5 Typedef e struct

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione della definizione di strutture e di tipi, oltre ad un riepilogo sulla codifica dei numeri con complemento a due.

I tipi di dato strutturato sono dichiarabili in C tramite la parola chiave `struct`

```
#define LEN n_elementi

void main(){

struct {
        tipo_campo1 nome_campo1;
        tipo_campo2 nome_campo2;
        ...
} nome_struttura;

istruzioni;
}
```

A questo punto potremo utilizzare `nome_struttura` come se fosse una variabile. Per accedere ai campi della struttura si utilizza la sintassi `nome_struttura.nome_campo`. Grazie ad esso possiamo accedere (leggere o scrivere) nei campi della struttura. L'assegnamento tra strutture è possibile solo se le strutture sono dello stesso tipo (stessi campi, per nome e tipo, con stesso ordine di dichiarazione). Non è possibile fare confronti tra strutture in maniera automatica.

Oltre ad i tipi nativi del C (ad esempio `int`, `char` o `float`) è possibile dichiarare dei tipi definiti dall'utente tramite la parola chiave `typedef`. Ad esempio, volendo dichiarare dei vettori della stessa dimensione `n_elementi`:

```
#define LEN n_elementi

void main(){

typedef tipo_variabile nome_variabile[LEN];

istruzioni;
}
```

È buona norma dichiarare nuovi tipi con nomi che inizino con una **lettera maiuscola**, con un prefisso o con un suffisso scelto.

La tecnica della dichiarazione di tipo si può combinare con la dichiarazione di strutture. Ad esempio:

```
#define LEN n_elementi

void main(){

typedef tipo_variabile Nome_tipo[LEN][LEN];

typedef struct {
        tipo_campo1 nome_campo1;
        tipo_campo2 nome_campo2;
        ...
} Nome_tipo_struttura;

istruzioni;
}
```

La definizione di un tipo non implica l'istanziazione di una variabile di quel tipo. La dichiarazione della variabile avverrà di seguito.

Enum Gli enumerated types contengono un elenco di costanti che possono essere indirizzate con valori integer. Per dichiarare tali tipi si utilizza `enum`; vengono dichiarati i tipi e le variabili come nell'esempio che segue:

```
        enum colori {rosso, giallo, verde, blu} pennarello;
        enum giorni {lun,mar,mer,gio,ven,sab,dom} settimana;
        enum colori pulsante, nastro;
```

In tale esempio viene dichiarato `colori` come enumerated type e la variabile `pennarello` con 4 valori accettabili definiti, mentre la variabile `settimana` di tipo`giorni` ha 7 valori accettabili definiti. Le variabili `pulsante` e `nastro` sono di tipo `colori`. Ogni item nell'elenco di valori accettabili è detto enumeration constant. Il C mappa ogni enumeration constant ad un'intero, per cui è ad esempio possibile scrivere:

```
        settimana = verde;
```

che come risultato fa si che settimana abbia valore 2, perchè di default a ogni membro dell'elenco di variabili è assegnato un valore incrementale partendo da 0 per il primo valore. È comunque possibile definire valori diversi agli elementi:

```
        enum colori {rosso=10, giallo=30, verde, blu=giallo};
```

Un ulteriore esempio relativo all'assegnazione di valori diversi è il seguente:

```
enum escapes {bell='\a', backspace='\b', tab='\t', newline='\
    n', vtab='\v', return='\r'};
```

È anche possibile iniziare ad assegnare i valori da un valore iniziale diverso da 0:

```
enum months (jan=1,feb,mar,...dec);
```

dove è implicito che febbraio sia corrispondente al valore 2, marzo al 3 e così via.

Riepilogo: CP2 Non cambia la codifica dei numeri positivi. La cifra più significativa ha significato -2^{m-1} se il numero è rappresentato da m bit.

Decimale -> CP2

- Controllo se il numero è rappresentabile con m bit
- Se è positivo, converto il numero in binario con l'algoritmo delle divisioni successive
- Se è negativo, considero il suo opposto, lo codifico in binario, inverto ogni bit, sommo 1

CP2 -> Decimale

- Se è positivo, converto il numero da binario a decimale (somma di esponenti di 2)
- Se è negativo, inverto ogni bit, aggiungo 1, converto in decimale, cambio di segno

La somma di numeri espressi in complemento a due si esegue normalmente, si ignora il carry (cifra significativa che esce dagli m bit). Abbiamo overflow se c'è inconsistenza nell'operazione (per esempio se la somma di negativi dà un numero positivo).

