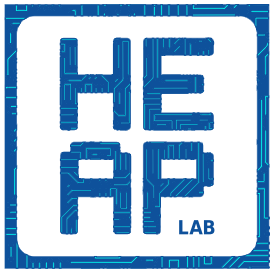
Programmazione in C

Stefano Cherubin*



19/10/2017

[**Informatica A**] Esercitazione #**5**

corso per Ing. Gestionale a.a. 2017/18

*<nome>.<cognome>@polimi.it

Indice

1 Conversione da decimale a binario

Dato un numero positivo Q, scrivere la sua rappresentazione in binario naturale, indicando anche il minimo numero di bit utilizzato.
Esempio:

Input 19_{10}

Output 5 bit, 10011_2

1.1 Approccio alla soluzione

È richiesto che l'output sia dato dalla codifica binaria naturale del numero in base 10 fornito in ingresso. È noto che per ottenere il numero in binario esiste un algoritmo per divisioni successive per 2 che fornisce il risultato in ordine inverso. Questo algoritmo può essere alla base di una soluzione oppure può essere modificato per fornire l'output in modo più conveniente.

Esistono diversi approcci a questo problema: di seguito ne sono riportati 3.

1.2 Soluzione 1 - Countdown to destruction!

Questa soluzione conta quante divisioni per 2 sono necessarie per azzerare Q. Questo numero coincide con il numero di bit che saranno necessari per rappresentare il numero in binario.

Per ogni nuovo bit si procede poi alla distruzione per divisioni successive di una copia del numero in input Q.

Listato 1: Conversione da decimale a binairo: Soluzione

```
1  #include <stdio.h>
2  int main () {
3    int Q, Qaux, current, i, n;
4    do {
5      printf("\nNumero intero : Q = ");
6      scanf("%d", &Q);
7    } while (Q < 0);
8    Qaux = Q;
9    n = 0;
10   do {
11     Qaux = Qaux / 2;
12     n += 1;
13   } while (Qaux != 0);
14   printf("\nCodifica di Q = %d con %d bit = ", Q, n);
15   for (current = n - 1; current >= 0; current--) {
16     Qaux = Q;
17     for (i = 0; i < current; i++)
18       Qaux = Qaux / 2;
19     printf("%d", Qaux % 2);
20   }
21   return 0;
22 }
```

1.3 Soluzione 2 - Raddrizzamento dell'output con algoritmo più semplice

Questa soluzione utilizza l'algoritmo per divisioni successive che fornisce l'output ordinato in ordine inverso e, tramite un accumulatore, raddrizza il risultato.

Listato 2: Conversione da decimale a binairo: Soluzione

```
1  #include <stdio.h>
2  int main () {
3    int dec, Q, n, base;
4    do {
5      printf("\nNumero intero Q non negativo : Q = ");
6      scanf("%d", &Q);
7    } while (Q < 0);
8    dec = 0;  /* accumulatore  */
9    n = 0;    /* posizione bit */
10   /* n a fine ciclo conterrà il # di bit */
11   base = 1; /* accumulatore delle potenze di 10 */
12   while (Q > 0){
13     dec = dec + (Q % 2) * base;
14     Q = Q / 2 ;
15     n = n + 1 ;
16     base = base * 10;
17   }
18   printf("\nNum. dec. emula la sequenza ");
19   printf("binaria di %d bit : %d ",n, dec);
20   return 0;
21 }
```

1.4 Soluzione 3 - Smash by power of 2

L'algoritmo da utilizzare è quello delle divisioni per decrescenti potenze di 2. Si inizia a dividere il numero per la più grande potenza di 2 che non eccede il numero in input. Si procede a dividere i resti della divisione intera per potenze di 2 decrescenti. L'output è dato dal quoziente della divisione così calcolato. L'algoritmo termina quando le potenze decrescenti di 2 arrivano a 2^0.

Il problema principale è identificare qual è la più grande potenza di due minore di un numero. Questa soluzione calcola la più piccola potenza di 2 maggiore di Q. Decrementando di uno è poi possibile calcolare la più grande potenza di 2 contenuta in Q.

