File di testo con stringhe

Testo

L'utente inserisce una serie di nomi (stringhe prive di spazi); la stringa "#" indica la fine della fase di inserimento.
L'elaboratore scrive i nomi in un file di testo.
I nomi sono letti dal file, nell'ordine di inserimento, e visualizzati.

Implementazione

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

void scriviInFile (char nomeFile[])
{
     char terminatore[] = "#";
     FILE *fileDiTesto;
     char nome[20];
     fileDiTesto = fopen (nomeFile, "w");
     printf ("Nome: ");
     scanf ("%s", nome); /* no spazi */
     while (strcmp (nome, terminatore) != 0)
     {
          fprintf (fileDiTesto, "%s\n", nome);
          printf ("Nome: ");
          scanf ("%s", nome);
     }
     fclose (fileDiTesto);
}
void leggiDaFile (char nomeFile[])
{
     FILE *fileDiTesto;
     char nome[20];
     fileDiTesto = fopen (nomeFile, "r");
     fscanf (fileDiTesto, "%s", nome); /* no spazi */
     while (!feof(fileDiTesto))
     {
          printf ("Nome: %s\n", nome);
          fscanf (fileDiTesto, "%s", nome);
     }
     fclose (fileDiTesto);
}
void main()
{
     char nomeArchivio [] = "pippo.txt";
     scriviInFile (nomeArchivio);
     leggiDaFile (nomeArchivio);
}
```

File di testo con stringhe ed interi

Testo

L’utente inserisce una serie di nomi (stringhe prive di spazi) e di età (interi senza segno); la stringa “#” indica la fine della fase di inserimento.
L’elaboratore scrive i nomi e le età in un file di testo.
I nomi e le età sono letti dal file, nell’ordine di inserimento, e visualizzati.

Implementazione

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

void scriviInFile (char nomeFile[])
{
     char terminatore[] = "#";
     FILE *fileDiTesto;
     char nome[20];
     unsigned int eta;
     fileDiTesto = fopen (nomeFile, "w");
     printf ("Nome: ");
     scanf ("%s", nome); /* no spazi */
     printf ("Età: ");
     scanf ("%d", &eta);
     while (strcmp (nome, terminatore) != 0)
     {
          fprintf (fileDiTesto, "%s\n", nome);
          fprintf (fileDiTesto, "%u", eta);
          printf ("Nome: ");
          scanf ("%s", nome);
          printf ("Età: ");
          scanf ("%d", &eta);
     }
     fclose (fileDiTesto);
}
void leggiDaFile (char nomeFile[])
{
     FILE *fileDiTesto;
     char nome[20];
     unsigned int eta;
     fileDiTesto = fopen (nomeFile, "r");
     fscanf (fileDiTesto, "%s", nome); /* no spazi */
     fscanf (fileDiTesto, "%d", &eta);
     while (!feof(fileDiTesto))
     {
          printf ("Nome: %s\n", nome);
          printf ("Eta: %u\n", eta);
          fscanf (fileDiTesto, "%s", nome);
          fscanf (fileDiTesto, "%d", &eta);
     }
     fclose (fileDiTesto);
}

void main()
{
     char nomeArchivio [] = "pippo.txt";
     scriviInFile (nomeArchivio);
     leggiDaFile (nomeArchivio);
}
```

File di testo con stringhe e caratteri

Testo

L'utente inserisce una serie di nomi (stringhe prive di spazi) e relativo sesso (carattere 'm' o 'f'); la stringa "#" indica la fine della fase di inserimento.
L'elaboratore scrive il tutto in un file di testo.
Nomi e sesso sono letti dal file, nell'ordine di inserimento, e visualizzati.