5.1 Esercizi

Esercizio 5.1

Un campionato di calcio è composto da venti squadre. Ciascuna squadra ha un nome (lungo al piú 30 caratteri), un numero di punti accumulati durante la stagione e al più 23 calciatori. Ciascun calciatore ha un cognome (lungo al piú 20 caratteri), un'etá, un numero di maglia, un numero di presenze in stagione ed un numero di gol realizzati in stagione.

1. Specificare i tipi Campionato, Squadra e Calciatore.
2. Viene data una variabile di nome serie_a di tipo Campionato, giá inizializzata e riempita dall'utente con tutte le squadre del campionato. Scrivere un frammento di codice che stampi il cognome, il numero di gol realizzati e la squadra di appartenenza del calciatore che ha realizzato piú gol nel campionato.

Esercizio 5.2

Scrivere un programma che permetta all'utente di gestire un libretto di voti di esami, per un massimo di 20 esami. Ogni esame sostenuto ha i seguenti attributi:

- nome del corso (stringa di massimo 256 caratteri)
- voto (minimo 18, massimo 30)
- data (in formato gg/mm/aaaa)
- codice del corso (codice di massimo 6 cifre)

Il programma permette all'utente di svolgere le seguenti operazioni:

inserimento: inserisce un esame in fondo alla lista degli esami eventualmente presenti.

stampa: stampa tutti gli esami presenti, con i dettagli di cui sopra.

ricerca: chiede all'utente di inserire un codice e cerca nel libretto la presenza di un esame corrispondente a quel codice. Se presente, stampa tale esame.

uscita: esce dal programma.

tramite un menù di scelta di questo tipo:

```
[i] inserimento nuovo esame
[s] stampa del libretto
[r] ricerca per codice
[x] uscita
```

Il programma deve continuare a presentare tale menù, fino a quando l'utente non preme il tasto per uscire.

Esercizio 5.3

(TdE November 2007) Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda,

..., quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (edificio diviso in 20 piani ognuno con 40 uffici).

```
/* definizioni dei tipi */
typedef struct {
  char nome[20], cognome[20];
  int cat; // contiene valori tra 1 e 5
  int stipendio;
} Impiegato;

typedef enum{
  nord, nordEst, est, sudEst, sud, sudOvest, ovest, nordOvest
} Esposizione;

typedef struct {
  int superficie; /* in m^2 */
  Esposizione esp;
  Impiegato occupante;
} Ufficio;

/* definizioni delle variabili */
Ufficio torre[20][40];

/* rappresenta un edificio di 20 piani con 40 uffici per piano */
```

1. Si scriva un frammento di codice (che includa eventualmente anche le dichiarazioni di ulteriori variabili) che stampi il cognome, lo stipendio e la categoria di tutte e sole le persone che occupano un ufficio orientato a sud oppure a sudEst e avente una superficie compresa tra 20 e 30 metri quadri;
2. Visualizzi a schermo i piani che non hanno neanche un ufficio esposto a nord;
3. Stampi lo stipendio medio degli impiegati in questi piani che si chiamano "Giacomo" di nome.

Esercizio 5.4

1. Si definiscano le seguenti variabili che specificano le strutture dati per rappresentare i messaggi scambiati attraverso Facebook e ai profili degli utenti:
   - una struttura che definisca un'utente, con nome, cognome e e-mail, tutti composti da caratteri alfanumerici;
   - una struttura che definisca un messaggio, con un contenuto alfanumerico, un mittente scelto tra gli utenti, un numero fissato di destinatari e i destinatari, anch'essi scelti tra gli utenti (fino ad un massimo di 256).

2. Definire, usando il linguaggio C, un'appropriata variabile per memorizzare le informazioni relative a 10 utenti e 10 messaggi. Queste variabili dovranno essere chiamate `utenti_facebook` e `messaggi_mandati`.

