

Laboratorio di Programmazione

Laurea in Bioinformatica

II Quadrimestre

20 febbraio 2008

1 JAVA: il tipo char e il costrutto switch

Esercizio Q2_1

La classe CharCfr contenga un programma che:

1. Dichiarare due variabili c e d di tipo char e ne acquisisce i valori da tastiera;
2. A fronte di opportuni confronti stampa una o più frasi fra le seguenti:
 - "c e d contengono lo stesso carattere <valore di c,d> (codice Unicode: <unicode di c,d>)"
 - "c e d contengono caratteri differenti"
 - "nella codifica Unicode il carattere <c> (<unicode di c>) precede <d> (<unicode di d>)"
 - "nella codifica Unicode il carattere <c> (<unicode di c>) precede immediatamente <d> (<unicode di d>)"

Esercizio Q2_2

Si scriva un'applicazione per la simulazione di una semplice calcolatrice. Il programma deve scrivere il risultato dopo aver ricevuto da tastiera due numeri e il simbolo dell'operazione desiderata.

Esercizio Q2_3

Si scriva un'applicazione che stampa tutte le lettere minuscole dell'alfabeto facendo uso dell'operatore postfisso di incremento. Nel codice appena scritto si sostituisca all'operatore postfisso l'operatore prefisso e si osservi il risultato.

Esercizio Q2_4

Si scriva un'applicazione che permette di specificare somme di numeri interi inserendo da tastiera stringhe come "1+13+6+123". [sugg.: si ricorra ai metodi della classe String che permettono di individuare l'indice di un carattere (+ in questo caso) e di estrarre sottostringhe]

2 JAVA: metodi statici e campi statici

Esercizio Q2_5

Si scriva un programma per la risoluzione delle equazioni di secondo grado
[sugg. si usi il metodo sqrt della class Math per il calcolo della radice quadrata]

Esercizio Q2_6

Si consideri un cerchio di raggio r il cui centro coincide con l'origine di una coppia di assi cartesiani. Si scriva un programma che:

1. generando una sequenza di N punti a caso di coordinate x e y comprese fra 0 ed r calcoli la frequenza dei punti che cadono nel quarto di cerchio di circonferenza (cioè il rapporto fra i punti che cadono nel quarto di circonferenza e il numero totale di punti generati)
2. usando il metodo al punto precedente calcolare un'approssimazione di π sapendo che la frequenza dei punti che cadono nel quarto di cerchio di circonferenza costituisce un'approssimazione del rapporto fra l'area A_q del quarto di cerchio e l'area del quadrato di lato r . Si ricordi poi che $e' 4 * A_q = \pi * r^2$.
3. si confronti il valore stimato di π con il valore della costante statica Math.PI .

3 JAVA: array e collezioni

Esercizio Q2_7

La classe OrdinaStringhe contenga un programma che:

1. inizializza un array di stringhe acquisite da tastiera
2. ordina l'array per lunghezza crescente delle stringhe, adoperando una variabile di tipo stringa come contenitore temporaneo
3. stampa l'array così ordinato

Esercizio Q2_8

La classe OrdinaStringhe2 contenga un programma che:

1. inizializza un array di stringhe acquisite da tastiera
2. ordina gli elementi dell'array per lunghezza crescente delle stringhe in un nuovo array, senza adoperare alcun contenitore temporaneo
3. stampa il nuovo array

[SUGG.: copiare nel secondo array e poi escludere la stringa più corta del primo array via via fino a suotare il primo array]

4 JAVA: array di array

Esercizio Q2_9

La classe OpMatrici contenga un programma che:

1. crea due matrici 2x2, la prima con righe [1 2] e [3 4], la seconda con righe [5 6] e [7 8].
2. esegue la visita completa delle due matrici, prima per RIGA e poi per COLONNA, stampandone il risultato.
3. ne esegue il prodotto matriciale. Per verifica, si noti che il risultato corretto e' una matrice 2x2 con righe [19 22] e [43 50].