Implementazione

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

void scriviInFile (char nomeFile[])
{
     char terminatore[] = "#";
     FILE *fileDiTesto;
     char nome[20];
     char sesso;
     fileDiTesto = fopen (nomeFile, "w");
     printf ("Nome: ");
     scanf ("%s", nome); /* no spazi */
     printf ("Sesso (m/f): ");
     /* elimina il car. '\n' (ENTER) rimasto in stdin */
     scanf ("%*c");
     /* legge il carattere che rappresenta il sesso */
     scanf ("%c", &sesso);
     while (strcmp (nome, terminatore) != 0)
     {
          fprintf (fileDiTesto, "%s\n", nome);
          fprintf (fileDiTesto, "%c", sesso);
          printf ("Nome: ");
          scanf ("%s", nome);
          printf ("Sesso (m/f): ");
          scanf ("%*c"); /* elimina il carattere '\n' */
          scanf ("%c", &sesso);
     }
     fclose (fileDiTesto);
}

void leggiDaFile (char nomeFile[])
{
     FILE *fileDiTesto;
     char nome[20];
     char sesso;
     fileDiTesto = fopen (nomeFile, "r");
     fscanf (fileDiTesto, "%s", nome); /* no spazi */
     fscanf (fileDiTesto, "%*c"); /* elimina '\n' */
     fscanf (fileDiTesto, "%c", &sesso);
     while (!feof(fileDiTesto))
     {
          printf ("Nome: %s\n", nome);
          printf ("Sesso: %c\n", sesso);
          fscanf (fileDiTesto, "%s", nome);
          fscanf (fileDiTesto, "%*c"); /* elimina '\n' */
          fscanf (fileDiTesto, "%c", &sesso);
     }
```

```
     fclose (fileDiTesto);
}

void main()
{
     char nomeArchivio[] = "pippo.txt";
     scriviInFile (nomeArchivio);
     leggiDaFile (nomeArchivio);
}
```

File binari - *Archivio rilevamenti geografici*

Testo

Il programma dovrà permettere all'utente di inserire una serie di rilevamenti geografici (latitudine e longitudine). L'inserimento di una latitudine pari a 100 indicherà la fine della fase di inserimento.
L'elaboratore salverà i dati in un file binario.
Successivamente l'utente inserirà una latitudine e l'elaboratore, cercando nel fine, troverà e visualizzerà tutti i rilevamenti caratterizzati da quel valore di latitudine.
Infine, l'elaboratore visualizzerà il numero di rilevamenti trovati.

Implementazione

```
#include <stdio.h>
typedef struct
{
     unsigned int latitudine;
     unsigned int longitudine;
} Rilevamento;

void inserisci (char nomeFile[])
{
     const unsigned int TERMINATORE = 100;
     FILE *archivio;
     Rilevamento nuovoRilev;
     archivio = fopen (nomeFile, "wb");
     printf ("Latitudine (100 per finire): ");
     scanf ("%u", &nuovoRilev.latitudine);
     if (nuovoRilev.latitudine != TERMINATORE)
     {
          printf ("Longitudine: ");
          scanf ("%u", &nuovoRilev.longitudine);
     }
     while (nuovoRilev.latitudine != TERMINATORE)
     {
          fwrite (&nuovoRilev, sizeof(Rilevamento), 1, archivio);
          printf ("Latitudine (100 per finire): ");
          scanf ("%u", &nuovoRilev.latitudine);
          if (nuovoRilev.latitudine != TERMINATORE)
          {
               printf ("Longitudine: ");
               scanf ("%u", &nuovoRilev.longitudine);
          }
     }
     fclose (archivio);
}
unsigned int cerca (char nomeFile[], unsigned int latidCercata)
{
     FILE *archivio;
     Rilevamento rilev;
     unsigned int cont = 0;
     archivio = fopen (nomeFile, "rb");
     fread (&rilev, sizeof(Rilevamento), 1, archivio);
     while (!feof(archivio))
     {
          if (rilev.latitudine == latidCercata)
          {
```