3. Scrivere in linguaggio C, aggiungendo eventualmente opportune dichiarazioni di variabili, un frammento di codice che permetta di visualizzare a video l'indirizzo email di tutti gli utenti che hanno spedito almeno un messaggio e che non siano tra i destinatari di tale messaggio. Si presupponga che la variabile `messaggi_inviati` sia inizializzata correttamente con le informazioni relative a 10 messaggi. Si presupponga inoltre che l'indirizzo email sia identificatore univoco di un utente: pertanto si può utilizzare l'indirizzo email per verificare se due utenti sono lo stesso utente.

Esercizio 5.5

(a) Si dica qual è l'intervallo di valori interi rappresentabile con la codifica in complemento a due a 9 bit.

(b) Con riferimento a tale codifica indicare, giustificando brevemente le risposte, quali delle seguenti operazioni possono essere effettuate correttamente:

- $-254 - 255$
- $+254 - 253$
- $-18 + 236$
- $+217 + 182$

(c) Mostrare in dettaglio come avviene il calcolo delle operazioni (i) e (ii), evidenziando il bit di riporto e il bit di overflow così ottenuti.

Esercizio 5.6

1. Si dica quale dei seguenti numeri è il maggiore, motivando con calcoli e ragionamenti la risposta:

   A. 34 (numero in base 10);

   B. 110110 (numero naturale in binario);

   C. 57 (numero in base 10);

   D. (numero in Complemento a 2);

   E. (numero in Complemento a 2).

2. Si indichi quindi il numero n di bit che permette di codificare tutti i numeri A - E in Complemento a 2 e li si codifichi tutti in Complemento a 2 usando n bit, riportando i passaggi fondamentali dei calcoli.

3. Si eseguano (riportando i calcoli) le seguenti operazioni utilizzando n bit e si indichino eventuali bit di carry (riporto) e overflow. Si segnalino i casi in cui l'operazione non puó essere eseguita utilizzando gli n bit identificati nel punto 2:

   I. A - B;

   II. (C + D) + E;

   III. 2A + E.

Soluzioni

Soluzione dell'esercizio 5.1

1.
```c
# define MAX_SQUADRE 20
# define MAX_COGNOME 20
# define MAX_NOME 30
# define MAX_CALCIATORI 23

typedef struct{
    char cognome[MAX_COGNOME + 1];
    int eta;
    int maglia;
    int presenze;
    int gol;
}Calciatore;

typedef struct{
    char nome[MAX_NOME + 1];
    int punti;
    Calciatore calciatore[MAX_CALCIATORI];
}Squadra;

typedef Squadra Campionato[MAX_SQUADRE];
```

2.
```c
Campionato serie_a;
Calciatore capocannoniere;
Squadra squadra_capocannoniere;
int i, j;

// inizializzazione capocannoniere
capocannoniere = serie_a[0].calciatore[0];
squadra_capocannoniere = serie_a[0];

// ricerca capocannoniere
for(i = 0; i < MAX_SQUADRE; i++)
    for(j = 0; j < MAX_CALCIATORI; j++)
        if(serie_a[i].calciatore[j].gol > capocannoniere.gol){
            capocannoniere = serie_a[i].calciatore[j];
            squadra_capocannoniere = serie_a[i];
        }

// stampa capocannoniere
printf("\ncapocannoniere: %s\n", capocannoniere.cognome);
printf("squadra: %s\n", squadra_capocannoniere.nome);
printf("numero di gol: %d\n\n", capocannoniere.gol);
```

