i].autore, elenco[i].anno);
    }
}

void ordina(libro elenco[], int nlibri) {
    int i, j;
    libro tmp;
    for (i = 0; i < nlibri-1; i++) {
        for (j = 0; j < nlibri - 1 - i; j++) {
            if (elenco[j].anno > elenco[j+1].anno) {
                tmp = elenco[j];
                elenco[j] = elenco[j+1];
                elenco[j+1] = tmp;
            }
        }
    }
}

void scriviSuFile(libro elenco[], int nlibri, char nomefile[]) {
    FILE *fp;
    int i;
    fp = fopen(nomefile, "w");
    if (fp == NULL) {
        printf("Errore scrittura");
```

```
        return;
    }

    for (i = 0; i < nlibri; i++) {
        fprintf(fp, "%s;%s;%d\n", elenco[i].titolo, elenco[i].autore, elenco[i].anno);
    }
    printf("Scrittura completata\n");
}

libro piuRecente(libro elenco[], int nlibri) {
    int i, imax;
    imax = 0;
    for (i = 1; i < nlibri; i++) {
        if (elenco[i].anno > elenco[imax].anno) {
            imax = i;
        }
    }
    return elenco[imax];
}

int secoloConPiuLibri(libro elenco[], int nlibri) {
    int secoli[21];
    int i, secolo, imax;

    for (i = 0; i < 21; i++) {
        secoli[i] = 0;
    }

    for (i = 0; i < nlibri; i++) {
        secolo = elenco[i].anno / 100;
        secoli[secolo]++;
    }

    imax = 0;
    for (i = 0; i < 21; i++) {
        if (secoli[i] > secoli[imax]) {
            imax = i;
        }
    }
    return imax*100;
}

int main() {
    FILE *fp;
    char riga[200];
    libro elenco[N], recente;
    int nlibri;
    int i, j;

    fp = fopen("6-2-5.txt", "r");
    if (fp == NULL) {
        printf("Errore");
        return 1;
    }

    for (nlibri = 0; !feof(fp); nlibri++) {
        fgets(riga, 200, fp);
        sscanf(riga, "%[^;];%[^;];%d", elenco[nlibri].titolo, elenco[nlibri].autore,
&elenco[nlibri].anno);
    }

    // stampa di controllo di tutti i dati
    printf("Elenco originale:\n");
    stampaElenco(elenco, nlibri);

    ordina(elenco, nlibri);
    printf("\n\nElenco ordinato:\n");
```

```
    stampaElenco(elenco, nlibri);

    scriviSuFile(elenco, nlibri, "6-2-5-out.txt");

    recente = piuRecente(elenco, nlibri);
    printf("\n\nIl libro piu' recente: %s;%s;%d\n", recente.titolo, recente.autore,
recente.anno);


    printf("\n\nIl secolo con piu' libri e' il %d", secoloConPiuLibri(elenco, nlibri));
}
```

6.2.6 Scrivi un programma che sia in grado di leggere da file i dati riportati di seguito e salvarli in opportune strutture dati (punto e triangolo). Il programma deve poi stampare sia su schermo che in un altro file i dati riguardanti ogni triangolo e se esso è scaleno, isoscele o equilatero. Per decidere il tipo di triangolo devono essere usate tre diverse funzioni che controllano ognuna se il triangolo passato come parametro è di uno dei tre tipi richiesti.

```
Input
(1, 0) (3, 3) (-0.6, 3.23)
(1, 0) (1, 4) (-2.46, 2)
(2, 0) (5, 3) (2, 6)
(3, 0) (4.4, 3.3) (2.35, 4.8)
(2, 2) (4, 2) (4, 6)

Output
(1.000000, 0.000000) (3.000000, 3.000000) (-0.600000, 3.230000) - equilatero
(1.000000, 0.000000) (1.000000, 4.000000) (-2.460000, 2.000000) - equilatero
(2.000000, 0.000000) (5.000000, 3.000000) (2.000000, 6.000000) - isoscele
(3.000000, 0.000000) (4.400000, 3.300000) (2.350000, 4.800000) - scaleno
(2.000000, 2.000000) (4.000000, 2.000000) (4.000000, 6.000000) - scaleno
```

6.2.7 Il seguente testo (che devi inserire in un file di testo) descrive i più celebri virologi Italiani nell'epoca covid

(Nome e Cognome - Ospedale - città - (O) Ottimista (P) Pessimista

Roberto Burioni - San Raffaele - Milano (P)

Matteo Bassetti - San Martino - Genova (O)

Alberto Zangrillo - San Raffaele - Milano (O)

Massimo Galli - Luigi Sacco - Milano (P)

MariaRita Gismondo - Luigi Sacco - Milano (O)

Andrea Crisanti - Università di Padova - Padova (P)

Ilaria Capua - One Health - Florida (P)

Antonella Viola - Città della Speranza - Padova (P)

Fabrizio Pregliasco - Galeazzi - Milano (O)

Walter Ricciardi - Policlino Gemelli - Roma (P)

Franco Locatelli - Bambin Gesù - Roma (P)

1. stampare nome e cognome dei virologi ordinati per cognome;
2. stampare il numero di virologi ottimisti e di quelli pessimisti.

Soluzione

```
// 6.2.7    Il seguente testo (che devi inserire in un file di testo) descrive i
// più celebri virologi Italiani nell'epoca covid (Nome e Cognome - Ospedale -
// città - (O) Ottimista (P) Pessimista
```