```
            printf ("Latitudine: %u\n", rilev.latitudine);
            printf ("Longitudine: %u\n", rilev.longitudine);
            cont++;
            }
            fread (&rilev, sizeof(Rilevamento), 1, archivio);
      }
      fclose (archivio);
      return cont;
}
void main()
{
      char nomeArchivio[] = "numeri.dat";
      unsigned int latitudine;
      inserisci (nomeArchivio);
      printf ("Quale latitudine? ");
      scanf ("%u", &latitudine);
      printf ("Totale: %u\n", cerca (nomeArchivio, latitudine));
}
```

Media e Varianza di un array di numeri reali

Testo

- Scrivere un programma che, utilizzando le funzioni, calcoli la media e la varianza di un array di numeri reali
- Utilizzare le funzioni:
  - leggi(): permette di inserire il vettore di numeri
  - media(): calcola e ritorna la media
  - varianza(): calcola e ritorna la varianza
  - scrivi(): visualizza il vettore, la media e la varianza

Implementazione

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

void leggi (float vet[], unsigned int lungh);
float media (float vet[], unsigned int lungh);
float varianza (float vet[], unsigned int lungh, float media);
void scrivi (float vet[], unsigned int lungh, float media,
float var);

void main()
{
     const unsigned int MAX = 3;
     float array[MAX];
     float m, v;
     int i;
     leggi (array, MAX);
     m = media (array, MAX);
     v = varianza (array, MAX, m);
     scrivi (array, MAX, m, v);
}

void leggi (float vet[], unsigned int lungh)
{
     unsigned int i;
     for (i = 0; i < lungh; i++)
     {
     printf ("Numero: ");
     scanf ("%f", &vet[i]);
     }
}

float media (float vet[], unsigned int lungh)
{
     unsigned int i;
     float m = 0;
     for (i = 0; i < lungh; i++)
     {
     m = m + vet[i];
     }
     return m / lungh;
}
```

```
float varianza (float vet[], unsigned int lungh, float media)
{
     unsigned int i;
     float v = 0;
     for (i = 0; i < lungh; i++)
     {
     v = v + pow(vet[i] - media, 2.0);
     }
     return v / (lungh - 1);
}

void scrivi (float vet[], unsigned int lungh, float med,
float var)
{
     unsigned int i;
     for (i = 0; i < lungh; i++)
     {
     printf ("%f\n", vet[i]);
     }
     printf ("Media: %f, varianza: %f", med, var);
}
```

Indirizzi Internet

Testo

Scrivere un programma basato su funzioni che:

- Chieda all'utente di inserire degli indirizzi Internet (per es: "www.google.it, ftp.pippo.com") controllando che siano lunghi almeno quattro caratteri
- Completi l'indirizzo con il relativo protocollo (es: "http://www.google.it, ftp://ftp.pippo.com")
- Scriva gli indirizzi sul monitor. Se l'indirizzo da stampare è "http://www.google.it", il programma dovrà visualizzare "SITO DI GOOGLE".

Implementazione

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

typedef char Stringa[100];
void leggiIndirizzi (Stringa indirizzi[], unsigned int nInd);
void scriviIndirizzi (Stringa indirizzi[], unsigned int nInd);

void main()
{
      const unsigned int NUMIND = 4;
      Stringa indirizzi[NUMIND];
      leggiIndirizzi (indirizzi, NUMIND);
      scriviIndirizzi (indirizzi, NUMIND);
}

void leggiIndirizzi (Stringa indirizzi[], unsigned int nInd)
{
      unsigned int i;
      Stringa protocollo;
      for (i = 0; i < nInd; i++)
      {
            do
            {
                  printf ("Indirizzo %d: ", i);
                  scanf ("%s", indirizzi[i]); /* senza & */
            } while (strlen (indirizzi[i]) < 4); /* almeno a.it */
            if (strstr (indirizzi[i], "ftp.") != NULL)
            {
                  strcpy (protocollo, "ftp://");
            }
            else
            {
                  strcpy (protocollo, "http://");
            }
            strcat (protocollo, indirizzi[i]);
            strcpy (indirizzi[i], protocollo);
      }
}

void scriviIndirizzi (Stringa indirizzi[], unsigned int nInd)
{
      unsigned int i;
      printf ("---------\n");
      for (i = 0; i < nInd; i++)
      {
            printf ("Indirizzo: %s ", indirizzi[i]);
            if (strcmp (indirizzi[i], "http://www.google.it") == 0)
```

```
            {
                  printf ("E' il sito di Google");
            }
            printf ("\n");
      }
}
```

Ricerca binaria

Testo

Scrivere un programma che, utilizzando le funzioni, permetta di:

- Inserire i dati della carta d'identità di alcune persone.
- Dato un cognome, cercare se corrisponde ad una persona nel vettore. In caso positivo, visualizzare la posizione nel vettore e tutti i dati personali.

Utilizzare l'algoritmo di ricerca binaria.
Si assume che le carte d'identità siano inserite già ordinate per cognome. In caso di cognome duplicato, la ricerca si fermerà alla prima occorrenza trovata.

Pseudocodice

funzione cercaBin()
[*inizio* indica la prima cella del vettore *v*]
[*fine* indica l'ultima cella del vettore *v*]

