6 Typedef, matrici e codifica

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

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

```
#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;
}
```

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

Le matrici in C vengono definite come vettori di vettori. Ad esempio per dichiarare una matrice bidimensionale dovremo scrivere:

```
#define DIM1 n_righe
#define DIM2 n_colonne

void main(){

tipo_variabile nome_variabile[DIM1][DIM2];

istruzioni;
}
```

dove `n_righe` e `n_colonne` sono degli interi che specificano il numero di righe e colonne della matrice. Nel caso di dimensioni superiori a 2, dovremo mettere tante parentesi quadre e specificare le grandezze massime per ogni dimensione.

La tecnica della dichiarazione di strutture si può combinare con la dichiarazione di matrici. 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.

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).

Riepilogo: IEEE 754-1985 standard Decimale -> IEEE standard

- Definisco il bit di segno $S = 0$ per positivo e $S = 1$ per negativo
- Codifico in virgola fissa in base 2, parte frazionaria e parte intera
- Porto il numero binario in forma normalizzata in base 2
- Definisco a 23 bit come la mantissa M senza il primo bit (sempre 1)
- Calcolo l'esponente E (aggiungo 127 a quello che ho trovato durante la normalizzazione)
- Compongo il numero $S/E/M$

IEEE standard -> Decimale

- Decompongo il numero $S/E/M$
- Calcolo l'esponente E e sottraggo 127
- Sposto la virgola in base all'esponente alla mantissa M
- Converto in decimale la parte intera
- Converto in decimale la parte frazionaria
- Definisco il bit di segno $S = 0$ per positivo e $S = 1$ per negativo

6.1 Esercizi

Esercizio 6.1

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 6.2

Scrivere un programma che chieda all'utente la dimensione effettiva di una matrice quadrata, che dovrà poi essere acquisita dal programma.

Successivamente, il programma dovrà controllare se la matrice è diagonale (i.e., nessuno zero sulla diagonale principale e solo zeri fuori da essa).

Estendere il programma per controllare anche se la matrice inserita è simmetrica.

Esercizio 6.3

(TdE Febbraio 2012, variante) Data m una matrice di dimensione $n \times n$ (costante simbolica) di numeri interi nell'intervallo $[0, 9]$:

1. si scriva un frammento di programma in linguaggio C (con le relative dichiarazioni di variabili necessarie al corretto funzionamento), che trovi il numero più frequente (si ipotizzi che tale numero sia unico). Il programma deve stampare a schermo tutti i numeri presenti nella matrice con valore minore del valore più frequente e successivamente tutti quelli con valore maggiore di quello più frequente;
2. Aggiungere al programma di cui sopra un frammento di codice che legga tutti e soli i valori memorizzati nel secondo dei due vettori (che potrebbero essere meno della lunghezza massima del vettore) e stampi a video se sono o meno monotoni crescenti, ovvero se tutti gli elementi adiacenti sono ordinati $a_i \leq a_{i+1}$.

Esercizio 6.4

(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 6.5

(TdE November 2012)

1. Si determini la codifica binaria dei numeri -12.625 e 16.65 secondo lo Standard IEEE 754-1985 a precisione singola, riportando i calcoli effettuati.

2. L'uso della rappresentazione IEEE 754-1985 a precisione singola ha comportato delle approssimazioni?

3. L'uso della rappresentazione IEEE 754-1985 a precisione doppia avrebbe portato ad una rappresentazione più precisa dei due numeri considerati?

Si ricorda che lo standard IEEE 754-1985 a precisione singola ha le seguenti caratteristiche: 1 bit per il segno, 23 bit per la mantissa, 8 per l'esponente (K=127) Si ricorda che lo standard IEEE 754-1985 a precisione doppia invece ha le seguenti caratteristiche: 1 bit per il segno, 52 bit per la mantissa, 11 per l'esponente (K=1023)

Esercizio 6.6

Scrivere un programma che converta un numero intero positivo da decimale a binario avendo a disposizione 10 bit, se possibile.

Soluzioni

Soluzione dell'esercizio 6.1

```
#include <stdio.h>

#define DIM_LIBR 20
#define DIM_NOME 256
//#define MIN_VOTO 18
//#define MAX_VOTO 30
#define DIM_DATA 10
#define DIM_CODI 6

void main(){

char menu;
int n_esami = 0;
int i,j;

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

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);
    fflush(stdin);

    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 6.2

```
#include <stdio.h>

#define MAX_DIM 20


void main(){

typedef enum{falso, vero} Booleano;

int matrice[MAX_DIM][MAX_DIM];
int dim_matr;
int i,j;