Esercizio Q2_10

La classe `LetturaVerticale` contenga un programma che:

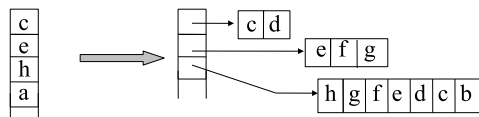
1. legge e salva in un array di stringhe una sequenza di stringhe da input fintantoché non viene immessa la stringa "FINE"
2. colloca le stringhe in una matrice di simboli avente un numero di colonne pari alla lunghezza della stringa minore, in cui la riga n-esima contiene i simboli della stringa n-esima
3. presenta a video una nuova sequenza, formata leggendo la matrice colonna dopo colonna.

Si risolva il problema prima adoperando un array di array di caratteri, poi lavorando col solo array di stringhe iniziale (ovvero senza creare la matrice di caratteri)

Esercizio Q2_11

La classe `CaratteriNelMezzo` contenga un programma che:

1. inizializza un array di 10 simboli UNICODE scelti a caso tra i codici decimali 1 e 150 adoperando il metodo statico `random()` di `Math`
2. sostituisce col carattere 'a' gli elementi nell'array che non sono alfabetici minuscoli
3. inserisce carattere 'z' nell'ultimo elemento dell'array, e infine stampa gli elementi dell'array risultante
4. per ogni elemento dell'array escluso l'ultimo, costruisce un nuovo array che contiene tutti i caratteri intermedi tra quello contenuto in quell'elemento (compreso) e quello contenuto nell'elemento di indice superiore (escluso), in avanti o all'indietro a seconda che l'elemento di indice superiore contenga un carattere rispettivamente posteriore o anteriore nell'ordinamento fissato dal codice UNICODE. Ad esempio:



5. stampa, riga dopo riga, ciascuna lista di caratteri contenuti nei nuovi array così definiti

Esercizio Q2_12

1. un costruttore Rettangolo (double b, double h), il quale costruisce un rettangolo di base b e altezza h
2. un costruttore Rettangolo (double b), il quale costruisce un rettangolo di base e altezza b (ovvero un quadrato)
3. un metodo double area(), che restituisce l'area del rettangolo
4. un metodo double perimetro(), che restituisce il perimetro del rettangolo
5. un metodo double diagonale(), che restituisce la diagonale del rettangolo
6. un metodo double diagonale(), che restituisce la diagonale del rettangolo
7. un metodo String toString(), che restituisce una rappresentazione del rettangolo nel formato

8. un metodo main che, dato un valore di partenza del perimetro, individua il rettangolo che a parità di perimetro possiede area massima al variare di base e altezza e di conseguenza ne stampa i valori di base, altezza e area nonché una sua rappresentazione. Per determinare l'area massima il metodo adopererà una procedura numerica che fa una ricerca di massimo partendo da 10000 valori scelti a caso della base compatibili col perimetro dato.

La classe contenga:

- Si progetti infine la classe `UsaCarte` contenente un metodo `main` che, appoggiandosi alle risorse della classe `CarteFrancesi` contenuta nel package `myclasses`, simula il gioco delle due carte. Il giocatore che dà le carte rimescoli 20 volte il mazzo secondo la regola di mescolamento fornita dalla classe.

6 JAVA: ricorsione

Esercizio Q2_14

Si scriva una classe che dato un intero k calcoli, sia in versione iterativa che in versione ricorsiva, il k -mo elemento della seguente serie (serie di Fibonacci): 1,1,2,3,5,8,13,21,... La serie è tale che il primo e il secondo numero sono uguali a 1, mentre ciascun elemento successivo è uguale alla somma dei due numeri che lo precedono.

Esercizio Q2_15

Si scriva una classe che si serve di metodi ricorsivi per il calcolo del fattoriale di un numero intero n inserito dall'utente e per il calcolo della potenza m^n , dove m è un secondo numero intero fornito dall'utente.

Esercizio Q2_16

Si scriva una classe che si serve di un metodo ricorsivo per la stampa di una stringa fornita dall'utente. Il metodo ricorsivo sia del tipo:

```
public static void stampainvert(String s, int i, ConsoleOutputManager out)
```

dove i parametri passati sono la stringa da stampare, un indice che determina l'indice del carattere da cui iniziare a stampare, e un oggetto di tipo ConsoleOutputManager per la stampa.