```
do
{
```
    [calcola punto di mezzo tra *inizio* e *fine*: *i = (inizio + fine) / 2*]
```
    if  ([la stringa cercata è minore del numero in v[i])
    {
```
        [la *stringa cercata* si troverà nella zona da *inizio* a *i – 1*, allora: *fine = i – 1*]
```
    }
    else if  ([la stringa cercata è maggiore della stringa in v[i])
    {
```
        [la *stringa cercata* si troverà nella zona da *i + 1* a *fine*, allora: *inizio = i + 1*]
```
    }
    else
    {
```
        [la *stringa cercata* è stata trovata in posizione *i*]
```
    }
} while  ([la stringa cercata non è stato trovato AND l' indicatore inizio è minore o uguale dell'indicatore fine e ciò significa che non è escluso che la stringa cercata ci sia])
if  ([la stringa cercata è stata trovata])
{
```
    [ritorna la posizione *i*]
}
else
{
```
    [ritorna *–1* che assumerà il significato di "non trovato"]
```
}
```
```

Implementazione

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

typedef char Stringa[30];
typedef struct
{
     Stringa nome, cognome;
     unsigned int eta;
     Stringa indirizzo;
} CartaIdentita;

/* Ricerca binaria (array ordinato) */
int cercaBin (CartaIdentita persone[], unsigned int lungh,
Stringa cercato)
{
     typedef enum {falso, vero} Booleano;
     int iniz, fine, medio;
     Booleano trovato;
     iniz = 0;
     fine = lungh - 1;
     trovato = falso;
     do
     {
          medio = (iniz + fine) / 2;
          if (strcmp (cercato, persone[medio].cognome) < 0)
          {
               fine = medio - 1;
          }
          else if (strcmp (cercato, persone[medio].cognome) > 0)
               {
                    iniz = medio + 1;
               }
               else
               {
                    trovato = vero;
               }
     } while (!trovato && iniz <= fine);
     if (trovato)
     {
          return medio; /* Si ferma alla prima */
          /* occorrenza trovata */
     }
     else
     {
          return -1;
     }
} /* Fine funzione cercaBin() */

void stampa (int pos, CartaIdentita persone[])
{
     if (pos != -1)
     {
          printf ("Trovato in posiz: %d\n", pos);
          printf ("Cognome: %s\n", persone[pos].cognome);
          printf ("Nome: %s\n", persone[pos].nome);
          printf ("Età: %u\n", persone[pos].eta);
          printf ("Indirizzo:%s\n", persone[pos].indirizzo);
     }
```

```
        else
        {
                printf ("Non trovato\n");
        }
}
void main()
{
        const int LUNGH = 5;
        CartaIdentita listaPersone[LUNGH];
        Stringa cognomeCercato;
        int posiz;
        unsigned int i;
        for (i = 0; i < LUNGH; i++)
        {
                printf ("Cognome: ");
                scanf ("%s", listaPersone[i].cognome); /* senza & */
                printf ("Nome: ");
                scanf ("%s", listaPersone[i].nome); /* senza & */
                printf ("Età: ");
                scanf ("%u", &listaPersone[i].eta); /* con & */
                printf ("Indirizzo: ");
                scanf ("%s", listaPersone[i].indirizzo); /* senza & */
        }
        printf ("Cognome da cercare: ");
        scanf ("%s", cognomeCercato);
        posiz = cercaBin (listaPersone, LUNGH, cognomeCercato);
        stampa (posiz, listaPersone);
}
```

Elimina tag HTML

Testo

Il programma legge un file di testo contenente una pagina HTML, elimina i tag presenti e salva il testo in un altro file. I tag scartati saranno visualizzati sul monitor.

