Fondamenti di Informatica T-1

Argomenti di questa esercitazione

- Stringhe
- Strutture

ESERCIZIO 1
(stringhe)

- Scrivere una funzione che date tre stringhe A, B e C concateni in C il contenuto di A e B e restituisca il numero di caratteri copiati in C

```
int conc(char A[],char B[],char C[]);
```

Esercizio 1 – Soluzione
(stringhe)

```
int conc (char s1[], char s2[], char dest[]) {
   int result;
   strcpy(dest, s1);
   strcpy(&dest[strlen(dest)], s2);
   return strlen(dest);
}

int conc2 (char s1[], char s2[], char dest[]) {
   int result=0;
   while(*s1 != '\0') {
        *dest = *s1;
        dest++; s1++; result++;
   }
   while(*s2 != '\0') {
        *dest = *s2;
        dest++; s2++; result++;
   }
   *dest = '\0';
   return result;
}
```

Esercizio 1 – Soluzione
(stringhe)

```
#define DIM 128

int main(void) {
   char s1[] = "Che bel ";
   char s2[] = "castello!";
   char s3[DIM];

   conc(s1, s2, s3);

   printf("%s\n", s3);
   return 0;
}
```

ESERCIZIO 2
(stringhe)

- Al fine di stampare degli indirizzi su delle buste, è necessario comporre la prima parte dell'indirizzo come "Cognome Nome" o "Cognome N."
- Si realizzi una funzione che riceva come parametri:
  - il cognome
  - il nome
  - una stringa che conterrà la prima parte dell'indirizzo
  - la lunghezza massima della stringa indirizzo

ESERCIZIO 2
(stringhe)

- La funzione deve copiare/concatenare nell'indirizzo il cognome seguito dal nome, avendo cura di rispettare le dimensioni della stringa indirizzo. Qualora la stringa indirizzo sia troppo piccola per contenere entrambi, la funzione provi a comporre la stringa come "Cognome N."

- Qualora neanche ciò sia possibile, la funzione ritorni un codice di errore opportuno (ad esempio -1)

- Se non si verifica nessun errore la funzione deve restituire il numero di caratteri nella stringa Indirizzo

ESERCIZIO 2
(stringhe)

- Si realizzi una funzione che riceva come parametri:
  - il cognome
  - il nome
  - una stringa che conterrà la prima parte dell'indirizzo
  - la lunghezza massima della stringa indirizzo

```
int indirizzo(char Cognome[], char Nome[],
  char Indirizzo[], int dim);
```

Esempio: Se il cognome è Rossi e il nome è Mario e la dimensione dim = 15 allora la stringa Indirizzo sara' "Rossi Mario". Se invece la dimensione fosse 8 allora la stringa indirizzo sarebbe "Rossi M.". Se la dimensione fosse 5 allora verrebbe restituito un codice di errore -1

ESERCIZIO 2
(stringhe)

- Per la risoluzione di questo esercizio si possono utilizzare le funzioni disponibili nella libreria standard <string.h>.
  - strlen() per determinare la lunghezza di una stringa
  - strcat() o strcpy() per comporre in indirizzo la nuova stringa

- Si provi ad organizzare il progetto su più file...

ESERCIZIO 2 - Soluzione
(stringhe)

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


#define TRUE    1
#define FALSE   0

#define RESULT_OK 1
#define RESULT_ADDRESS_TOO_LONG -1
#define RESULT_COMPRESSED_NAME -2

#define MAX 50

typedef int BOOL;
typedef int resultType;
```

ESERCIZIO 2 - Soluzione
(stringhe)

```
resultType componiIndirizzo(char * cognome, char * nome, char * indiriz, int maxChars) {
    int requiredChars;
    int size;
    resultType result;

    strcpy(indirizzo, ""); //inizializzazione...
    requiredChars = strlen(cognome) + 1 + strlen(nome);
    if (requiredChars < maxChars)
          result = RESULT_OK;
    else {
          requiredChars = strlen(cognome) + 3;
          if (requiredChars < maxChars)
                    result = RESULT_COMPRESSED_NAME;
          else
                    result = RESULT_ADDRESS_TOO_LONG;
    }
    if (result ==  RESULT_OK || result == RESULT_COMPRESSED_NAME) {
          strcat(indirizzo, cognome);
          strcat(indirizzo, " ");
          if (result == RESULT_OK)
                    strcat(indirizzo, nome);
          else {
                    size = strlen(indirizzo);
                    indirizzo[size] = nome[0];
                    indirizzo[size+1] = '.';
                    indirizzo[size+2] = '\0';     }
    }
    return result;
}
```

ESERCIZIO 2 - Soluzione
(stringhe)

