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.