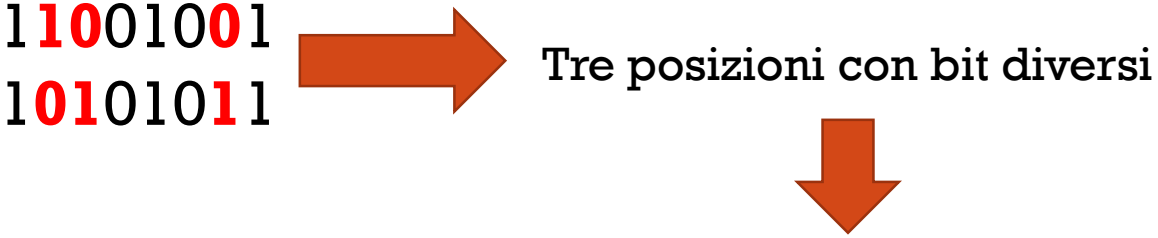
IL LINGUAGGIO C

Esercizio: calcolo della distanza di Hamming

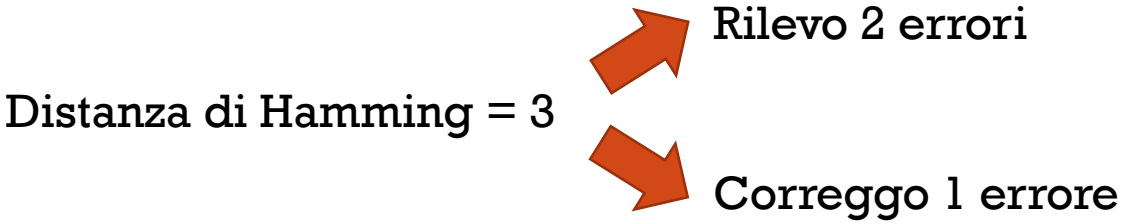
DISTANZA DI HAMMING

- **Definizione**: tra due stringhe (codeword) di ugual lunghezza è il numero di posizioni nelle quali i simboli corrispondenti sono diversi.
- **Misura**:
  - il numero di *sostituzioni* necessarie per convertire una stringa nell'altra;
  - il numero minimo di *errori* che possono aver portato alla trasformazione di una stringa nell'altra.



DISTANZA DI HAMMING - PROPRIETÀ

- **Definizione**: dato un insieme di codeword (codice), la distanza di Hamming del codice è la minima distanza (di Hamming) tra le sue codeword.
- **Proprietà**: dato un codice con distanza di Hamming pari a "d" e indicato con "k" il numero degli errori si ha che:
  - per poter rilevare "k" errori si deve avere $d = k + 1$;
  - per poter correggere "k" errori si deve avere $d = 2k + 1$