Listato 3: Conversione da decimale a binairo: Soluzione

```
1  #include <stdio.h>
2  int main( ) {
3    int i, d, Q, n = 0;
4    do {
5      printf("\nInserisci un numero positivo Q: ");
6      scanf("%d", &Q);
7    } while (Q <= 0);
8    d = 1;
9    do {
10     n = n + 1;
11     d = d * 2;
12   } while (Q > d);
13 /* d è la più piccola potenza di 2 maggiore di Q */
14 /* n ha il valore dell'esponente della potenza di 2
       memorizzata in d, e coincide con il numero di bit
       minimo per rappresentare Q. */
15 /* (n-1) è il valore dell'esponente più significativo
       presente nella rappresentazione del numero Q */
16   printf("\n%d in decimale, con %d bit = ", Q, n);
17   n = n - 1;
18   d = d / 2;
19   do {
20     if (Q >= d) {
21       printf("1");
22       Q = Q - d;
23     } else {
24       printf("0");
25     }
26     n = n - 1;
27     d = d / 2;
28   } while ( n >= 0 );
29   return 0;
30 }
```

2 Intervallo di caratteri

Si scriva un programma in linguaggio C che risolva il problema seguente.

Ricevere dallo standard input due caratteri alfabetici, convertirli in maiuscolo e stampare a video ordinatamente tutti i caratteri dell'alfabeto fra essi compresi, estremi inclusi.

Esempio: dati 'g' e 'M' stampa a video la sequenza GHIJKLM.

2.1 Approccio alla soluzione

2.1.1 Input di due caratteri alfabetici

Occorre verificare che i caratteri inseriti siano caratteri alfabetici. Sfruttando la tabella della codifica ASCII è possibile identificare gli intervalli su cui fare il controllo.

```
1 do {
2   printf("Inserisci due caratteri alfabetici separati
       da spazio: ");
3   scanf("%c %c", &c1, &c2);
4 } while ((c1 < 'A') || (c2 < 'A') || (c1 > 'z') || (c2
     > 'z') || (c1 > 'Z' && c1 < 'a') || (c2 > 'Z' &&
    c2 < 'a'));
```

2.1.2 Lowercase to uppercase

Nota

In ASCII si hanno le seguenti corrispondenze tra i caratteri alfanumerici e le loro codifiche:

$'0' \div' 9' \;\; 48 \div 57$

$'A' \div' Z' \;\; 65 \div 90$

$'a' \div' z' \;\; 97 \div 122$

Per passare da minuscolo a maiuscolo bisogna sottrarre al carattere, la differenza tra i due range.

$$'a' -' A' = 97 - 65 = 32$$

```
if (c1 >= 'a') {
  c1 = c1 - 32;
}
```

Nel caso non ci fosse disponibile la tabella ASCII per calcolare la differenza tra i due intervalli, è possibile farla calcolare al programma.

```
const int diff = 'A' - 'a';
if (c1 < 'a' || c1 > 'z') {
  c1 = c1 + diff;
}
```

2.2 Soluzione C

Listato 4: Intervallo di caratteri

```
1  #include <stdio.h>
2  int main( ) {
3    char x, y; /* dati in ingresso  */
4    char t;    /* ausiliaria per scambiare */
5    int i;     /* contatore  */
6    do {
7      printf("\nInserisci il carattere alfabetico x: ");
8      scanf("%c", &x);
9    } while (!(((x >= 'a') && (x <= 'z')) || ((x >= 'A')
         && (x <= 'Z'))));
10   /* lowercase to uppercase */
11   if ((x >= 'a') && (x <= 'z')) {
12     x = x - ('a'-'A');
13   }
14   do {
15     printf("\nInserisci il carattere alfabetico y: ");
16     scanf("%c", &y);
17   } while (!(((y >= 'a') && (y <= 'z')) || ((y >= 'A')
         && (y <= 'Z'))));
18   /* lowercase to uppercase */
19   if ((y >= 'a') && (y <= 'z'))
20     y = y - ('a'-'A');
21   /* ordina i due caratteri x, y in ordine crescente
        */
22   if (x > y) {
23     t = x;
24     x = y;
25     y = t;
26   }
27   printf("\nLa sequenza di caratteri richiesta e': ");
28   i = (int)x;
29   while (i <= (int)y) {
30     printf("%c", (char)i);
31     /* operatori di casting opzionali */
32     i = i + 1;
33   }
34   return 0;
35 }
```