Soluzione dell'esercizio 5.2

```c
#include <stdio.h>

#define DIM_LIBR 20
#define DIM_NOME 256
#define DIM_DATA 10
#define DIM_CODI 6
```

```c
typedef struct{
    char nome_corso[DIM_NOME+1];
    int voto;
    char data[DIM_DATA+1];
    char codice[DIM_CODI+1];
} Esame;

void main(){

    //dichiarazioni
    char menu;
    int n_esami = 0;
    int i,j;
    char codice[DIM_CODI];
    int is_equal;
    char cestino;
    //dichiarazione con tipo user-defined
    Esame libretto[DIM_LIBR];

    do {
        system("cls"); //WINDOWS
        //system("clear"); //UNIX
        printf("Scegliere una delle seguenti opzioni:\n");
        printf("[i] inserimento nuovo esame\n");
        printf("[s] stampa del libretto\n");
        printf("[r] ricerca per codice\n");
        printf("[x] uscita\n");
        scanf("%c", &menu);
        scanf("%c", &cestino);

        switch (menu) {
            case 'i':
                //Istruzioni
                printf("Inserisci nome esame: ");
                scanf("%s",libretto[n_esami].nome_corso);
                printf("Inserisci voto: ");
                scanf("%d",&libretto[n_esami].voto);
                printf("Inserisci data: ");
                scanf("%s",libretto[n_esami].data);
                printf("Inserisci codice: ");
                scanf("%s",libretto[n_esami].codice);
                n_esami++;

                break;
            case 's':
                //Istruzioni
                for (i = 0; i < n_esami; i++)
                   // printf() ....

                break;
            case 'r':
                //Istruzioni
                printf("Inserisci il codice da controllare: ");
                scanf("%s",codice);
                for (i = 0; i < n_esami; i++) {
                    //libretto[i].codice == codice;
                    is_equal = 1;
                    for (j = 0; j < DIM_CODI; j++)
                        if (libretto[i].codice[j] != codice[j])
                            is_equal = 0;

                    if (is_equal){
```

```
                        //Stampare esame
                    }
                }

                break;

            default:

                break;
        }
    } while (menu != 'x');

}
```

Soluzione dell'esercizio 5.3

```
int p, u; /* indice di piano nelledificio e di ufficio nel piano */

for (p = 0; p < 20; p++)
    for (u = 0; u < 40; u++)
        if ((torre[p][u].esp == sudEst || torre[p][u].esp == sud) &&
            (torre[p][u].superficie >=20 && torre[p][u].superficie<=30)) {
            printf("\n il Signor %s è impiegato di categoria %d",
                   torre[p][u].occupante.cognome,
                   torre[p][u].occupante.cat);
            printf("e ha uno stipendio pari a %d euro \n",
              torre[p][u].occupante.stipendio);
        }
```

```
int uffNord; /* uffNord fa da flag*/

for (p = 0; p < 20; p++) {
    uffNord = 0; //per ogni piano assumo 0
    for (u = 0; u < 40 && !uffNord; u++)
        if(torre[p][u].esposizione == nord)
            uffNord = 1;

    /* se qui vale ancora 0 vuol dire che non ci sono uffici a nord*/
    if (!uffNord)
        printf("il piano %d  non ha edifici esposti a nord", p);
 }

/* è corretto anche senza !uffNord nel for(), anche se non è efficiente. */
```

```
for(i = 0; i < 20; i++) { //scorre i piani
    noUfficiNord = 1; //versione con flag inversa

    for(j = 0; j < 40; j++) //senza flag arriva fino a 39
        if(torre[i][j].esposizione == nord)
            noUfficiNord = 0;

    if(noUfficiNord) {
        printf("il piano %d non ha uffici a nord", i);
        stipendioMedio = 0;
        cnt = 0;
        for(j = 0; j < 40; j++)
            if(strcmp(torre[i][j].occupante.nome, "Giacomo") == 0){
                stipendioMedio += torre[i][j].occupante.stipendio;
                cnt++;
```

```
            }
        printf("Lo stipendio medio dei Giacomo nel "
               "piano è %f ", stipendioMedio / cnt);
    }
 }
```

Soluzione dell'esercizio 5.4

```
typedef struct {
  char nome[24];
  char cognome[24];
  char email[64];
} Utente;

typedef struct {
  char contenuto[256];
  Utente mittente;
  int numDestinatari;
  Utente destinatari[256];
} Messaggio;
```

```
Utente utenti_facebook[10];
Messaggio messaggi_inviati[10];
```

Gli utenti che hanno spedito almeno un messaggio sono necessariamente presenti come mittenti nella variabile `messaggi_inviati`:

```
void main () {
  int i, j;
  Utente utenti_facebook[10];
  Messaggio messaggi_inviati[10];

  for (i = 0; i < 10; i++) {
    j = 0;

    //ricerca destinatario.email != mittente.email con strcmp()
    while (j < Messaggi[i].numDestinatari &&
              strcmp(Messaggi[i].mittente.email,
                     Messaggi[i].destinatari[j].email) != 0) {
      j++;
    }

    if (j == Messaggi[i].numDestinatari) //non trovato
      printf("%s", Messaggi[i].mittente.email);
}
```

Soluzione dell'esercizio 5.5

(a) I valori rappresentabili vanno da $-2^{m-1} = -256$ a $2^{m-1} - 1 = +255$ compresi.