```
void handleError(resultType result) {
    switch (result) {
          case RESULT_OK:
                printf("Nessun errore occorso!\n"); break;
          case RESULT_ADDRESS_TOO_LONG:
                printf("L'indirizzo e' troppo lungo...\n"); break;
          case RESULT_COMPRESSED_NAME:
                printf("Il nome e' stato compresso...\n"); break;
          default:
                printf("Unknown Error!\n");
    }
}

int main(void)
{
    char indirizzo[MAX];
    resultType result;
    char cognome[10];

    scanf("%s", cognome);
    result = componiIndirizzo("Chesani", "Federico", indirizzo, MAX-1);
    if ((result == RESULT_OK) || (result == RESULT_COMPRESSED_NAME)) {
          printf("%s\n", indirizzo);
          printf("Lunghezza indirizzo: %d\n", strlen(indirizzo));
    }
    else
          handleError(result);
    return (0);
}
```

ESERCIZIO 3
(stringhe)

- Scrivere una funzione C che, data una stringa A ed una stringa B, calcoli il numero di occorrenze della stringa A in B.
- `int occorrenze(char A[], char B[]);`

- Ad esempio, se B="tre tigri contro tre tigri" ed A="tr", deve restituire 3.

ESERCIZIO 3 - Soluzione
(stringhe)

```
int occorrenze(char a[], char b[]) {
    int result = 0;
    int trovato = 0;
    int i=0, j=0, temp;

    while (b[j] != '\0') {
         if (b[j] == a[i]) {
                  trovato = 1;
                  temp = j;
                  while (a[i] != '\0' && trovato) {
                           if (a[i] == b[j]) {
                                    i++;
                                    j++;
                           }
                           else {
                                    trovato = 0;
                           }
                  }
                  if (trovato)
                           result++;
                  j = temp;
                  i=0;
         }
         j++;
    }
    return result; }
```

ESERCIZIO 3 - Soluzione
(stringhe)

```
int main(void) {
   int s1[] = "tr";
   int s2[] = "tre tigri contro tre tigri";

   printf("Numero di occorrenze: %d\n", occorrenze(s1, s2));

   return 0;
}
```

ESERCIZIO 4
(stringhe)

Scrivere una procedura ricorsiva:

void printchar(char stringa[])

che stampi, *ricorsivamente*, tutti i caratteri contenuti in **stringa**, un carattere per linea, assumendo che **stringa** sia *ben formata*.

ESERCIZIO 4 - Soluzione
(stringhe)

```
void printchar (char stringa[]) {
   if (stringa[0] == '\0')
       return;
   else {
       printf("%c\n", stringa[0]);
       printchar(&(stringa[1]));
   }
}

int main(void) {
   char s1[] = "Marcondirondirondello!";
   printchar(s1);
   return 0;
}
```

ESERCIZIO 5
(stringhe)

Stampa di numeri reali con "dettaglio" a piacimento

- ***Si realizzi una funzione stampaDettagli(…) che riceva come parametri un numero reale e due interi (che indicano, rispettivamente, il numero di cifre per la parte intera e per la parte decimale)***
- ***La funzione stampi a video il numero secondo le indicazioni ricevute come parametri***
- Come? Tramite una opportuna stringa di formato (es. "%6.2f" significa stampare un float con 6 cifre per la parte intera e due per la parte decimale)
- La funzione componga dinamicamente una opportuna stringa di formato, utilizzando la funzione sprintf(…) ed una stringa allocata staticamente (si controlli di non eccedere la dimensione della stringa già allocata)

ESERCIZIO 5
(stringhe)

- Pseudo algoritmo:
  - Creo una stringa di dimensione prefissata MAX, dove comporrò il formato
  - Calcolo quanti caratteri sono necessari per comporre la stringa di formato
  - Se ho “spazio a sufficienza”, con sprintf() scrivo nella stringa di formato il formato che desidero
  - Altrimenti restituisco un apposito codice di errore

ESERCIZIO 5 - Soluzione
(stringhe)

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


#define TRUE    1
#define FALSE   0
#define BOOLEAN int;

#define MAX_FORMATO 128
#define RESULT_OK 1
#define RESULT_TOO_MANY_CHARS -1
#define RESULT int;
```

ESERCIZIO 5 - Soluzione
(stringhe)

```
RESULT stampaDettagli(     float num,
                      int cifreParteIntera,
                      int cifreParteDecimale) {
  char stringaFormato[MAX_FORMATO];
  int nCarInt, nCarDec;
  RESULT result;

  nCarInt = ((int) log10(cifreParteIntera)) + 1;
  nCarDec = ((int) log10(cifreParteDecimale)) + 1;

  if ((nCarInt + nCarDec + 6) >= MAX_FORMATO)
     result = RESULT_TOO_MANY_CHARS;
  …
```

ESERCIZIO 5 - Soluzione
(stringhe)

```
    else
    {
        sprintf(    stringaFormato,
                    "%%%d.%df\\n",
                    cifreParteIntera,
                    cifreParteDecimale);
        printf("Formato: %s\n",stringaFormato);
        printf(stringaFormato, num);
        result = RESULT_OK;
    }
    return result;
}
```