Booleano is_diagonale, is_simmetrica;

do {
    printf("Inserire la dimensione della matrice quadrata: ");
    scanf("%d",&dim_matr);
} while (dim_matr < 0 || dim_matr > MAX_DIM);

for(i = 0; i < dim_matr; i++)
    for(j = 0; j < dim_matr; j++) {
        printf("Inserire l'elemento in posizione [%d][%d]: ",i+1,j+1);
        scanf("%d",&matrice[i][j]);
    }

is_diagonale = 1;

for (i = 0; i < dim_matr; i++) {
    if (matrice[i][i] == 0)
        is_diagonale = 0;
    }

if (is_diagonale) {
    for(i = 0; i < dim_matr; i++)
```

```
        for(j = 0; j < dim_matr; j++) {
            if (i != j && matrice[i][j] != 0)
                is_diagonale = 0;
        }
}
is_simmetrica = 1;
for(i = 0; i < dim_matr; i++) {
    for(j = i+1; j < dim_matr; j++) {
        if (matrice[i][j] != matrice[j][i])
            is_simmetrica = 0;
    }
}

is_diagonale = -2;

if (is_diagonale)
    printf("La matrice e' diagonale e simetrica\n");
else {
    if (is_simmetrica)
        printf("La matrice non e' diagonale, ma simmetrica\n");
    else
        printf("La matrice non e' diagonale, ne' simmetrica\n");
}


}
```

Soluzione dell'esercizio 6.3

```
#define N 5       // dimensione matrice
#define RANGE 10  // da 0 a 9

#include <stdio.h>

int main () {
    int m[N][N];
    int v1[N*N], v2[N*N];
    int freq[RANGE];
    int fmax = 0;
    int valmax;

    int u, z;

    int i, j;

    //inizializzo matrice con valori crescenti (non necessario
    // se non per evitare di inserire a mano i valori)
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            m[i][j] = i * j;
            printf("%2d ", m[i][j]);
        }
        printf("\n");
    }

    //inizializzo vettore frequenze
    for (i = 0; i < RANGE; i++)
        freq[i] = 0;

    for (i = 0; i < N; i++) {
```

```
        for (j = 0; j < N; j++) {
            freq[m[i][j]]++;
        }
    }

    //ricerco valore piu frequente
    for (i = 0; i < RANGE; i++)
        if (i == 0 || freq[i] > fmax) {
            valmax = i;    //valore
            fmax = freq[i]; //frequenza
        }

    //spostamento valori
    u = z = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (m[i][j] < valmax) {
                v1[u] = m[i][j];
                u++;
            }

            if (m[i][j] > valmax) {
                v2[z] = m[i][j];
                z++;
            }
        }
    }

    //stampa valori
    printf("fmax = %d, valmax = %d\n", fmax, valmax);
    for (i = 0; i < u; i++)
        printf("%d ", v1[i]);

    printf(" (%d elementi)\n", u);

    for (i = 0; i < z; i++)
        printf("%d ", v2[i]);

    printf(" (%d elementi)\n", z);

    //controllo se v2 e' monotono
    i = 0;
    while (i < z-1 && v2[i] <= v2[i+1])
        i++;

    //uscita prematura dal ciclo?
    if (i < z-1)
        printf("Vettore NON monotono crescente.");
    else
        printf("Vettore monotono crescente.");
}
```

Soluzione dell'esercizio 6.4

(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 6.5

Per quanto riguarda -12.625

- Definisco il bit di segno: $S = 1$
- Codifico in virgola fissa in base 2, parte frazionaria $(0.625)_{10} = (0.101)_2$ e parte intera $(12)_{10} = (1100)_2$ ovvero 1100.101
- Porto il numero binario in forma normalizzata: 1.100101×2^3
- Definisco a 23 bit come la mantissa $M = 10010100000000000000000$
- Calcolo l'esponente $E = 127 + 3 = (130)_{10} = (010000010)_2$
- Compongo il numero 1 010000010 10010100000000000000000

Il numero è già rappresentato in maniera esatta in precisione singola. La precisione doppia non cambierà il numero (non aumenterà la precisione della rappresentazione).

Per quanto riguarda 16.65

- Definisco il bit di segno: $S = 0$
- Codifico in virgola fissa in base 2, parte frazionaria $(0.65)_{10} = (0.10\overline{1001})_2$ e parte intera $(16)_{10} = (10000)_2$ ovvero $10000.10\overline{1001}$
- Porto il numero binario in forma normalizzata: $1.00001010\overline{1001} \times 2^5$
- Definisco a 23 bit come la mantissa $M = 00001010011001100110011$
- Calcolo l'esponente $E = 127 + 5 = (132)_{10} = (010000100)_2$
- Compongo il numero $0\ 010000100\ 00001010011001100110011$

Il numero non è rappresentato in maniera esatta in precisione singola, ed essendo periodico non lo può essere neanche in precisione doppia. La precisione doppia avrà una rappresentazione più precisa del numero.

Soluzione dell'esercizio 6.6

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

#define MAX_DIGIT 10

void main() {

int n_10, temp, i, nDigit, maxVal;
int n_2[MAX_DIGIT];

maxVal = pow(2 , MAX_DIGIT) - 1;
do
{
    printf("Inserire il nr base 10: ");
    scanf("%d" , &n_10);
}while(n_10 > maxVal || n_10 <0);

temp = n_10;
nDigit = 0;

while(temp > 0)
{
    n_2[nDigit] = temp % 2;
    temp = temp / 2;
    nDigit++;
}

printf("%d in base 2 diventa: " , n_10);
for(i = nDigit - 1 ; i >= 0 ; i--)
    printf("%d", n_2[i]);

}
```