Esercizio Q2_17

Si scriva una classe che si serve di un metodo ricorsivo per ordinare un array di interi in $[0,9]$ fornito dall'utente. Il metodo ricorsivo implementi la seguente strategia in tre passi (algoritmo "mergesort"):

1. Divide l'array non ordinato in due sotto-array di uguale lunghezza (a meno di uno per lunghezze dispari)
2. Ordina ricorsivamente ognuno dei due sotto-array fino a ottenere il caso di lunghezza 1 (in questo caso viene restituito l'array stesso)
3. Ricomponi, attraverso un metodo merge di servizio, i due sotto-array ordinati in un unico array ordinato

7 PERL: variabili numeriche e stringhe

Esercizio Q2_18

Scrivere un programma PERL che calcola l'area di un triangolo di base 3 e altezza 7, e stampa il risultato a schermo.

Esercizio Q2_19

Si modifichi l'esercizio precedente in modo che venga chiesto all'utente di specificare la base e l'altezza del triangolo.

Esercizio Q2_20

Scrivere un programma che, dopo aver chiesto all'utente di inserire una stringa alfanumerica e un intero n., stampa a video la stringa secondo lo schema seguente:

```
stringastringastringa ... (n volte)

stringa stringa stringa ... (n volte)

stringa
stringa (n volte)
stringa
...
```

Esercizio Q2_21

Scrivere un programma che, dopo aver chiesto all'utente di inserire due stringhe alfanumeriche, determina se sono uguali (e stampa un opportuno messaggio a video) o quale delle due stringhe precede lessicograficamente l'altra se sono diverse. In quest'ultimo caso, stampa a video le due stringhe in ordine lessicografico.

Si risolva l'esercizio prima facendo uso del costrutto "if", poi cercando una soluzione alternativa che ne eviti l'uso.

Esercizio Q2_22

Scrivere un programma che converte una cifra fornita dall'utente da euro a lire (sapendo che 1 euro è pari a 1936.27 lire)

Esercizio Q2_23

Dopo aver chiesto all'utente di inserire un carattere da tastiera, si stampi un messaggio che indica se il carattere è un numero fra 0 e 9, una lettera minuscola, una lettera maiuscola, o un tipo diverso di carattere.

8 PERL: array e costrutti di controllo per l'iterazione

Esercizio Q2_24

Si realizzino in Perl le seguenti operazioni su array:

1. Si inizializzi un array, di nome @array1 e di lunghezza 5, con i seguenti valori: aa, zeta, mio, cicala, fff.
2. Si stampino a schermo i valori dell'array su una riga e separati da uno spazio
3. Si stampino i valori dell'array in seconda, terza e quarta posizione.
4. Si eseguano nell'ordine le seguenti operazioni sull'array (usare gli operatori pop, push, etc.): eliminazione dalla coda dell'array degli ultimi tre elementi; inserimento in coda dell'elemento "tuo".
5. Si stampi la nuova lunghezza dell'array e l'array stesso in ordine sia crescente che decrescente.

Esercizio Q2_25

Si modifichi lo script precedente in modo che l'utente costruisca l'array elemento per elemento.

[Sugg: si utilizzi un costrutto while con istruzione di interruzione ("last") in caso di valore inserito pari ad es. a 0]

Esercizio Q2_26

Si scriva uno script che realizza le seguenti operazioni:

1. Ricava il numero di lettere maiuscole (dimensione dell'intervallo [A,Z]) e lo comunica all'utente. Stampa inoltre su una riga tutte le lettere maiuscole.
2. Chiede all'utente di inserire un numero intero k compreso tra 2 e il numero di lettere maiuscole ricavato (itera la richiesta fino a quando l'utente non inserisce un intero nel range richiesto)
3. Stampa su una riga le prime k lettere maiuscole
4. Crea un nuovo array che viene usato per contenere le prime k lettere maiuscole raggruppate in stringhe di lunghezza crescente. Ad es. per k=6, st2[0]=A, st2[1]=BC, st2[2]=DEF.
5. Stampa il nuovo array.