```c
//     Roberto Burioni - San Raffaele - Milano (P)
//     Matteo Bassetti - San Martino - Genova (O)
//     Alberto Zangrillo - San Raffaele - Milano (O)
//     Massimo Galli - Luigi Sacco - Milano (P)
//     MariaRita Gismondo - Luigi Sacco - Milano (O)
//     Andrea Crisanti - Università di Padova - Padova (P)
//     Ilaria Capua - One Health - Florida (P)
//     Antonella Viola - Città della Speranza - Padova (P)
//     Fabrizio Pregliasco - Galeazzi - Milano (O)
//     Walter Ricciardi - Policlino Gemelli - Roma (P)
//     Franco Locatelli - Bambin Gesù - Roma (P)
// 1.   stampare nome e cognome dei virologi ordinati per cognome;
// 2.   stampare il numero di virologi ottimisti e di quelli pessimisti.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 100
#define M 300

typedef struct {
    char nome[N];
    char cognome[N];
    char ospedale[N];
    char citta[N];
    char orientamento;
} virologo;

// funzione di confronto, cosa che non avete visto, da spiegare
int confrontaVirologi(const void *a, const void *b) {
    return strcmp(((virologo *)a)->cognome, ((virologo *)b)->cognome);
}

// ordinamento con bubble sort standard
void ordinaVirologi(virologo lista[], int n) {
    int i, j;
    virologo tmp;

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - 1 - i; j++) {
            if (strcmp(lista[j].cognome, lista[j + 1].cognome) > 0) {
                tmp = lista[j];
                lista[j] = lista[j + 1];
                lista[j + 1] = tmp;
            }
        }
    }
}

int main() {
    FILE *fp;
    virologo lista[N];
    char riga[M];
    int nvirologi, i;

    fp = fopen("6-2-7.txt", "r");
    if (fp == NULL) {
        printf("Errore\n");
        return 1;
    }

    for (i = 0; !feof(fp); i++) {
        fgets(riga, M, fp);
        if (strlen(riga) > 0) {
            sscanf(riga, "%s %s - %[^-]- %s (%c)", lista[i].nome,
                   lista[i].cognome, lista[i].ospedale, lista[i].citta,
```

```
                    &lista[i].orientamento);
            // printf("%s %s - %s - %s (%c)\n", lista[i].nome, lista[i].cognome,
            //     lista[i].ospedale, lista[i].citta, lista[i].orientamento);
        }
    }
    nvirologi = i;

    // 1.   stampare nome e cognome dei virologi ordinati per cognome;
    qsort(lista, nvirologi, sizeof(virologo), confrontaVirologi);
    // ordinaVirologi(lista, nvirologi);
    for (i = 0; i < nvirologi; i++) {
        printf("%s %s - %s - %s (%c)\n", lista[i].nome, lista[i].cognome,
               lista[i].ospedale, lista[i].citta, lista[i].orientamento);
    }

    // 2.   stampare il numero di virologi ottimisti e di quelli pessimisti.
    int ottimisti = 0, pessimisti = 0;
    for (i = 0; i < nvirologi; i++) {
        if (lista[i].orientamento == 'O') {
            ottimisti++;
        } else if (lista[i].orientamento == 'P') {
            pessimisti++;
        }
    }
    printf("\nOttimisti: %d\nPessimisti: %d\n", ottimisti, pessimisti);
}
```

6.2.8 Un file di testo contiene i seguenti dati:

```
Marco Polo 1254 1324
Cristoforo Colombo 1451 1506
Amerigo Vespucci 1454 1512
Francisco Pizarro 1475 1541
Ferdinando Magellano 1480 1521
Hernan Cortez 1485 1547
Walter Raleigh 1552 1618
Henry Hudson 1570 1611
James Cook 1728 1779
Charles Darwin 1809 1882
Kit Carson 1809 1866
David Livingstone 1813 1873
Charles Foucauld 1858 1916
Ronald Amundsen 1872 1928
Ernest Shakleton 1874 1922
```

Scrivi un programma che legga i dati contenuti nel file, li memorizzi opportunamente, e poi scrivi e utilizza le seguenti funzioni:

1. una funzione che stampa tutti i dati.
2. una funzione che ordina i dati in ordine alfabetico secondo il nome degli esploratori
3. una funzione che ordina i dati in ordine crescente secondo le date di nascita degli esploratori
4. una funzione che calcola la durata della vita di ogni esploratore e aggiunge questo dato agli altri
5. una funzione che ordina gli esploratori in base alla durata della loro vita in ordine decrescente e in caso di parità in ordine crescente secondo la data di nascita
6. una funzione che riscrive tutti i dati in un secondo file

Soluzione

```c
// 6.2.8 - Un file di testo contiene i seguenti dati:
// Marco Polo 1254 1324
// Cristoforo Colombo 1451 1506
// Amerigo Vespucci 1454 1512
// Francisco Pizarro 1475 1541
// Ferdinando Magellano 1480 1521
// Hernan Cortez 1485 1547
// Walter Raleigh 1552 1618
// Henry Hudson 1570 1611
// James Cook 1728 1779
// Charles Darwin 1809 1882
// Kit Carson 1809 1866
// David Livingstone 1813 1873
// Charles Foucauld 1858 1916
// Ronald Amundsen 1872 1928
// Ernest Shakleton 1874 1922