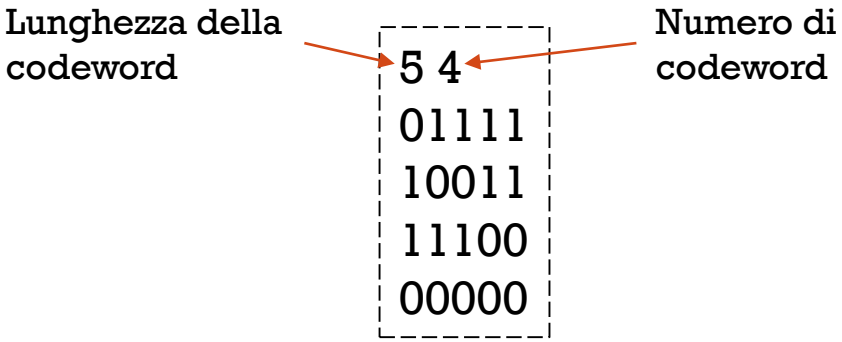


CALCOLO DELLA DISTANZA DI HAMMING

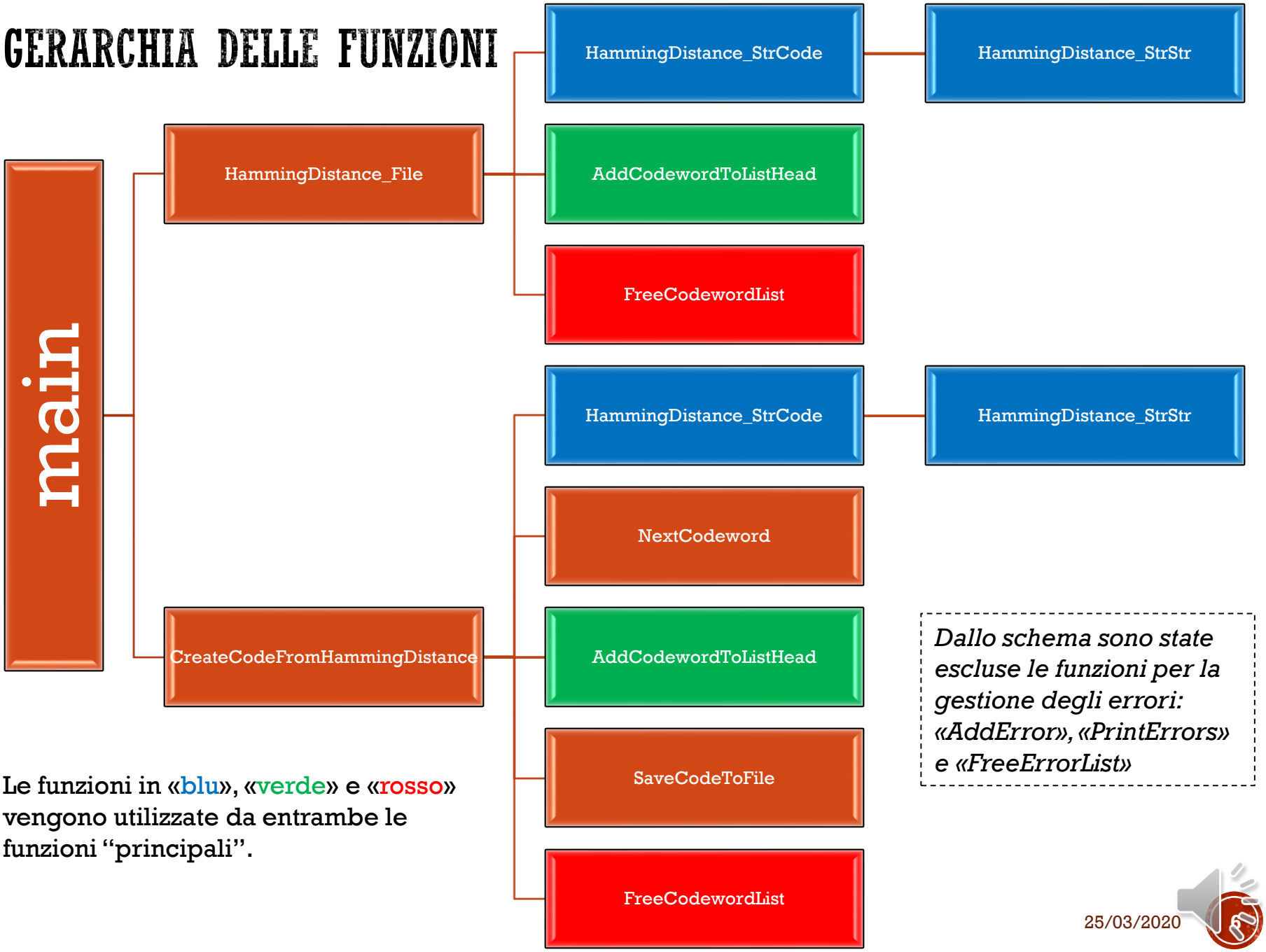
- Si sviluppi un programma, scritto in C, che:
  1. calcoli e visualizzi la distanza di Hamming di un codice binario memorizzato su un file di testo;
  2. generi e salvi su un file di testo un codice composto da tutte le possibili codeword binarie che rispettino i seguenti vincoli forniti dall'utente:
     1. codeword lunghe N bit (max 32 bit);
     2. distanza di Hamming del codice pari a D.