(b) Le soluzioni delle operazioni sono:

- $-254-255$ NO si ottiene un valore negativo troppo grande in valore assoluto

- $+254 - 253$ SI si ottiene un valore piccolo in valore assoluto
- $-18 + 236$ SI si ottiene un valore positivo, grande in valore assoluto ma nei limiti
- $+217 + 182$ NO si ottiene un valore positivo troppo grande in valore assoluto

(c)

- In complemento a 2 a $m = 9$ bit, $(-254)_{10} = (100000010)_{CP2}$ (perchè $(254)_{10} = (011111110)_2$). Da questo possiamo ricavare che $(-255)_{10} = (100000001)_{CP2}$ essendo $-255 = -254 - 1$. La somma diventa:

		(1)								
		1	0	0	0	0	0	0	1	0	+
		1	0	0	0	0	0	0	0	1	=
[1]	(1)	0	0	0	0	0	0	0	1	1	

- 254 in complemento a due è $(011111110)_{CP2} = (011111110)_2$ essendo positivo. Invece $-253 = -254 + 1 = (100000011)_{CP2}$, quindi:

		(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)		
		0	1	1	1	1	1	1	1	0	+
		1	0	0	0	0	0	0	1	1	=
[0]	(1)	0	0	0	0	0	0	0	0	1	

ignorando il carry abbiamo il risultato esatto $(000000001)_{CP2} = (1)_{10}$.

Soluzione dell'esercizio 5.6

1. Tra i numeri rappresentati in base 10 possiamo immediatamente decidere che il numero maggiore è 57. Il numero 1111111 è chiaramente negativo avendo il bit piú significativo pari a 1, quindi è certamente minore di 57. Possiamo convertire in base 10 anche gli altri due numeri per prendere la decisione finale:
   - 110110: 32+16+4+2 = 54;
   - 0011011: 16+8+2+1 = 27;

   si noti che 0011011 è in complemento a 2 (CP2) ma è un numero positivo (il bit piú significativo è pari a 0), di conseguenza, è possibile applicare la regola di

conversione che vale anche per i numeri naturali. Possiamo quindi concludere che il numero maggiore è 57.

2. Il numero maggiore, 57, necessita di 7 bit per la conversione, infatti n deve essere tale che $57 \leq 2^{n-1}$. Per n = 7 si ha che 2^{n-1} è pari a 64. Se avessimo scelto n pari a 6, avremmo avuto che $2^{n-1} = 32$, che è chiaramente minore di 57. Anche l'unico numero negativo è espresso su 7 cifre binarie, quindi 7 è il numero di bit adeguato a convertire tutti i numeri considerati. Convertendo tutti i numeri in CP2 a 7 bit si ha:

   A. 34 in base 10 $\Rightarrow$ 0100010 in CP2 su 7 bit;

   B. 110110 in binario $\Rightarrow$ 0110110 in CP2 su 7 bit;

   C. 57 in base 10 $\Rightarrow$ 0111001 in CP2 su 7 bit;

   D. 1111111 in CP2 su 7 bit;

   E. 0011011 in CP2 su 7 bit.

3. I. Prima calcolo $-B$ = 10001010 ed ho:

      0 1 0 0 0 1 0 + 1 0 0 1 01 0 = [0] 1 1 0 1 1 0 0;

   II. Eseguo le due somme:

      0 1 1 1 0 0 1 + 1 1 1 1 1 1 1 = 0 1 1 1 0 0 0 + 0 0 1 1 0 1 1 = [1] 1 0 1 0 0 1 1;

      In questo caso abbiamo overflow.

   III. Questa operazione non è fattibile in quanto 2A = 68 che darebbe giá overflow.