Implementazione

```
#include <stdio.h>
void eliminaTag (char nomeFileHTML[],char nomeFileTesto[])
{
      typedef enum {tag, testo, errore} TipoStato;
      FILE *docHTML, *docTesto;
      char car;
      TipoStato stato = testo;
      docHTML = fopen (nomeFileHTML, "r");
      if (docHTML == NULL)
      {
            printf ("File %s non trovato\n", nomeFileHTML);
            return;
      }
      docTesto = fopen (nomeFileTesto, "w");
      fscanf (docHTML, "%c", &car);
      while (!feof(docHTML) && stato != errore)
      {
            if (stato == testo)
            {
            switch (car)
            {
                  case '<': stato = tag;
                              break;
                  case '>': stato = errore;
                              break;
                  default: fprintf (docTesto, "%c", car);
                              break;
            }
      }
      else if (stato == tag)
            {
                  switch (car)
                  {
                        case '<': stato = errore;
                              break;
                        case '>': stato = testo;
                              break;
                        default: printf ("%c", car);
                              break;
            }
      }
      fscanf (docHTML, "%c", &car);
      }
      fclose (docTesto);
      fclose (docHTML);
}

void main()
{
```

```
    char nomeFileHTML[255], nomeFileTesto[255];
    printf ("Nome file HTML: ");
    scanf ("%s", nomeFileHTML);
    printf ("Nome file testo: ");
    scanf ("%s", nomeFileTesto);
    eliminaTag (nomeFileHTML, nomeFileTesto);
}
```

1.3 File binario ordinato

Testo

L'utente inserisce una serie di numeri float **ordinati** in senso crescente; l'inserimento è terminato immettendo uno zero.
L'elaboratore inserisce questi numeri in un file binario.L'utente inserisce un numero float e l'elaboratore verifica se è presente nel file.
Utilizzare l'algoritmo di ricerca binaria.

Implementazione

```
#include <stdio.h>
#include <string.h>
/* si suppone inserimento in ordine */
void inserisci (char nomeFile[])
{
	FILE *archivio;
	float numero;
	archivio = fopen (nomeFile, "wb");
	printf ("numero diverso da 0: ");
	scanf ("%f", &numero);
	while (numero != 0)
	{
		fwrite (&numero, sizeof(float), 1, archivio);
		printf ("numero diverso da 0: ");
		scanf ("%f", &numero);
	}
	fclose (archivio);
}
void cerca (char nomeFile[])
{
	typedef enum {falso, vero} Booleano;
	Booleano trovato = falso;
	FILE *archivio;
	float corrente, cercato;
	int inizio, fine, medio;
	printf ("Num cercato: ");
	scanf ("%f", &cercato);
	archivio = fopen (nomeFile, "rb");
	inizio = 0;
	fseek (archivio, -sizeof(float), SEEK_END);
	fine = ftell (archivio) / sizeof(float);
	do
	{
		medio = (inizio + fine) / 2;
		fseek (archivio, medio * sizeof(float), SEEK_SET);
		fread (&corrente, sizeof(float), 1, archivio);
		if (cercato < corrente)
		{
			fine = medio - 1;
		}
		else if (cercato > corrente)
			{
				inizio = medio + 1;
			}
			else
			{
				trovato = vero;
			}
```

```
      } while (!trovato && inizio <= fine);
      if (trovato)
      {
            printf ("Trovato in posiz %u", medio);
      }
      else
      {
            printf ("Non trovato!");
      }
      fclose (archivio);
}
void main()
{
      char nomeArchivio[] = "numeri.dat";
      inserisci (nomeArchivio);
      cerca (nomeArchivio);
}
```

1.4 Archivio film

Testo

Il programma dovrà permettere all'utente, tramite un menù, di gestire un archivio di film.
Dovranno essere disponibili le seguenti operazioni:
1. L'utente inserisce un certo numero di film:
a. Codice identificativo numerico e titolo (senza spazi).
b. I film dovranno essere inseriti con il codice già **ordinato** in senso crescente.
c. L'inserimento della lista di film è terminato immettendo un codice pari a zero.
2. L'utente inserisce un codice e l'elaboratore verifica se il corrispondente film è presente nel file (ed in questo caso ne scrive i dati sul monitor).
Utilizzare l'algoritmo di ricerca binaria.