Esercizio Q2_27

Si realizzi un programma che permette di gestire un array in modo interattivo, fornendo all'utente il seguente menu di scelte:

che operazione si vuole fare? i (inserisci), c (cancella), s (sort), 0 (termina):

A seguito di una data scelta, il programma esegua le operazioni necessarie e stampi il nuovo array. Nel caso "i" si gestisca interattivamente l'inserimento di un nuovo elemento.

Esercizio Q2_28

Si cheda all'utente di immettere una stringa di lunghezza arbitraria costituita dalle sole quattro lettere A,C,G e T (lettere diverse sono comunque ammesse). Si mutino poi tutte le lettere T in U e tutte le eventuali lettere diverse da A,C,G e T, in N. Si stampi infine la stringa risultante.

[Sugg.: si usi la funzione substr (\$stringa,\$i,\$j) per estrarre da \$stringa la sottostringa di lunghezza \$j che inizia dalla posizione \$i]

Esercizio Q2_29

Si costruisca un vettore @basi contenente i caratteri A, T, G, e C. Si costruisca poi un vettore di lunghezza 10 formato da una combinazione scelta a caso degli elementi di @basi. Si iteri il passo precedente un numero prefissato di volte concatenando tutti i vettori generati in una variabile stringa. Si stampino i singoli vettori uno per riga e infine la variabile stringa contenente la concatenazione dei vettori.

[Sugg: per generare un valore intero scelto a caso nell'intervallo [0,\$len] si usi l'istruzione "int rand(\$len)"].

9 Java: esercizi di ricapitolazione

Esercizio Q2_30

Si progetti una classe di nome `Treno`, la quale istanzia oggetti in grado di rappresentare la potenza di traino della locomotiva, il numero di vagoni e il peso di un treno. Si assuma che l'oggetto abbia senso solo se il peso del treno non supera la potenza e se il numero di vagoni è inferiore a 10: quest'ultimo campo dovrà essere pubblicamente accessibile a tutte le classi che intendano adoperare le risorse fornite dalla classe `Treno`. La classe contenga:

1. un costruttore `Treno(int p)`, il quale costruisce un oggetto che modella un treno formato dalla sola locomotiva di potenza `p`
2. un costruttore `Treno(int p, int n, int m)`, il quale se possibile costruisce un oggetto che modella un treno formato dalla locomotiva di potenza `p` più `n` vagoni ciascuno pesante `m`, altrimenti restituisce la sola locomotiva di potenza `p`
3. un metodo booleano `accodaVagone(int m)`, che se possibile aggiunge al treno modellato un vagone di massa `m`, e in tal caso restituisce vero; altrimenti restituisce falso
4. un metodo `int sganciaVagone()`, che se possibile elimina il vagone di coda del treno modellato e in tal caso restituisce la sua massa; altrimenti restituisce 0
5. un metodo `int quantoPesa()`, che restituisce il peso del treno modellato
6. un metodo `int quantoResta()`, che restituisce il peso che può essere ancora aggiunto al treno
7. un metodo `int quantiVagoni()`, che restituisce il numero dei vagoni che attualmente formano il treno
8. un metodo `String toString()`, che restituisce il modello del treno nel formato "`<p>m1:m2:...`", in cui `p` è la potenza della locomotiva e ciascun elemento tra caratteri due punti contiene la massa `m` del vagone `i`-esimo.

Per modellare i vagoni si consiglia di adoperare un array di interi `mass[i]`, in cui ciascun elemento contiene la massa `m` del vagone `i`-esimo. La classe dovrà far parte di un package di nome `myclasses` accessibile da ambiente Java.

Si scriva infine una classe di nome `TestTreno` che contenga delle semplici istruzioni per la verifica del corretto funzionamento dei metodi della classe `Treno`.