3 Sequenze pari crescenti

Si codifichi un programma C che legge dallo standard input una sequenza (di lunghezza arbitraria) di interi positivi terminata dal valore 0 e, al termine della sequenza, visualizza su standard output un messaggio che indica quante terne di numeri consecutivi diversi e pari sono contenute nella sequenza.

Esempio:

input 2 50 13 16 8 6 4 6 18 6 6 16 4 1 25 0

output 4

Le sequenze sono 16-8-6 8-6-4 4-6-18 6-16-4

3.1 Approccio alla soluzione

3.1.1 La sentinella

Quando si tratta di prendere in input una sequenza di numeri, è necessario stabilire quanti numeri in input è necessario acquisire.

- È possibile che i numeri siano in un numero prestabilito noto a priori.
- È possibile chiedere all'utente la lunghezza della sequenza.
- È possibile che la sequenza abbia un terminatore che ne delimita la fine (la sentinella).

La sentinella può essere richiesta all'utente prima che l'utente inserisca il primo elemento oppure può essere nota a priori.

3.1.2 Flag

Quando una condizione è molto complessa e viene valutata in diversi punti del programma, può essere utile utilizzare per controlli una variabile ausiliaria. Questa variabile ausiliaria viene impostata come *attiva* o *disattiva* una sola volta, quando si verifica che la condizione sia vera o falsa. Per convenzione questo tipo di variabili ausiliarie vengono chiamate *flag*, bandiere di segnalazione, che si alzano e si abbassano a seconda che la condizione sia vera o falsa.

Di natura le flag sono di tipo booleano (solo vero o falso). In c99 non esiste un tipo che rappresenti una variabile booleana quindi si utilizzano variabili intere. Il valore 0 significa falso mentre un valore $\neq 0$ rappresenta il valore logico vero (di solito si utiliza 1).

3.2 Soluzione C

Listato 5: Conteggio terne positive

```
1  #include <stdio.h>
2  #define SENTINELLA 0
3  int main() {
4    int cont = 0; /* conteggio delle terne valide */
5    int prev2, prev1 = 1, cur = 1;
6    /* Le inizializzazioni di cur e prev1 a 1 sono
        innocue perché impostate con valori positivi
        dispari
7    cur, prev1, prev2: rappresentano rispettivamente il
        valore corrente e i due precedenti */
8    int cond = 0; /* flag */
9    printf("\nSeq. di int separati da spazi ");
10   printf("(termina con %d)\n\n", SENTINELLA);
11   do {
12     prev2 = prev1;
13     prev1 = cur;
14     scanf("%d", &cur);
15     cond = (prev2 > 0 && prev1 > 0 && cur >= 0);
16     cond = cond && !(prev2 == prev1 || prev2 == cur ||
           prev1 == cur);
17     cond = cond && (prev2 % 2 == 0 && prev1 % 2 == 0
          && cur % 2 == 0);
18     if (cond) {
19       cont++;
20       printf("\nNuova terna: ");
21       printf("(%d, %d, %d)", prev2, prev1, cur);
22     }
23   } while ( cur != SENTINELLA );
24   printf("\nLe terne di numeri cercati sono %d", cont)
       ;
25   return 0;
26 }
```

Licenza e crediti

Crediti

Quest'opera contiene elementi tratti da materiale di Gerardo Pelosi redatto per il corso di Fondamenti di Informatica per Ingegneria dell'Automazione a.a. 2013/14.

Licenza beerware[1]

Quest'opera è stata redatta da Stefano Cherubin. Mantenendo questa nota, puoi fare quello che vuoi con quest'opera. Se ci dovessimo incontrare e tu ritenessi che quest'opera lo valga, in cambio puoi offrirmi una birra.

[1] http://people.freebsd.org/~phk/