// Scrivi un programma che legga i dati contenuti nel file, li
// memorizzi opportunamente, e poi scrivi e utilizza le seguenti funzioni:
// 1.	una
// funzione che stampa tutti i dati.
// 2.	una
// funzione che ordina i dati in ordine alfabetico secondo il nome degli
// esploratori
// 3.	una
// funzione che ordina i dati in ordine crescente secondo le date di nascita degli
// esploratori
// 4.	una
// funzione che calcola la durata della vita di ogni esploratore e aggiunge questo
// dato agli altri
// 5.	una
// funzione che ordina gli esploratori in base alla durata della loro vita in
// ordine decrescente e in caso di parità in ordine crescente secondo la data di
// nascita
// 6.	una
// funzione che riscrive tutti i dati in un secondo file

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 100
#define M 300

typedef struct {
    char nome[N];
    char cognome[N];
    int nascita, morte, durata;
} esploratore;

// 4.	una
// funzione che calcola la durata della vita di ogni esploratore e aggiunge questo
// dato agli altri
void calcolaEta(esploratore elenco[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        elenco[i].durata = elenco[i].morte - elenco[i].nascita;
    }
}

// 1.	una
// funzione che stampa tutti i dati.
void stampaElenco(esploratore elenco[], int n) {
    int i;
```

```
    for (i = 0; i < n; i++) {
        printf("%s %s %d %d %d\n", elenco[i].nome, elenco[i].cognome, elenco[i].nascita,
elenco[i].morte, elenco[i].durata);
    }
}

// 2.     una
// funzione che ordina i dati in ordine alfabetico secondo il nome degli
// esploratori
int confAlfabetico(void *a, void *b) {
    return strcmp(((esploratore *)a)->nome, ((esploratore *)b)->nome);
}



// 3.     una
// funzione che ordina i dati in ordine crescente secondo le date di nascita degli
// esploratori
// 4.     una
// funzione che calcola la durata della vita di ogni esploratore e aggiunge questo
// dato agli altri
// 5.     una
// funzione che ordina gli esploratori in base alla durata della loro vita in
// ordine decrescente e in caso di parità in ordine crescente secondo la data di
// nascita
// 6.     una
// funzione che riscrive tutti i dati in un secondo file

int main() {
    FILE *fp;
    esploratore lista[N];
    char riga[M];
    int  i, nesploratori,letti;

    fp = fopen("6-2-8.txt", "r");
    if (fp == NULL) {
        printf("Errore\n");
        return 1;
    }

    for (i = 0; !feof(fp); i++) {
        fgets(riga, M, fp);
        // if (strlen(riga) > 0) {
        //     sscanf(riga, "%s %s %d %d", lista[i].nome, lista[i].cognome, &lista[i].nascita,
&lista[i].morte);
        // }
        letti = sscanf(riga, "%s %s %d %d", lista[i].nome, lista[i].cognome, &lista[i].nascita,
&lista[i].morte);
        if (letti != 4) {
            i--;
        }
    }
    nesploratori = i;

    calcolaEta(lista, nesploratori);
    stampaElenco(lista, nesploratori);
    qsort(lista, nesploratori, sizeof(esploratore), confAlfabetico);
    printf("----------------------\n");
    stampaElenco(lista, nesploratori);
}
```

6.2.9 Dato il seguente file di dati in input:

A1: Milano - Napoli: Autostrade per l'Italia: 759

A2: Salerno - Reggio Calabria: Anas: 442

A3: Napoli - Salerno: Autostrade per l'Italia: 52

A4: Torino - Trieste: Autostrade per l'Italia: 524
A6: Torino - Savona: Gruppo Gavio: 124
A7: Milano - Genova: Enti Pubblici: 133
A8: Milano - Varese: Autostrade per l'Italia: 43
A10: Savona - Ventimiglia: Gruppo Gavio: 113
A11: Firenze - Pisa: Autostrade per l'Italia: 81
A12: Genova - Cecina: Gruppo Gavio: 210
A13: Bologna - Padova: Autostrade per l'Italia: 116
A14: Bologna - Taranto: Autostrade per l'Italia: 743
A15: Parma - La Spezia: Gruppo Gavio: 108
A16: Napoli - Canosa: Autostrade per l'Italia: 172
A19: Palermo - Catania: Anas: 191
A21: Torino - Piacenza - Brescia: Gruppo Gavio: 238
A22: Brennero - Modena: Enti Pubblici: 315
A23: Palmanova - Tarvisio: Enti Pubblici: 119
A24: Roma - Teramo: Gruppo Toto: 159
A25: Torano - Pescara: Gruppo Toto: 115
A26: Genova - Gravellona: Autostrade per l'Italia: 197

1. Stampare il gestore dell'autostrada A8
2. stampare il gestore dell'autostrada "Savona - Ventimiglia"
3. stampare l'autostrada più lunga, con il suo percorso e il suo gestore
4. Stampare quanti differenti gestori sono presenti nell'elenco
5. Per ognuno dei gestori presenti stampare il numero di autostrade gestite, e il numero di km gestiti

Soluzione

```
// 6.2.9    Dato il seguente file di dati in input:
// A1: Milano -  Napoli: Autostrade per l'Italia: 759
// A2: Salerno - Reggio Calabria: Anas: 442
// A3: Napoli - Salerno: Autostrade per l'Italia: 52
// A4: Torino - Trieste: Autostrade per l'Italia: 524
// A6: Torino - Savona: Gruppo Gavio: 124
// A7: Milano - Genova: Enti Pubblici: 133
// A8: Milano - Varese: Autostrade per l'Italia: 43
// A10: Savona - Ventimiglia: Gruppo Gavio: 113
// A11: Firenze - Pisa: Autostrade per l'Italia: 81
// A12: Genova - Cecina: Gruppo Gavio: 210
// A13: Bologna - Padova: Autostrade per l'Italia: 116
// A14: Bologna - Taranto: Autostrade per l'Italia: 743
// A15: Parma - La Spezia: Gruppo Gavio: 108
// A16: Napoli - Canosa: Autostrade per l'Italia: 172
// A19: Palermo - Catania: Anas: 191
// A21: Torino - Piacenza - Brescia: Gruppo Gavio: 238
// A22: Brennero - Modena: Enti Pubblici: 315
// A23: Palmanova - Tarvisio: Enti Pubblici: 119
// A24: Roma - Teramo: Gruppo Toto: 159
// A25: Torano - Pescara: Gruppo Toto: 115
// A26: Genova - Gravellona: Autostrade per l'Italia: 197