CALCOLO DELLA DISTANZA DI HAMMING

- Informazioni sul file contenente il codice:
  - file testuale in cui ogni codeword è memorizzata come una stringa di *"0"* e *"1"*.
  - la prima riga del file, intestazione, è composta da due numeri interi positivi separati da uno spazio che rappresentano rispettivamente:
    - la lunghezza delle codeword (numero di cifre binarie);
    - il numero di codeword presenti nel file.
  - le righe successive alla prima contengono una codeword ciascuna.



GERARCHIA DELLE FUNZIONI



Le funzioni in «blu», «verde» e «rosso» vengono utilizzate da entrambe le funzioni "principali".

Dallo schema sono state escluse le funzioni per la gestione degli errori: «AddError», «PrintErrors» e «FreeErrorList»

FUNZIONE: «main»

- Si occupa dell'interazione con l'utente:
  - funzione da eseguire;
  - richiesta dei nomi dei file e dei parametri necessari all'elaborazione;
  - visualizzazione dei risultati e degli, eventuali, errori.
- Crea un menù con tre voci e consente la scelta all'utente:
  1. *«Calcola la distanza di Hamming del file»*
  2. *«Genera un file con distanza di Hamming definita»*
  3. *«Esci»*
- Gestisce il menù con un ciclo post-condizionale *"do-while"* e uno *"switch"*.

FUNZIONE: «HammingDistance_File»

- **Definizione**: *int HammingDistance_File(FILE* fp).*
- **Obiettivo**: calcola la distanza di Hamming delle codeword contenute nel file passato come parametro.
- **Strutture utilizzate**: lista in cui vengono progressivamente inserite le codeword presenti nel file.

```
typedef struct codeword_list_element
{
        char* codeword;
        struct codeword_list_element* next;
} codewordListElement;
```


FUNZIONE: «HammingDistance_File»

- **Passaggi**:
  1. Legge dalla prima riga del file:
     - la lunghezza delle codeword;
     - il numero di codeword presenti.
  2. Esegue un ciclo in cui legge tutte le codeword presenti e per ognuna di esse:
     i. chiama la funzione "*HammingDistance_StrCode*" e ottiene la distanza di Hamming tra la codeword letta e tutte quelle già inserite nella lista;
     ii. verifica se la distanza ottenuta è quella minima rilevata fino ad ora e, se necessario, aggiorna il minimo;
     iii. chiama la funzione "*AddCodewordToListHead*" che aggiunge la codeword in testa alla lista delle codeword.
  3. All'uscita del ciclo libera la memoria allocata e ritorna la distanza ottenuta.

FUNZIONE: «HammingDistance_StrCode»

- **Definizione**: *int HammingDistance_StrCode(char* codeword, codewordListElement* head, int codewordLength).*
- **Obiettivo**: calcola la distanza di Hamming tra la codeword ("*codeword*") passata come primo parametro e quelle inserite nella lista puntata da "*head*".
- **Passaggi**:
  1. Esegue un ciclo in cui scorre tutte le codeword presenti nella lista e per ognuna di esse:
     i. chiama la funzione "*HammingDistance_StrStr*" e ottiene la distanza di Hamming tra la codeword passata come parametro e quella attualmente puntata nella lista;
     ii. verifica se la distanza ottenuta è quella minima rilevata fino ad ora e, se necessario, aggiorna il minimo.
  2. All'uscita del ciclo ritorna la distanza ottenuta.

FUNZIONE: «HammingDistance_StrStr»

- **Definizione**: *int HammingDistance_StrStr(*char* codeword_1, char* codeword_2, int codewordLength*).*
- **Obiettivo**: calcola la distanza di Hamming tra la codeword ("*codeword_1*" e "*codeword_2*") passate come parametro.
- **Passaggi**:
  1. Esegue un ciclo in cui scorre, ordinatamente, tutti i «bit» delle due codeword confrontandoli tra loro. Se i due «bit» sono diversi, aggiorna la distanza di Hamming tra le codeword, incrementandola di uno.
  2. All'uscita del ciclo ritorna la distanza ottenuta.