Implementazione

```
#include <stdio.h>
typedef struct
{
     unsigned int codice;
     char titolo[100];
} Film;
/* si suppone inserimento con il codice ordinato in
senso crescente */
void inserisci (char nomeFile[])
{
     FILE *archivio;
     Film nuovoFilm;
     /* apertura file per scrivere in append */
     archivio = fopen (nomeFile, "ab");
     printf ("Codice film diverso da 0: ");
     scanf ("%u", &nuovoFilm.codice);
     if (nuovoFilm.codice != 0)
     {
          printf ("Titolo film: ");
          scanf ("%s", nuovoFilm.titolo);
     }
     while (nuovoFilm.codice != 0)
     {
          fwrite (&nuovoFilm, sizeof(nuovoFilm), 1, archivio);
          printf ("Codice film diverso da 0: ");
          scanf ("%u", &nuovoFilm.codice);
          if (nuovoFilm.codice != 0)
          {
               printf ("Titolo film: ");
               scanf ("%s", nuovoFilm.titolo);
          }
     }
     fclose (archivio);
}

void cerca (char nomeFile[])
{
     typedef enum {falso, vero} Booleano;
     Booleano trovato = falso;
     FILE *archivio;
     Film filmCorrente;
     unsigned int codCercato;
```

```
	int inizio, fine, medio;
	printf ("Codice cercato: ");
	scanf ("%u", &codCercato);
	archivio = fopen (nomeFile, "rb");
	inizio = 0;
	fseek (archivio, -sizeof(Film), SEEK_END);
	fine = ftell (archivio) / sizeof(Film);
	do
	{
		medio = (inizio + fine) / 2;
		fseek (archivio, medio * sizeof(Film), SEEK_SET);
		fread (&filmCorrente, sizeof(filmCorrente), 1,
		archivio);
		if (codCercato < filmCorrente.codice)
		{
			fine = medio - 1;
		}
		else if (codCercato > filmCorrente.codice)
			{
				inizio = medio + 1;
			}
			else
			{
				trovato = vero;
			}
	} while (!trovato && inizio <= fine);
	if (trovato)
	{
		printf ("Trovato in posiz: %u\n", medio);
		printf ("Codice: %u\n", filmCorrente.codice);
		printf ("Titolo: %s\n", filmCorrente.titolo);
	}
	else
	{
		printf ("Non trovato!\n");
	}
	fclose (archivio);
}
unsigned int menu ()
{
	unsigned int scelta;
	printf ("1. Inserisci elenco film\n");
	printf ("2. Cerca film\n");
	printf ("3. Esci\n");
	printf ("\nScelta: ");
	scanf ("%u", &scelta);
	return scelta;
}
void main()
{
	char nomeArchivio[] = "numeri.dat";
	unsigned int n;
	do
	{
		n = menu();
		switch (n)
		{
			case 1: inserisci (nomeArchivio);
				break;
```

```
                case 2: cerca (nomeArchivio);
                        break;
            }
        } while (n != 3);
}
```

2.1 Operazioni con i puntatori

Dato il seguente frammento di programma C:

```
int *p1, *p2, x, y;
x = 10;
y = 20;
p1 = &x;
p2 = p1;
*p1 = 5;
p2 = &y;
y = 30;
*p2 = *p1 + *p2;
*p1 = *p1 + x;
```

disegnare, a fianco di ciascuna riga, il valore di tutte le variabili al termine dell'esecuzione della riga stessa.

Le variabili `x` e `y` si trovano, rispettivamente, nelle celle di memoria di indirizzo 1000 e 2000.

Soluzione esercizi 2.1 e 2.2

Espressione	P1	X(1000)	Y(2000)	P2
`int *p1, *p2, x, y`	?	?	?	?
`x = 10`	?	10	?	?
`y = 20`	?	10	20	?
`p1 = &x`	1000	10	20	?
`p2 = p1`	1000	10	20	1000
`*p1 = 5`	1000	5	20	1000
`p2 = &y`	1000	5	20	2000
`y = 30`	1000	5	30	2000
`*p2 = *p1 + *p2`	1000	5	35	2000
`*p1 = *p1 + x`	1000	10	35	2000

4.1 Battaglia navale

Testo

Scrivere un programma che consenta di giocare a “battaglia navale”. Ogni giocatore avrà a disposizione una campo di 5 x 5 celle, le navi da sistemare sono:
- Una fregata (una cella)
- Un incrociatore (due celle)
- Una portaerei (tre celle)

Il numero massimo di mosse ammesse sarà stabilito e fissato all’interno del programma..