ESERCIZIO 5 - Soluzione
(stringhe)

```
void handleError(RESULT result, char *str)
{
   switch (result)
   {
        case RESULT_OK:
              strcpy(str, "");
              break;
        case RESULT_TOO_MANY_CHARS:
              strcpy(str, "Stringa di formato con troppi caratteri\n");
              break;
        default:
              strcpy(str, "Risultato non gestito.\n");
   }
}
```

ESERCIZIO 5 - Soluzione
(stringhe)

```
int main(void)
{
   RESULT result;
   char resultStr[MAX_FORMATO];
   result = stampaDettagli(2.0, 3, 8);

   if (result!=RESULT_OK)
       handleError(result, resultStr);
   printf("errore %s\n", resultStr);
   return (0);
}
```

ESERCIZIO 6
(stringhe)

Codice segreto nella pagina di un libro

- Sono date due stringhe, una denominata msg e una denominata secret (non più lunga di msg) di caratteri tutti minuscoli
- Si vuole sapere se tutti i caratteri di secret sono contenuti nello stesso ordine (ma eventualmente intervallati da altri caratteri) nella stringa msg
- In caso positivo, il programma deve restituire una copia del msg originale, dove però ad ogni lettera riconosciuta come facente parte di secret viene sostituita la lettera maiuscola
  - Es: msg = “ma che bel castello”, secret = “cestello”
  - Risultato: **SI** e stringa **“ma ChE bel caSTELLO”**

ESERCIZIO 6 - Soluzione
(stringhe)

```
BOOLEAN identifica(       char * msg,
                          char * secret,
                          char * result) {

   while ((*msg != '\0') && (*secret != '\0')) {
         if (*msg == *secret) {
                  secret++;
                  *result = *msg - 'a' + 'A';
         }
         else
                  *result = *msg;
         msg++;
         result++;
   }

   if (! *secret)
         while (*msg != '\0') {
                   *result = *msg;
                  msg++;
                  result++;
         }
   return (! *secret);
}
```

ESERCIZIO 7
(stringhe)

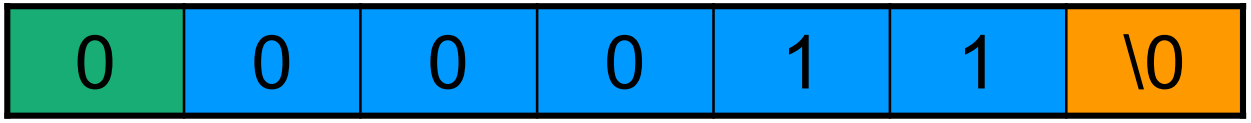
Conversione di numeri interi in rappresentazione a modulo 2 e operazioni

- Si realizzi un programma capace di
  - Effettuare la conversione da numeri interi (con segno!) alla corrispondente rappresentazione in modulo 2 e viceversa
  - Effettuare la somma e sottrazione di due numeri interi utilizzando la rappresentazione in modulo 2
- Si utilizzino le stringhe per la rappresentazione in modulo 2

ESERCIZIO 7
(stringhe)

In particolare, si definiscano le seguenti funzioni

- `int convertiBin(char* bin);`
  - Ottenere la codifica binaria del valore assoluto e convertire quella
- `RESULT convertiInt(int value, char* res);`
  - Dove RESULT può valere
    - OK se la conversione è stata effettuata con successo
    - SIZE_OVERFLOW se la dimensione fisica di res non è abbastanza grande per contenere la conversione
      - si utilizzi il logaritmo in base 2 per calcolare lo spazio necessario, e si considerino segno e terminatore
- Facciamo in modo che le stringhe siano riempite sempre per tutta la dimensione fisica
  - Es: se la dimensione fisica è 7, la rappresentazione del numero 3 dovrà essere



ESERCIZIO 7
(stringhe)

- `void invertiSegno(char *num, char *res);`
  - Inverte il segno di num (ovvero realizza l'algoritmo del modulo 2)
    - PASSO 1: inversione dei bit
    - PASSO 2: aggiunta di uno
      - Suggerimento: si converta l'intero 1 in binario e si utilizzi la funzione sum (vedi sotto)
- `RESULT sum(char *add1, char *add2, char *res);`
  - Somma i due numeri binari (controllare che abbiano la stessa lunghezza)
  - Restituisce SIZE_OVERFLOW se l'operazione causa overflow (x-or dei due carry più significativi)
- NOTA: si faccia in modo che le funzioni calcolino un risultato corretto anche passando come risultato una delle variabili date in input
  - Es: **`invertiSegno(num, num)`** deve invertire correttamente il segno di **`num`** (dopo l'invocazione, **`num`** contiene il risultato)

