Tecniche di Programmazione

Esercitazione 2

Tipi di dato primitivi

Esercizio 2.1

Definire una variabile per ogni tipo primitivo (char, short, int, long, float, double) e stampare la dimensione in byte di ciascuna di esse.

Esercizio 2.2

Date le seguenti variabili:

```
unsigned char b;
short s;
int i;
long l;
float f;
double d;
char c;
```

Scrivere un programma che prenda in input le suddette variabili e stampi il risultato delle seguenti espressioni:

1. `b+10l`
2. `(b+i)*l`
3. `(b+i)*l+f`
4. `s/f + sin(f)`
5. `c == 'b'`
6. `l+1.5f`
7. `i<10`
8. `d * 3.14159`

Nota: quando chiediamo di leggere un numero in input, scanf() automaticamente ignora eventuali caratteri bianchi iniziali ' ', '\n', '\t'. In particolare, viene ignorato il newline che abbiamo usato per terminare l'input precedente. Attenzione: ciò non avviene con "%c". Come richiedere che spazi precedenti vengano consumati?

Esercizio 2.3

Scrivere un programma che stampi l'intero set dei caratteri ASCII, con la struttura: "carattere" ; "codice carattere".

Esercizio 2.4

Scrivere un programma che calcoli il numero più grande possibile che una variabile di tipo `int` e una di tipo `long` possono immagazzinare.
N.B: ricorda che il tipi long e int comprendono il segno.

Esercizio 2.5

Scopri il destino legato al tuo nome secondo la numerologia. I numeri del destino sono quelli compresi fra 1 e 9 più 11 e 22 (maggiori informazioni qui). Il numero del destino si ottiene sommando i codici ASCII delle lettere del nome e poi sommando le cifre di tale somma finché non viene un numero del destino. Scrivere un programma che legge uno alla volta i caratteri del nome e calcola il corrispondente numero del destino. Calcola il tuo numero.

Esercizio 2.6

Dati due interi *i*, *j* il cui valore e' **preso da tastiera**, si calcoli il risultato della divisione *k* = *i/j* di tipo double.
In seguito, si iteri sui primi decimali di *k* (massimo 10), ciascuno a distanza *p* rispetto alla virgola (la prima cifra decimale starà a distanza 0 dalla virgola), e si stampi il carattere alfanumerico associato in ASCII dopo aver aggiunto il corrispondente valore *p*.

Esempio:
Dati i seguenti valori di i,j,k:

```
int i=2
int j=3;
double k=i/j;  // = 0.6666666

Il risultato sara':
6
7
8
9
:
;
<
=
>
?
```

Esercizio 2.7

Si consideri il seguente calcolo:

```
float sum = 0;
for (int i = 0; i < 10; ++i)
{
  sum += 0.1f;
}
```

Si stampi il valore di `sum`, e si controlli se `(sum == 1.0f)`. In caso quest'ultimo test fallisca, si sostituisca l'uguaglianza con un test appropriato per confrontare numeri in virgola mobile.

Puntatori

Esercizio 2.8

Completare il seguente programma in modo tale da assegnare alla variabile `j` il valore della variabile i usando solo puntatori a char e senza usare l'istruzione di assegnamento tra interi (ad es., l'istruzione `j = i`; è proibita).

```
int i = 10;
int j = -1;
char *p, *q;
// Inserire codice qui (senza j = ...)
// ...
printf("%d == %d\n", i, j);
```

Esercizio 2.9

Definite due variabili intere `a` e `b`, calcolare la distanza `dist` in memoria tra queste variabili (tramite differenza di puntatori) e modificare il contenuto di `a` scrivendo una espressione che contiene solo il puntatore `b` e `dist`.

Esercizio 2.10

Scrivere un programma che inizializzi in memoria un puntatore a intero `p`, ne determini il valore (valore dell'indirizzo) e scelga di conseguenza la più piccola variabile che può contenere questo valore, scegliendo fra:

```
unsigned int
unsigned long int
unsigned long long int
```