// 1.   Stampare il gestore dell'autostrada A8
// 2.   stampare il gestore dell'autostrada "Savona - Ventimiglia"
```

```c
// 3.   stampare l'autostrada più lunga, con il suo percorso e il suo gestore
// 4.   Stampare quanti differenti gestori sono presenti nell'elenco
// 5.   Per ognuno dei gestori presenti stampare il numero di autostrade gestite,
// e il numero di km gestiti

#include <stdio.h>
#include <string.h>

#define M 4
#define N 100
#define K 200

typedef struct {
    char nome[M];
    char tratta[N];
    char gestore[N];
    int km;
} Autostrada;

int main() {
    FILE *fp;
    Autostrada elenco[N];
    int i, nautostrade, imax;
    char riga[K];

    fp = fopen("6-2-9.txt", "r");
    if (fp == NULL) {
        printf("errore\n");
        return 1;
    }

    for (i = 0; !feof(fp); i++) {
        fgets(riga, K, fp);
        // A26: Genova - Gravellona: Autostrade per l'Italia: 197
        sscanf(riga, "%[^:]: %[^:]: %[^:]: %d", elenco[i].nome,
               elenco[i].tratta, elenco[i].gestore, &elenco[i].km);
    }
    nautostrade = i;

    // stampa di controllo
    for (i = 0; i < nautostrade; i++) {
        printf("%s: %s: %s: %d\n", elenco[i].nome,
               elenco[i].tratta, elenco[i].gestore, elenco[i].km);
    }

    // 1.   Stampare il gestore dell'autostrada A8
    for (i = 0; i < nautostrade; i++) {
        if (strcmp(elenco[i].nome, "A8") == 0) {
            printf("\n\nGestore dell'autostrada A8: %s\n", elenco[i].gestore);
        }
    }

    // 2.   stampare il gestore dell'autostrada "Savona - Ventimiglia"
    for (i = 0; i < nautostrade; i++) {
        if (strcmp(elenco[i].tratta, "Savona - Ventimiglia") == 0) {
            printf("\nGestore dell'autostrada Savona - Ventimiglia: %s\n", elenco[i].gestore);
        }
    }

    // 3.   stampare l'autostrada più lunga, con il suo percorso e il suo gestore (io stampo
tutto)
    imax = 0;
    for (i = 0; i < nautostrade; i++) {
        if (elenco[i].km > elenco[imax].km) {
            imax = i;
        }
    }
```

```
    printf("L'autostrada piu' lunga:\n - %s: %s: %s: %d\n", elenco[imax].nome,
            elenco[imax].tratta, elenco[imax].gestore, elenco[imax].km);

    // 4 e 5 richiedono le mappe per essere fatti in modo semplice, se no sono lunghi e difficili
}
```

6.2.10 Dato il file contenente le seguenti informazioni:

Atalanta: Bergamo, Italia
Hellas: Verona, Italia
Sampdoria: Genova, Italia
Genoa: Genova, Italia
Liverpool: Liverpool, Regno Unito
Everton: Liverpool, Regno Unito
Espanol: Barcellona, Spagna
Barcellona: Barcellona, Spagna
Siviglia: Siviglia, Spagna
Betis: Siviglia, Spagna
Juventus: Torino, Italia
Young Boys: Berna, Svizzera
Chievo: Verona, Italia
Inter: Milano, Italia
Milan: Milano, Italia
Reggina: Reggio Calabria, Italia

1. Leggi i dati e salvali in una struttura dati opportuna
2. Trovare la città dell'Hellas, e stampare il nome dell'altra squadra della stessa città
3. stampare in ordine alfabetico tutte le squadre italiane
4. Stampare le sole squadre italiane, ordinate in base alla città, e a parità di città in ordine alfabetico

Soluzione

```
// 6.2.10   Dato il file contenente le seguenti informazioni:
// Atalanta: Bergamo, Italia
// Hellas: Verona, Italia
// Sampdoria: Genova, Italia
// Genoa: Genova, Italia
// Liverpool: Liverpool, Regno Unito
// Everton: Liverpool, Regno Unito
// Espanol: Barcellona, Spagna
// Barcellona: Barcellona, Spagna
// Siviglia: Siviglia, Spagna
// Betis: Siviglia, Spagna
// Juventus: Torino, Italia
// Young Boys: Berna, Svizzera
// Chievo: Verona, Italia

// 1.   Leggi i dati e salvali in una struttura dati opportuna
// 2.   Trovare la città dell'Hellas, e stampare il nome dell'altra squadra della
// stessa città 3.  stampare in ordine alfabetico tutte le squadre italiane 4.
// Stampare le sole squadre italiane, ordinate in base alla città, e a parità di
// città in ordine alfabetico
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>

#define M 50
#define N 100
#define K 300

typedef struct {
    char nome[M];
    char citta[M];
    char nazione[M];
} Squadra;