ESERCIZIO 7 - Soluzione
(stringhe)

```
int convertiBin(char* bin) {
   int res = 0, i, segno = 1;
   char absBin[MAXSIZE];
   if(bin[0]=='1') {
        invertiSegno(bin, absBin);
        segno = -1;
   }
   else
        strcpy(absBin, bin);

   for(i = strlen(absBin)-1; i >= 0; i--) {
        if(absBin[i] == '1')
              res = res + pow(2, strlen(absBin)-i-1);
   }
   return segno * res;
}
```

Il numero è negativo, quindi derivo la codifica del valore assoluto facendone il modulo due

Il numero è positivo, quindi ho già la codifica del valore assoluto

ESERCIZIO 7 - Soluzione
(stringhe)

```
RESULT convertiInt(int num, char *bin) {
   int i, absNum;
   absNum = abs(num);
   if((int)log2(absNum) >= MAXSIZE-2)
      return SIZE_OVERFLOW;

   for(i = 0; i < MAXSIZE - 1; i++)
      bin[i] = '0';
   bin[MAXSIZE - 1] = '\0';

   ...
```

Tolgo un posto per il bit di segno e un posto per il terminatore

Riempio tutte le celle a 0 (così non avrò posizioni scoperte)

ESERCIZIO 7 - Soluzione
(stringhe)

```
    i = 0;
    while(absNum != 0) {
        bin[i] = '0' + (absNum % 2);
        absNum = absNum / 2;
        i++;
    }
    inverti(bin);
    if(num < 0)
        invertiSegno(bin, bin);
    return OK;
}
```

Codifico il valore assoluto

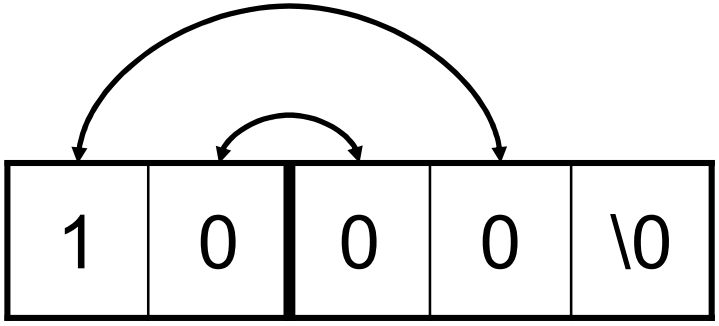
Devo ottenere la stringa "speculare" (oppure riempire il vettore bin a partire dal fondo - tanto ne conosciamo la dimensione)

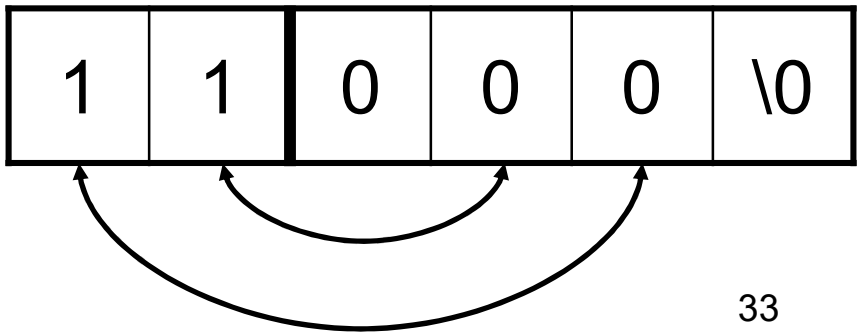
Se il numero è negativo ne devo invertire il segno (modulo 2)

ESERCIZIO 7 - Soluzione
(stringhe)

```
void inverti(char* str)
{
   int i;
   char temp;
   for(i = 0; i < strlen(str)/2; i++)
   {
        temp = str[i];
        str[i] = str[strlen(str) - i - 1];
        str[strlen(str) - i - 1] = temp;
   }
}
```

È l'intero inferiore rispetto alla metà (in caso di elementi dispari quello in mezzo rimane dov'è)





ESERCIZIO 7 - Soluzione
(stringhe)

```
void invertiSegno(char *num, char *res)
{
  int i;
  char one[MAXSIZE];
  convertiInt(1, one);
  for(i = 0; i < strlen(num); i++)
  {
     res[i] = '0' + (num[i] == '0' ? 1 : 0);
  }
  res[MAXSIZE - 1] = '\0';
  sum(res, one, res);
}
```

Inversione dei bit

Aggiunta di 1

ESERCIZIO 7 - Soluzione
(stringhe)

```
RESULT sum(char *add1, char *add2, char *res)
{
   int i;
   int carry = 0;
   int curCarry = 0;
   int value;
   BOOLEAN overflow;
   if( strlen(add1) != strlen(add2))
       return GENERIC_ERROR;
   …
```