Esercizio Q2_31

Si scriva una classe di nome `UsaTreno` contenente un metodo `main` che, appoggiandosi alle risorse della classe `Treno` contenuta nel package `myclasses`, simula la creazione di due treni dotati di locomotive di potenza di traino uguale a 50, a cui sono accodati al più 10 vagoni di peso casuale variabile tra 1 e 10. Il metodo successivamente scambia i vagoni in modo da bilanciare quanto più possibile il peso dei due treni secondo una strategia "greedy", ovvero sganciando tutti i vagoni dai due treni e poi assegnando via via il vagone più pesante al treno in quel momento più leggero.

Esercizio Q2_32

Si progetti una classe di nome `ZeroUno`, la quale istanzia oggetti che modellano successioni formate da zero e uno. La classe mantiene la successione in un campo stringa privato (già dato nel testo) e fornisce:

1. un costruttore che, invocato senza parametri, costruisce un oggetto che rappresenta una successione vuota
2. un costruttore che, ricevuto un parametro `s` di tipo `String`, costruisce un oggetto che rappresenta la più grande successione di zero e uno contenuta in `s` nell'ordinamento dato dalla stringa. Ad es., se `s` è "0er10rt0rt1ghf10h", il campo `zerouno` dell'oggetto varrà "0100110"
3. un costruttore che, ricevuto un parametro `n` di tipo `int`, costruisce un oggetto che rappresenta una successione lunga `n` di zero e uno casualmente distribuiti
4. un metodo `toString()` che restituisce la stringa che rappresenta la successione di zero e uno in cui ogni termine è seguito da virgola: se, ad esempio, la successione è 0100 allora la stringa che la rappresenta sarà "0,1,0,0,"
5. un metodo `toArray()` che restituisce un array di boolean in cui l'elemento di indice `i` è `true` o `false` a seconda che il termine `i`-esimo della successione (numerata partendo da zero) sia rispettivamente uguale a uno o zero
6. un metodo `somma()` che RICORSIVAMENTE somma i termini della successione e restituisce un intero contenente la somma. Se ad esempio la successione è 001100010 allora `somma()` restituirà il valore 3

La classe inoltre contenga un metodo `main()` il quale, costruiti due oggetti `ZeroUno` rispettivamente adoperando una stringa e un intero come parametri per la creazione delle successioni a essi associate, dà successivamente la rappresentazione in stringa e la somma di entrambe le successioni.

Esercizio Q2_33

Si progetti una classe di nome `GrandiIntPositivi`, la quale istanzia oggetti che modellano numeri interi positivi di 20 cifre. La classe mantiene cifre comprese tra 0 e 9 in un campo array privato `int[]` cifre di dimensione costante `MAX` posta a 20. La classe fornisce:

1. un costruttore che, invocato senza parametri, costruisce un oggetto che rappresenta lo zero, ovvero inizializza l'array ponendo tutti gli elementi a zero;
2. un costruttore che, ricevuto un parametro di tipo `String`, costruisce un oggetto che rappresenta il numero contenuto nella stringa. Per semplicità si assuma che la stringa contenga solo cifre decimali. Inoltre, lo stesso costruttore sia eventualmente in grado di scartare la coda di stringhe lunghe più di 20 cifre;
3. un metodo `GrandiIntPositivi piu(GrandiIntPositivi g)` che restituisce, in forma di un nuovo oggetto, la somma dell'oggetto che esegue il metodo con quello passato come parametro. Se la stessa somma supera le 20 cifre allora restituisce l'oggetto che rappresenta il più grande intero positivo di 20 cifre;
4. un metodo `GrandiIntPositivi perInt(int a)` che restituisce il prodotto dell'oggetto che esegue il metodo per l'intero positivo passato come parametro. Il metodo deve essere implementato in forma RICORSIVA, facendo uso del metodo "piu".
5. un metodo `toString()` che stampa in forma di stringa lunga 20 caratteri il numero rappresentato dall'oggetto che utilizza il metodo;
6. un metodo booleano `equals(GrandiIntPositivi g)` che verifica l'uguaglianza tra due oggetti in base al numero rappresentato.

La classe contenga infine un metodo `main` con semplici istruzioni di test della classe (somma fra due oggetti, prodotto per un intero positivo e stampa).