FUNZIONE: «AddCodewordToListHead»

- **Definizione**: *int AddCodewordToListHead(char* codeword, int codewordLength, codewordListElement** p_head).*
- **Obiettivo**: inserisce la nuova codeword ("*codeword*"), passata come primo parametro, in un nuovo elemento della lista puntata da ("*p_head*"). L'inserimento avviene in testa alla lista.

FUNZIONE: «FreeCodewordList»

- **Definizione**: *int FreeCodewordList(codewordListElement** p_head).*
- **Obiettivo**: libera la memoria allocata per la lista.

FUNZIONE: «CreateCodeFromHammingDistance»

- **Definizione**: *int CreateCodeFromHammingDistance(FILE* fp, int codewordLength, int hammingDistance).*
- **Obiettivo**: genera e salva sul file passato, come primo parametro, un codice composto da codeword lunghe "*codewordLength*" bit che hanno tra loro una distanza di Hamming almeno pari a "*hammingDistance*".
- **Strutture utilizzate**: lista in cui vengono progressivamente inserite le codeword «valide» che comporranno progressivamente il codice.

```
typedef struct codeword_list_element
{
        char* codeword;
        struct codeword_list_element* next;
} codewordListElement;
```

FUNZIONE: «CreateCodeFromHammingDistance»

- **Passaggi**:
  1. Esegue un ciclo in cui, grazie alla funzione "*NextCodeword*", genera tutte le possibili codeword della lunghezza richiesta e per ognuna di esse:
     i. chiama la funzione "*HammingDistance_StrCode*" e ottiene la distanza di Hamming tra la codeword generata e tutte quelle già inserite nella lista delle codeword «valide» appartenenti al codice;
     ii. verifica se la distanza ottenuta è maggiore o uguale alla distanza di Hamming richiesta e, in caso affermativo, chiama la funzione "*AddCodewordToListHead*" che aggiunge la codeword generata in testa alla lista delle codeword «valide».
  2. All'uscita del ciclo:
     i. chiama la funzione "*SaveCodeToFile*" per salvare sul file il codice creato.
     ii. libera la memoria allocata e ritorna il numero di codeword «valide».

FUNZIONE: «NextCodeword»

- **Definizione**: *int NextCodeword(char* codeword, int codewordLength).*
- **Obiettivo**: genera tutte le possibili codeword di "*codewordLength*" «bit».
- **Utilizzo**:
  - ad ogni chiamata genera una nuova codeword, diversa da tutte le precedenti, e la copia in "*codeword*";
  - la funzione deve essere inizializzata prima di essere utilizzata passando come primo parametro il valore "*NULL*".
- **Valori di ritorno**:
  - -1: errore;
  - 0: nessuna ulteriore codeword generabile;
  - 1: nuova codeword generata;
  - 2: inizializzazione riuscita.

FUNZIONE: «NextCodeword»

- **Passaggi**:
  1. Verifica se il primo parametro "*codeword*" è "*NULL*":
     - se «VERO»: alloca e inizializza un vettore, puntato da un puntatore dichiarato «*static*», di lunghezza opportuna che conterrà l'ultima codeword generata. Gli elementi del vettore vengono inizializzati a '0'.
     - se «FALSO»:
       a. verifica che non sia già stata generata l'ultima codeword possibile e, in questo caso, lo segnala ("*return 0*"), altrimenti:
          i. copia l'ultima codeword generata nel vettore passato come primo parametro "*codeword*";
          ii. genera la nuova codeword sommando uno (somma binaria) al valore dell'ultima codeword generata;
          iii. verifica se si sia generato un overflow e, in questo caso, setta un flag interno ("*noMoreCodeword*") che è indica che è stata generata l'ultima codeword possibile.

FUNZIONE: «SaveCodewordToFile»

- **Definizione**: *int SaveCodeToFile(FILE* fp, codewordListElement* head, int codewordLength, int codewordCount).*
- **Obiettivo**: salva la lista delle codeword sul file passato come primo parametro ed aggiunge, nella prima riga del file, un'intestazione contenente due valori, separati da uno spazio, che rappresentano rispettivamente:
  - la lunghezza delle codeword;
  - il numero di codeword presenti.