Verifico che le stringhe abbiano la stessa lunghezza

ESERCIZIO 7 - Soluzione
(stringhe)

```
for(i = strlen(add1) - 1; i >= 0; i--)
   {
      curCarry = carry + (add1[i]-'0') + (add2[i]-'0') >= 2;
      res[i] = '0' + (carry != (add1[i]-'0' != add2[i]-'0' ?
                        1 : 0) ? 1: 0);
       if(i==0)
              overflow = curCarry != carry ? 1 : 0;
       carry = curCarry;
   }
   res[MAXSIZE - 1] = '\0';
   if(overflow)
        return SIZE_OVERFLOW;
   else
        return OK;
}
```

Procedo dal bit meno significativo fino al bit di segno

Prima il calcolo del nuovo carry se voglio poter sovrascrivere un addendo (se res è add1 o add2, in questo passo sovrascrivo l'i-mo bit)
NOTA: res[i] si calcola facendo xor tra i due bit i-mi degli addendi e il carry

Se i == 0 il carry corrente è quello del segno, quindi faccio x-or con il carry precedente per l'overflow

ESERCIZIO 7 – Sullo XOR
(stringhe)

- L'algoritmo può diventare più chiaro introducendo una funzione che esegua l'exclusive or fra due bit

```
int xor(char bit1, char bit2)
{
     return bit1 == bit2 ? 0 : 1;
}
```

Esercizio: integrare la funzione di cui sopra nel codice…

ESERCIZIO 1
(strutture)

Si realizzi un programma C che legga da utente i dati relativi ad alcuni corsi. In particolare, per ogni corso vengono dati:

- ***denominazione*** *del corso: una stringa di 20 caratteri che riporta il nome del corso;*
- ***cognome*** *del docente: una stringa di 15 caratteri che rappresenta il cognome del docente del corso;*
- ***iscritti****: un intero che indica il numero di studenti che frequentano il corso.*

Il programma deve stampare la denominazione del corso e il cognome del docente relativi a tutti i corsi che hanno il numero di iscritti maggiore o uguale alla media aritmetica degli iscritti (calcolata su tutti i corsi).

ESERCIZIO 1
(strutture)

Attenzione: abbiamo bisogno di un ARRAY di strutture !!!!!

Esempio: l'utente inserisce i seguenti dati per 3 corsi

analisi
obrecht
55
fond.inf
milano
40
geometria
ferri
37

analisi	***fond.inf***	***geometria***
obrecht	***milano***	***ferri***
55	***40***	***37***

La media è di 44 quindi il programma stamperà:

analisi
obrecht

ESERCIZIO 1 - Soluzione
(strutture)

```
#include <stdio.h>
#define N 30
typedef struct stud {
            char denominazione[21];
            char cognome_docente[16];
            int studenti;
    } Corso;

int main() {
    int i, nc;
    float somma media;
    Corso corsi[N];
    printf("Inserisci il numero dei corsi ");
    scanf("%d", &nc);
    /* inserimento dati */
    for (i=0; i<nc && i<N; i++) {
            printf("Inserisci il nome del corso ");
            scanf("%s",corsi[i].denominazione);
            printf("Inserisci il cognome del docente ");
            scanf("%s",corsi[i].cognome_docente);
            printf("Inserisci il numero degli iscritti");
            scanf("%d",&corsi[i].studenti); }
```

Continua...

ESERCIZIO 1 - Soluzione
(strutture)

```
    somma=0;
    for (i=0; i< nc; i++)
            somma=somma + corsi[i].studenti;
    media= somma/nc;

    for (i=0; i< nc; i++)
            if (corsi[i].studenti>=media)
                    printf("%s %s\n",corsi[i].denominazione,
                                    corsi[i].cognome_docente);
}
```

ESERCIZIO 2
(strutture)

1) Si scriva un programma C che legga una serie di dati e li memorizzi in un vettore SQUADRE (di dimensione statica prefissata) contenente strutture (**`struct squadra`**) del tipo:

- ***nome squadra*** *(stringa di lunghezza 20)*
- ***codice squadra*** *(intero)*
- ***goal fatti*** *(intero)*
- ***goal subiti*** *(intero)*

2) Stampi a terminale tutti i nomi e codici delle squadre che hanno fatto un numero di goal maggiore del numero dei goal subiti.

3) Letto a terminale un codice di una squadra stampi a video il nome della squadra, si stampino i goal fatti e i goal subiti.

ESERCIZIO 2
(strutture)

Attenzione: abbiamo bisogno di un ARRAY di strutture !!!!!