Implementazione

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

const unsigned int DIM = 5, MAXMOSSE = 20;
typedef enum {orizzontale, verticale} Direzione;
typedef enum {falso, vero} Booleano;
typedef Booleano Matrice[DIM][DIM];
typedef struct
{
     Matrice campo;
     unsigned int aSegno, aVuoto;
} Giocatore;
typedef struct
{
     unsigned int x, y;
} Punto;

void visualizza (Matrice campo)
{
     unsigned int i, j;
     for (j = 0; j < DIM; j++)
     {
          for (i = 0; i < DIM; i++)
          {
               if (campo[i][j] == vero)
               {
                    printf ("X\t");
               }
               else
               {
                    printf (".\t");
               }
          }
          printf ("\n");
     }
}

void inserisciNavi (Matrice campo, unsigned int lunghNave,
unsigned int gioc)
{
     Punto p;
     Direzione d;
     unsigned int i;
     clrscr(); /* Cancella lo schermo; definita in conio.h */
     printf("Gioc: %u. Inserisci nave %u\n", gioc, lunghNave);
```

```
        printf ("Coord (x y): ");
        scanf ("%u %u", &p.x, &p.y);
        printf ("0=orizz, 1=vert: ");
        scanf ("%u", &d);
        if (d == orizzontale)
        {
                for (i = p.x; i < p.x + lunghNave; i++)
                {
                        campo[i][p.y] = vero;
                }
        }
        else
        {
                for (i = p.y; i < p.y + lunghNave; i++)
                {
                        campo[p.x][i] = vero;
                }
        }
        visualizza (campo);
        scanf ("%*c%*c");
}

void sparaColpo (Matrice campo, unsigned int *aSegno,
unsigned int *aVuoto, unsigned int gioc)
{
        Punto p;
        clrscr();
        printf ("Gioc. %u. Coord (x y): ", gioc);
        scanf ("%d %d", &p.x, &p.y);
        if (campo[p.x][p.y] == vero)
        {
                (*aSegno)++;
                campo[p.x][p.y] = falso;
                printf ("Colpito! A segno:%u a vuoto:%u", *aSegno,
                *aVuoto);
        }
        else
        {
                (*aVuoto)++;
                printf ("A vuoto! A segno:%u a vuoto:%u", *aSegno,
                *aVuoto);
        }
        scanf ("%*c%*c");
}

Booleano vittoria (Matrice campo)
{
        unsigned int x,y;
        for (x = 0; x < DIM; x++)
        {
                for (y = 0; y < DIM; y++)
                {
                        if (campo[x][y] == vero)
                        {
                                return falso;
                        }
                }
        }
        return vero;
```

```
}

void inizializza (Giocatore gioc[])
{
     unsigned int i, j;
     for (i = 0; i < DIM; i++)
     {
          for (j = 0; j < DIM; j++)
          {
               gioc[0].campo[i][j] = falso;
               gioc[1].campo[i][j] = falso;
          }
     }
     gioc[0].aVuoto = 0;
     gioc[0].aSegno = 0;
     gioc[1].aVuoto = 0;
     gioc[1].aSegno = 0;
}

void main()
{
     Giocatore gioc[2];
     unsigned int aSegnoA, aVuotoA, aSegnoB, aVuotoB;
     unsigned int nGioc, i, mosse;
     Booleano haVinto;
     inizializza (gioc);
     for (i = 1; i <= 3; i++)
     {
          inserisciNavi (gioc[0].campo, i, 0);
          inserisciNavi (gioc[1].campo, i, 1);
     }
     nGioc = 0;
     mosse = 0;
     do
     {
          sparaColpo (gioc[1 - nGioc].campo, &gioc[nGioc].aSegno,
          &gioc[nGioc].aVuoto, nGioc);
          haVinto = vittoria(gioc[1 - nGioc].campo);
          nGioc = 1 - nGioc;
          mosse++;
     }while (!haVinto && mosse < MAXMOSSE);
     if (haVinto)
     {
          printf ("Vittoria di %u\n", 1 - nGioc);
          printf ("Colpi a segno: %u, a vuoto: %u",
          gioc[1 - nGioc].aSegno, gioc[1 - nGioc].aVuoto);
     }
}
```