// // 2.    Trovare la città dell'Hellas, e stampare il nome dell'altra squadra
// della stessa città int trovaCitta(Squadra elenco[], int nsquadre, char
// nome[], Squadra *ris) {
//     int i;
//     for (i = 0; i < nsquadre; i++) {
//         if (strcmp(elenco[i].nome, nome) == 0) {
//             *ris = elenco[i];
//             return 1;
//         }
//     }
//     return 0;
// }

int confrontoPerNome(void *a, void *b) {
    return strcmp(((Squadra *)a)->nome, ((Squadra *)b)->nome);
}
int confrontoPerCitta(void *a, void *b) {
    int confcitta;
    confcitta = strcmp(((Squadra *)a)->citta, ((Squadra *)b)->citta);
    if (confcitta == 0) {
        return strcmp(((Squadra *)a)->nome, ((Squadra *)b)->nome);
    } else {
        return confcitta;
    }
}

int main() {
    FILE *fp;
    Squadra elenco[N];
    int i, nsquadre;
    char riga[K];

    fp = fopen("6-2-10.txt", "r");
    if (fp == NULL) {
        printf("errore\n");
        return 1;
    }

    // 1.   Leggi i dati e salvali in una struttura dati opportuna
    for (i = 0; !feof(fp); i++) {
        fgets(riga, K, fp);
        if (strlen(riga) > 1) {
            sscanf(riga, "%[^:]: %[^,], %[^\n]", elenco[i].nome,
                   elenco[i].citta, elenco[i].nazione);
        } else {
            i--;
        }
    }
    nsquadre = i;

    for (i = 0; i < nsquadre; i++) {
        printf("%s: %s, %s\n", elenco[i].nome, elenco[i].citta,
               elenco[i].nazione);
    }
```

```
    // 2.   Trovare la città dell'Hellas, e stampare il nome dell'altra squadra
    // della stessa città
    for (i = 0; i < nsquadre; i++) {
        if (strcmp(elenco[i].nome, "Hellas") == 0) {
            printf("Citta' dell'Hellas: %s\n", elenco[i].citta);
            break;
        }
    }
    if (i >= nsquadre) {
        printf("Non ho trovato la citta dell'Hellas\n");
    }

    // 3.   stampare in ordine alfabetico tutte le squadre italiane
    qsort(elenco, nsquadre, sizeof(Squadra), confrontoPerNome);
    printf("\nSquadre ordinate per nome:\n");
    for (i = 0; i < nsquadre; i++) {
        if (strcmp(elenco[i].nazione, "Italia") == 0) {
            printf("%s: %s, %s\n", elenco[i].nome, elenco[i].citta,
                   elenco[i].nazione);
        }
    }

    // 4.   Stampare le sole squadre italiane, ordinate in base alla città, e a
    // parità di città in ordine alfabetico
    qsort(elenco, nsquadre, sizeof(Squadra), confrontoPerCitta);
    printf("\nSquadre ordinate per citta':\n");
    for (i = 0; i < nsquadre; i++) {
        if (strcmp(elenco[i].nazione, "Italia") == 0) {
            printf("%s: %s, %s\n", elenco[i].nome, elenco[i].citta,
                   elenco[i].nazione);
        }
    }

}
```

6.2.11 Devi assegnare ad ogni persona della classe un identificativo univoco formato dalle prime tre consonanti del nome seguite dalle prime tre consonanti del cognome seguito da una serie di 3 caratteri alfanumerici casuali. Se i nomi o i cognomi non contengono 3 consonanti aggiungi vocali a caso per arrivare alle 3 lettere richieste sia per il nome che per il cognome. Nel seguente file di testo è riportato l'elenco dei nomi. In un secondo file riscrivi i nomi seguiti dal codice identificativo.

```
Banani, Fahd
Brambilla, Carlo
Brambilla, Federico
Carrera, Aurora
Crespi, Ryan
Germanò, Riccardo
Grassi, Arianna
Hoxha, Visar
Kuka, Blerina
La Rosa, Abraham
Livieri, Lorenzo
Meterangelo, Chiara
Miele, Serena
Puertas Moreno, Alessandra
Pulvirenti, Giorgia
Roncoroni, Luca Davide
Rossetti, Gabriele
Sambo, Luigi
Sfondrini, Diego
Tarboush, Andrea
Tieni, Ilaria
Tornaghi, Francesca
```

Ad esempio Tieni Ilaria è associata a lrotnaf5u, lr sono scelti dal nome, o è a caso, tn presi dal cognome, a a caso, f5u a caso.

Soluzione

```c
// 6.2.11 -  Devi assegnare ad ogni persona della classe un
// identificativo univoco formato dalle prime tre consonanti del nome seguite
// dalle prime tre consonanti del cognome seguito da una serie di 3 caratteri
// alfanumerici casuali. Se i nomi o i cognomi non contengono 3 consonanti
// aggiungi vocali a caso per arrivare alle 3 lettere richieste sia per il nome
// che per il cognome. Nel seguente file di testo è riportato l'elenco dei nomi.
// In un secondo file riscrivi i nomi seguiti dal codice identificativo.