Esempio: l'utente inserisce i seguenti dati per 3 squadre

juventus
1
10
12
milan
2
7
6
inter
3
13
11

juventus	***milan***	***inter***
1	***2***	***3***
10	***7***	***13***
12	***6***	***11***

2) Viene stampato a video

milan 2
inter 3

3) Se l'utente digita 1 viene stampato

juventus 10 12

ESERCIZIO 2 - Soluzione
(strutture)

```
#include <stdio.h>
#define N 30

typedef struct squadra{
        char nome[20];
        int codice;
        int goal_fatti, goal_subiti;
} Squadra;

void main() {
        int i, ns, cod, T;
        Squadra squadre[N];

        printf("Inserisci il numero delle squadre");
        scanf("%d", &ns);

        /* inserimento dati */
        for (i=0; i<ns; i++) {
                printf("Inserisci nome, codice, goal fatti e subiti \n");
                scanf("%s", squadre[i].nome);
                scanf("%d", &squadre[i].codice);
                scanf("%d", &squadre[i].goal_fatti);
                scanf("%d", &squadre[i].goal_subiti);
        }
```

ESERCIZIO 2 - Soluzione
(strutture)

```
/* punto 2 */
    for (i=0; i<ns; i++) {
        if(squadre[i].goal_fatti> squadre[i].goal_subiti)
                printf("%s\n", squadre[i].nome);
                printf("%d\n", squadre[i].codice);
        }

/* punto 3 */
    printf("Inserisci un codice ");
    scanf("%d", &cod);
    i=0; T=0;
    while ((i < ns)&& (T==0)) {
        if (squadre[i].codice == cod)  {
                printf("%s\n", squadre[i].nome);
                printf("%d\n", squadre[i].goal_fatti);
                printf("%d\n", squadre[i].goal_subiti);
                T=1;}
        i++;
    }
    if (T==0) printf("codice non trovato");
}
```

ESERCIZIO 3
(strutture)

Si vuole implementare un programma per il calcolo dell'inflazione su determinati prodotti commerciali.

A tal scopo ogni prodotto è rappresentato tramite una struttura **`item`**, definita da una stringa **`name`** con il nome del prodotto, e da due float **`old_price`** e **`new_price`** rappresentanti i prezzi.

ESERCIZIO 3
(strutture)

a) Si scriva una funzione **`lettura()`** che riceva come parametri di ingresso un vettore **`prezzi`** di strutture **`item`**, la dimensione fisica **`max`** del vettore **`prezzi`**, e un puntatore a intero **`num`** che rappresenta la dimensione logica del vettore. La funzione deve leggere da standard input il nome del prodotto ed i due prezzi, e deve copiare tale informazione nella prima posizione libera nel vettore **`prezzi`**.

ESERCIZIO 3
(strutture)

La funzione deve terminare se l'utente inserisce come nome del prodotto il termine "fine", oppure se viene raggiunta la dimensione fisica del vettore.

La dimensione logica del vettore **`prezzi`** così riempito deve essere restituita tramite il parametro **`num`** (passato appunto per riferimento). Al termine della lettura dei dati la funzione deve restituire il valore 0.

ESERCIZIO 3
(strutture)

b) Si scriva un programma **`main`** che, dopo aver definito un vettore di strutture **`item`** (di dimensione massima **`MAX_ITEM`**), invochi la funzione **`lettura()`** per riempire tale vettore.

Il programma stampi poi a video nome e tasso d'inflazione per ogni prodotto, utilizzando la formula:

$$infl_i = \left(\frac{new\_price_i}{old\_price_i} - 1 \right) * 100$$

ESERCIZIO 3 - Soluzione
(strutture)

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

#define DIM 21
#define MAX_ITEM 100

typedef struct {
    char name[DIM];
    float old_price;
    float new_price;
} Item;
...
```

ESERCIZIO 3 - Soluzione
(strutture)

```
int lettura (Item prezzi[], int max, int * num) {
     char name[DIM];
     *num = 0;

     printf("Inserire nome prodotto: ");   scanf("%s", name);
     while ((strcmp(name, "fine")) && (*num < max)) {
         strcpy(prezzi[*num].name, name);
         printf("Inserire old price: ");
         scanf("%f", &prezzi[*num].old_price);
         printf("Inserire new price: ");
         scanf("%f%*c", &prezzi[*num].new_price);
         (*num)++;

         printf("Inserire nome prodotto: ");
         scanf("%s", name);
     }
     return 0;
}
...
```

ESERCIZIO 3 - Soluzione
(strutture)

```
int main() {
     Item v[MAX_ITEM];
     int num, i, result;
     float infl;

     result = lettura(v, MAX_ITEM, &num);

     if (result!=0) {
         printf("Problemi durante la lettura...\n");
     }
     else {
         for (i=0; i < num; i++) {
                  infl = (v[i].new_price/v[i].old_price -1)*100;
                  printf("Inflazione del prodotto %s: %6.2f%%\n",v[i].name, infl);
         }
     }
     return 0;
}
```

ESERCIZIO 4
(strutture)

- Sia data la struttura

```
struct time
{
  int hour, minute, second;
};
```

- Per semplicità si può definire il tipo Time

