

Laboratorio di Informatica

Corso di laurea triennale in Fisica

CARLO MEREGHETTI

Scopo di questa esercitazione sarà innanzitutto approfondire la dichiarazione, definizione e manipolazione di variabili di tipi diversi in C++. Impareremo come operare l'input di tali variabili mediante lo stream `cin` ed implementeremo operazioni su variabili di tipo eterogeneo. Successivamente, prenderemo in esame le *strutture di controllo* definite nell'ambito dalla *programmazione strutturata*, proponendo la realizzazione di programmi di una certa complessità.

Tutti i file (sorgenti ed eseguibili) prodotti andranno salvati sotto la directory `Lab2`.

Variabili

1. Scrivere un programma che dichiari due variabili `a` e `b` di tipo intero e due variabili `c` e `d` di tipo `float`. Il programma deve successivamente riempire tali variabili con dati opportunamente chiesti all'utente. Dichiarare anche le variabili `x` e `y` di tipo opportuno che contengano rispettivamente la somma di `a` con `c` ed il prodotto di `b` con `d`; stampare i contenuti di `x` e `y`. Stampare direttamente (senza porre il risultato in variabili d'appoggio) la divisione tra `a` e `b` e tra `c` e `d`; notare la differenza nel risultato. Come far apparire eventuali cifre decimali nella divisione tra `a` e `b`?

Strutture di controllo

1. Realizzare il programma discusso a lezione che chiede tre interi `a`, `b` e `c` e stampa le soluzioni dell'equazione $ax^2 + bx + c = 0$. Per il calcolo della radice quadrata, potete utilizzare la funzione `sqrt`: l'invocazione `sqrt(d)` restituisce la radice quadrata di `d`. Tale funzione appartiene alla libreria `cmath` che deve essere opportunamente inclusa all'inizio del programma mediante la direttiva `#include <cmath>`.
2. Realizzare il programma `MediaPesata.C` che calcola la media pesata (per numero di cfu) dei vostri voti. Per terminare l'inserimento dei dati, l'utente inserirà 0 come voto. Come miglioramento del programma, controllate che l'utente inserisca esclusivamente voti sufficienti e numero di cfu pari esclusivamente a 6 o 12.
3. Realizzare il programma `MinMax.C` che chiede all'utente una sequenza di interi terminata da 0 e stampi l'intero massimo e l'intero minimo inseriti.
4. Per utilizzare il tipo `string`, è necessario includere la libreria `string` mediante la direttiva `#include <string>` posta all'inizio del programma. La variabile `string s` può contenere una sequenza contigua di caratteri sino al primo spazio o newline e viene letta usualmente come `cin >> s`. Attenzione che se l'utente inserisse `Ciao pippo`, la variabile `s` conterrebbe solo `Ciao`. Volendo leggere una riga di testo, spazi compresi, sino al newline è necessario usare la funzione `getline(cin, s)`.

Sulle stringhe (che in realtà sono oggetti) possiamo invocare diversi metodi. Ad esempio, la lunghezza di `s` viene restituita da `s.length()`, mentre il carattere di posto `n` si ottiene scrivendo `s[n]`. È importante notare che *le posizioni dei caratteri vengono numerate a partire da 0*, per cui l'ultimo carattere di `s` si trova alla posizione `s.length() - 1`.

Una stringa è *palindroma* se letta da sinistra a destra o viceversa rimane identica. Ad esempio `anna`, `osso`, `ailatiditalia` sono stringhe palindrome. Scrivete il programma `Palindroma.C` che accetta in ingresso una stringa e ne testa la palindromia.

5. Realizzare il programma `Tabellina.C` che stampa le tabelline da quella dell'uno a quella del 10. I numeri devono essere perfettamente incolonnati e a questo proposito ricordate il carattere di tabulazione `\t`. L'output del programma dovrebbe essere come segue:

```

1  2  3  4  5  6  7  8  9  10
2  4  6  8 10 12 14 16 18 20
:
:
10 20 30 40 50 60 70 80 90 100

```

6. Il numero naturale x è *amico* del numero naturale y se e solo se la somma di tutti i divisori di x (escluso x stesso) è uguale ad y . Ad esempio, 10 è amico di 8 avendo come divisori 1, 2, 5 la cui somma è proprio 8; notare che il viceversa non è vero.

Realizzare il programma `Amici.C` che chiede all'utente due numeri e stampa la relazione di amicizia tra i due numeri (scrive, cioè, se il primo è amico del secondo, se è vero il contrario, oppure se i due numeri sono vicendevolmente nemici). Il programma deve inoltre controllare che entrambi i numeri interi inseriti siano naturali (cioè, interi non negativi), emettendo eventualmente un opportuno messaggio d'errore.

7. Realizzare il programma `TabellaAmici.C` che chiede all'utente due numeri naturali r ed s e stampa tutte le coppie (x, y) tali che $0 \leq x \leq r$, $0 \leq y \leq s$ e x è *amico* di y .

8. Realizzare il programma `SmallestNonDivisor.C` che chiede all'utente un numero naturale n e stampa il *più piccolo numero che non divide n*. È facile notare che tale programma stamperà sempre 2 se n è un numero dispari; ecco invece un esempio d'esecuzione su input pari (in neretto i dati inseriti dall'utente):

```

Inserisci un numero naturale:  720
Il più piccolo numero che non divide 720 e' 7

```

9. Realizzare il programma `Primo.C` che chiede all'utente un numero naturale e segnali la *primatà* o meno del numero. Prendere poi spunto da tale programma per realizzare il programma `IstogrammaPrimi.C` che chiede all'utente un numero naturale n e stampa un istogramma a barre le cui righe siano indicizzate dai numeri da 0 ad n e alla destra dell'indice x di ogni riga compaiano *tanti * quanti sono i numeri primi minori o uguali a x*. Ecco un esempio di esecuzione (in neretto i dati inseriti dall'utente):

```

Limite superiore dell'istogramma:  11

0:
1:
2:  *
3:  **
4:  **
5:  ***
6:  ***
7:  ****
8:  ****
9:  ****
10:  ****
11:  *****

```