// Ad esempio Tieni Ilaria è associata a lrotnaf5u, lr sono
// scelti dal nome, o è a caso, tn presi dal cognome, a a caso, f5u a caso.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define N 100

int main() {
    FILE *fpin, *fpout;
    char riga[N], nome[N], cognome[N], codice[10];
    char vocali[] = "aeiou";
    char alfanum[] = "abcdefghijklmnopqrstuvwxyz0123456789";
    int i, j;

    srand(time(NULL));

    fpin = fopen("6-2-11.txt", "r");
    if (fpin == NULL) {
        printf("errore lettura\n");
        return 1;
    }

    fpout = fopen("6-2-11-out.txt", "w");
    if (fpout == NULL) {
        printf("errore scruttura\n");
        return 2;
    }

    while (!feof(fpin)) {
        fgets(riga, N, fpin);
        if (strlen(riga) > 1) { // controllo se la riga non è vuota
            sscanf(riga, "%[^,], %[^\n]", cognome, nome);

            // prime tre consonanti del nome
            j = 0;
            for (i = 0; i < strlen(nome) && j < 3; i++) {
                nome[i] = tolower(nome[i]);
                if ((nome[i] >= 'a' && nome[i] <= 'z') && nome[i] != 'a' &&
                    nome[i] != 'e' && nome[i] != 'i' && nome[i] != 'o' &&
                    nome[i] != 'u') {
                    codice[j] == nome[i];
                    j++;
                }
            }
            while (j < 3) {
                codice[j] = vocali[rand()%5];
                j++;
            }

            for (i = 0; i < strlen(cognome) && j < 6; i++) {
```

```
                cognome[i] = tolower(cognome[i]);
                if ((cognome[i] >= 'a' && cognome[i] <= 'z') && cognome[i] != 'a' &&
                    cognome[i] != 'e' && cognome[i] != 'i' && cognome[i] != 'o' &&
                    cognome[i] != 'u') {
                    codice[j] == cognome[i];
                    j++;
                }
            }
            while (j < 6) {
                codice[j] = vocali[rand()%5];
                j++;
            }

            while (j < 9) {
                codice[j] = alfanum[rand()%strlen(alfanum)];
                j++;
            }

            // da finire
        }
    }

    fclose(fpin);
    fclose(fpout);
}
```

6.2.12 Sei un giocatore di giochi di ruolo e ti serve avere delle tabelle che ti indichino le probabilità di ottenere i vari punteggi con dei tiri di diversi dadi. Non conosci molto la matematica e il calcolo combinatorio (e oltretutto calcolare tutto a mano è lungo) ma sai programmare bene, pensi quindi di stimare le probabilità con un programma e di scrivere dei file ognuno che idica le diverse probabilità ottenibili con un diverso dado. I dadi che usi sono: d4 (dado a quattro facce), d6, d8, d10, d12, d20. Per ogni dado vuoi sapere le probabilità di ottenere ogni diverso risultato possibile tirando il dado una volta, o 2, o 3… fino a 10 volte.

Alla fine devi generare 6 file (uno per dado) e ogni file deve avere una prima riga che indica il dado (ad esempio “d6”) e poi diverse righe, ognuna inizia indicando il numero di dadi tirati (da 1 a 10 quindi 10 righe) seguito da un “:” e poi un dizionario che associa ad ogni valore ottenibile la probabilità associata.

Soluzione

```
// 6.2.12   Sei un giocatore di giochi di ruolo e ti serve avere delle tabelle che
// ti indichino le probabilità di ottenere i vari punteggi con dei tiri di
// diversi dadi. Non conosci molto la matematica e il calcolo combinatorio (e
// oltretutto calcolare tutto a mano è lungo) ma sai programmare bene, pensi
// quindi di stimare le probabilità con un programma e di scrivere dei file
// ognuno che idica le diverse probabilità ottenibili con un diverso dado. I
// dadi che usi sono: d4 (dado a quattro facce), d6, d8, d10, d12, d20. Per ogni
// dado vuoi sapere le probabilità di ottenere ogni diverso risultato possibile
// tirando il dado una volta, o 2, o 3… fino a 10 volte. Alla fine devi generare
// 6 file (uno per dado) e ogni file deve avere una prima riga che indica il
// dado (ad esempio “d6”) e poi diverse righe, ognuna inizia indicando il numero
// di dadi tirati (da 1 a 10 quindi 10 righe) seguito da un “:”  e poi un
// dizionario che associa ad ogni valore ottenibile la probabilità associata.

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int main() {
```

```c
    int dadi[] = {4,6,8,10,12,20};
    int ndadi = 6;
    int i, j, k, l, ris;
    FILE *fp;
    char nomefile[30];
    float prob[201];
    int nlanci = 10000;

    srand(time(NULL));

    for (i = 0; i < ndadi; i++) {
        sprintf(nomefile, "d%d.txt", dadi[i]);
        fp = fopen(nomefile, "w");
        if (fp == NULL) {
            printf("errore");
            return 1;
        }

        for (j = 1; j <= 10; j++) {
            fprintf(fp, "%2d. ", j); // qui stampo quanti dadi tiro
            printf("%d\n", j);
            for (k = 0; k < 201; k++){
                prob[k] = 0;
            }
            for (k = 0; k < nlanci; k++){
                ris = 0;
                for (l = 0; l < j; l++) {
                    ris += rand() % dadi[i] + 1;
                }
                prob[ris] += 1;
            }
            for (k = 0; k < 201; k++){
                prob[k] /= nlanci;
                // k mi rappresenta il numero venuto lanciando j dadi insieme
                if (prob[k] > 0) {
                    fprintf(fp, "%d: %.2f  ", k, prob[k]*100); // qui stampo la singola prob
                }
            }
            fprintf(fp, "\n");
        }
        fclose(fp);
    }
}
```

6.2.13 Crea un programma di gestione di una agenda. L'agenda deve permettere di visualizzare e inserire i dati riguardanti i propri contatti. Per ogni persona devono essere memorizzati nome cognome e numero di telefono. La visualizzazione dei dati deve avvenire in ordine alfabetico per nome o cognome (scegli tu). Tutti i dati devono essere salvati in un file in modo da poterli caricare ogni volta che si apre il programma.

Rendi il programma più modulare possibile dividendo le funzionalità in funzioni separate. Scrivi anche una funzione che ti permetta di cercare un contatto in base a nome o cognome o entrambi.

Variante: L'agenda è ordinata per cognome e la funzione di ricerca cerca il contatto a partire dal cognome con una ricerca dicotomica. Se hai fatto le funzioni ricorsive la funzione deve essere fatta sia nella versione iterativa che ricorsiva

6.2.14 Il seguenti dati, che dovrai inserire in un file di testo, descrivono quante ore hanno lavorato in una settimana, una serie di dipendenti: ad esempio Mr White ha lavorato 8 ore il lunedì, 9 ore il martedì, 8 ore il mercoledì, 9 ore il giovedì e 4 ore il venerdì