```
typedef struct time Time;
```

ESERCIZIO 4
(strutture)

Si realizzi in un modulo **tempo.h/tempo.c** un insieme di funzioni per la gestione del tipo Time. In particolare:

- Si realizzi una funzione

```
Time leggiTime()
```

che legga da input ore, minuti e secondi, e restituisca una struttura di tipo Time opportunamente inizializzato coi valori letti

- Si realizzi una funzione

```
int leggiMoreTimes(Time v[], int dim)
```

La funzione deve leggere da input delle strutture Time (a tal scopo si utilizzi la funzione definita sopra) e salvarle nel vettore v, di dimensione fisica dim. La funzione deve restituire il numero di elementi letti. La lettura termina se l'utente inserisce un tempo con ora negativa.

ESERCIZIO 4
(strutture)

- Si progetti una funzione in grado di calcolare la differenza fra due strutture **`Time`** e che restituisca il risultato in termini di una nuova struttura **`Time`**

- L'interfaccia della funzione è facilmente desumibile dalle specifiche:

```
Time subtract(Time t1, Time t2);
```

- Due possibili approcci:
  1. Trasformare in secondi, eseguire la differenza, trasformare in ore, minuti, secondi
  2. Eseguire la sottrazione direttamente tenendo conto dei riporti

ESERCIZIO 4 - Soluzione
(strutture)

File "Tempo.h"

```
#include <stdio.h>

struct time {
     int hour, minute, second;
};

typedef struct time Time;

Time leggiTime();
int leggiMoreTimes(Time v[], int dim);
Time subtract1(Time t1, Time t2);
Time subtract2(Time t1, Time t2);
```

ESERCIZIO 4 - Soluzione
(strutture)

File “Tempo.c”

```
#include "tempo.h"


Time leggiTime() {
     Time result;

     printf("Ore: ");
     scanf("%d", &result.hour );
     printf("Minuti: ");
     scanf("%d", &result.minute );
     printf("Secondi: ");
     scanf("%d", &result.second );

     return result;
}
```

…

ESERCIZIO 4 - Soluzione
(strutture)

File "Tempo.c"

```
…
int leggiMoreTimes(Time v[], int dim) {
     int result;
     Time temp;

     result = 0;
     do {
        temp = leggiTime();
        if (temp.hour>=0 && result<dim) {
               v[result] = temp;
               result++;
        }
     } while (temp.hour>=0 && result<dim);
     return result;
}
…
```

ESERCIZIO 4 - Soluzione
(strutture)

```
File "Tempo.c"
…
Time subtract1(Time t1, Time t2)
{
    int s1, s2, sResult;
    Time result;

    s1 = t1.hour * 3600 + t1.minute * 60 + t1.second;
    s2 = t2.hour * 3600 + t2.minute * 60 + t2.second;
    sResult = s1 - s2;

    result.hour = sResult / 3600;
    sResult = sResult % 3600;
    result.minute = sResult / 60;
    sResult = sResult % 60;
    result.second = sResult;

    return result;
}
```

ESERCIZIO 4 - Soluzione
(strutture)

File "Tempo.c"

```
…
Time subtract2(Time t1, Time t2) {
    Time result;
    int carry;
    result.second = t1.second - t2.second;
    carry = 0;
    if (result.second < 0) {
        result.second = 60 + result.second;
        carry = -1;
    }
    result.minute = t1.minute - t2.minute + carry;
    carry = 0;
    if (result.minute < 0) {
        result.minute = 60 + result.minute;
        carry = -1;
    }
    result.hour = t1.hour - t2.hour + carry;
    return result;
}
```

ESERCIZIO 5
(strutture)

Una compagnia di autobus che effettua servizio su lunghe distanze vuole realizzare un programma di controllo delle prenotazioni dei posti.

A tal scopo rappresenta ogni prenotazione tramite una struttura **`booking`** contenente nome del cliente (al massimo 1023 caratteri, senza spazi) e numero del posto prenotato (un intero).

Le prenotazioni effettuate vengono registrate tramite un array (di dimensione prefissata **`DIM`**) di strutture **`booking`**, di dimensione logica iniziale pari a 0.

Si realizzi il modulo C gestione.h/gestione.c, contenente la struttura dati booking e le seguenti funzioni…

ESERCIZIO 5
(strutture)

a) Si realizzi una funzione:

```
int leggi(booking * dest);
```

La funzione legge da input una struttura di tipo **`booking`** (nome cliente, numero posto) rappresentante una richiesta di prenotazione, e provvede a memorizzarla in dest.

Se l'utente inserisce come nome del cliente la stringa "**`fine`**" o la stringa "**`stampa`**", allora la funzione deve restituire i valori 0 e -1, rispettivamente.

Altrimenti la funzione deve restituire 1, a significare l'inserimento corretto di una nuova richiesta di prenotazione.

ESERCIZIO 5
(strutture)

b) Si realizzi una funzione:

```
int assegna( booking list[],
             int dim,
             int * lengthList,
             char * name,
             int pref)
```

La funzione riceve in ingresso l'array di prenotazioni e la sua dimensione fisica e logica, e poi il nome del cliente ed il posto da lui indicato. La funzione deve controllare che il posto indicato non sia già stato assegnato, ed in caso contrario deve restituire il valore 0.

ESERCIZIO 5
(strutture)

Qualora invece il posto sia ancora libero, la funzione deve assegnare tale posto al cliente copiando i dati della prenotazione nell'ultima posizione libera nell'array, e deve provvedere ad aggiornare correttamente la dimensione logica dell'array. In questo secondo caso la funzione deve invece restituire come valore 1, indicante il successo nella prenotazione.

Al fine di copiare il nome del cliente, si utilizzi la funzione di libreria

```
char * strcpy(char * s, char * ct)
```

che copia ct in s (terminatore compreso).

ESERCIZIO 5
(strutture)

c) Si realizzi un programma main (file main.c) che chieda all'operatore di inserire una richiesta di prenotazione (a tal fine si usi la funzione di cui al punto a)).

Il programma deve cercare di registrare la prenotazione tramite la funzione `assegna`; qualora l'operazione di prenotazione fallisca (perché il posto risulta essere già assegnato), il programma provveda a chiedere all'operatore di inserire una nuova richiesta prenotazione, finché non si riesca ad effettuare con successo una prenotazione.

ESERCIZIO 5
(strutture)

Qualora l'operatore inserisca il nome "**`fine`**", il programma deve terminare; qualora invece venga inserita la stringa "**`stampa`**", il programma deve stampare a video le prenotazioni già effettuate. Si noti che la funzione di cui al punto (a) restituisce un codice specifico per indicare quale stringa sia stata inserita.

ESERCIZIO 5 - Soluzione
(strutture)

File "gestione.h":

```
#include <stdio.h>
#include <string.h>
#define MAX 1024
#define DIM 10

typedef struct {
     char name[MAX];
     int seat;
} booking;

int leggi(booking * dest);
int assegna( booking list[], int * lengthList, int dim, char * name, int pref);
int assegna2( booking list[], int * lengthList, int dim, booking temp);
int stampaBooking(booking list[], int lengthList);
```

ESERCIZIO 5 - Soluzione
(strutture)

File “gestione.c”:

```
#include "gestione.h"


int leggi(booking * dest) {
      printf("Inserire il nome: ");
      scanf("%s", (*dest).name );

      if (strcmp("fine", dest->name)==0)
          return 0;
      else if (strcmp("stampa", dest->name)==0)
                    return -1;
          else {
                    printf("Posto preferito: ");
                    scanf("%d", &(dest->seat));
                    return 1;
          }
}
```

ESERCIZIO 5 - Soluzione
(strutture)

```
int assegna(   booking list[], int * lengthList,
               int dim, char * name, int pref) {

   int i=0;
   int trovato = 0;

   while (( i < *lengthList) && !trovato) {
        if (list[i].seat == pref)
                trovato = 1;
        i++;
   }

   if (!trovato && *lengthList<dim) {
        list[*lengthList].seat = pref;
        strcpy(list[*lengthList].name, name);
        (*lengthList)++;
        return 1;
   }
   else
        return 0;
}
```

ESERCIZIO 5 - Soluzione
(strutture)

```
int assegna2( booking list[], int * lengthList, int dim, booking temp) {
      int i=0;
      int trovato = 0;

      while ( (i < *lengthList) && !trovato) {
          if ( list[i].seat == temp.seat  )
                    trovato = 1;
          i++;
      }
      if (!trovato && *lengthList<dim) {
          list[*lengthList] = temp;
          (*lengthList)++;
          return 1;
      }
      else
          return 0;
}
```

ESERCIZIO 5 - Soluzione
(strutture)

```
int stampaBooking(booking list[], int lengthList) {
     int i=0;

     for (i=0; i<lengthList; i++)
        printf("%s: %d\n", list[i].name, list[i].seat);

     return 0;
}
```

ESERCIZIO 5 - Soluzione
(strutture)

File “main.c”:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gestione.h“

int main(void) {
    booking list[DIM];
    int lengthList = 0;
    booking temp;
    int resultLeggi;
...
```

ESERCIZIO 5 - Soluzione
(strutture)

```
    ...
    do {
        resultLeggi = leggi(&temp);
        if (resultLeggi == 1)) {
                if (assegna2(list, &lengthList, DIM, temp)==0)
                        printf("Posto gia' occupato, pren. non effettuata!\n");
                else
                        printf("Prenotazione efffettuata con successo!\n");
        }
        else {
                if (resultLeggi == -1 || lengthList==DIM)
                        stampaBooking(list, lengthList);
        }
    } while (resultLeggi!=0 && resultLeggi!=-1 && lengthList<DIM);


    return (0);
}
```