```
Mr White: 8, 9, 8, 9, 4
Mr Brown: 0, 8, 7, 10, 8
Mr Blonde: 8, 8, 8, 9, 9
Mr Black: 6, 9, 9, 8, 8
Mr Red: 8, 7, 8, 8, 8
Mr Green: 4, 8, 8, 8, 4
```

1. Stampare la classifica dei lavoratori, in base al numero di ore lavorate
2. Stampare il giorno con più ore lavorate, e quante ore sono state lavorate in quel giorno

Soluzione

```c
// 6.2.14 - Il seguenti dati, che dovrai inserire in un file di
// testo, descrivono quante ore hanno lavorato in una settimana, una serie di
// dipendenti: ad esempio Mr White ha lavorato 8 ore il lunedì, 9 ore il
// martedì, 8 ore il mercoledì, 9 ore il giovedì e 4 ore il venerdì

// Mr White: 8, 9, 8, 9, 4
// Mr Brown: 0, 8, 7, 10, 8
// Mr Blonde: 8, 8, 8, 9, 9
// Mr Black: 6, 9, 9, 8, 8
// Mr Red: 8, 7, 8, 8, 8
// Mr Green: 4, 8, 8, 8, 4
// 1.     Stampare
// la classifica dei lavoratori, in base al numero di ore lavorate
// 2.     Stampare
// il giorno con più ore lavorate, e quante ore sono state lavorate in quel
// giorno

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define N 20
#define M 100

typedef struct {
    char nome[N];
    int ore[5];
    int somma;
} Lavoratore;

int confronto(void *a, void *b) {
    return ((Lavoratore *)a)->somma < ((Lavoratore *)b)->somma;
}

int main() {
    FILE *fp;
    int n, i, j, maxj, max, somma;
    char riga[M];
    Lavoratore elenco[N];
    char giorni[][20] = {"Lunedi'", "Martedi'", "Mercoledi'", "Giovedi'", "Venerdi'"};

    fp = fopen("6-2-14.txt", "r");
    if (fp == NULL) {
        printf("errore lettura\n");
        return 1;
    }

    for (n = 0; !feof(fp); n++) {
        fgets(riga, M, fp);
        // printf("%s\n", riga);

        sscanf(riga, "%[^:]: %d, %d, %d, %d, %d", elenco[n].nome,
```

```
            &elenco[n].ore[0], &elenco[n].ore[1], &elenco[n].ore[2],
            &elenco[n].ore[3], &elenco[n].ore[4]);

        elenco[n].somma = elenco[n].ore[0] + elenco[n].ore[1] +
                          elenco[n].ore[2] + elenco[n].ore[3] +
                          elenco[n].ore[4];

        // printf("%10s: %2d %2d %2d %2d %2d - %2d\n", elenco[n].nome,
        //        elenco[n].ore[0], elenco[n].ore[1], elenco[n].ore[2],
        //        elenco[n].ore[3], elenco[n].ore[4], elenco[n].somma);
    }
    fclose(fp);
    // 1.     Stampare
    // la classifica dei lavoratori, in base al numero di ore lavorate
    qsort(elenco, n, sizeof(Lavoratore), confronto);
    for (i = 0; i < n; i++) {
        printf("%10s: %2d %2d %2d %2d %2d - %2d\n", elenco[i].nome,
               elenco[i].ore[0], elenco[i].ore[1], elenco[i].ore[2],
               elenco[i].ore[3], elenco[i].ore[4], elenco[i].somma);
    }

    // 2.     Stampare
    // il giorno con più ore lavorate, e quante ore sono state lavorate in quel
    // giorno
    maxj = 0;
    max = 0;
    for (i = 0; i < n; i++) {
        max += elenco[i].ore[0];
    }
    for (j = 1; j < 5; j++) {
        somma = 0;
        for (i = 0; i < n; i++) {
            somma += elenco[i].ore[j];
        }
        if (somma > max) {
            max = somma;
            maxj = j;
        }
    }
    printf("Il giorno in cui si e' lavorato di piu' e' %s\n", giorni[maxj]);
}
```

6.2.15 Il seguente file di testo contiene le informazioni riguardanti le provincie d'Italia raggruppate per regione:

```
Valle d'Aosta: Aosta
Piemonte: Torino, Alessandria, Asti, Biella, Cuneo, Novara, Verbano-Cusio-Ossola,
Vercelli
Liguria: Genova, Imperia, La Spezia, Savona
Lombardia: Milano, Bergamo, Brescia, Como, Cremona, Lecco, Lodi, Mantova, Monza
e Brianza, Pavia, Sondrio, Varese
Trentino-Alto Adige: Trento, Bolzano
Veneto: Venezia: Belluno, Padova, Rovigo, Treviso, Verona, Vicenza
Friuli-Venezia Giulia: Trieste Gorizia, Pordenone, Udine
Emilia-Romagna: Bologna, Ferrara, Forlì-Cesena, Modena, Parma, Piacenza, Ravenna,
Reggio Emilia, Rimini
Toscana: Firenze, Arezzo, Grosseto, Livorno, Lucca, Massa Carrara, Pisa, Pistoia,
Prato, Siena
Umbria: Perugia, Terni
Marche: Ancona, Ascoli Piceno, Fermo, Macerata, Pesaro-Urbino
Lazio: Roma, Frosinone, Latina, Rieti, Viterbo
Abruzzo: L'Aquila, Chieti, Pescara, Teramo
Molise: Campobasso, Isernia
Campania: Napoli, Avellino, Benevento, Caserta, Salerno
Puglia: Bari, Andria, Barletta, Brindisi, Foggia, Lecce, Taranto, Trani
Basilicata: Potenza, Matera
Calabria: Catanzaro, Cosenza, Crotone, Reggio Calabria, Vibo Valentia
```

```
Sicilia: Palermo, Agrigento, Caltanissetta, Catania, Enna, Messina, Ragusa,
Siracusa, Trapani
Sardegna: Cagliari, Nuoro, Oristano, Sassari, Sud Sardegna
```

La prima provincia indicata per ogni regione è il capoluogo di regione.

Scrivi un programma che memorizzi opportunamente tutti i dati, poi:

1. Scrivi e usa una funzione che calcola quante provincie sono presenti mediamente in ogni regione
2. Ordina le regioni per numero di provincie
3. Scrivi i nomi delle regioni con più provincie e con meno provincie

Soluzione

```c
// 6.2.15 - Il seguente file di testo contiene le informazioni
// riguardanti le province d'Italia raggruppate per regione:

// La prima provincia indicata per ogni regione è il capoluogo
// di regione.
// Scrivi un programma che memorizzi opportunamente tutti i
// dati, poi:
// 1. Scrivi e usa una funzione che calcola quante province sono presenti
// mediamente in ogni regione
// 3. Scrivi i nomi delle regioni con più province e con
// meno province

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define N 50
#define M 200

typedef struct {
    char nome[N];
    char province[N][N];
    int nprovince;
} Regione;

void stampaRegioni(Regione elenco[], int nregioni) {
    int i, j;
    for (i = 0; i < nregioni; i++) {
        printf("%s: ", elenco[i].nome);
        for (j = 0; j < elenco[i].nprovince; j++) {
            printf("%s, ", elenco[i].province[j]);
        }
        printf("%d\n", elenco[i].nprovince);
    }
}

float mediaprovince(Regione elenco[], int nregioni) {
    float ris = 0;
    int i;
    for (i = 0; i < nregioni; i++) {
        ris += elenco[i].nprovince;
    }
    ris /= nregioni;
    return ris;
}

int confrontaRegioniPerNprovince(void *a, void *b) {
    return ((Regione *)a)->nprovince > ((Regione *)b)->nprovince;
}
```

```c
int main() {
    FILE *fp;
    int i, ntoken, j;
    char riga[M], *token;
    Regione elenco[N];
    int nregioni;

    fp = fopen("6-2-15.txt", "r");
    if (fp == NULL) {
        printf("errore lettura\n");
        return 1;
    }

    for (i = 0; !feof(fp); i++) {
        fgets(riga, M, fp);
        if (riga[strlen(riga) - 1] == '\n') {
            riga[strlen(riga) - 1] = 0;
        }

        // if (strlen(riga) > 1) {
        //     ntoken = sscanf(riga,
        //                     "%[^:]: %[^,], %[^,], %[^,], %[^,], %[^,], %[^,], "
        //                     "%[^,], %[^,], %[^,], %[^,], %[^,], %[^,], %[^,]",
        //                     elenco[i].nome, elenco[i].province[0],
        //                     elenco[i].province[0], elenco[i].province[1],
        //                     elenco[i].province[2], elenco[i].province[3],
        //                     elenco[i].province[4], elenco[i].province[5],
        //                     elenco[i].province[6], elenco[i].province[7],
        //                     elenco[i].province[8], elenco[i].province[9],
        //                     elenco[i].province[10], elenco[i].province[11],
        //                     elenco[i].province[12]);
        //     elenco[i].nprovince = ntoken - 1;
        // }

        if (strlen(riga) > 1) {
            token = strtok(riga, ": ");
            strcpy(elenco[i].nome, token);

            token = strtok(NULL, ", ");
            for (j = 0; token != NULL; j++) {
                strcpy(elenco[i].province[j], token);
                token = strtok(NULL, ", ");
            }
            elenco[i].nprovince = j;
        }


    }
    nregioni = i;

    stampaRegioni(elenco, nregioni);

    printf("Media province per regione: %g\n\n",
           mediaprovince(elenco, nregioni));

    qsort(elenco, nregioni, sizeof(Regione), confrontaRegioniPerNprovince);
    printf("Regione con piu' province: %s\n", elenco[nregioni-1].nome);
    printf("Regione con meno province: %s\n", elenco[0].nome);

    // stampaRegioni(elenco, nregioni);
}
```

7. Divisione programmi in progetti di più files

Per apprendere questo argomento è possibile modificare gli esercizi contenenti funzioni in modo da spostare la dichiarazione e l'implementazione delle funzioni in file diversi da quelli contenenti il main.